

18TBIP-
C144194
-01

보안 과제(), 일반 과제(○) / 공개(○), 비공개()발간등록번호()
국토교통기술화지원 사업 최종 보고서

R&D / 18TBIP-C144194-01

지하공간
재난상황
인지 IoT
플랫폼 및
재난대응
긴급대피
경로 유도
플랫폼
개발
최
종
보
고
서

2020

국
토
교
통
과
학
기
술
부
진
흥
원

지하공간 재난상황 인지 IoT 플랫폼 및 재난대응 지능형 긴급대피 경로 유도 플랫폼 개발 최종보고서

2020.09.14.

주관연구기관 / (주)승화기술정책연구소
공동연구기관 / 미래아이티(주)

국 토 교 통 부
국토교통과학기술진흥원

제출문

제 출 문

국토교통과학기술진흥원장 귀하

‘지하공간 재난상황 인지 IoT 플랫폼 및 재난대응 긴급대피 경로 유도 플랫폼 개발’(연구개발
기간 : 2018. 04. ~ 2020. 08.) 과제의 최종보고서 10부를 제출합니다.

2020.09.14.

주관연구기관명 : (주)승화기술정책연구소 박미연 (인)

공동연구기관명 : 미래아이티(주) 송정화 (인)

주관연구기관책임자: 임종권

공동연구기관책임자: 박길주

국토교통부소관 연구개발사업 운영규정 제29조에 따라 최종보고서 열람에
동의합니다.

보고서 요약서

과제 고유 번호	18TBIP-C14 4194-01	해당 단계 연구 기간	2019.01.01. ~ 2020.08.31	단계구분	2차년 / 2차년
연구사업명	중사업명	국토교통기술사업화지원			
	세부사업명	국토교통기술사업화지원			
연구과제명	대과제명	-			
	세부과제명	지하공간 재난상황인지 IoT 플랫폼 및 재난대응 지능형 긴급 대피 경로유도 플랫폼 개발			
연구책임자	임종권	해당단계 참여연구원 수	총: 37명 내부: 명 외부: 명	해당단계 연구비	정부:943,000천원 민간:377,200천원 정부외: 천원 상대국: 천원 계:1,320,000천원
		총 연구기간 참여연구원 수	총: 37명 내부: 명 외부: 명	총 연구비	정부:1,054,054천원 민간:420,000천원 정부외: 천원 상대국: 천원 계:1,470,000천원
연구기관명 및 소속 부서명	(주)승화기술정책연구소			공동연구기관 미래아이티(주)	
국제공동연구	상대국명:			상대국 연구기관명:	
위탁연구	연구기관명:한국철도공사			연구책임자:백인철	

※ 국내·외의 기술개발 현황은 연구개발계획서에 기재한 내용으로 같음

연구개발성과의 보안등급 및 사유	
-------------------------	--

9대 성과 등록·기탁번호

구분	논문	특허	보고서 원문	연구시설 · 장비	기술요약 정보	소프트 웨어	화합물	생명자원		신품종	
								생명 정보	생물 자원	정보	실물
등록·기 탁 번호		9				3					

국가과학기술종합정보시스템에 등록된 연구시설·장비 현황

구입기관	연구시설 · 장비명	규격 (모델명)	수량	구입연월일	구입가격 (천원)	구입처 (전화)	비고 (설치장소)	NTIS 등록번호

요약(연구개발성과를 중심으로 개조식으로 작성하되, 500자 이내로 작성합니다)

보고서 면수

116면

요약문

연구의 목적 및 내용	지하공간 재난상황 인지 IoT 플랫폼 및 재난대응 지능형 긴급 대피 경로 유도 플랫폼 개발 지하 구조물에서 발생하는 재난의 경우, 감지 및 예측에 대한 자료 수집이 센서에 의해 제공되고, 이를 통해 수집된 자료를 분석하거나 판단하여 최종 대비 및 대응이 이루어지고 있다. 센서 기기에 데이터 판단 로직을 적용하여 긴급 재난 상황을 인식 및 전달, 대피 및 피난지원기능을 동시에 할 수 있는 기술을 적용한 제품과 S/W를 개발하여 테스트 베드에 설치하여 실용성을 검증하고 상용화를 위한 문제점을 보완하여 최종 플랫폼 제품으로 개발 할 계획이다. 지하공간(지하철) 재난 발생 시 신속, 정확한 상황 인지를 위한 모듈 제작 및 인명 피해를 최소화하기 위한 가이드라인을 제시하는 IoT 플랫폼 개발과 재난으로 인한 제한적인 상황에서 주어진 정보를 바탕으로 최대한 정확한 상황인지 및 판단을 위한 의사결정 지원 플랫폼 및 솔루션 개발을 수행한다. 이때 효율적인 의사결정을 위하여 다양하게 수집된 빅데이터 정보를 바탕으로 위험성을 평가하여 대피경로를 분산하는 모듈을 개발한다. 그리고 중앙 관리 시스템에서 재난전조감지 및 재난 발생 시 관할 구역에 있는 이용객이 안전하게 대피할 수 있는 위치정보를 전달하고 대피를 유도할 수 있는 플랫폼 개발과 실용성을 위한 현장 대응 가이드 및 시나리오 개발하고 현장 테스트를 수행한다.				
연구개발성과	1) 센서 및 카메라 모듈 IoT 플랫폼 개발 - 센서 및 카메라 모듈 개발 및 제작 - 유무/무선/LTE 3중 네트워크 망 구성 2) 모니터링 및 상황인지 모듈 개발 3) 의사결정 지원 플랫폼 및 솔루션 개발 - 위험 상황전파 및 대피의사결정 지원(주의/경계/관심/심각) 모듈 - 재난상황에 대한 의사결정 지원을 위하여 재난 전조 상황 표출 및 대응 솔루션 개발 4) 위치정보 지원 플랫폼 - 유닛별/현장별 위치정보 DB 구축 - 공간 및 위치 기반 분석 모듈 개발 5) 긴급대피 유도 플랫폼 개발 - 시나리오별/상황별 패널 안내메시지 표출 - 대피경로 분석 및 대피 경로 유도 안내메시지 표출 - 유도등, 대피경로안내패널 고도화 6) 빅데이터 정보 기반 대피 분산 모듈 개발 7) 현장대응 관리 시스템 개발 8) 현장 대응 훈련 시나리오 개발 및 실용화 테스트				
연구개발성과의 활용계획 (기대효과)	본 연구를 통해 개발되는 지하공간(지하철)에 대한 재난 대응 시스템의 경우는 재난발생 시 해당 센서를 통해 수집되는 자료를 바탕으로 자체적으로 대응 행동을 디스플레이 할 수 있는 기능이 포함될 것이며 이를 통해 중앙관제센터에서 하달하는 대응 정보가 전송되지 않는 극한 상황에서도 자체적으로 대응 절차를 특정다수에게 원활하게 전달할 수 있도록 개발하여 다중이용시설에 대한 재난대응기술을 확보할 수 있을 것으로 판단된다.				
국문핵심어 (5개 이내)	대피경로	지하공간	재난예방	사물인터넷(IoT)	스마트 센싱
영문핵심어 (5개 이내)	Evacuation Plan	Underground	Hazard mitigation	Internet of things	Smart Sensing

※ 국문으로 작성(영문 핵심어 제외)

< 목 차 >

1. 연구개발과제의 개요	6
2. 연구수행내용 및 성과	11
3. 목표 달성도 및 관련 분야 기여도	116
4. 연구개발성과의 활용 계획 등	120

1. 연구개발과제의 개요

가. 사업화 추진 배경

○ 국내·외 지하공간에서 예기치 못한 재난 시 초기 대응을 위한 사물인터넷 기반의 재난 안전 시스템 구축 사례 및 4차산업혁명 기술(IoT, 빅데이터 등)을 접목하여 긴급 대응이 가능한 시스템은 안전·재난 분야에서는 많지 않은 상태임

○ 미래 성장동력 지능형 사물인터넷 기술과 재난안전관리 스마트 시스템 기술을 동시에 확보하여 장기적 산업시장에서 기업 경쟁력 강화와 창조적 산업생태계 구축을 선도할 수 있는 선도 기술 확보

○ 현재 운영되고 있는 국내 전국 700여개의 기존 지하철 역사, 장대 철도 터널구간, 대심도 터널 구간 등의 소방 방재 시스템(화재탐지기, 유독가스탐지 등)과 연동하여 활용 가능

○ 또한, 최근 밀양 00병원 화재 등 다양한 안전사고 및 재난 발생 시 나타나고 있는 국가 재난 대응 체계의 미흡함은 국민들의 불안을 초래하고 있어 건축물 등 다양한 분야에 적용 가능한 시스템으로 활용 가능

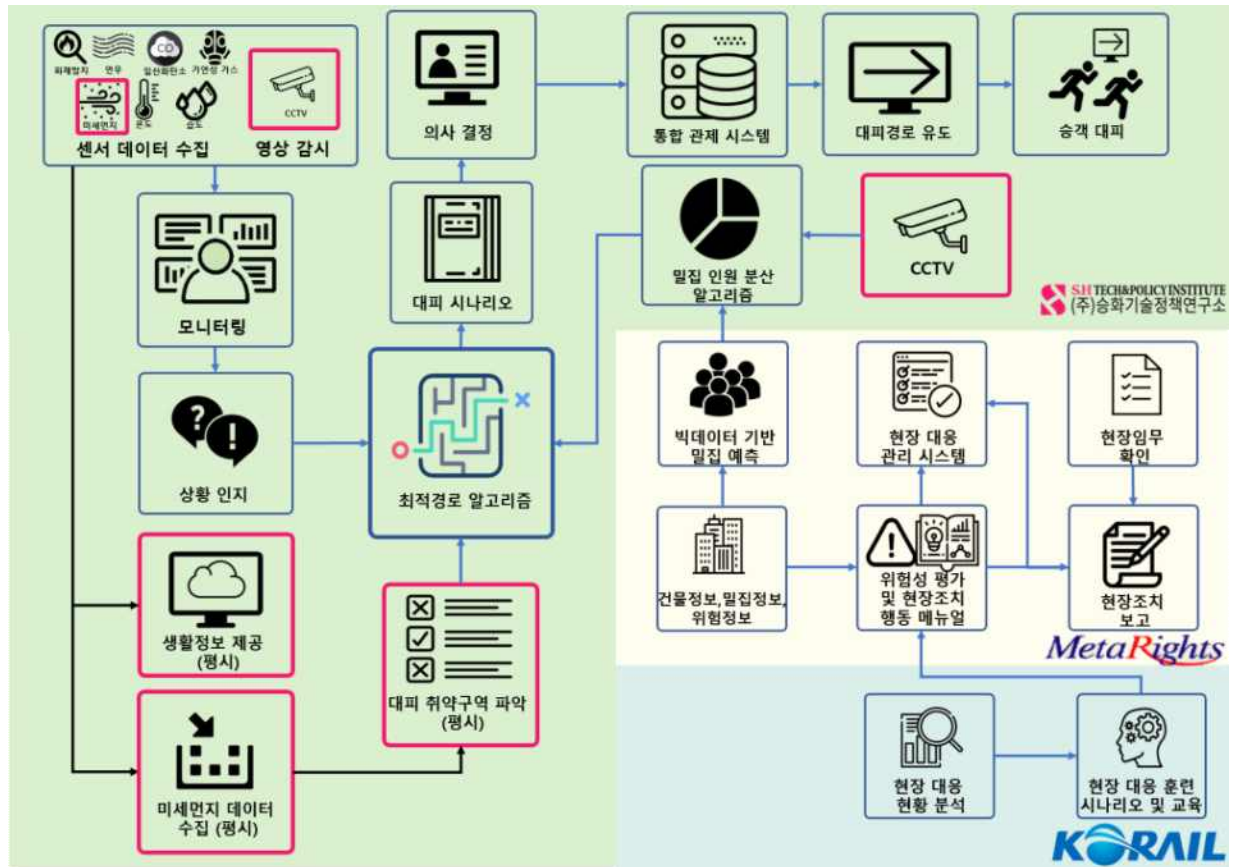
나. 사업화 대상기술 요약

구분	기존 보유기술 또는 이전받은 기술	사업화 기술
▪과제명	M2M 기반 지하공간(지하철) 재난 대응 대화형 스마트네트워크 시스템	지하공간 재난상황 인지 IoT 플랫폼 및 재난대응 지능형 긴급 대피 경로 유도 플랫폼 개발
▪사업화 기술 및 분야	■ 자체보유 □ (이전 받은 기관명)에서 기술 이전	□ 공공기술 중소기업 이전 및 사업화 ■ 중소기업 보유기술 및 아이디어 사업화 □ 국토교통 안전기술 사업화 □ 서비스 R&D 융합기술 사업화

■사업화 대상이 되는 기술의 지식재산권 * 특허, 실용신안 등 유형별 기재	(특허번호) 1015697230000		(특허번호) 10-2018-0125314	
	(특허명) 다점 송출식 다중이용 시설 가스 탐지 방법		(특허명) 공기질 지속 측정을 이용한 지하공간 대피 유도 방 법	
	(권리자) 주식회사 승화기술정 책연구소		(권리자) (주)승화기술정책연구소	
■기존기술과 비교하여 성능 개선 등 차별화 내용		기존	신규	차별성
	센서	산소, 온도, 습 도, 기압, CO, CO2, 시안화 탄소, 이산화질 소	산소, 온도, 습 도, CO, 메탄, 가연성가스(시 안화탄소 포 함), 불꽃탐지, 연무(이산화질 소 포함), 미세 먼지	CCTV 추가되 어 영상 데이 터 수집 및 분 산, 현장감시.
	대피경로 알고 리즘	MNWST알고리 즘	aStart알고리즘	최단경로 계산 에서 최적 경 로 계산으로 변경.
	통신 방식	LoRA를 이용 한 무선 통신 방식.	RJ45(유선)통 신 및 무선 통 신의 혼용방식.	고주파수 무선 통신은 복잡한 공간에서 음영 지역이 발생할 수밖에 없음.

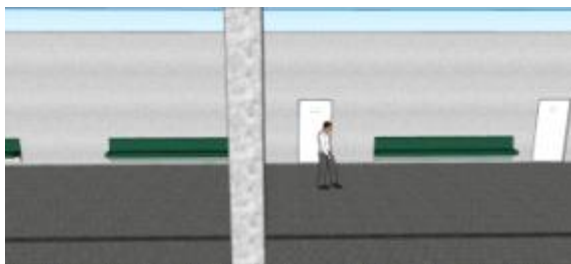
○ 사업화 대상 기술의 개념도

시스템도

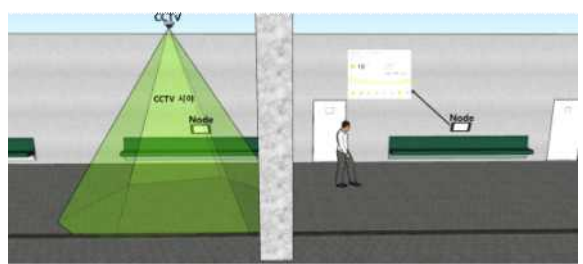


평시

시스템 미구축

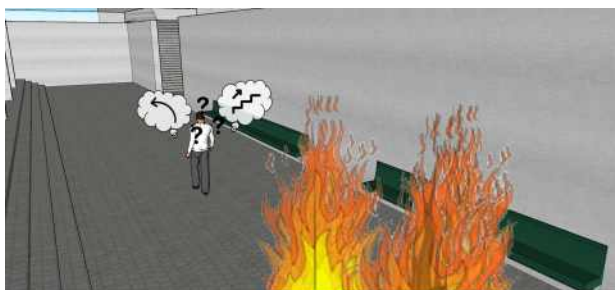


시스템 구축

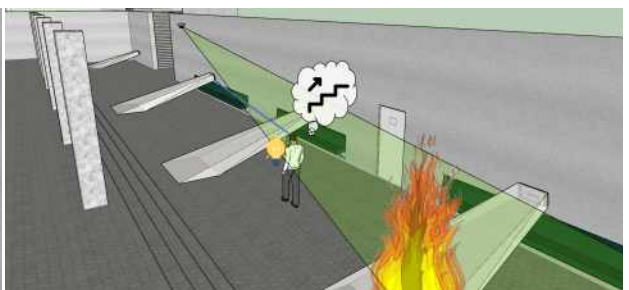


재난시

시스템 미구축

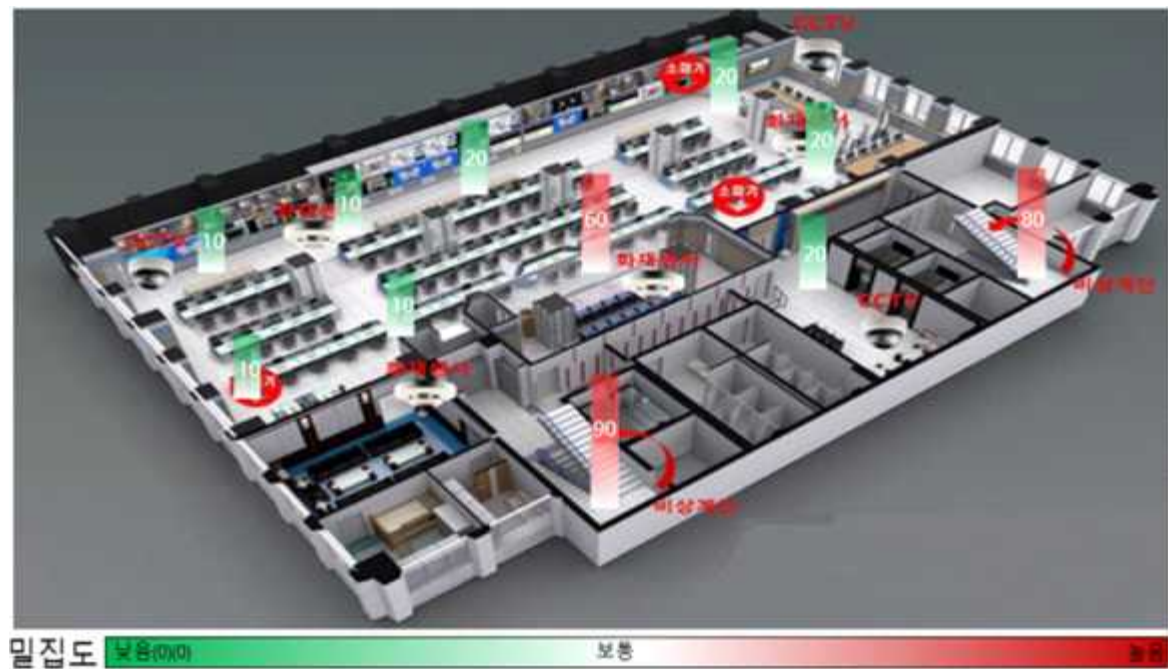


시스템 구축





밀집도 분석 예시



○ 국내외 유사기술과 수준 비교

구분	해당기술	CONERS	국외 (지멘스 Total Building Solution)
데이터 수집	▪온습도 ▪불꽃 탐지 ▪가연성 가스 ▪연무 ▪미세먼지 센서 ▪CCTV 영상 데이터 취득	▪일산화탄소 ▪연무 ▪담배 ▪유기용제	▪공기질 센서(Symaro) ▪이슬점 센서(Symaro) ▪유량 센서(Symaro) ▪온습도 센서(Symaro) ▪압력 센서(Symaro) ▪태양열 센서(Symaro) ▪방사선 센서(Symaro) ▪바람 센서(Symaro)
센서 종류	▪8개	▪4개	▪n개
모듈화	▪기성 하우스 (2차년도 자체 개발 예정)	▪자체 개발	▪자체 개발
네트워 크	▪유/무선	▪무선(LoRA)	▪무선(LoRA)
대피경 로 표출	▪일체형 모니터 ▪피난 방향 유도선	▪센서 및 안내등 ▪피난 유도등 ▪음성안내 ▪지면 매립형 안내 모듈	▪형광등 이용 플래싱 ▪출입구 양옆 안내등
분석방 식	▪분산식	▪중앙 집중식	▪중앙 집중식
혼잡도 분석	▪빅데이터 및 장치를 이용한 예측 및 실시간 혼잡도 분석	▪병목현상 예측 및 음성안내	
취약구 역 판단	▪센서를 이용한 환기 성능 분석 가능.	▪없음	▪없음
피난 유도	▪최단, 최적 회피/우회 경로 산출. ▪안전 대피로 생성시간 1초이내	▪안전 대피로 생성시간 1초이내	▪안전 대피로 생성시간 6.65초 ILOG CPLEX(IBM)
금액	▪모듈당 비용 약 60만원. ▪시공비 2600만원. (2016년 개포동역 시공 기준.)		▪해외 기술 도입시 비용 및 유지관리 비용이 과도하게 발생.

2. 연구수행내용 및 성과

가. IoT피난대피 유도 플랫폼 개발

(1) IoT Node 모듈 개발

IoT Node모듈 개발은 이용객 이용 공간에 설치되어 설치된 구역의 (반)실시간 온도, 습도, 메탄가스, 연무, 가연성 가스, 불꽃탐지, 일산화탄소, 대기오염도, 미세먼지등의 데이터를 수집 및 유/무선 네트워크를 사용하여 관리서버(마스터)측 데이터베이스에 전송하고, 평/재난 상황 알림 및 재난 상황시 피난 유도가 가능한 H/W와 S/W가 통합된 모듈 제작을 목표로 제작되었으며 아래와 같은 과정을 통하여 제작되었다.

(가) 설계

①함체 설계

일반적으로 함체라 함은, 흔히 Case로 불리며, 각종 전자제품, 반도체장비 및 통신용 장비의 외장케이스로서 옥외형과 옥내형 두 종류가 있으며, 주요 기능은 방수/방습, 용량에 맞는 방열 구조, 내장된 부품을 보호하는 기능을 내제한다. 위와 같은 정의에 따라, 본 과제에서는 지하공간의 다습하며, 먼지가 많은 상황을 고려하여 부식에 강한 SUS 함체에 분체 도장으로 마무리 하는 방식으로 제작하였다.

○ 요구사항 정리

IoT모듈 함체 내에 내장되어야 할 부품의 목록은 23인치 디스플레이, 센서 7종(가연성가스, 일산화탄소, 공기질, 화재탐지, 연무, 온습도, 미세먼지), 제어용 SBC, 통신용 RJ45 Hub, 전원 공급용 SMPS(Switching Multi Power Supply)로 선정되었으며, 현장에 설치될 방식에 대하여는 1차로 브라켓을 벽면에 설치하고, 2차로 IoT모듈의 후면에 홈 내지는 브라켓을 장착하여 결합하는 방식으로 설계하였다. 이 과정을 통하여 1차로 설계된 함체 외형은 그림 1. 과 같다.

○ 설계 변경

1차 설계안의 설계를 통하여 시제품 제작중, SUS를 파내는 방식으로 제작할 시, 모듈의 무게가 이론중량으로 시제품의 케이스 무게만((0.63(폭)x0.4(길)x0.1(두께))x7.98(316(L)= 0.201096)200kg에 가깝게 나오는 상황으로 벽체 브라켓이 IoT모듈의 무게를 견딜수 없어 소재의 변경이 필요하였으며, Sus와 오염과 먼지에 강하면서도 비중이 가벼운 알루미늄을 이용하여 제작하는 방식으로 변경하였으며, 관련하여 좀 더 유연한 디자인 설계가 가능해짐에 따라 추가로 디자인 설계 작업을 통하여 그림 2. 와 같이 2차 모듈 설계를 실시 하였으며, 최종 안으로 선택하였다.

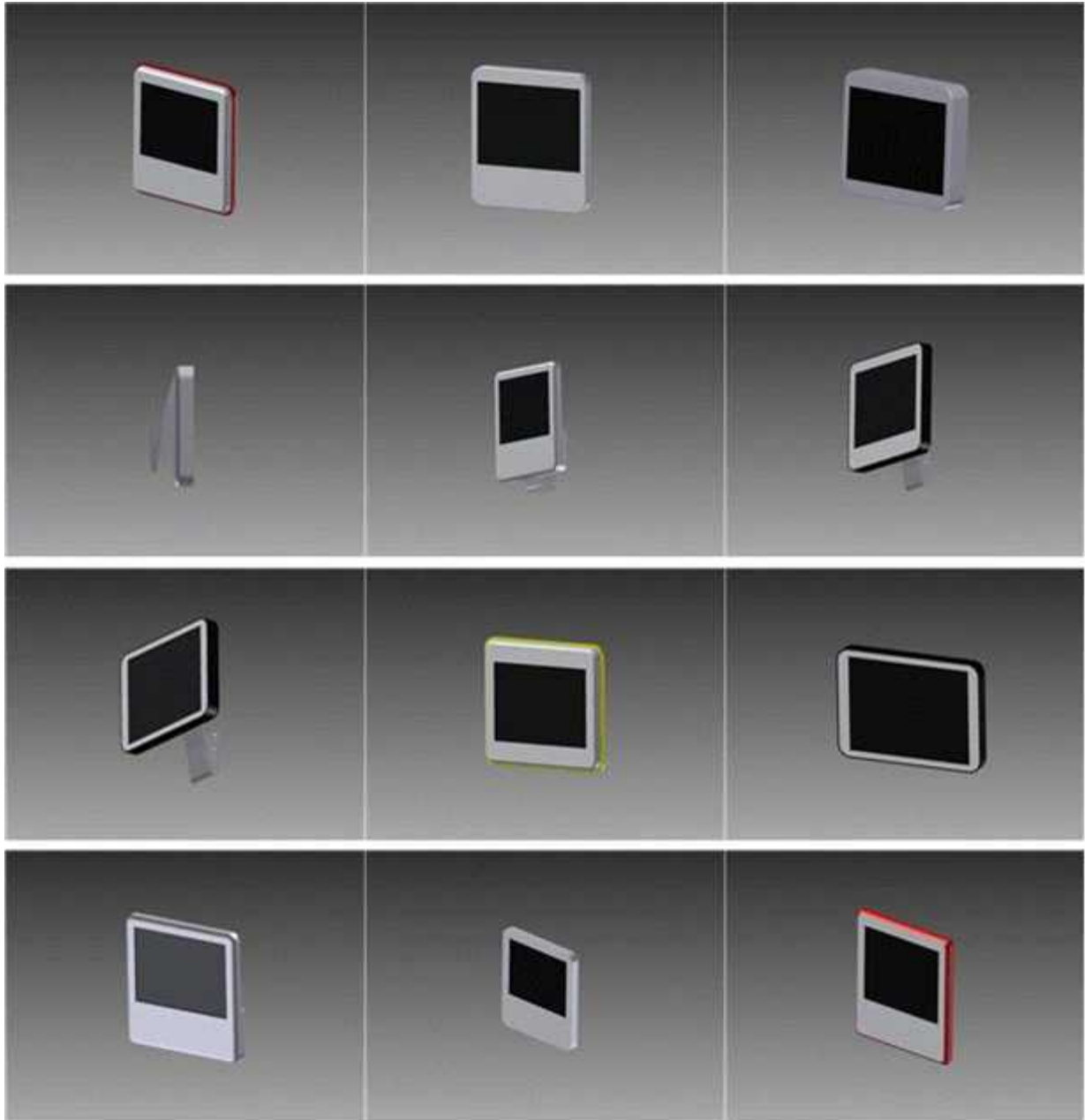


그림 1. IoT 모듈1차 설계안





그림 2. IoT 모듈 2차(최종) 설계안

○ 디스플레이 선정

최종적으로 설계된 그림2의 IoT모듈의 설계안에 따라 IoT모듈상에 23인치의 디스플레이를 장착하게 되었으며, 소방법상 축광표지의 성능인증 및 제품검사의 기술기준 의 9조에 의한 기준을 충족할 수 있도록 200ccd(칸델라)의 밝기를 가진 Newsync社의 B230h패널을 사용하였다.

- 제9조(축광시점) 축광무도표지 및 축광위표지의 표시면은 0h 상태에서 1시간이상 발광한 후 300h 밝기와 광원으로 20분간 조사시킨 상태에서 다시 무원조도를 0h로 하여 축광시점을 실시하는 경우 다음 각 호에 적합하여야 한다.
1. 5분간 발광시킨 후의 휘도는 1cd/m² 이상이어야 한다.
 2. 10분간 발광시킨 후의 휘도는 1cd/m² 90mcd/m² 이상이어야 한다.
 3. 20분간 발광시킨 후의 휘도는 1cd/m² 24mcd/m² 이상이어야 한다.
 4. 60분간 발광시킨 후의 휘도는 1cd/m² 7mcd/m² 이상이어야 한다.

그림 3. 소방법상 축광표지의 성능인증 및 제품검사의 기술기준

SPECIFICATION

모델명	비트엠 Newsync B230H 그레이스
패널	광시야각 패널
화면크기	58.42cm(23") / 16 : 9
최적해상도	1920 x 1080 [FULL HD]
컬러수	16.7M
동적명암비	3,000 : 1
밝기	200cd/m ²
응답속도	8ms
시야각	178도 / 178도
스피커	2W+2W
입력단자	HDMI, D-Sub, Audio, 전원in
소비/대기전력	18W / 0.5W
VESA	2홀 / 75mm
구성품	HDMI 케이블, 전원 어댑터, 전원케이블
인증번호	MSIP-REI-btm-B230H

그림 4. 제조사 고시 디스플레이 모듈 성능표

② 센서 선정

모니터링 모듈에 들어갈 센서는 일산화탄소, 가연성가스, 공기질, 화재탐지, 연무, 온습도, 미세먼지 센서이며 각 센서에 대한 설명 및 본 과제에서의 활용은 아래와 같다.

○ 일산화탄소(CO)센서

- 일반사항

일반적으로 CO의 기준범위는 1시간 지속 노출 시 허용농도는 25ppm 이하이며 800ppm 이상일 경우 2시간 내 의식을 잃고 사망할 가능성이 있으며 3,200ppm 이상일 경우 30분 이내로 사망할 수 있다. 본 연구에서 사용한 CO 센서는 HANWEI ELECTRONICS社의 MQ-7를 사용하였다. MQ-7은 크기가 작으며 알코올 및 습도의 영향이 적다. 또한, 저전력 소모와 긴 사용수명으로 경제적인 장점을 가지고 있다. 일산화 탄소 센서의 자세한 Spec과 Hardware는 다음과 같다.

SPECIFICATIONS			
A. Standard work condition			
Symbol	Parameter name	Technical condition	Remark
V_s	circuit voltage	$5V \pm 0.1$	Ac or Dc
$V_H(H)$	Heating voltage (high)	$5V \pm 0.1$	Ac or Dc
$V_H(L)$	Heating voltage (low)	$1.4V \pm 0.1$	Ac or Dc
R_L	Load resistance	Can adjust	
R_0	Heating resistance	$33 \Omega \pm 5\%$	Room temperature
$T_H(H)$	Heating time (high)	60 ± 1 seconds	
$T_H(L)$	Heating time (low)	90 ± 1 seconds	
PRI	Heating consumption	About 350mW	
b. Environment conditions			
Symbol	Parameters	Technical conditions	Remark
T_{stg}	Storage temperature	$-20^\circ\text{C} \sim 50^\circ\text{C}$	
T_{op}	Working temperature	$-20^\circ\text{C} \sim 50^\circ\text{C}$	Advice: using scope
RH	Relative humidity	Less than 95%(RH)	
O ₂	Oxygen concentration	21%(stand condition) the oxygen concentration can affect the sensitivity characteristic	Maximum value is over 2%
c. Sensitivity characteristic			
Symbol	Parameters	Technical parameters	Remark
R_s	Surface resistance OI sensitive body	2~20k	In 100ppm Carbon Monoxide
$\pm (300 \sim 100\text{ppm})$	Concentration slope rate	Less than 0.5	$R_0 (100\text{ppm})/R_s (100\text{ppm})$
Standard working condition	Temperature: $-20^\circ\text{C} \sim 2^\circ\text{C}$ relative humidity: $65\% \pm 5\%$	$R_L 10k \Omega \pm 5\%$	
	$V_s 5V \pm 0.1V$ $V_H 5V \pm 0.1V$ $V_H 1.4V \pm 0.1V$		
Prelent time	No less than 48 hours	Detecting range: 20ppm~2000ppm carbon monoxide	

그림 5. MQ-7 센서 Spec

- Calibration

MQ-7 센서의 민감도 특성 및 온도, 습도와 관계는 다음의 그래프와 같으며 이 관계식에서 도출한 Calibration 값을 일산화탄소 감지 로직에 적용하였다.

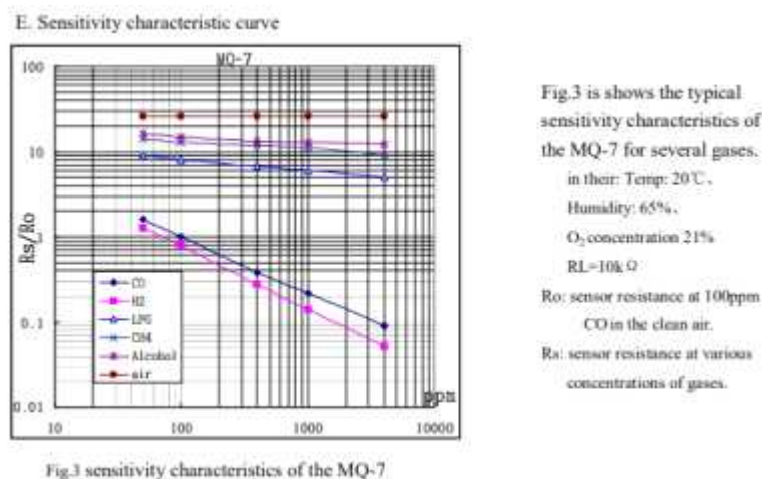


그림 6. 일산화탄소 응답 특성

○ 가연성 가스 센서

- 일반사항

가연성 가스의 생물학적 노출기준 범위는 국내 규정상, 프로판의 경우 TWA - 1,000ppm, 1800mg/m³, 노말부탄의 경우 TWA - 800ppm, 1900mg/m³, 이소부탄 TWA - 800ppm, 1900mg/m³이다. 본 연구에서 사용한 CO 센서는 HANWEI ELECTRONICS社의 MQ-6를 사용하였으며, 가연성 가스 센서의 자세한 Spec과 Hardware는 다음과 같다.

A. Standard work condition			
Symbol	Parameter name	Technical condition	Remarks
V_C	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	AC OR DC
P_L	Load resistance	20K Ω	
R_H	Heater resistance	33 Ω ± 5%	Room Tem
P_H	Heating consumption	less than 750mw	
B. Environment condition			
Symbol	Parameter name	Technical condition	Remarks
T_{ao}	Using Tem	-10℃~50℃	
T_{as}	Storage Tem	-20℃~70℃	
R_H	Related humidity	less than 95%RH	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%
C. Sensitivity characteristic			
Symbol	Parameter name	Technical parameter	Remarks
R_s	Sensing Resistance	10K Ω - 60K Ω (1000ppm LPG)	Detecting concentration scope: 200-10000ppm LPG , iso-butane, propane, LNG
α (1000ppm/4000ppm LPG)	Concentration slope rate	≤ 0.6	
Standard detecting condition	Temp: 20℃ ± 2℃ Humidity: 65%±5%	V_C :5V±0.1 V_H : 5V±0.1	
Preheat time	Over 24 hour		
D. Strucyure and configuration, basic measuring circuit			

그림 7. 가연성 가스 Spec

- Calibration

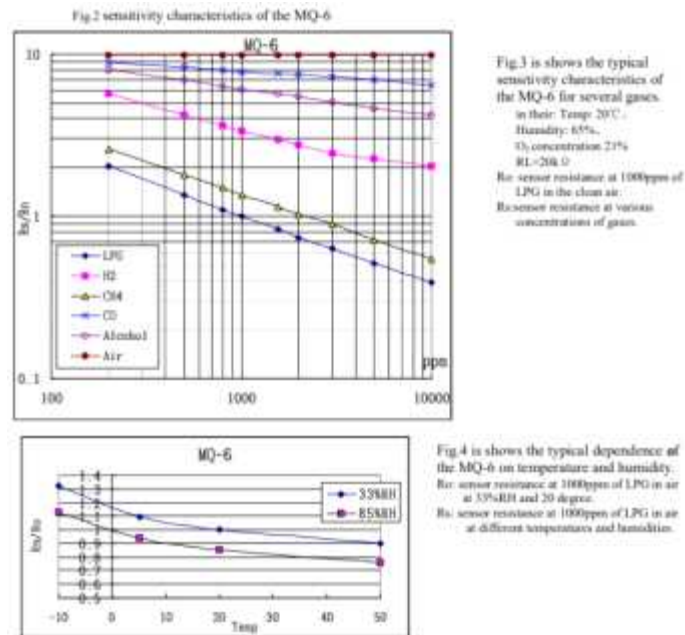


그림 8. 가연성 가스 센서 응답 특성

MQ-6 센서의 민감도 특성 및 온도, 습도와의 관계는 다음의 그래프와 같으며 이 관계식에서 도출한 Calibration 값을 가연성 가스 감지 로직에 적용하였다.

○ 공기질 센서

- 일반사항

공기질 센서는 암모니아, 벤젠, 알콜을 감지하며, 국내 안전기준은 벤젠의 경우 1ppm이하, 암모니아는 TWA - 25ppm, STEL - 35ppm이며 본 연구에서 사용한 공기질 센서는 HANWEI ELECTRONICS社의 MQ-135를 사용하였으며, 가연성 가스 센서의 자세한 Spec과 Hardware는 다음과 같다.

SPECIFICATIONS			
A. Standard work condition			
Symbol	Parameter name	Technical condition	Remarks
V_c	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	AC OR DC
R_L	Load resistance	can adjust	
R_H	Heater resistance	33Ω ± 5%	Room Temp
P_H	Heater consumption	less than 800mw	
B. Environment condition			
Symbol	Parameter name	Technical condition	Remarks
T_{ao}	Using Temp	-10°C~45°C	
T_{st}	Storage Temp	-20°C~70°C	
R_H	Relative humidity	less than 95%Rh	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%
C. Sensitivity characteristic			
Symbol	Parameter name	Technical parameter	Remark 2
R_a	Sensing Resistance	30KΩ~200KΩ (100ppm NH ₃)	Detecting concentration scope: 10ppm~300ppm NH ₃ 10ppm~1000ppm Benzene 10ppm~300ppm Alcohol
α (200/50) NH ₃	Concentration Slope rate	≤0.65	
Standard Detecting Condition	Temp: 20°C ± 2°C V_c : 5V±0.1 Humidity: 65%±5% V_H : 5V±0.1		
Preheat time	Over 24 hour		

그림 9. 공기질 Spec

- Calibration

MQ-135 센서의 민감도 특성 및 온도, 습도와의 관계는 다음의 그래프와 같으며 이 관계식에서 도출한 Calibration 값을 공기질 감지 로직에 적용하였다.

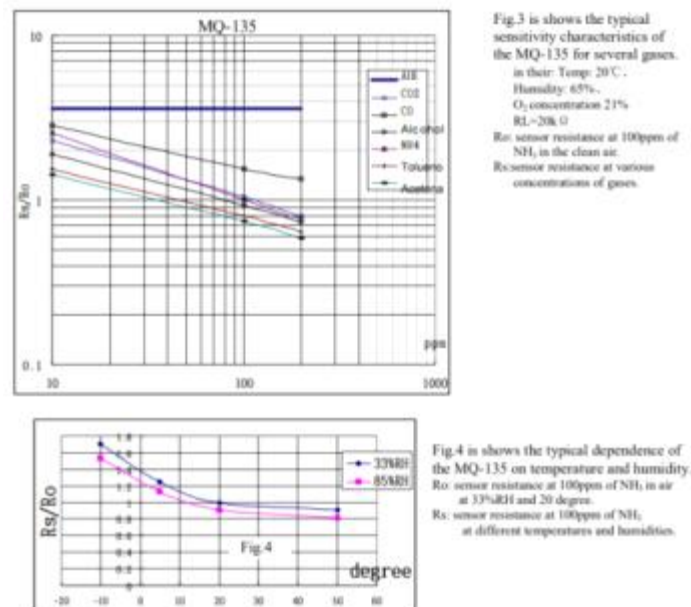


그림 10. 공기질 응답 특성

○ 연무 센서

- 일반사항

연무센서는 연기를 탐지하며, 본 연구에서 사용한 연무탐지 센서는 HANWEI ELECTRONICS社의 MQ-135를 사용하였으며, 연무 센서의 자세한 Spec과 Hardware는 다음과 같다.

Model No.			MQ-2
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite (Black Bakelite)
Detection Gas			Combustible gas and smoke
Concentration			300-10000ppm (Combustible gas)
Circuit	Loop Voltage	V_c	$\leq 24V$ DC
	Heater Voltage	V_H	$5.0V \pm 0.2V$ AC or DC
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	$31\Omega \pm 3\Omega$ (Room Tem.)
	Heater consumption	P_H	$\leq 900mW$
	Sensing Resistance	R_s	$2K\Omega - 20K\Omega$ (in 2000ppm C_3H_8)
	Sensitivity	S	$R_s(\text{in air})/R_s(1000ppm \text{ isobutane}) \geq 5$
	Slope	α	$\leq 0.6(R_{5000ppm}/R_{3000ppm} CH_4)$
Condition	Tem. Humidity		$20^\circ C \pm 2^\circ C$; $65\% \pm 5\% RH$
	Standard test circuit		$V_c: 5.0V \pm 0.1V$; $V_H: 5.0V \pm 0.1V$
	Preheat time		Over 48 hours

그림 11. 연무 센서 Spec

○ 온습도 센서

- 일반사항

온습도 센서는 온도, 습도를 감지하며, 대한민국 기상청에서는 $33^\circ C$ 이상의 기온이 2일 이상 지속될 경우 폭염 주의보를, $35^\circ C$ 이상의 기온이 2일 이상 지속되면 폭염 경보를 발령한다. 본 연구에서 사용한 온도 센서는 SENSIRION社의 SHT1X를 사용하였으며, 온습도 센서의 자세한 Spec과 Hardware는 다음과 같다.

Parameter	Conditions	min	typ	max	Units
Power supply DC ¹⁰		2.4	3.3	5.5	V
Supply current	measuring		0.55	1	mA
	average ¹¹	2	28		μA
	sleep		0.3	1.5	μA
Low level output voltage	$I_{OL} < 4 \text{ mA}$	0		250	mV
High level output voltage	$R_P < 25 \text{ k}\Omega$	90%		100%	VDD
Low level input voltage	Negative going	0%		20%	VDD
High level input voltage	Positive going	80%		100%	VDD
Input current on pads				1	μA
Output current	on			4	mA
	Tri-stated (off)		10	20	μA

그림 12. 온습도 Spec

- Calibration

SHT1X 센서의 민감도 특성 및 온도, 습도와의 관계는 다음의 그래프와 같으며 이 관계식에서 도출한 Calibration 값을 온습도 감지 로직에 적용하였다.

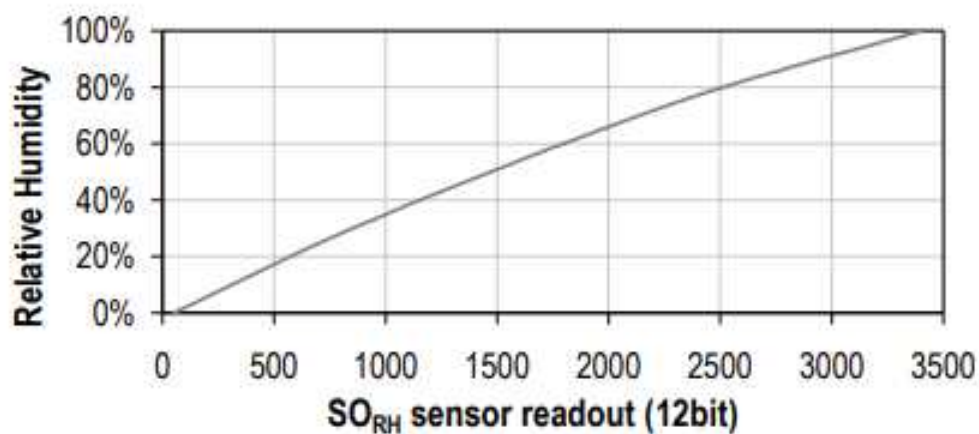


그림 13. 온습도 응답 특성

○ 미세먼지 센서

- 일반사항

미세먼지 센서는 미세먼지를 감지하며, 대한민국 기상청에서는 미세먼지가 PM2.5 75 $\mu\text{g}/\text{m}^3$ 이상 또는 PM10 150 $\mu\text{g}/\text{m}^3$ 이상 지속될 경우 미세먼지 주의보를, PM2.5 150 $\mu\text{g}/\text{m}^3$ 이상 또는 PM10 300 $\mu\text{g}/\text{m}^3$ 이상 지속되면 폭염 경보를 발령한다. 본 연구에서 사용한 온도 센서는 DFROBOT社의 SEN0177을 사용하였으며, 미세먼지 센서의 자세한 Spec과 Hardware는 다음과 같다.

Specification

Basic	Feature
Operating voltage :4.95 ~ 5.05V Maximum electric current: 120mA Measuring pm diameter: 0.3~1.0、1.0~2.5、2.5~10 (um) Measuring pm range : 0~500 ug/m3 Standby current: ≤200 uA Response time: ≤10 s Operating temperature range: -20 ~ 50C Operating humidity range: 0 ~ 99% RH Maximum size: 65 × 42 × 23 (mm) MTBF: ≥ 5 years	Quick response Standard serial input word output Second-order multi-point calibration curve The minimum size of 0.3 micron resolution

그림 14. 미세먼지 Spec

○ 화재탐지(불꽃탐지) 센서

- 일반사항

화재탐지(불꽃탐지)센서는 760nm - 1100nm파장대의 빛을 감지하여, 불꽃을 탐지하며, 본 연구에서 사용한 화재탐지 센서는 DFROBOT社의 SEN0177을 사용하였으며, 미세먼지 센서의 자세한 Spec과 Hardware는 다음과 같다.

Spectrum range: 760nm ~ 1100nm

Detection angle: 0 - 60 degree

Power: 3.3V ~ 5.3V

Mounting hole dia.: 3.1mm

PCB size: 4.7 * 1.5cm

Weight: 2g

Package size: 8.0 * 6.0 * 1.0cm

Package weight: 2g

그림 15. 화재탐지(불꽃탐지) Spec

(나) 제작

위와 같은 과정을 거쳐서 피난대피 IoT 모니터링(Node)모듈의 시제품을 제작하였으며 최종적으로 제작된 제품은 그림 16. 과 같다.



그림 16. IoT모듈

최종적으로 제작된 모듈 온/습도, 메탄가스, 연무, 일산화탄소, 공기질, 미세먼지 센서 및 제어용 SBC(Single Board Computer), RJ45외부 포트, USB외부 포트, 통풍 팬, 네트워크 Hub, RJ-485 컨버터를 내장하며 공구의 선단 및 Wire등으로부터 보호되며, 수직에서 15도 범위에서 떨어지는 물방울로부터 내부의 부품을 보호 할 수 있도록 제작하였다.

(2) 피난대피 유도모듈 개발

이용객 이용 공간에 설치되고, 모니터링 모듈에 의해 제어되는 재난상황시에 활성화 되어 실시간 피난 안내가 가능한 피난 유도 H/W 모듈을 아래와 같은 과정을 통하여 제작하였다.

(가)함체 설계

함체는 기본적으로 매립 공사가 어려운 지하 다중이용시설의 특성상, 기존의 시설물에 매립하는 것이 아닌, 벽체 하단의 걸레받이 상단에 부착하는 방향으로 설계되었으며, 승객의 파손 및 역사내 물청소등에 영향이 없고, 함체 내부에 전력선, 통신선, Led Matrix, 컨트롤 보드, 통신 모듈, SMPS 매립이 가능하도록 설계 되었다.

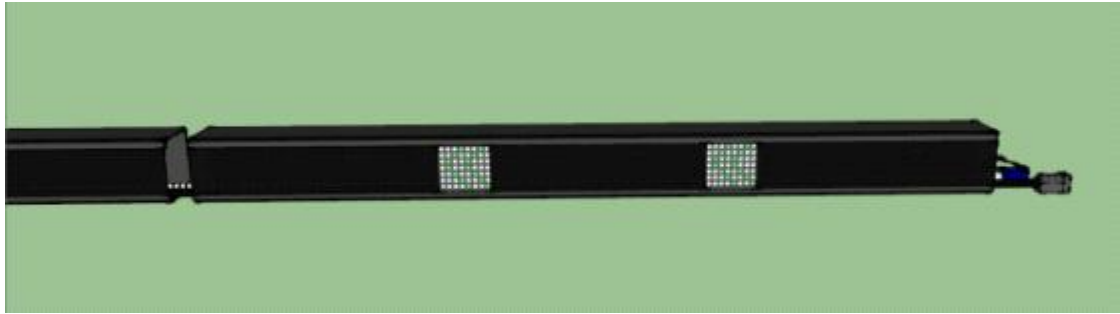


그림 17. 피난대피유도모듈 모델링

(나) 내장 회로 설계

내장 회로는 각 지역/모듈별 컨트롤이 가능하도록 통신이 가능한 통신 모듈과 LED 매트릭스의 좌향/우향 제어를 담당하는 컨트롤보드 및 7*8 Ledmatrix, PLC를 통한 제어를 받는 컨트롤러로 구성되어있다.



그림 18. 내장 회로

(다)제작

본격적인 제작으로는 테스트베드인 이매역의 설치 구간이 약 3.8km에 달할 것으로 계산되어 모듈을 케이블 및 기타 부품류의 보호를 위한 케이스와 LED매트릭스를 이용하여 피난 방향을 안내하는 1M 모듈의 2종류로 이원화하여 제작하였으며, 최종적으로 제작 및 테스트베드 설치된 피난대피유도모듈의 모습은 그림 19. 와 같다.



그림 19. 피난대피 유도모듈

(3) 상황인지 재난대응 S/W(Sh-Monitoring β)개발

(가) 시스템 구성 기획

Sh-Monitoring β(가제)는, 지하공간(지하철) 시설물에 설치된 센서를 통해 여러가지 환경 정보 데이터를 수집하여 승객 대피 의사결정을 지원한다.

관리자의 의사결정에 따라 재난의 경우 Sh-Monitoring β(가제)는 센서링값에 의해 재난 모드로 전환이 되어 이용객의 대피를 유도한다.

Sh-Monitoring β(가제)는 자신의 상태를 다른 Sh-Monitoring β(가제)와 공유하여 쏘구조자에게 정확한 대피로를 안내한다.

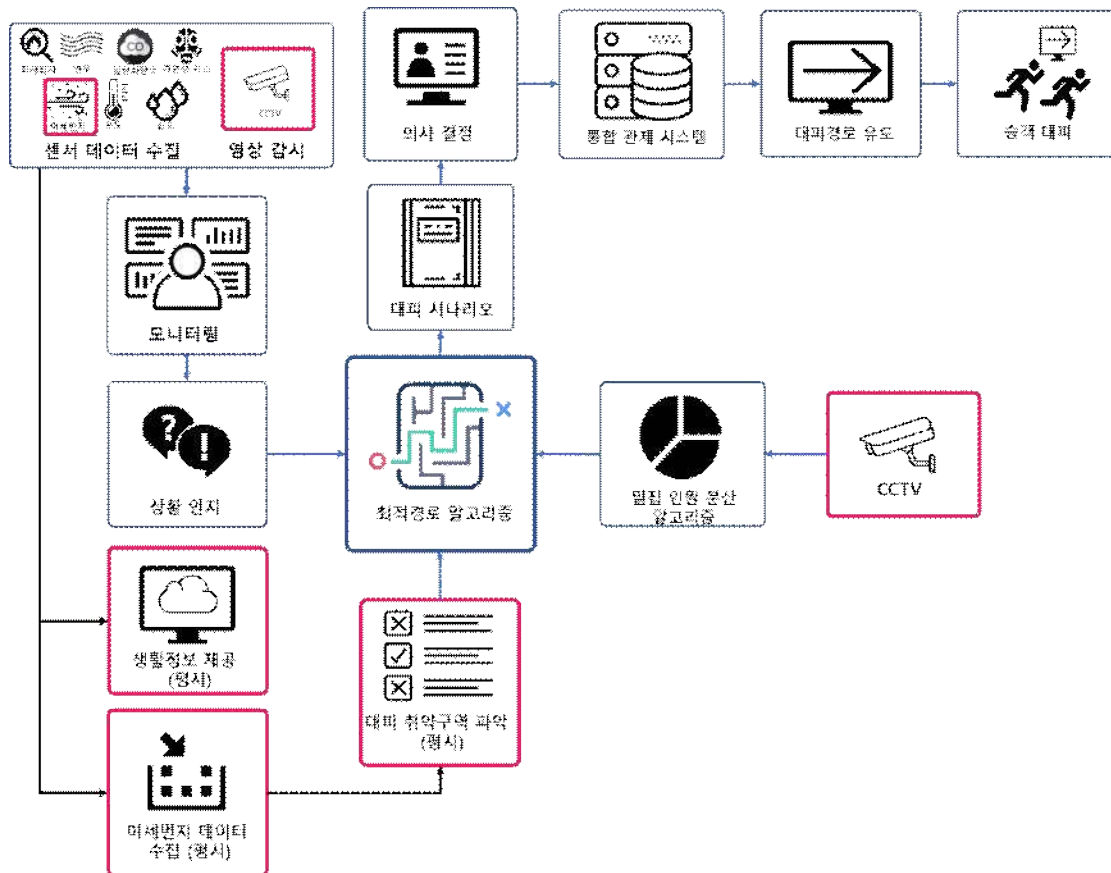


그림 20. 시스템 개요도

(나) 평시모드 작동 시나리오

Sh-Monitoring β(가제)의 평시모드는 센서 데이터를 수집하여 분석한 후 데이터를 서버의 의사결정 학습 모듈에 전달하여 분석을 진행하며, 분석된 데이터를 수신받아 업데이트한다.

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

(다) 재난모드 작동 시나리오

Sh-Monitoring β(가제)의 재난모드는 서버로부터 재난발생 상황을 수신한 상태이며 UI가 재난모드로 변경이 되어 要구조자에게 현재 발생한 재난의 종류와 대피로의 경로를 기본으로 표시한다.

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

(라) Sh-Monitoring β(가제) UI 구성

① 개요

Sh-Monitoring β(가제)는 중앙 관리 시스템과 네트워크상에서 연결이 되고 있을 경우에 관리하는 각종 센서의 정보를 중앙 관리 시스템으로 전달하고 要구조자에게 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 要구조자를 안전하게 대피시킬 수 있는 SOP 가이드로서 역할을 하기 위한 재난모드가 있다.



그림 21. Sh-Monitoring β(가제) 노드 구성

② 평시모드

Sh-Monitoring β(가제)의 평시모드에서는 재난 시 대피요령에 대한 안내, 평시 안전관련

수칙등의 정보를 순차적으로 제공한다.

③ 재난모드

Sh-Monitoring β(가제)의 재난모드에서는 현재 발생한 재난의 종류와 대피로의 경로를 기본으로 표시하고 사용자의 선택에 따라 대피방향, 현재 위치를 표시한다.



그림 22. 평시 모드 사진 예시



그림 23. 재난 모드 사진 예시

④ 방향표지예시

그림 24. 와 같이 다양한 피난방향을 표지할 수 있도록 다양한 방향표지 디자인을 시스템에 내장시킨다.

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

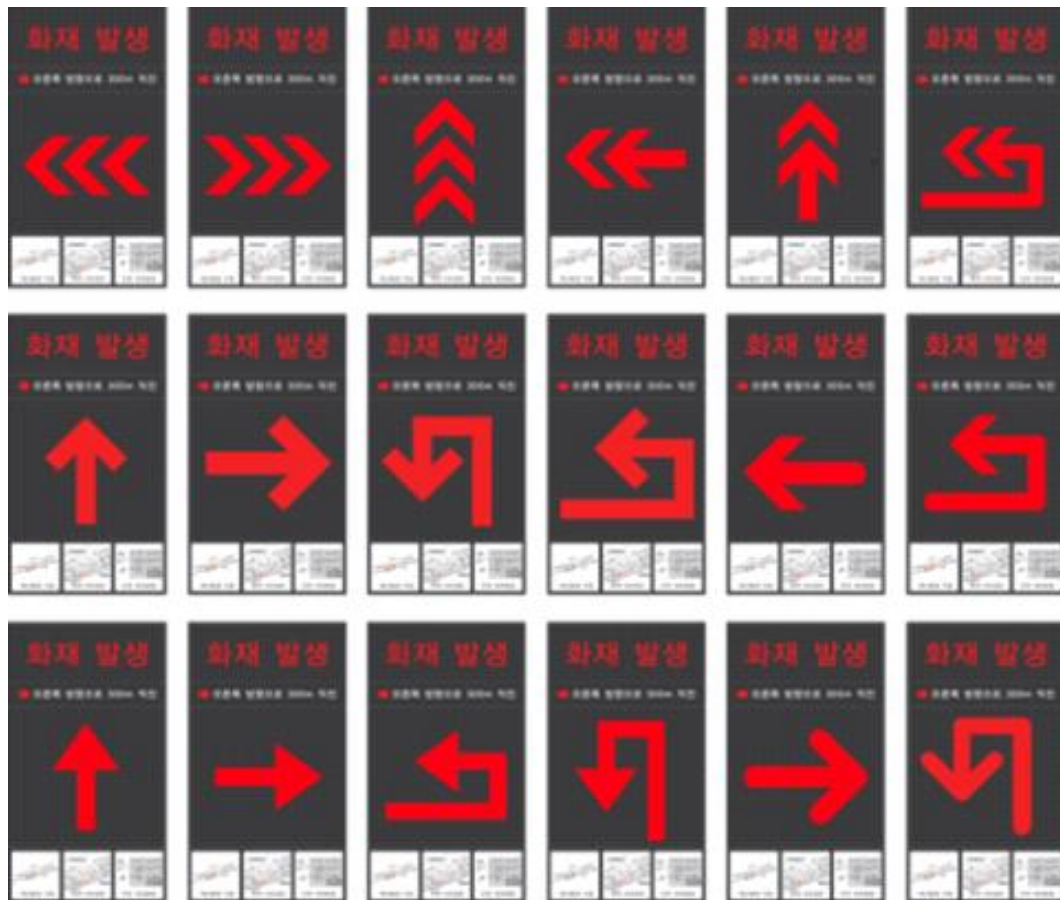


그림 24. 그림 방향 표지 예시

(3-1) 대피 경로 분석/대피 유도 모듈 β 버전 개발

(가) 개요

재난 발생 시 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.

(나) 성능목표

○ 기존기술 수준

기존 지하철 공간 피난은 피난 공간에 대한 정량적 분석 기반이 아닌, 경험적 판단에 의한 경로 설정이 일반적이었다.

이러한 경험적 경로 설정에 대한 객관적 근거 제시가 미흡했으며, 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.

(다) 공간의 그래프화

지하철 역사에 재난 발생 시 최적의 경로로 피난을 유도하여야 하나, 최적의 피난 경로를 정량적으로 구하는 방법론은 아직까지 사회적으로 충분히 개발되어 있지 않은 실정이다.

일반적인 방법으로는 건물 내 피난 시뮬레이션 S/W를 사용하는 방법이 있으나, 이는 대상 공간의 크기에 따라서 많은 소요시간이 필요하다는 단점이 있다.

그러므로 본 연구에서는 시뮬레이션 이외에 그래프 이론(graph theory)를 이용해서 최적 피난 경로를 찾을 수 있는 알고리즘을 개발하였고, 다음의 단계를 거쳐서 공간의 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.

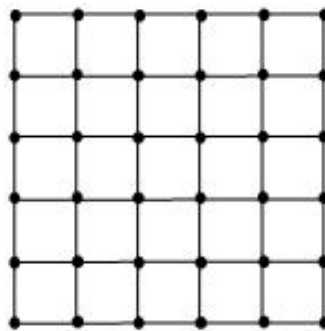


그림 25. 격자형 평면 그래프

이러한 평면 그래프는 수직으로 중첩함으로써 평면에서 공간으로 확장할 수 있는데, 그림 26. 과 같이 공간에서 평면 그래프로 이해될 수 있다.

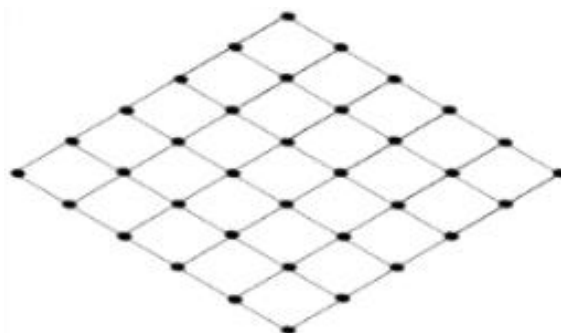


그림 26. 공간에서의 평면 그래프

실제 공간에서는 벽체가 존재하며 아래 그림 27. 과 같이 벽체가 존재하면 그 벽체가 공간의 연결을 끊게 된다.

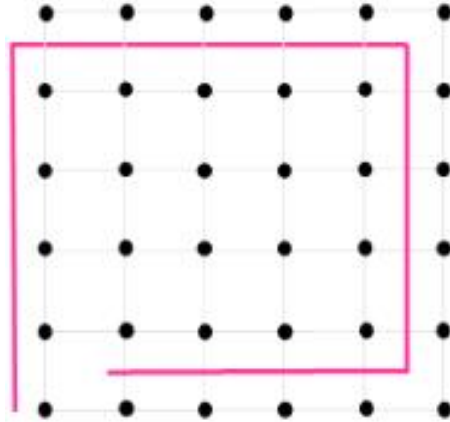


그림 27. 벽체가 공간의 연결을 끊는 예시

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

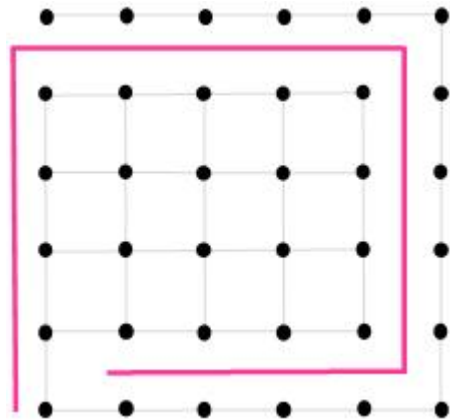


그림 28. 공간을 그래프로 변환한 결과 그래프

상기 기술한 과정을 일반적인 공간 도면에 적용하면 그림 29. 과 같은 결과가 얻어진다.

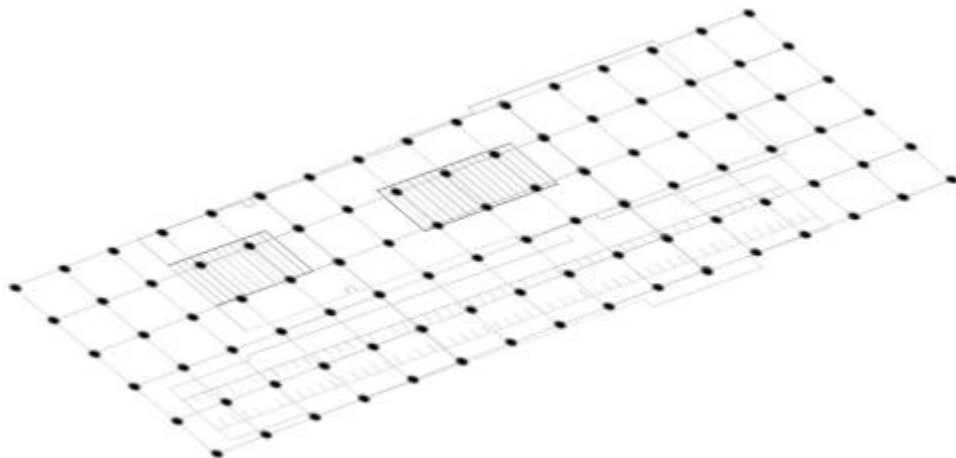


그림 29. 공간 도면과 그래프 데이터의 매칭

(라) A-Star Algorithm

① 개요

제안 알고리즘은 주어진 출발 꼭짓점에서부터 목표 꼭짓점까지 가는 최단 경로를 찾아내는(다시 말해 주어진 목표 꼭짓점까지 가는 최단 경로임을 판단할 수 있는 테스트를 통과하는) 그래프/트리 탐색 알고리즘 중 하나이다. 이 알고리즘은 각 꼭짓점 x 에 대해 그 꼭짓점을 통과하는 최상의 경로를 추정하는 순위값인 "휴리스틱 추정값" $h(x)$ 을 매기는 방법을 쓴다. 이 알고리즘은 이 휴리스틱 추정값의 순서로 꼭짓점을 방문한다. 그러므로 A* 알고리즘을 너비 우선 탐색의 한 예로 분류할 수 있다.

이 알고리즘은 1968년 피터 하트, 닐스 닐슨, 버트램 라펠이 처음 제안하였다. 그 3명의 논문에서, 이 알고리즘은 A 알고리즘(algorithm A)이라고 불렸다. 적절한 휴리스틱을 가지고 이 알고리즘을 사용하면 최적(optimal)이 된다. 그러므로 A* 알고리즘이라고 불린다.

A* 알고리즘은 출발 꼭짓점으로부터 목표 꼭짓점까지의 최적 경로를 탐색하기 위한 것이다. 이를 위해서는 각각의 꼭짓점에 대한 평가 함수를 정의해야 한다. 이를 위한 평가 함수 $f(n)$ 은 다음과 같다.

- $f(n)=g(n)+h(n)$
- $g(n)$: 출발 꼭짓점으로부터 꼭짓점 n 까지의 경로 가중치
- $h(n)$: 꼭짓점 n 으로부터 목표 꼭짓점까지의 추정 경로 가중치

② 타 알고리즘과 비교

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

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

③ 알고리즘

제안 알고리즘은 다음과 같다.

```
if (!_isCreated) return;

if (startTile == null || endTile == null) return;
// 기존 정보 초기화
_tiles.ForEach(o => o.Text = null);
_openList.Clear();
_closeList.Clear();
```

```

_path.Clear();

if (startTile == null && endTile == null)
{
    startTile = _tiles[0];
    endTile = _tiles[_tiles.Count - 1];
}

_openList.Add(startTile);
Tile tile = null;
do
{
    if (_openList.Count == 0) break;

    tile = _openList.OrderBy(o => o.F).First();
    _openList.Remove(tile);
    _closeList.Add(tile);

    if (tile == endTile) break;

    foreach (var target in _tiles)
    {
        if (target.IsBlock) continue;
        if (_closeList.Contains(target)) continue;
        if (!IsNearLoc(tile, target)) continue;

        if (!_openList.Contains(target))
        {
            _openList.Add(target);
            target.Execute(tile, endTile);
        }
        else
        {
            if (Tile.CalcGValue(tile, target) < target.G)
            {
                target.Execute(tile, endTile);
            }
        }
    }
}
while (tile != null);

if (tile != endTile)
{
    MessageBox.Show("길막힘");
}

```

```

        return;
    }

    do
    {
        _path.Add(tile);
        tile = tile.Parent;
    }
    while (tile != null);
    _path.Reverse();

    for (int i = 0; i < _path.Count; i++)
    {
        if (i == 0) _path[i].Text = "START";
        else if (i == _path.Count - 1) _path[i].Text = "END";
        else _path[i].Text = i.ToString();
    }
    _isStarted = true;

```

- ㉓ Tile : x,y 좌표를 가지는 정사각형의 바닥
- ㉔ Map : x,y 좌표를 갖고 Tile로 이루어진 공간
- ㉕ OpenList : 열린 목록 (이동 가능한 Tile)
- ㉖ ClosedList : 닫힌 목록 (다시는 검색할 필요가 없는 Tile)
- ㉗ 상하좌우 이동값=10, 대각선 이동값=14

④ 알고리즘 도식 해설

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 이동경로가 작성이 된다.

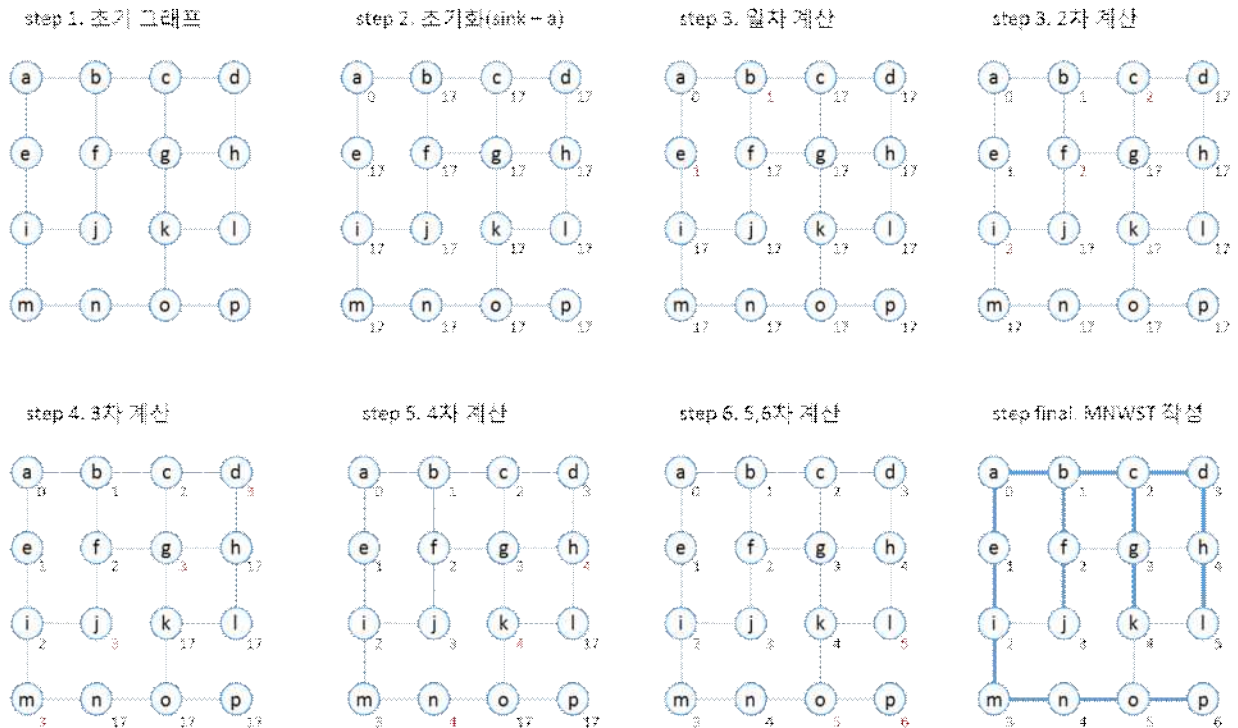


그림 30. AStar 경로계산 과정

⑤ Astart의 특징

모든 노드에서 최단 경로 검색이 가능하다. 각 노드의 값들은 Astar에서 탈출처인 싱크(sink)로의 최단거리를 나타낸다.

또한 필요에 따라서 각 노드가 이동경로로 이용되는 빈도로서 노드의 혼잡도는 각 노드의 자식 노드 수로서 구할 수 있다.

⑥ Astar 공간 적용

○ 그래프 데이터 생성

대상 공간을 그리드로 나누고 문이나 기둥에 의해서 단절되는 부분 이외를 그래프로 작성하고, 단절이 되지 않은 노드끼리만 연결을 하고 대상 사이트 내부에 대해서 그래프를 생성한다.

○ 대피처 설정

대피처를 하나 이상 설정한 후에 알고리즘에 의해서 각 노드의 웨이트를 계산한다.

○ Astar 계산

알고리즘에 의해서 각 노드의 가중치 및 스패닝 트리가 산출된다.

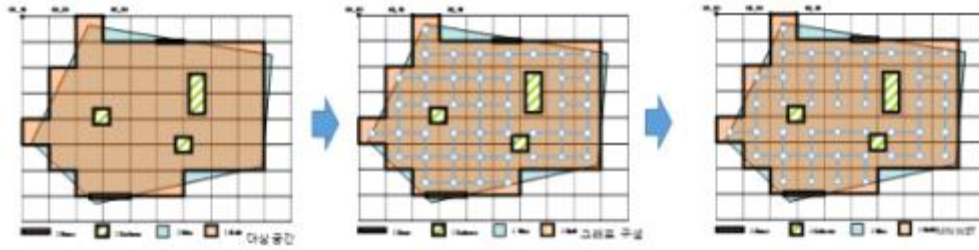


그림 31. 공간데이터의 Astar 적용 예

⑦ 테스트베드 Astar 적용

위 작업을 위해서 우선 한국철도공사에서 제공한 캐드도면에서 필요한 정보만을 추출하였다.

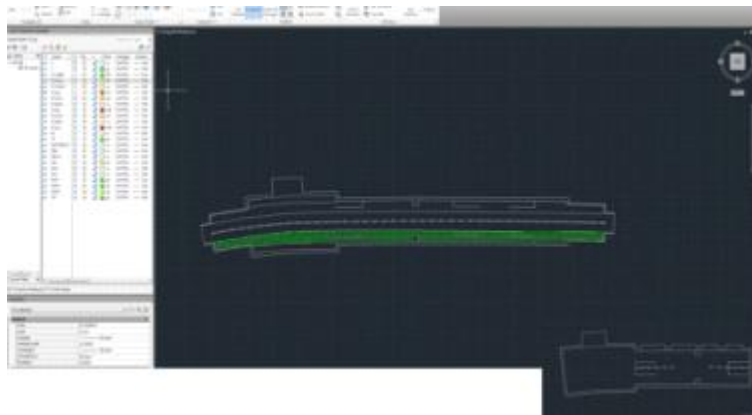


그림 32. 한국철도공사의 캐드 파일

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

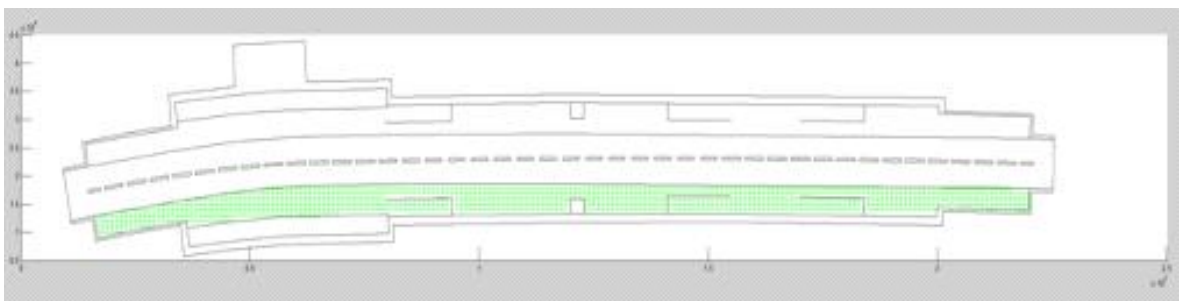


그림 33. 공간 그래프

공간 그래프 중 계단부를 확대해보면 아래 그림 33.과 같이 벽체가 있는 부분의 간선은

전부 삭제된 것을 알 수 있다.

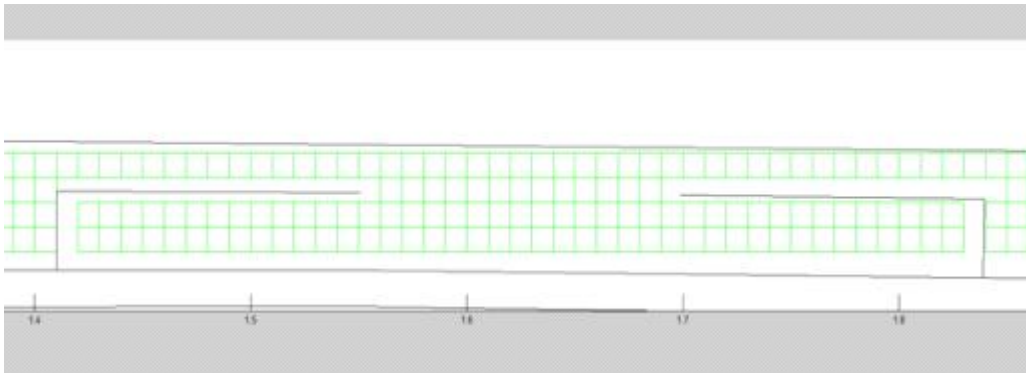


그림 34. 공간 그래프 중 계단부 확대

공간 그래프에 대해서 Astar알고리즘을 적용하여 다음의 결과를 얻었다.

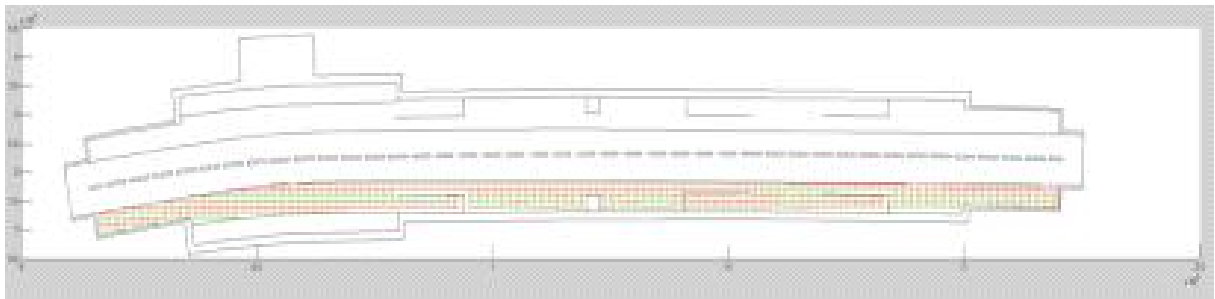


그림 35. 테스트베드 승강장의 공간그래프(녹색) + 벽체(붉은색)

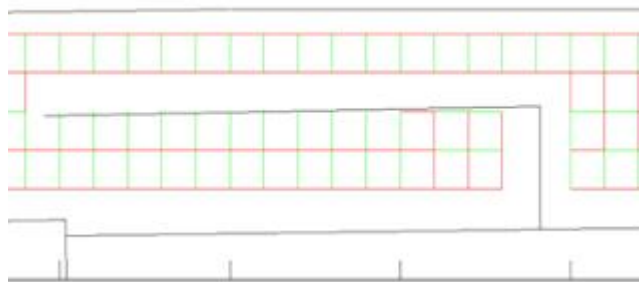


그림 36. 테스트베드 공간그래프 및 벽체 계단부 확대

⑧ Astar 알고리즘의 활용

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 사용 횟수를 구하거나 타임라인 별로 각 노드에서의 피난 에이전트 수를 구함으로써 간이 피난 시뮬레이션을 수행하는 것과 동일한 효과가 발생한다.

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 목적지를 계산한 그래프 즉 대피처까지의 거리를 속성으로 가지는 그래프를 작성한다.

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

⑨ 피난유도 SOP의 표출

피난유도 방향을 마스터에서 설정하고 저장을 하면 설정이 노드에 전파되어 동기화되며, 이 설정은 재난 발생 시 노드 화면에서 표출된다. 여기서 중요한 것은 공간데이터에 피난유도 알고리즘을 적용한 결과의 피난 유도 방향과 노드에서 보여주는 방향에는 차이가 있다는 점이다.

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

이에 노드에서 나타내는 방향을 설정하기 위해서는 실제 노드가 설치된 장소의 입면에서 보는 방향이 되기 때문에 각 노드 별로 방향을 설치하기 위해서는 현장에서 노드가 설치된 위치를 확인한 후에 방향을 설정해야 한다.

화살표는 시스템에서 코드화되어서 각 시나리오별로 노드와 코드가 매칭되어서 저장된다.

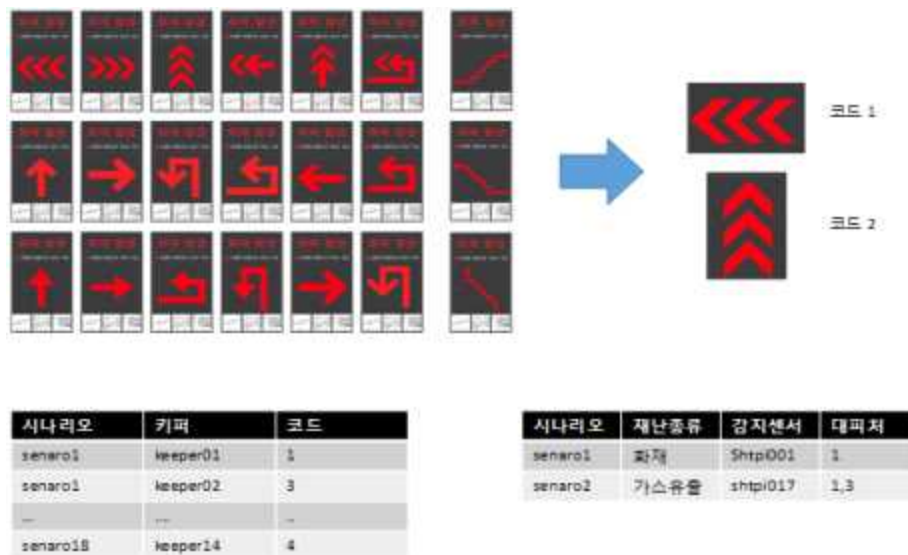


그림 37. 피난유도 화살표와 코드화

(3-2) Sh-Monitoring β(가제) 네트워크 구축

(가) Sh-Monitoring β(가제) 네트워크 구성

재난전조 대응 네트워크는 아래 그림 37. 과 같다. Sh-Monitoring β(가제)는 센서노드와 마스터 서버로 구성되어진다. 센서노드는 Wifi 및 RJ45케이블을 이용하여 프로토콜로 데이

터를 주고받는다.

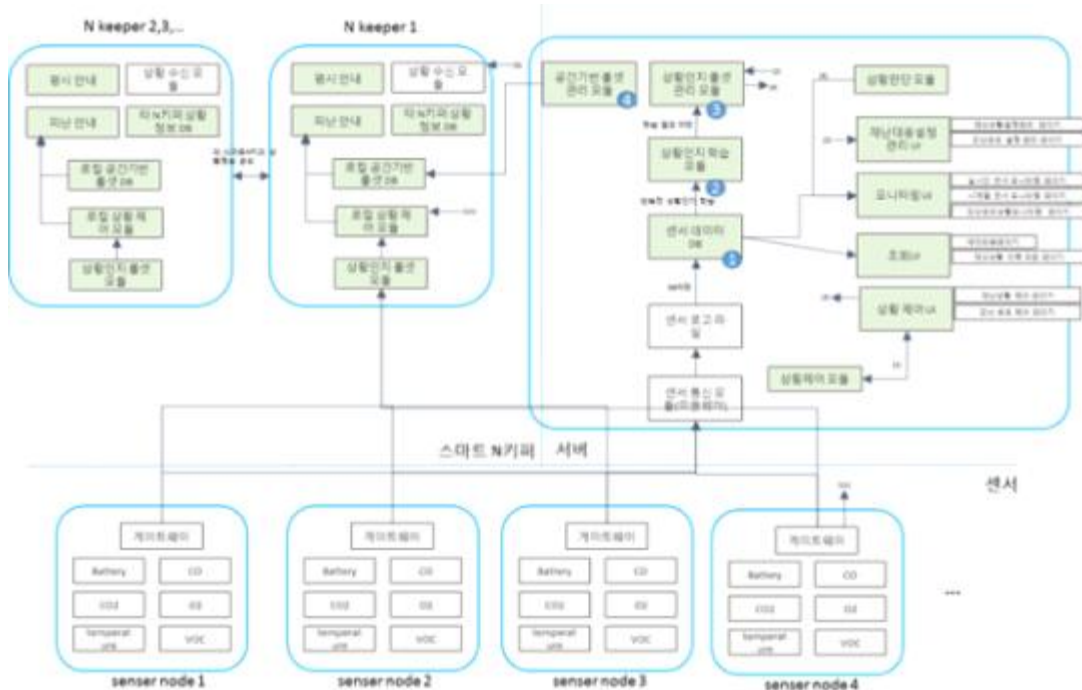


그림 38. SH_Monitoring(가제) 네트워크 구성도

(나) 네트워크 계층도

재난 전조 대응 모니터링 네트워크는 지하공간의 노드와 역무실의 마스터 서버로 구성되어진다.

○ 노드 : 노드는 주로 센싱작업과 데이터 송신작업을 한다. 각각의 가스센서의 concentration을 수행하며, 전력 절감을 위해 슬립모드를 수행한다. 페이로드로는 온도, 습도, 연무, 불꽃탐지, 메탄가스, 가연성가스, 일산화탄소, 대기오염값 및 미세먼지값을 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						

그림 39. 노드 통신 프레임 데이터 프로토콜 구조

○ 마스터 서버(Station server) : 노드에서 전송한 데이터를 마스터서버의 DB에 저장한다. 최초의 데이터는 버퍼에 저장되며, 누적된 콜드데이터는 mongodb로 이전한다. 마스터 서버는 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.

(다) 재난 전조 대응 노드 모듈 네트워크 구성

① 개요

시스템은 구성도에서 오른쪽 상단은 마스터의 기능을 나타내며, 왼쪽 상단은 노드의 기능을 나타낸다.

복수의 노드는 TCP/IP로 연결되고 네트워크로 연결된 센서 네트워크들이 각 노드 및 마스터의 센서 데이터 수용모듈을 통해서 각 노드모듈에 저장된다.

저장된 데이터는 바이너리 형식으로 각 머신에서 제공된다.

노드 및 마스터는 각 SW가 설치된 머신의 로컬호스트를 통해서 바이너리형식으로 센싱 데이터를 제공받아 자체DB에 저장을 하며 사용을 한다.

노드 및 마스터는 TCP/IP로 통신을 하며 필요한 정보를 교환한다.

② 주요 서비스 내용

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

㉠ 마스터

마스터는 센서 노드에서 올라온 정보들의 취합과 실시간으로 모니터링을 데이터를 표출하며, 센서 데이터의 조건에 의해서 정상, 이상, 재난이라는 3가지 상태를 실시간으로 상황을 판단한다.

상황판단 결과가 이상이나 재난을 나타내게 되면 시스템은 관리자에게 이 정보를 표출하고 관리자의 대응을 요청한다.

수집되는 센서 값들이 정상으로 판단된 경우, 시스템은 정상 모드로 돌아가며, 이전 상황을 로그에 기록하며, 이상이나 재난으로 관리자가 판단한 경우, 시스템은 이상 및 재난 상황 발생을 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.

㉡ 노드

노드는 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 피난유도를 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

㉓ 개발환경

노드 모듈은 윈도우 10 환경에서 .net FrameWork 4.7.1 환경에서 개발되었다.

마스터 모듈은 Java, Spring web framework 환경에서 개발되었다.

서비스는 웹으로 표출되며, 프론트엔드(웹페이지)는 크롬미움계열의 웹 브라우저를 지원한다.

㉔ 운용환경

운용환경은 노드는 .net FrameWork 4.7.1이 돌아가는 환경이면 OS에 상관없이 돌아갈 수 있다.

마스터는 Html 5.0 표준을 지원하면 사용 가능 하며 현재 운용중인 웹브라우저로는 크롬을 사용하였으며 표준 해상도는 1920x1080(Full HD)해상도 기반으로 개발하였다.

(3-3) 웹기반 마스터 관리 시스템 개발

재난 전조 대응 시스템의 관리자 웹 어플리케이션은 아래 그림 40.와 같다. 최상단의 화면에서 현재 들어오는 데이터의 센서노드 번호와 측정된 시간 그리고 센서노드 프레임 카운트 그리고 배터리량을 보여준다. 왼쪽에는 센서노드에서 도달한 값 총 9가지의 센서값(온도, 습도, 연무, 불꽃탐지, 메탄가스, 가연성가스, 일산화탄소, 대기오염도, 미세먼지)를 게이지레벨로 출력한다. 총 18대의 센서노드의 값이 들어오면서 화면은 계속 변경되며, 현재 재난 상태와 센서 상태를 인식할 수 있다. 그림 40 b와 그림 40 c는 센서노드 별로 센서데이터 선택하여 과거의 데이터 이력을 확인할 수 있다.



<그림 a. 노드 정보 조회>



<그림 b. 센서 정보 조회>



<그림 c. 센서 값 그래프 조회>



<그림 d. 임계 값 및 시간 설정>

그림 40. 시스템 관리자 웹 어플리케이션

(가) 웹 어플리케이션 구성

① 데이터 통신 / 센서노드 관리자 웹 어플리케이션 구성도

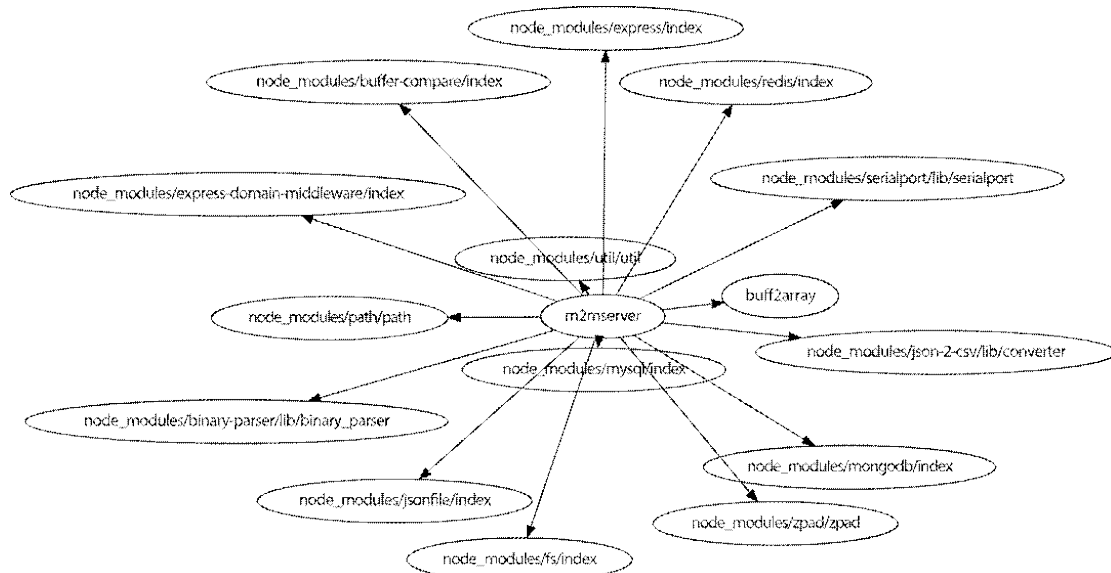


그림 41. 데이터 통신 / 센서노드 관리자 웹 프로그램 ERD

② 패키지 구성

패키지 및 의존성 관리 파일	package.json
<pre>{ "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"</pre>	

```

"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

<pre> +SensorName.k: string = TimeStampUnit +SensorTag.k: string = SENSOR_TST +SensorUnit.k: string = EpochTime +SensorValue.k: variable = TimeStamp +Value: variable = undefined +Value: variable = undefined +buffsize: number = 128 +client: variable = undefined +count: number = 0 +database: string = MeshlumiDB +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 = gkrdmcsjs +querscript: variable +querscript: variable +startindexcount: 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 querscript: 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 </pre>	<pre> +DataStruct(serialID, waspmoteID, NOB, FOS, TST, name, tag, value, unit) +FrameSequences(19, 20) +FrameType(4, 5) +LI() +NumberOfBytes(5, 6) +Separator(18, 19) +SerialID() +StartDelimiter() +TimeStamp() +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, waspid, wasp +index(data, indicator) +indicator() +nowVal() +nowarray(), +parity(parity) +parsedJSON(result) +parsedJSON(NULL) +qarray(qdata) +qdata(buffsize) +redisArgs(NULL) +redisClient() +sensor(sensor) +sensorData(i(NULL) +sensorID(ind) +sensorName(sensorName) +sensorNode(sensorNode) +serialConf(file) +serialPort(, true,) +server(), +serverTimeStamp() +startDataInStart() +stopBits(stopBits) +stopDataInStop() +timeStamp() +uptodate(NULL) +wstream() </pre>
---	--

그림 42. 데이터 구조 및 클래스

④ 의존성 패키지 구성도

가) 통신 모듈 (serialport)

통신 모듈 'serialport'는 'spring Framework'에서 시리얼포트를 제어할 수 있는 모듈이다. OS는 Linux, MacOS, Windows 및 RaspberryPi 등 다양한 환경에서 사용가능하며 모두 콜백함수 처리하여 비동기식 응용프로그램을 구성할 수 있다.

패키지 및 의존성 관리 파일	package.json
<pre> <?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <groupId>com.github.diegosep</groupId> <artifactId>spring-serial-port-connector</artifactId> <version>0.0.1-RELEASE</version> <packaging>jar</packaging> <name>spring-serial-port-connector</name> <description>spring-serial-port-connector</description> <parent> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-parent</artifactId> <version>1.5.7.RELEASE</version> <relativePath/> <!-- lookup parent from repository --> </parent> <properties> <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding> <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding> <java.version>1.8</java.version> <nrjavaserial.version>3.12.0</nrjavaserial.version> <lombok.version>1.16.18</lombok.version> </properties> <scm> <connection>scm:git:git://github.com/diegosep/spring-serial-port-connector.git</connection> <developerConnection>scm:git:ssh://github.com/diegosep/spring-serial-port-connector.git</developerConnecti on> <url>https://github.com/diegosep/spring-serial-port-connector</url> </scm> <licenses> <license> <name>The Apache License, Version 2.0</name> <url>http://www.apache.org/licenses/LICENSE-2.0.txt</url> </license> </licenses> <developers> <developer> <name>Claudia López</name> <email>colopezfuentes@gmail.com</email> <timezone>UTC-4</timezone> </developer> </pre>	

```

    <developer>
      <name>Diego Sepúlveda</name>
      <email>diegosepulvedabriones@gmail.com</email>
      <timezone>UTC-4</timezone>
    </developer>
  </developers>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <dependency>
      <groupId>org.bidib.com.neuronrobotics</groupId>
      <artifactId>nrjavaserial</artifactId>
      <version>${nrjavaserial.version}</version>
    </dependency>

    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>${lombok.version}</version>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-configuration-processor</artifactId>
      <optional>true</optional>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>

      <plugin>
        <groupId>org.sonatype.plugins</groupId>
        <artifactId>nexus-staging-maven-plugin</artifactId>
        <version>1.6.8</version>
        <extensions>true</extensions>
        <configuration>
          <serverId>ossrh</serverId>
          <nexusUrl>https://oss.sonatype.org/</nexusUrl>
          <autoReleaseAfterClose>true</autoReleaseAfterClose>
        </configuration>
      </plugin>

      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-source-plugin</artifactId>
        <version>3.0.1</version>
        <executions>
          <execution>

```

```

        <id>attach-sources</id>
        <goals>
          <goal>jar-no-fork</goal>
        </goals>
      </execution>
    </executions>
  </plugin>

  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-javadoc-plugin</artifactId>
    <version>3.0.0-M1</version>
    <executions>
      <execution>
        <id>attach-javadocs</id>
        <goals>
          <goal>jar</goal>
        </goals>
      </execution>
    </executions>
  </plugin>

  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-gpg-plugin</artifactId>
    <version>1.5</version>
    <executions>
      <execution>
        <id>sign-artifacts</id>
        <phase>verify</phase>
        <goals>
          <goal>sign</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>

<distributionManagement>
  <snapshotRepository>
    <id>ossrh</id>
    <url>https://oss.sonatype.org/content/repositories/snapshots</url>
  </snapshotRepository>
  <repository>
    <id>ossrh</id>
    <url>https://oss.sonatype.org/service/local/staging/deploy/maven2/</url>
  </repository>
</distributionManagement>
</project>

```

④ Express 미들웨어 API

‘express-domain-middleware’는 ‘express’웹서버 에러 핸들러 미들웨어이다. 시리얼포트 구성이나, 통신에러 및 DB 접속에러가 발생할 경우 웹서버를 종료시키지 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 'spring'의 웹서버 'express'에서 사용할 수 있는 대표적인 에러 핸들러이다.

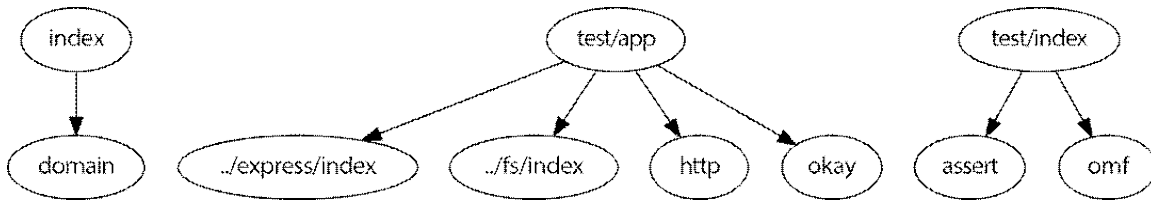


그림 43. 미들웨어 관계도

패키지 및 의존성 관리 파일	package.json
<pre>{ "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-middlew\nare.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? OR... Do you just let your process crash on any un\nhandled exception and restart it? Do you find it hard to pass `request.id` to 8 nested database calls so you\n\n can keep a context of what request you're working on? How do you associate log entries with a specific request? Passing the request around everywhere again? Domains can help. ## USE DOMAINS\n\nFirst, [read this](http://nodesj.org/api/domain.html). Second, realize once you enable domains `process.domain` will give yo\nu the active domain. Third, use this middleware to bind each request/response pair to its own domain. `js\n\nexpress.use(require('express-domain-middleware'));\n\n### express-domain-middleware api\n\n#### var domainMiddleware = require('express-domain-middleware');\n\n##### domainMiddleware = function(req, res, next) {\n\n exports a function matching the signature of express middleware. Binds incoming request & response from\nexpress to a new domain. Assigns the domain a unique id by calling `domainMiddleware.id(req)`. If the domain\nemits an error event, `domainMiddleware` will call `next(err)` with the error event from the domain. This allows existing\nexpress specific error handling middleware to function as if the error was handled by your application code. Allow\nme to demonstrate with an example:\n\n`js\n\n//old-school\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', function(err, rows) {\n if(err) return next(err);\n\n fs.readFile('alskdjflasd', function(err, contents) {\n if(err) return next(err);\n\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//with domain-middleware\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n if(err.domain) {\n //you should think about gracefully stopping & respawning your server\n\n //since an unhandled error might put your application into an unknown state\n });\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error', function(req, res, next) {\n db.query('SELECT happiness()', process.domain.intercept(function(rows) {\n fs.readFile('alskdjflasd', 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\nwill keep running.W');\n });\n });\n });\n});\n\n//WnWnWnWn 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\nWnWn``\n\nvar ok = require('okay');\n\napp.use(require('express-domain-middleware'));\n\napp.use(app.router);\n\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. :(W');\n\n WnWn``\n\n});\n\napp.get('/error',</pre>	

```

ext) {Wn db.query('SELECT happiness()', ok(next, function(rows) {Wn fs.readFile('asldkfjaskdf', ok(next, function(c
ontents) {Wn process.nextTick(ok(next, function() {Wn throw new Error(W"The individual request will be
passed to the express error handler, and your application will keep running.W");Wn }));Wn }));Wn }));Wn }));
Wn`WnWn## licenseWnMITWn`,
"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"
}

```

㉔ 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 전이종 통신이 가능하다. 이와 같은 특징으로 웹에서도 채팅이나 게임, 실시간 주식 차트와 같은 실시간이 요구되는 응용프로그램의 개발을 한층 효과적으로 구현할 수 있게 되었다. 본 과제에서는 Socket.io를 node에서 구현하여 각 N노드간의 브라우저로 양방향 통신을 구현한다.

패키지 및 의존성 관리 파일	package.json
<pre> { "name": "socket.io", "version": "1.4.6", "description": "spring 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", </pre>	

```

    "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-edition.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"
}

```

(3-4) 노드 어플리케이션 개발

(가) 개요

현장에 설치된 안내 시스템인 노드의 상태관리 및 각 노드 별 피난 경로 설정 기능 개발

함께 센서 데이터 기반 재난 발생 전조 관리 기능 개발 및 재난 발생 상황 전파 기능 구현을 수행하였다.

(나) 노드 기능

노드 S/W는 다음과 같은 기능을 수행한다.

- 실시간 센서 데이터 모니터링
- 이상/재난상황 판단



그림 44. 노드 UI

(다) 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.

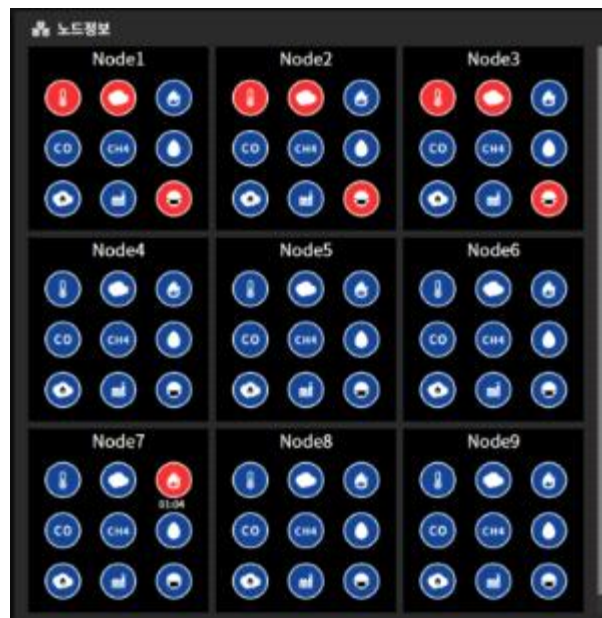


그림 45. 노드 대시보드 사진

(라) 이상/재난상황 판단

이상 및 재난상황 판단은 마스터 측에서 설정된 이상범위 값을 이용하여 현재 센서링 데이터값과 비교하여 판단하게 된다.

The screenshot displays a configuration window titled '46. 재난 상황 설정'. It is organized into two main columns of settings. Each row represents a different sensor type, such as '온도' (Temperature), '습도' (Humidity), 'CO', '가연성 가스' (Flammable Gas), '화재 연기' (Fire Smoke), and '가스 누출' (Gas Leak). For each sensor, there are three rows of thresholds: '정상' (Normal), '이상' (Abnormal), and '재난' (Disaster). Each threshold row includes a numerical input field, a unit selection dropdown (e.g., '°C', '%', 'ppm'), and a ratio selection dropdown (e.g., '100%', '50%', '20%'). The interface uses a color-coded system: green for normal, yellow for abnormal, and red for disaster. At the bottom, there are buttons for '저장' (Save) and '취소' (Cancel).

그림 46. 임계값 설정 사진

(마) 센서 데이터 수집 및 DB 전송

○ 데이터 저장

최근 센서 데이터를 읽어와 중복을 제거하고 DB에 저장된다.
마스터에서 사용한다.

○ 데이터 연동

실시간으로 RS-485를 이용한 통신을 이용하여 아날로그 감지기를 통한 데이터를 입력받으며, 모듈 상의 센서와는 flag를 이용하여 구분한다.

○ 최근 데이터 로딩

DB에 저장된 센서 데이터에 대해 시간순으로 정렬을 하여 마지막 데이터를 받아온다.
각 센서별로 계산을 하여 Binary 데이터로 리턴한다.

○ 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 sldx in range(1,sensorCount+1):
    subldx = ("0"+ str(sldx)) if sldx <10 else str(sldx)
    sensor = "SHTPI0"+subldx
    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
        ret.add Typeof[sensor]
return ret

def get_sensingDataFromRS485():
    ret={}
    sensingData_rows = sensingData.objects.all()
    sensorName = ["-BAT" , "485value,]
    sensorCount = 2
    for sldx in range(1,sensorCount+1):
        subldx = ("0"+ str(sldx)) if sldx <10 else str(sldx)
        sensor = "SHTPI0"+subldx
        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
            ret.add Typeof[sensor]
    return ret

```

○ URL을 통해 센서데이터를 가져오는 방법

노드에서 사용한다.

센서 데이터 서버IP는 외부주소(223.171.32.11)가 디폴트값으로 입력되어있으며 내부설
정시 localhost로 수정을 하면 속도가 상승된다.

settings.py 파일의 SHOST 변수 수정

○ 코드

```

def get_sensingDataFromURL():
    startldx = -1
    endlldx = -1
    sensorCount = 18
    list_sensors = ['BAT', 'GP_NI', 'GP_TC', 'GP_CO', 'GP_O2', 'GP_PRE', '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,startldx,endlldx)
        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(sldx,startIdx,endIdx):
    host = settings.SHOST
    id_wasp = ""
    if sldx<10:
        id_wasp = "0"
    sensorNode = "SHTPI0"+id_wasp+str(sldx)
    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

○ 데이터 로딩

노드는 센서링된 센서의 아날로그 및 디지털 값을 캘리브레이션 하여 마스터DB의 Node 테이블로 넘김.

1초에 한번씩 업데이트

DB의 Sensor 테이블 스키마

nodeseq

정수형

dust

정수형

humidity

정수형

temperature

정수형

airPollution

정수형

carbonMonoxide

정수형

inflammableGas

정수형

methane

정수형

mist

정수형

flame

정수형

regDate

날자/시간형

○ 코드

```

def updateText(request):
    ret = []
    informations_rows = informations.objects.filter(used=1)
    if informations_rows:

```

```

        for row in informations_rows:
            ret.append(row.iText)
    else:
        ret = ["준비중입니다."]
    return HttpResponse(json.dumps(ret))

```

○ 데이터 디스플레이

agauge API 사용.

데이터로 로딩 기능에 의해 넘어온 배열의 값을 파싱하여 텍스트화해 1초에 한번씩 순차적으로 표기함.

○ 코드

```

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

```

① 센서별 단계 설정 기능 내용 정리

○ 단계 설정값 관리

설정값은 마스터 서버상 관리자의 설정값이다.

각 센서별로 3단계의 경계값 설정은 마스터의 시스템설정-센서별 단계 설정 페이지에서 값을 저장한다.

값을 저장시 히스토리는 관리하지 않으며 기존의 데이터에 저장된 값을 업데이트한다.

설정에서 저장한 메시지는 정상 상태일시 약 2초후 노드로 자동 동기화가 된다.

○ 데이터 로딩

DB의 Node 테이블 스키마

name

문자열

max_length 40

state

정수형

default 0

사용여부(0 , 1, 2) (0 : 정상, 1 : 재난, 2 : 테러)

image

Blob

각개의 센서데이터를 계산 후 각 센서별로 단계값과 상태값을 DB상으로 리턴한다.

```

{GP_HUM: {grade: 4, status: 1}, GP_TC: {grade: 1, status: 0}, GP_NI: {grade: 1, status: 0},...}
▶ GP_CO: {grade: 2, status: 1}
▶ GP_HCN: {grade: 1, status: 0}
▶ GP_HUM: {grade: 4, status: 1}
▶ GP_NI: {grade: 1, status: 0}
▶ GP_O2: {grade: 4, status: 1}
▶ GP_TC: {grade: 1, status: 0}

```

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

```

(바) 센서 데이터를 이용한 재난전조 감지

○ 개요

센서에서 올라오는 실시간 데이터를 이용해서 재난 전조를 조기에 감지하면서, 센서의 일시적인 노이즈를 필터링하는 기능을 수행하는 이상전조 예측 모델을 생성하고 지속적인 센서 데이터를 이용한 학습을 통해서 안정적인 재난전조 예측 기술을 개발하고자 한다.

○ 센서 데이터의 개요

아래 그림 47.과 같이 일정한 범위의 값의 시계열 데이터에 정상상황과 이상상황의 데이터가 혼재되어서 들어오는 경우, 이상상황을 판별해 내는 방법론 총칭한다.

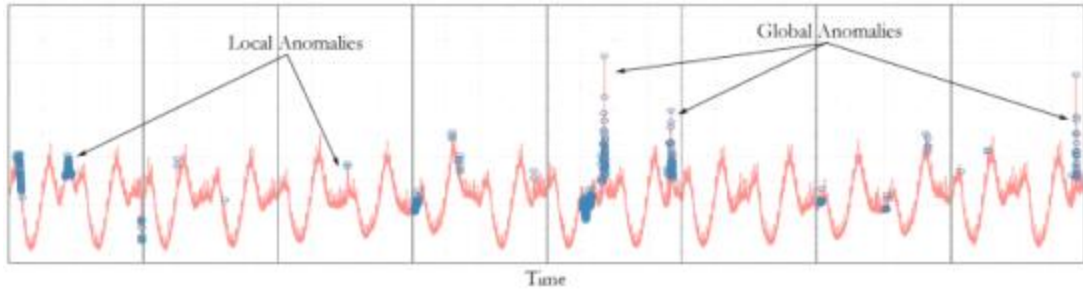


그림 47. 센서 데이터 노이즈 필터링

○ 센서 데이터 정상 범위 설정

기존의 센서 데이터를 각 센서 종류별로 시간별로 정리 후에 값의 범위를 설정해서 정상 범위를 선정한다.

마할로코프 거리 3을 정상범위로 설정해서 전체 데이터에 대해서 99%가 정상범위가 되도록 센서 값의 범위를 잡는다.

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 경우로 설정을 하도록 설정 한다.

○ 입력데이터

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

○ 입력 데이터 파싱

csv 모듈을 이용해서 데이터를 읽어들인다.

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 시간이다.

아래 파이썬 코드를 이용해서 읽어 들인다.

```
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 읽어 들여진다.

```

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

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

```

○ 데이터는 이중 리스트 형태

중요하진 않지만 일단 필요없는 첫번째 요소를 없애도록 한다.

가장 단순한 방법인 for-loop를 이용해서 새 변수에 필요한 것만 담도록 구현 한다.

아래와 같이 새변수 newMtxData에 필요변수를 할당 한다.

```

# 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

- c# set 함수를 이용해서 그룹 추출
- List()로 다시 리스트 변수로 변환

센서 별로 데이터를 아래와 같은 형식으로 리스트 모음으로 정의함.

#각 센서별로 데이터 파싱

```

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 한다.

마할라노비스 거리가 3이하의 범위를 각 센서에 대해서 정상범위로 설정한다.

○ 마할노비스 거리

- 정의 : 군과의 거리가 표준편차의 몇 배인지를 나타내는 값
- 공식 : $(\text{실측값} - \text{평균값}) / \text{표준편차}$
- 의의 : 어떤 값이 얼마나 일어나기 힘들 값이지, 또는 얼마나 이상한 값인지를 수치화 하는 방법이다.

○ 용도

- 가짜 데이터와 진짜 데이터의 구분용도 즉 outlier나 이상 검출에 활용된다.

적용 예

- 하루 교통량 평균이 20대이고 표준편차가 3대인 어느 날 차가 26대가 지나갔다고 하자. 자로 잰 평균과의 거리는 6이다. 그런데, Mahalanobis distance로는 $(26 - 20) / 3 = 2$ 이다. 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 비정상
의 정보를 출력한다.

○ 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 j 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, ~~거레가 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:
                    # 구해진 마할라노비스의 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')

```

(3-5) 공간 모델링

(가) 모델링

공간을 정의 및 계산하기 위하여는 한 공간을 선정하고, 선정된 공간에 대한 데이터를 컴퓨터가 이해할 수 있는 방식으로 취득하여 이 취득된 데이터를 통하여 컴퓨터 내에 공간을 구현하는 작업이 필요하다. 이것을 ‘공간 모델링’이라고 하며, 본 과제에서는 먼저 평면 데이터 상에서 정확한 치수를 추출하고, 추출된 치수를 사용하여 3D모델링 툴을 사용, 3D모델링 작업을 실시하여 피난 대피 동선 계산에 필요한 공간 데이터를 추출하였다.

○ 치수 추출

치수 추출은 한국철도공사의 테스트베드 도면을 사용하여 각 공간에 대한 데이터를 추출하는 방식으로 작업하였으며 예시된 도면은 아래 그림 48. 과 같다.

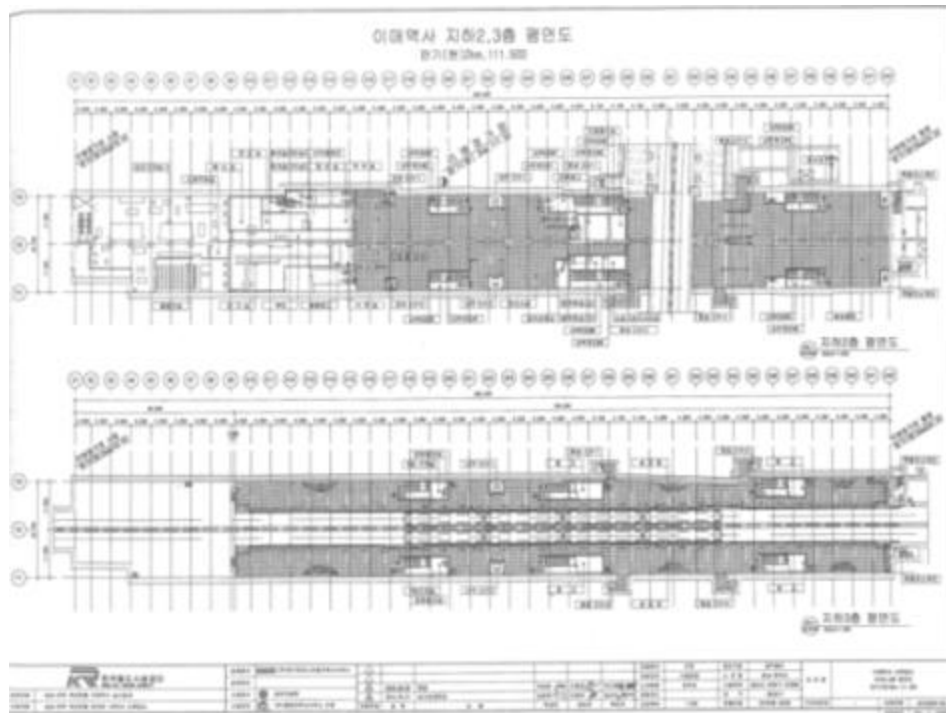


그림 48. 경강선 지하2, 3층 도면

○ 3차원 모델링

치수 추출 작업에서 정리된 데이터는 1차로 선별 작업을 거쳐서(평면도는 평면치수, 입면도는 높이치수)중간 파일로 정리하였으며, 중간 파일을 이용, 3D 모델링 작업을 진행하였고, 모델링된 데이터는 아래 그림과 같다.

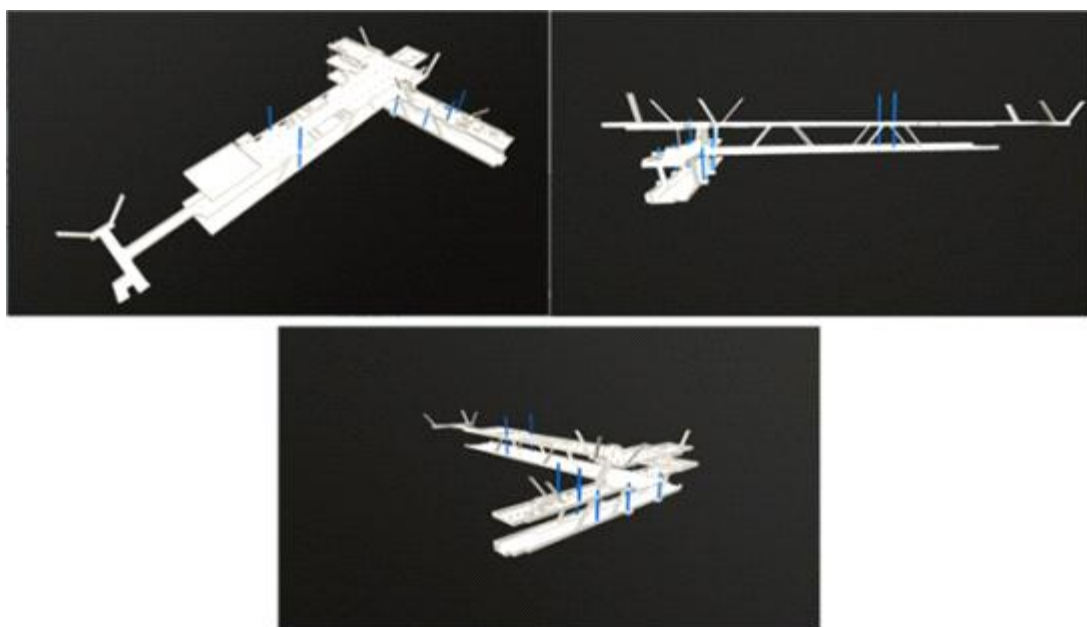


그림 49. 테스트베드 모델링 데이터

(3-6) 인증 시험

위와 같은 과정을 통하여 상황인지 재난대응 S/W(Sh-Monitoring β)를 개발하였으며 대구기계부품연구원을 통하여 2차에 걸쳐 3자 검사를 통하여 시험성적서를 발급 받았다.

(가) 1차 시험

1차 시험은 2020. 01. 02 - 03 기간 동안 이뤄졌으며 시험 항목 및 성적서는 아래와 같음.

표 1. 1차 3자검사 시험 항목

IoT 시스템 동작	IoT 모듈 동작 시험	합
	IoT 노드 - 서버간 통신 시험	합
피난대피 유도 모듈 동작	실시간 경로 안내 제어 시험	합



그림 50. 3자 검사 1차 시험 결과 및 내용

(나) 2차 시험

2차 시험은 2020. 05. 18 - 19 기간 동안 이뤄졌으며 시험 항목 및 성적서는 아래와 같음.

표 2. 2차 3자검사 시험 항목

경로 계산 시간	고위험지대 Only	0.244 Sec
	고위험지대 & 저위험지대	0.175 Sec
최적 경로 표출 (고위험 /저위험 지대 혼재 시 우회 경로 산출)		합
상황 감지용 센서 감지	1 점식	합
	다점식	합

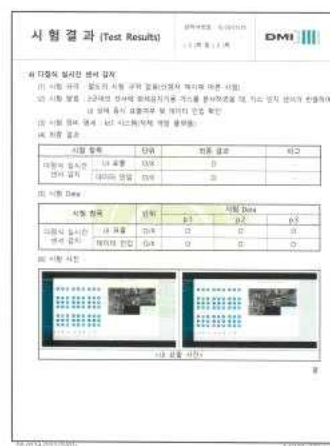
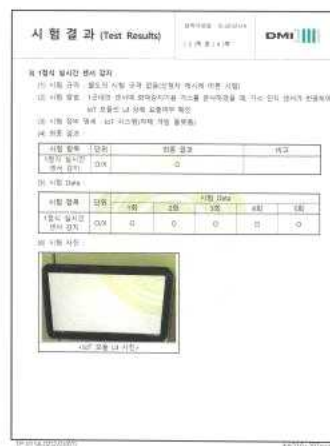
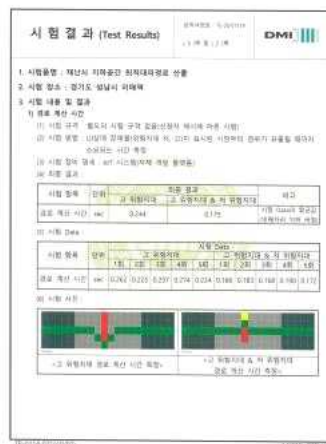


그림 51. 3자 검사 2차 시험 결과 및 내용

(4) 배터리 모듈 개발

일반적으로 다중이용시설에 재난상황이 발생하였을 경우, 주전원은 차단되게 되며, 비상 전력을 사용하게 된다. 본 과제에서도 주 전원이 차단된 상황을 상정하여 대응할 수 있는 비상 전력으로 이용키 위하여 배터리 모듈을 구상하였으며, 배터리 모듈에 요구되는 사항은 아래와 같았다.

- 주전원 차단시 IoT모듈 및 피난대피유도모듈의 전원 차단 없이 주전원 - 배터리 전력 간 전력 호환이 가능한 BMS혹은, 무정전 기능을 탑재할 것
- 평시상황에는 주전원(상전)에 연결되어 배터리를 충전할 것
- 배터리의 상태를 통신을 통하여 표출할 것
- 주전원(상전)의 단선시에도 40분 이상의 시스템 사용시간(500Wh이상의 용량)을 만족할 것
- 과출력, 과방전, 과충전등의 방지기능 탑재할 것

(가) 외함 설계

외함은 리튬이온배터리의 수분에 닿으면 폭발할 수 있는 위험성에 따라 완전 밀폐방식으로 제작되었으며, 재질은 SUS에 분체 도장 방식으로 제작되었고, 볼팅 고정을 위한 날개, 운반용 손잡이, RS-485통신 포트, 스위치, 전원연결용 터미널 등의 요구사항에 맞춰 그림 52. 와 같이 설계하였다.

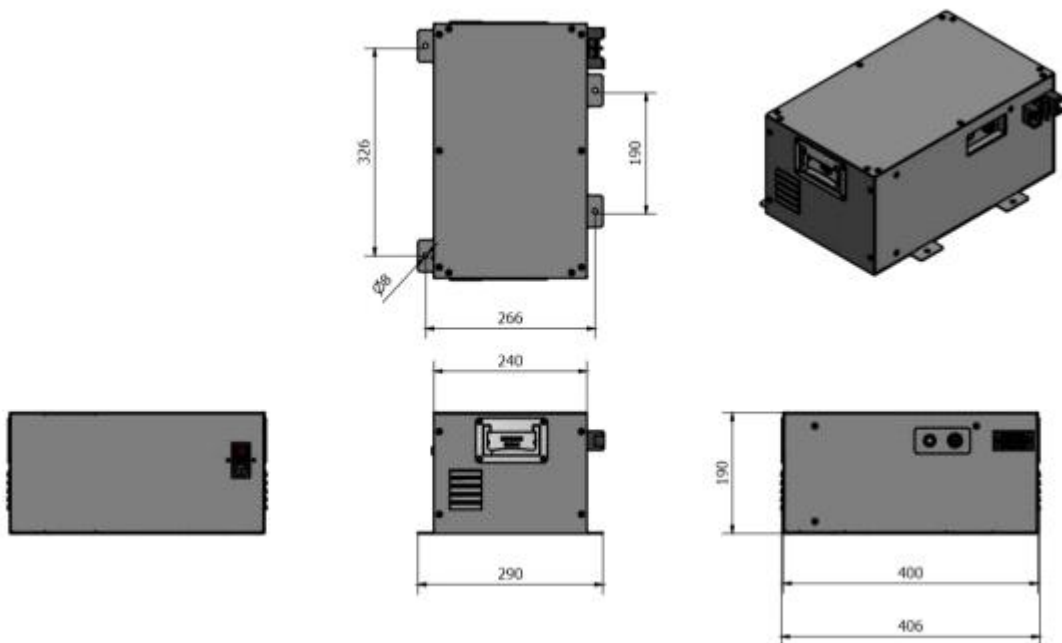


그림 52. 배터리 외함 3차원 및 2차원 설계도

(나) 통신 설계

배터리 모듈의 필요 사양 검토 시 부터 배터리의 정보 표출 및 작동 상태를 원격으로 확인해야 할 필요성이 있다고 검토되었으며, 이에 따라 피난대피 유도 모듈의 PLC제어와 기존의 소방장비와의 동일한 통신 방식을 고려하여 RS-485s 표준을 선정하였으며 상세한 제원은 Baudrate 9600, 1 Stopbit, No Parity, Little Endian type Byte order로 개발하였고, 상세한 프로토콜은 아래와 같이 개발되었으며

Section	Contents	Start Byte	Start Bit	Bit Length	Valid Range	Real Value Range	Resolution	Unit	Value Type	Description
Status Request (외부 -> BMS)	STX	0x02	0	8	-	-	-	-	Unsigned	
	Dummy Packet	4(0x52)	1	8	-	-	-	-		
	Dummy Packet	5(0x53)	2	16	-	-	-	-		
	Dummy Packet	4(0x54)	3	24	-	-	-	-		
	Dummy Packet	8(0x58)	4	32	-	-	-	-		
	Dummy Packet	5(0x55)	5	40	-	-	-	-		
	CRC16	-	6	48	16	-	-	-	Unsigned	STX, ETX 제외 Data의 CRC16 (고정값 0x1691)
	ETX	0x03	8	64	8	-	-	-	Unsigned	

Section	Contents	Start Byte	Start Bit	Bit Length	Valid Range	Real Value Range	Resolution	Unit	Value Type	Description
Status Response (BMS -> 외부)	STX	0x02	0	8	-	-	-	-	Unsigned	
	Station ID	-	1	8	0~255	0~255	1	-	Unsigned	Station ID
	전압	-	2	16	0~65535	0~100.0	0.1	V	Unsigned	
	전류	-	4	32	-32767~32767	-100.0~100.0	0.1	A	Signed	전류 값이 음수 -> 충전 전류 값이 양수 -> 방전
	SOC	-	6	48	0~255	0~100	1	%	Unsigned	
	충전 충전전 Cycle	-	7	56	0~65535	0~65535	1	-	Unsigned	
	Warning/Fault 정보 (상대 정보)	-	8	72	8	0~8	-	-	Unsigned	Bit 0 - Warning (0-정상, 1-알림) Bit 1 - Fault (0-정상, 1-알림)
	Warning 정보 #1	-	10	80	8	-	-	-	Unsigned	Bit 0 - Over Voltage Protection Bit 1 - Under Voltage Protection Bit 2 - Over Current Protection Bit 3 - Charge Over Current Protection Bit 4 - Over Temperature Protection Bit 5 - Charge Over Temperature Protection Bit 6 - Under Temperature Protection Bit 7 - Charge Under Temperature Protection Warning은 경고 신호이며 출력은 정상 동작
	Warning 정보 #2	-	11	88	8	-	-	-	Unsigned	Bit 0 - Cell Diff Protection Bit 1 - Voltage Sensing Protection Bit 2 - Temperature Sensing Protection Warning은 경고 신호이며 출력은 정상 동작
	Fault 정보 #1	-	12	96	8	-	-	-	Unsigned	Bit 0 - Over Voltage Protection Bit 1 - Under Voltage Protection Bit 2 - Over Current Protection Bit 3 - Charge Over Current Protection Bit 4 - Over Temperature Protection Bit 5 - Charge Over Temperature Protection Bit 6 - Under Temperature Protection Bit 7 - Charge Under Temperature Protection Fault는 차단 신호이며 출력은 정지
	Fault 정보 #2	-	13	104	8	-	-	-	Unsigned	Bit 0 - Cell Diff Protection Fault는 차단 신호이며 출력은 정지
	CRC16	-	14	112	16	-	-	-	Unsigned	STX, ETX 제외 Data의 CRC16
	ETX	0x03	16	128	8	-	-	-	Unsigned	

Section	Contents	Start Byte	Start Bit	Bit Length	Valid Range	Real Value Range	Resolution	Unit	Value Type	Description
Station ID Set (외부 -> BMS)	STX	0x02	0	8	-	-	-	-	Unsigned	
	Dummy Packet	4(0x52)	1	8	-	-	-	-		
	Dummy Packet	5(0x53)	2	16	-	-	-	-		
	Dummy Packet	4(0x54)	3	24	-	-	-	-		
	Dummy Packet	8(0x58)	4	32	-	-	-	-		
	Dummy Packet	5(0x55)	5	40	-	-	-	-		
	Dummy Packet	5(0x55)	6	48	-	-	-	-		
	Station ID	-	7	56	0~255	0~255	1	-	Unsigned	Setting Station ID
	CRC16	-	8	64	16	-	-	-	Unsigned	STX, ETX 제외 Data의 CRC16
	ETX	0x03	10	80	8	-	-	-	Unsigned	

그림 53. 배터리 통신 프로토콜

Addressable Firmware 수정을 거쳐 최종적으로 이매역 테스트베드에 설치된 모듈은 아래와 같다.



그림 54. 배터리 모듈

나. 테스트베드 구축

(1) 철도공사 협의

(가) 설치역사 선정

표 3. 설치 후보 역사 목록

역명	접근성*	복잡도**	설치 편의성	최종 점수
강남	상	상	하	10
가락시장	상	상	하	10
고속터미널	상	상	하	10
교대	상	중	중	20
도곡	중	상	상	15
동묘앞	중	중	중	10
모란	상	중	중	20
북정	상	하	상	30
사당	상	상	하	10
서울	하	상	하	0
선정릉	중	중	상	20
수서	상	중	중	20
수원	하	상	중	5
신도림	하	중	중	10
왕십리	중	상	하	5
양재	상	상	하	10
영등포	하	중	하	5
용산	중	중	하	10
이매	상	하	상	30
이수	중	상	하	5
잠실	하	상	하	0
합정	하	상	하	0
홍대	하	상	중	5

*.접근성 = 본사로부터의 이동 소요시간

**.복잡도 = 운행구간 + 연계시설

이번 과제를 위한 테스트 베드는, 2개 노선 이상이 지나가는 환승 역사를 고려하였으며, 추가로 접근성, 복잡도, 설치 편의성 등이 고려되었으며, 최종적으로는 표 3. 22개의 역사들중, 이매역으로 선정되었다.

(나) 설치품목 정리

테스트 베드 설치 품목은 표 4. 와 같이 정리되었다.

표 4. 테스트 베드 설치 품목

품목	수량	소모전력량
IoT모듈	28	52W
피난대피유도모듈	700	1.3W
CCTV	8	25W
배터리	2	840W
데이터 서버	2	300W

(다) 지역본부 협의

정리된 설치품목을 토대로 2019.06.18. 코레일 대전본부에서 동부분부와 협의 진행하였으며 최종적으로 분당선/경강선 환승역인 이매역으로 철도공사측의 승인되었다.



그림 55. 한국철도공사 광역(동부)본부 협의

테스트 베드 선정 이후 설치 방법에 관한 협의 및 전기 공사에 관련하여 분당 전기사업소와 협의 진행하였으며 2019.12.19. 이매역사에 대한 테스트베드 구성에 관련하여 최종 협의가 이루어졌다.

(라) 설치 계획 구성

동부분부 및 분당전기사업소와의 협의를 통하여 이매역사에 최종적으로 IoT모듈 28대, 피난대피유도모듈 총연장 3.8Km, CCTV 8대, 배터리 2개, 데이터 서버2대설치를 계획하였으며, 상세한 계획은 아래 그림 56. ~ 59. 와 같다.



그림 56. 경강선 이매역사 지하1층 평면도

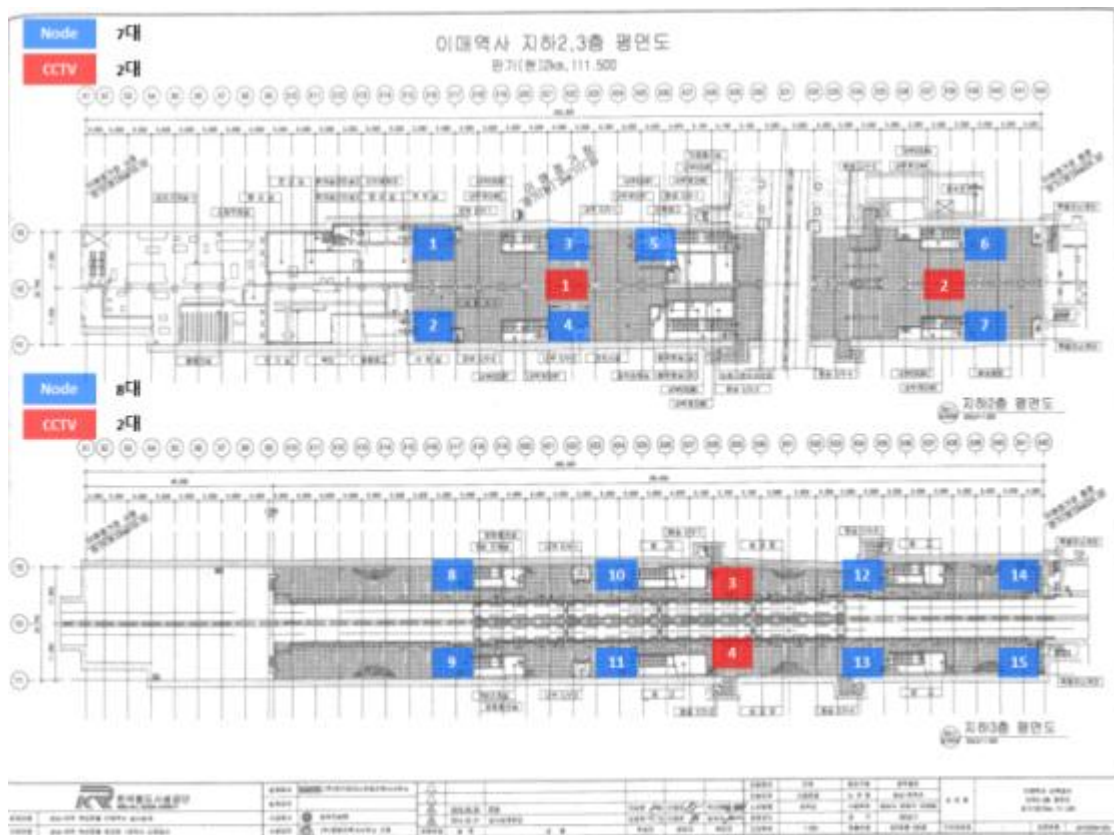


그림 57. 경강선 이매역사 지하2, 3층 평면도

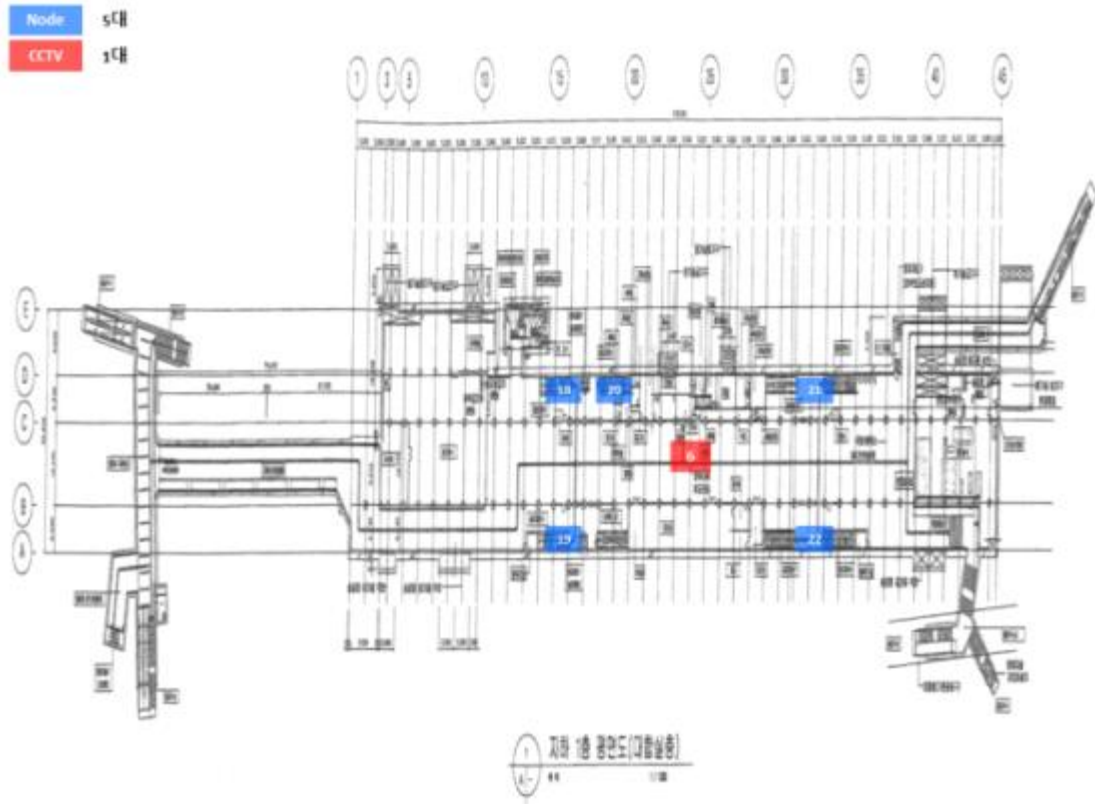


그림 58. 경강선 이매역사 지하1층 카메라 설치 위치

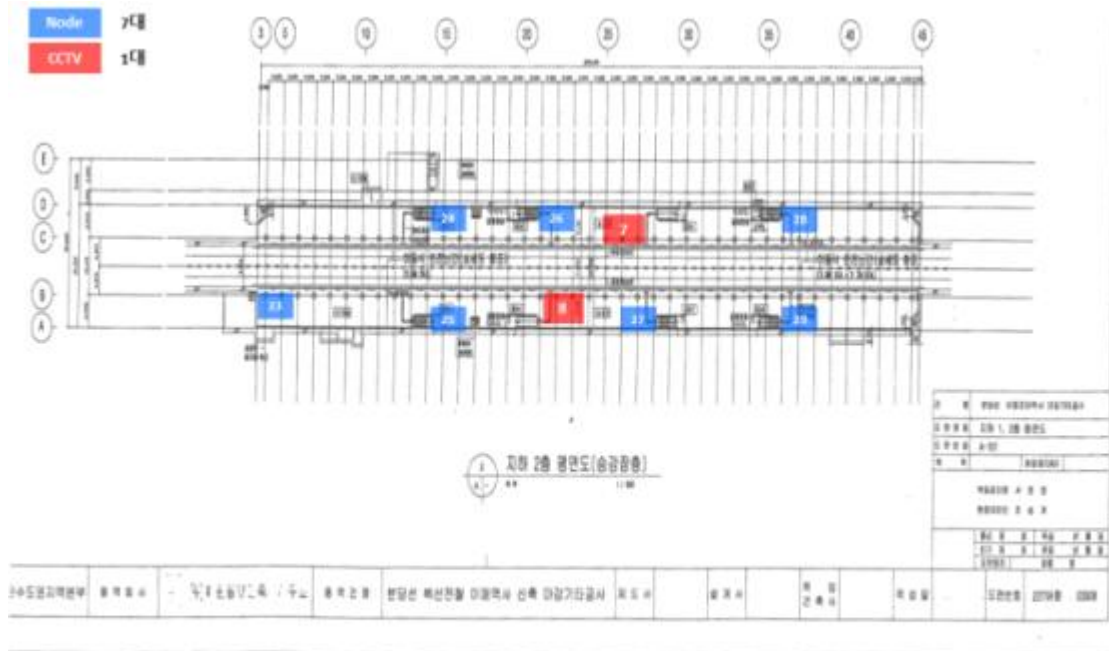


그림 59. 경강선 이매역사 지하2, 3층 카메라 설치 위치

(2) IoT모듈, 피난대피유도모듈 설치

지하역사의 특성상 주간(영업시간)에는 작업이 불가하였으며, 본 과제의 진행을 위하여 야간의 막차이후부터 첫차 시작 이전인 차단시간에 작업이 진행되었으며, 한국철도공사 시설처 및 동부분부, 이매역의 관리역사인 수서역 및 야탑역의 협조하에 진행되었다.

(가) 모듈 설치

IoT모듈은 직결 피스 및 칼블럭을 사용하여 시공되었으며, IoT모듈 28개소, 피난대피유도모듈 700세트, 총 연장 3.8km 시공하였으며, 작업 과정은 그림 60. 과 같다.



그림 60. 이매역 테스트베드 설치 공사

(나) 통신 공사

①RS-485

설치 작업 이후 서버실에 설치된 PLC - 피난대피유도모듈의 컨트롤러간 통신을 담당하는 RS-485케이블의 포설 공사가 이어졌으며 공사 방식은 기존의 설치된 피난대피유도모듈 내부에 RS-485케이블을 포설하는 방식으로 작업하였고, 스펙 상으로 총연장 1.8Km인 RS-485의 통신거리 한계로 인하여 RS-485 리피터를 중간 환승구간에 설치하여 테스트 베드를 구성하였다.



그림 61. RS-485 리피터



그림 62. RS-485 공사

②유/무선공사

RS-485공사를 통하여 피난대피유도모듈의 제어까지 가능하게 되었으며, 테스트 베드 구축에서 필요한 요소는 Node에서 들어오는 데이터들의 서버 저장 및 D/B연동, 생활 정보 표출, CCTV 영상 데이터 연동을 위하여 Ethernet공사가 필요하였으며, Ethernet 공사는 6 군데의 광허브설치 및 광케이블 포설, RJ45(UTP)케이블포설의 순서로 작업하였다.



그림 63. 이매역 유/무선 공사

다. 지능형 밀집예측 기술 개발

(1) 데이터의 수집

2018년도 코레일 월보데이터를 통해 일별/시간별 승,하차인원 합산데이터를 구축하였다.

	DT	STATION_NM	ON_CNT	OFF_CNT
1	2018-01-01 00:00:00	이대	2	3
2	2018-01-01 00:00:00	이대	4	12
3	2018-01-01 01:00:00	이대	0	0
4	2018-01-01 01:00:00	이대	0	0
5	2018-01-01 02:00:00	이대	0	0
6	2018-01-01 02:00:00	이대	0	0
7	2018-01-01 03:00:00	이대	0	0
8	2018-01-01 03:00:00	이대	0	0
9	2018-01-01 04:00:00	이대	0	0
10	2018-01-01 04:00:00	이대	4	0
11	2018-01-01 05:00:00	이대	11	1
12	2018-01-01 05:00:00	이대	49	37

그림 64. 월보데이터 예시

기상청의 지역별상세관측자료(AWS) 수집기를 작성하였다.

	TEMPERATURE	RAIN_15M	RAIN_3H	WIND1_SPD
1	-0.5	0	<null>	1.1
2	-0.8	0	<null>	3.2
3	-1	0	<null>	1.8
4	-1.5	0	<null>	2.9
5	-1.9	0	<null>	1.8
6	-1.9	0	<null>	2.9
7	-2.2	0	<null>	1.8
8	-1.8	0	<null>	2.9
9	-1.8	0	<null>	3.6
10	-0.8	0	<null>	3.6
11	0.8	0	<null>	2.5
12	3.4	0	<null>	2.5
13	4.9	0	<null>	5.8

그림 65. 기상데이터 예시

잠실 경기장의 야구, 축구, 농구 경기시간 데이터 수집기를 작성하였다.

	MEASURE_DT	FOOTBALL	BASEBALL	BASKETBALL
1	2017-01-01 15:00:00	0	0	1
2	2017-01-01 16:00:00	0	0	1
3	2017-01-01 17:00:00	0	0	1
4	2017-01-04 18:00:00	0	0	1
5	2017-01-04 19:00:00	0	0	1
6	2017-01-04 20:00:00	0	0	1
7	2017-01-07 13:00:00	0	0	1
8	2017-01-07 14:00:00	0	0	1
9	2017-01-07 15:00:00	0	0	1
10	2017-01-08 13:00:00	0	0	1
11	2017-01-08 14:00:00	0	0	1
12	2017-01-08 15:00:00	0	0	1

그림 66. 야구, 축구, 농구 경기시간 데이터 예시

공휴일 정보 수집기를 작성하였다. (공공데이터포털의 특일정보 API연동)

	DT	DATENAME	IS_HOLIDAY
1	2015-01-01	신정	Y
2	2015-02-18	설날	Y
3	2015-02-19	설날	Y
4	2015-02-20	설날	Y
5	2015-03-01	삼일절	Y
6	2015-05-05	어린이날	Y
7	2015-05-25	석가탄신일	Y

그림 67. 공휴일정보 데이터 예시

작성된 수집기는 과거의 데이터를 포함하여 실시간 데이터를 수집할 수 있도록 작성되었으며, 이매역의 분당선과 경강선에 대한 실제 현장조사 진행 (2019년 7월 첫주) 하였다.

현장조사의 영역은 분당선 상행/하행, 경강선 상행/하행을 4개 구역으로 구분하여 총 16개 구역으로 분할 하였으며, 현장조사는 열차의 도착 시간을 기준으로 하며 머무르고 있는 인원을 기준으로 잡기 위해 승차 인원 수치로 사용하였다.

(2) 수집데이터 분석

(가) 시간별 승하차 통계데이터 분석

아래의 데이터는 요일별 승하차인원 수치를 시각화한 것으로, 수치가 특정 패턴을 보임을 알 수 있다.

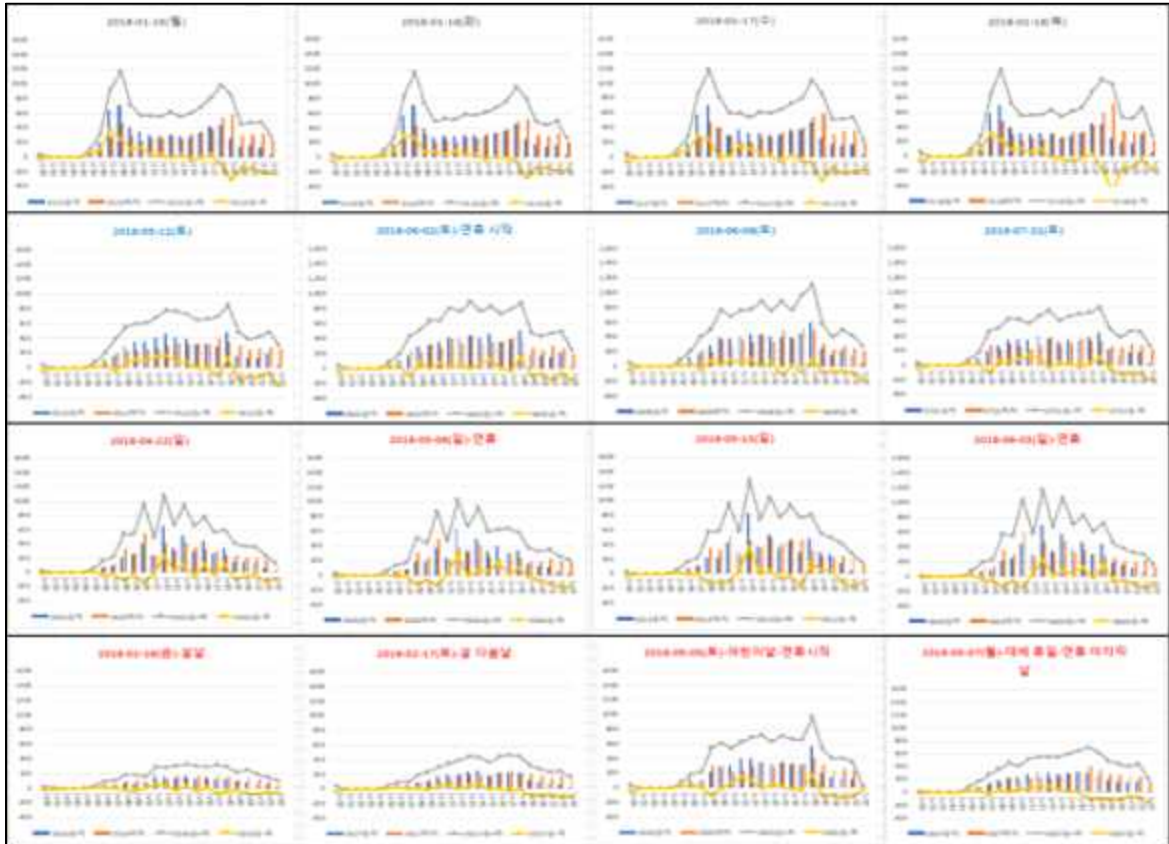


그림 68. 평일/토요일/일요일/공휴일의 승하차 인원 패턴 예시

아래는 일별기온과 승하차인원의 데이터를 시각화 한 것으로 패턴에 크지 않지만 환경변화도 다소간의 영향을 보임을 알 수 있다.

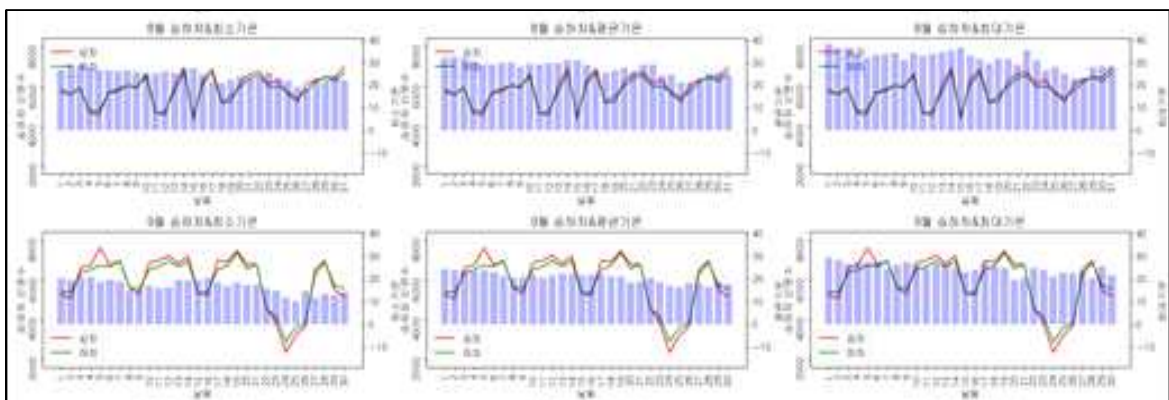


그림 69. 일별 기온과 승하차인원 비교예시

(나) 수집요소 상관관계 분석

아래의 상관관계 그래프와 같이 승하차 인원의 데이터는 요소들 중 시간 데이터의 영향이 가장 크게 미치는 것을 볼 수 있다.

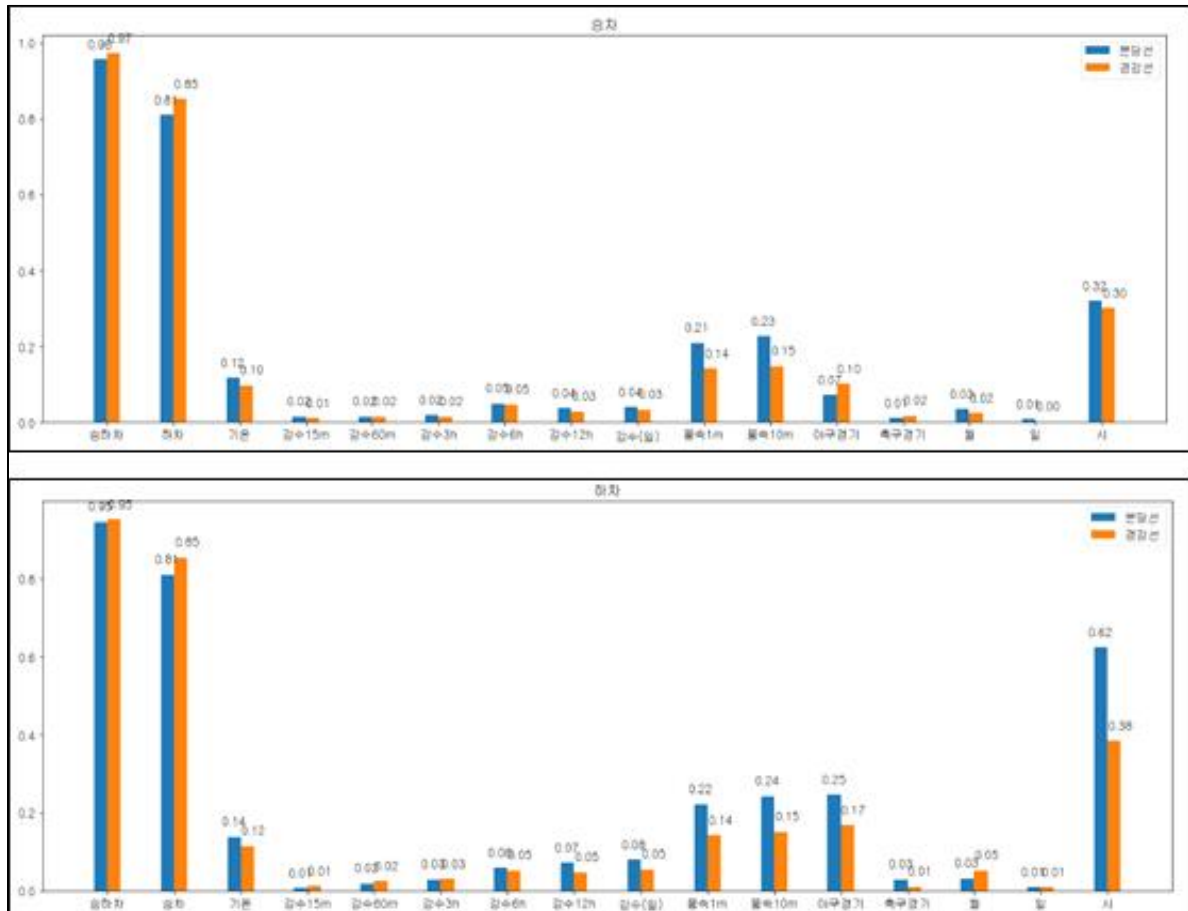


그림 70. 수집데이터와 승하차인원의 상관관계 그래프

(3) 현장수집 데이터 가공

(가) 현장수집 데이터 분석

현장의 데이터를 실시간으로 자동수집할 수 있는 수단이 존재하지 않는다. 따라서, 인력으로 각 승차장별 A, B, C, D의 4개 구역으로 나누어 현장데이터를 수집하였으며, 월보데이터의 시간별 승하차 인원과 유사한 패턴을 확인하였다. 그 결과 경강선과 분당선의 배차간격 차이가 크다는 것을 알 수 있었다.

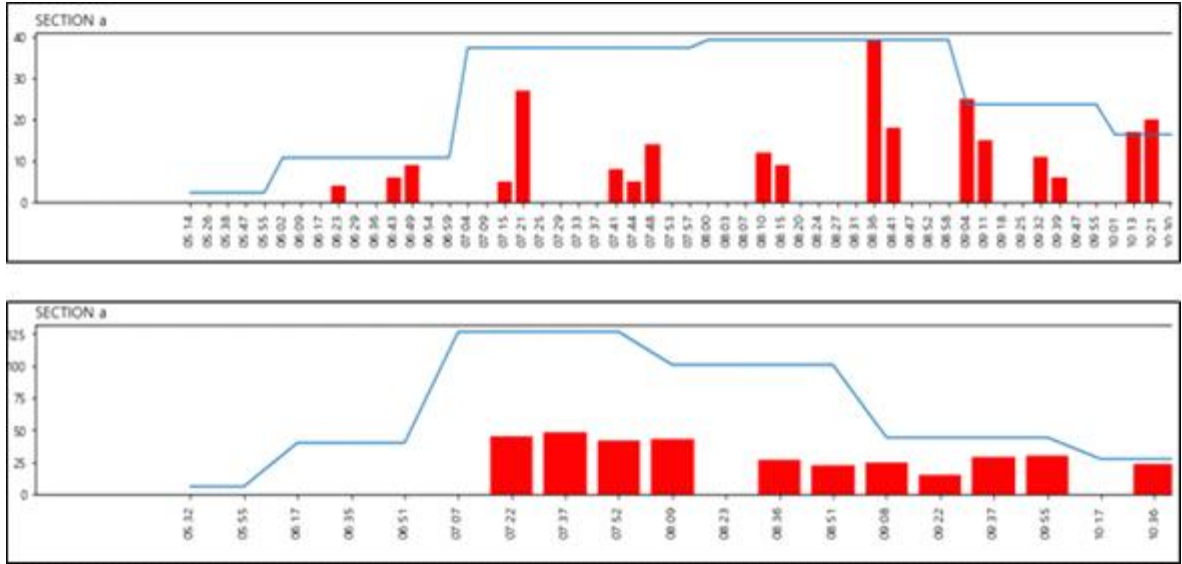


그림 71. 분당선(위)과 경강선(아래)의 현장조사 데이터 일부 (실선은 월보데이터)

분당선과 경강선 승강장 내 머무르는 인원은 환승역의 열차도착에 영향을 받는다. 분당선은 환승열차의 도착여부에 따라 인원수가 많은 경우 적은 경우로 나뉘어 진다.

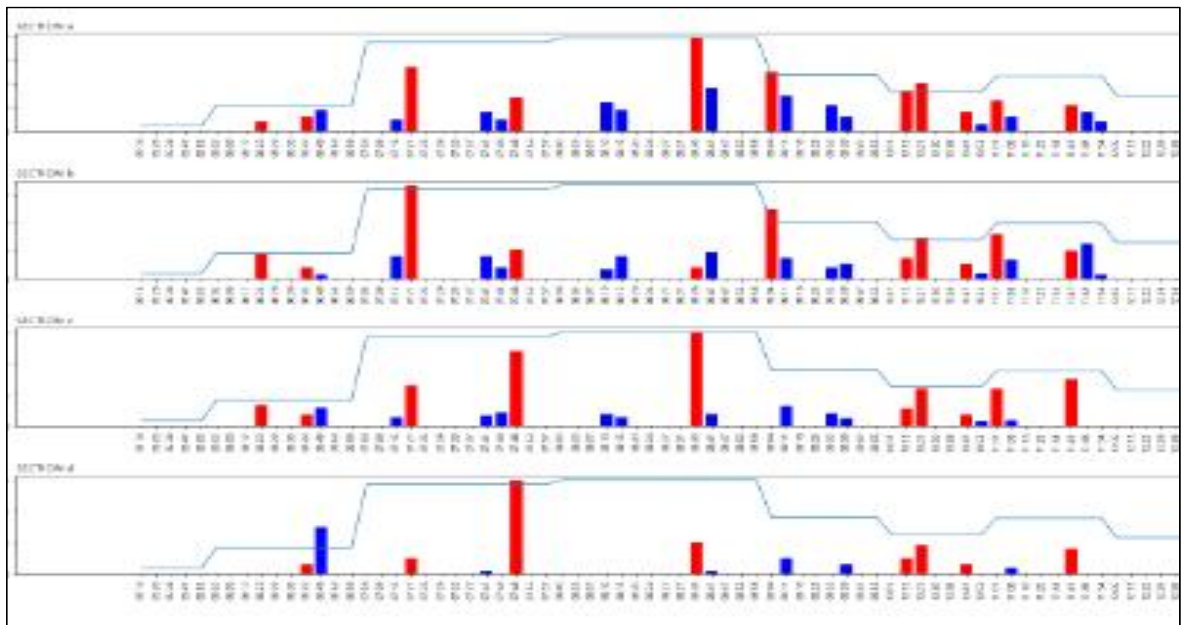


그림 72. 환승열차의 도착(붉은색)과 미도착(파란색)시의 차이

경강선의 경우 환승역 도착이 거의 균등하여 큰 의미는 찾기 힘들었다.

(나) 현장수집 데이터 조정

실제 데이터와 전년도 같은 기간 월보데이터와 비교확인 하였다. 각 시간별 높은 비율과 낮은 비율의 2가지 경우에 대한 비율데이터 생성하였으며, 조사데이터를 환승역 도착시와 미도착시의 데이터로 분리하였으며, 함월보데이터와 2가지 비율데이터를 각각 조합하였다.

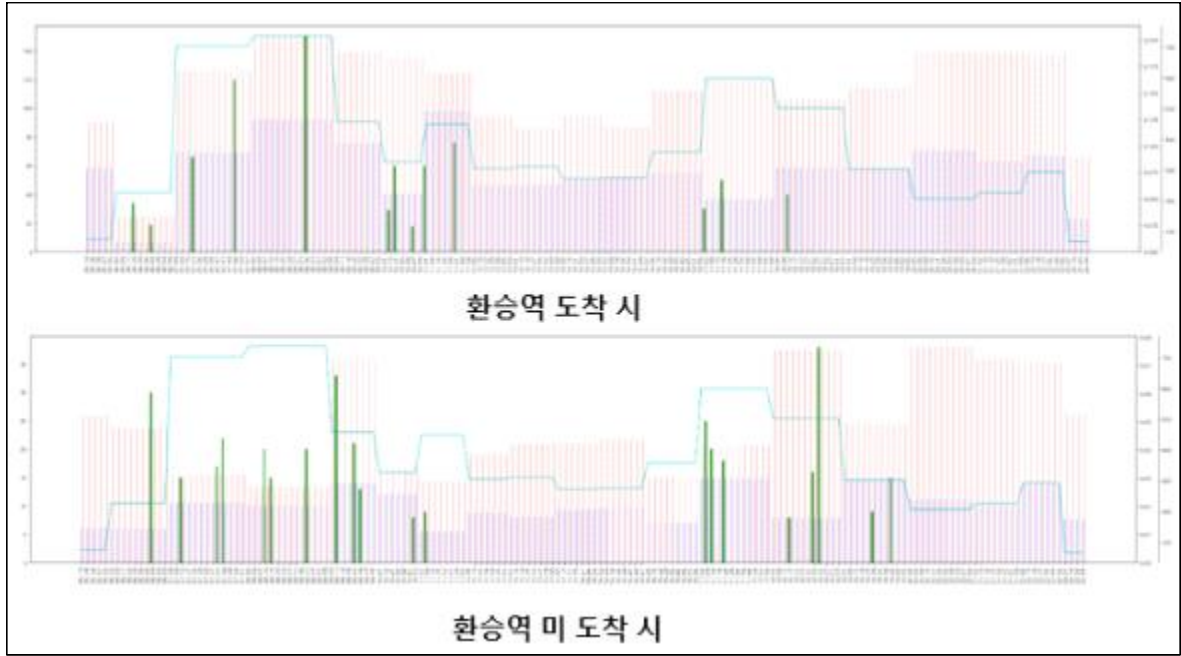


그림 73. 조사데이터, 월보데이터, 비율데이터의 시각화

각 승차장의 A, B, C, D 구역에 대해 데이터 생성 로직을 구현 하였으며, 비율 범위 내 난수 처리 및 증감에 따른 수치 변화경향을 보완처리 하였다.

분당선의 경우 환승열차 도착/미도착 구분하여 별도 생성 후 병합처리 하였다.

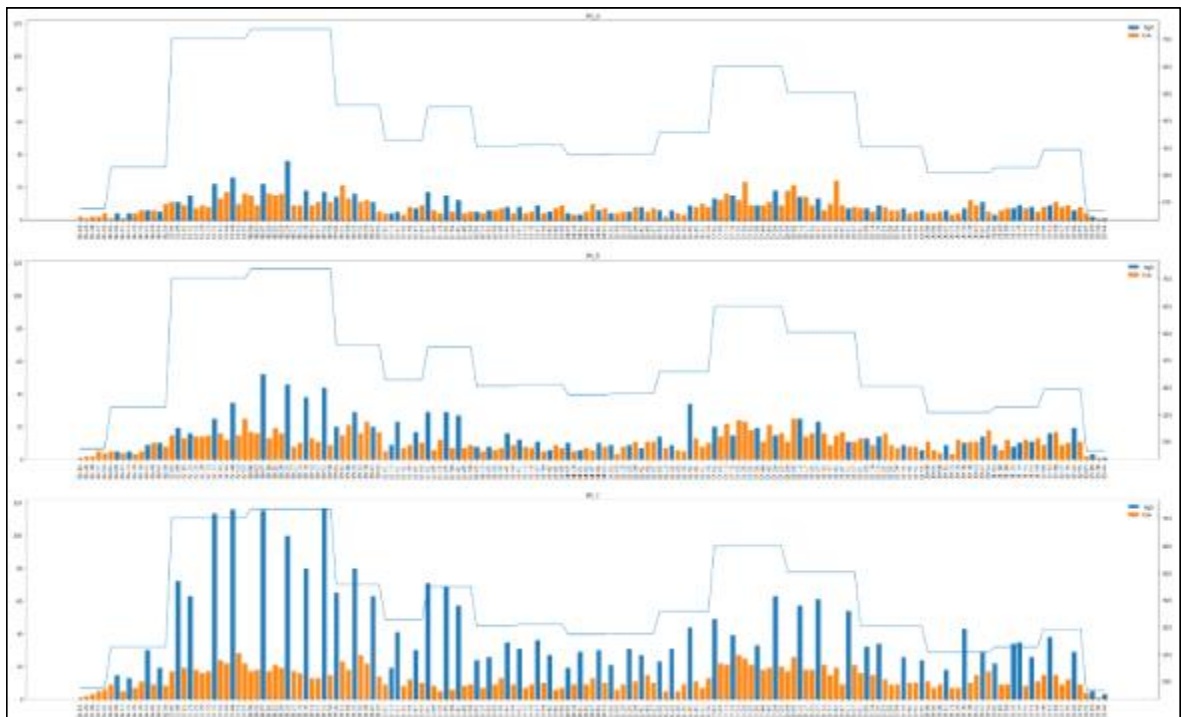


그림 74. 생성된 분당선 데이터 예시

(4) 지하공간 밀집도 예측 모델 개발

(가) 데이터의 통합

시간데이터는 월, 시간, 분으로 분리하였다. 시간을 하루중의 분으로만 표현한 시간데이터로 확장하여 평일, 금요일, 토요일, 일요일 데이터를 원 핫 인코딩 방식으로 변환 하였으며, 통합은 생성된 데이터의 열차 도착시간을 기준으로 하였다.

(나) 데이터 모델링

모델은 Dense Layer로 구성된 신경망을 구성하였으며, 전체 데이터셋에 대하여 Standard와 Min/Max를 각각 적용한 2개의 데이터셋을 생성하였다. 특성의 중요도를 각 3%, 7%, 10% 이상인 경우로 제한하여 학습을 진행하였으며, 학습을 제한하는 Early Stop의 적용/미적용 하였다. 전처리 방식에 따른 차이는 미미하지만, Standard가 조금 더 나은 모습을 보였다.

[standard]								
	val_loss	val_mae	val_mse	val_mape	loss	mae	mse	mape
995	7.48062	2.08174	7.48062	40.9806	6.95261	1.94939	6.95261	39.7992
996	6.98475	1.955	6.98475	35.2141	6.98056	1.94901	6.98057	39.5663
997	7.61781	2.03542	7.61781	35.1723	6.95675	1.94999	6.95675	39.344
998	7.35033	1.98786	7.35033	35.0814	6.95023	1.95035	6.95023	39.3972
999	7.04858	1.9545	7.04858	36.1357	7.0083	1.95729	7.0083	39.8832
[MinMax]								
	val_loss	val_mae	val_mse	val_mape	loss	mae	mse	mape
995	7.83691	2.19308	7.83691	48.443	7.2699	1.9834	7.2699	40.0524
996	7.27733	2.00567	7.27733	37.697	7.23481	1.98912	7.23481	40.245
997	7.29726	1.98639	7.29726	34.8174	7.24027	1.98722	7.24027	40.3038
998	6.95829	1.96219	6.95828	36.3712	7.24649	1.98039	7.24649	40.1171
999	7.09223	2.01926	7.09222	39.8467	7.24628	1.98507	7.24628	40.035
[standard - MinMax]								
	val_loss	val_mae	val_mse	val_mape	loss	mae	mse	mape
0	-0.356291	-0.111345	-0.356293	-7.46237	-0.31729	-0.0340167	-0.317288	-0.253193
1	-0.292583	-0.0506756	-0.292583	-2.48291	-0.254242	-0.0401181	-0.254241	-0.678726
2	0.320553	0.0490264	0.320554	0.354904	-0.283526	-0.0372267	-0.283525	-0.959713
3	0.392045	0.0256695	0.392047	-1.28978	-0.296259	-0.030043	-0.296258	-0.719971
4	-0.043647	-0.0647641	-0.0436459	-3.71098	-0.237981	-0.0277809	-0.237984	-0.151817

그림 75. 전처리 방식에 따른 성능 차이

중요도 10% 이상의 Early Stop방식의 모델이 가장 나은 모습을 보였다.

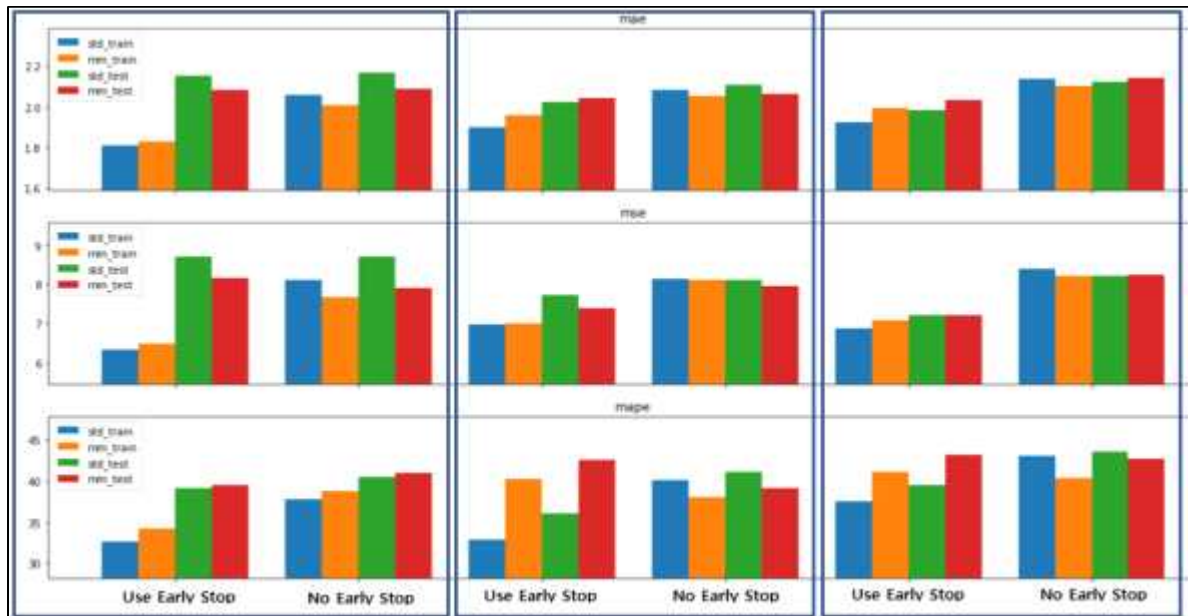


그림 76. 특성 중요도 3% 이상(좌측), 7% 이상(가운데), 10% 이상(우측)의 제한에 따른 결과

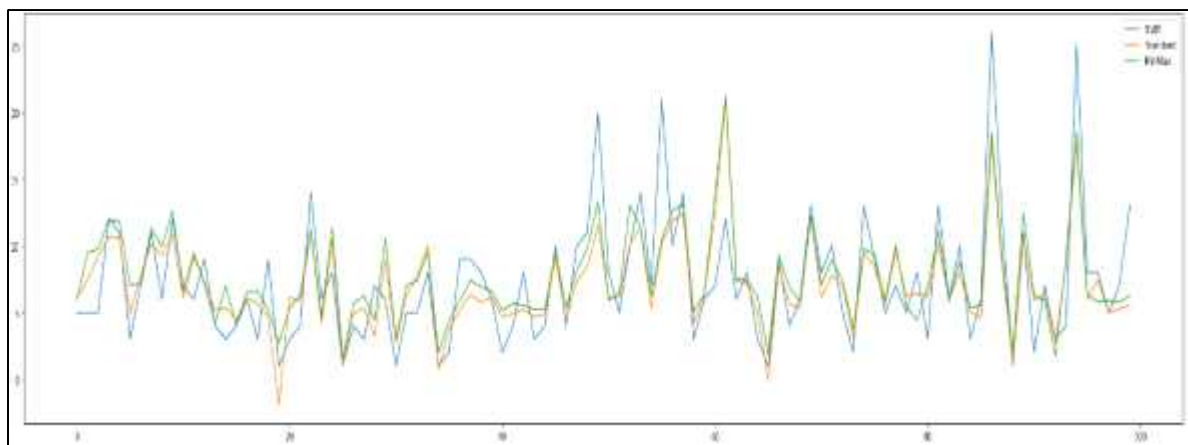


그림 77. 모델의 예측결과 비교(100개)

(5) 예측모델 변경 및 추정모델 적용

인위적으로 만들어진 데이터가 예측모델의 기본 바탕이 되는 점을 고려하여, 월보 데이터의 시간별 승하차인원 합산데이터를 기반으로 모델을 재생성하였다. 기존의 데이터 생성 로직을 활용하여 예측결과로 각 구역별 인원을 추정토록 하고, 각 구역별 밀집도 추정 모듈을 개발하였다.

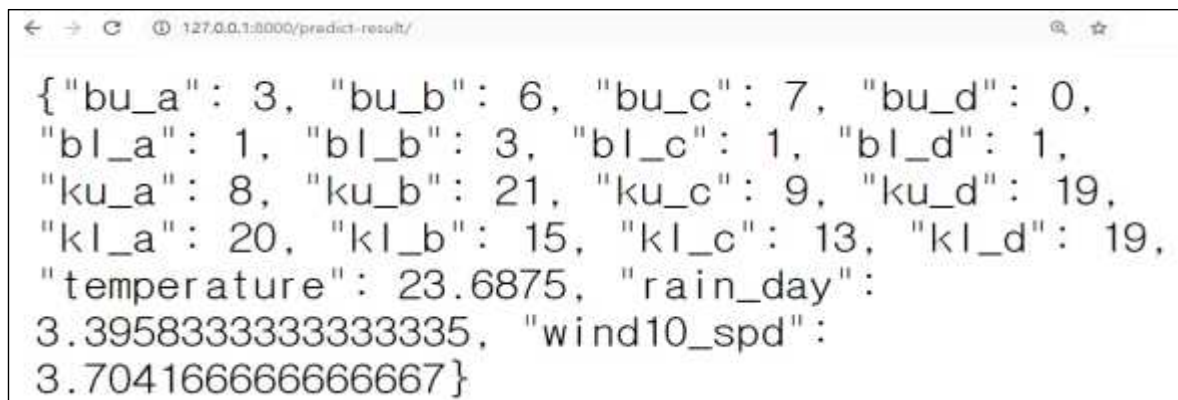
```

In[13]: test_dt = datetime(2019, 3, 25, 8, 6)
...: e = DensityEstimator()
...: dict_result = e.estimate(test_dt)
In[14]: dict_result
Out[14]:
{'bu_a': 17,
 'bu_b': 33,
 'bu_c': 95,
 'bu_d': 8,
 'bl_a': 42,
 'bl_b': 24,
 'bl_c': 18,
 'bl_d': 2,
 'ku_a': 6,
 'ku_b': 6,
 'ku_c': 4,
 'ku_d': 5,
 'kl_a': 21,
 'kl_b': 18,
 'kl_c': 11,
 'kl_d': 22}

```

그림 78. 추정모듈 사용 예시 및 결과

웹을 통해 추정결과를 받을 수 있도록 API를 작성하였다.



```

{"bu_a": 3, "bu_b": 6, "bu_c": 7, "bu_d": 0,
 "bl_a": 1, "bl_b": 3, "bl_c": 1, "bl_d": 1,
 "ku_a": 8, "ku_b": 21, "ku_c": 9, "ku_d": 19,
 "kl_a": 20, "kl_b": 15, "kl_c": 13, "kl_d": 19,
 "temperature": 23.6875, "rain_day":
 3.3958333333333335, "wind10_spd":
 3.7041666666666667}

```

그림 79. 브라우저에서 API를 직접 호출 한 예시

(6) 지하공간 밀집 추정 모델 시각화 개발

아래와 그림과 같이 지하공간 밀집 추정 모델 시각화 시스템을 개발 하였으며,

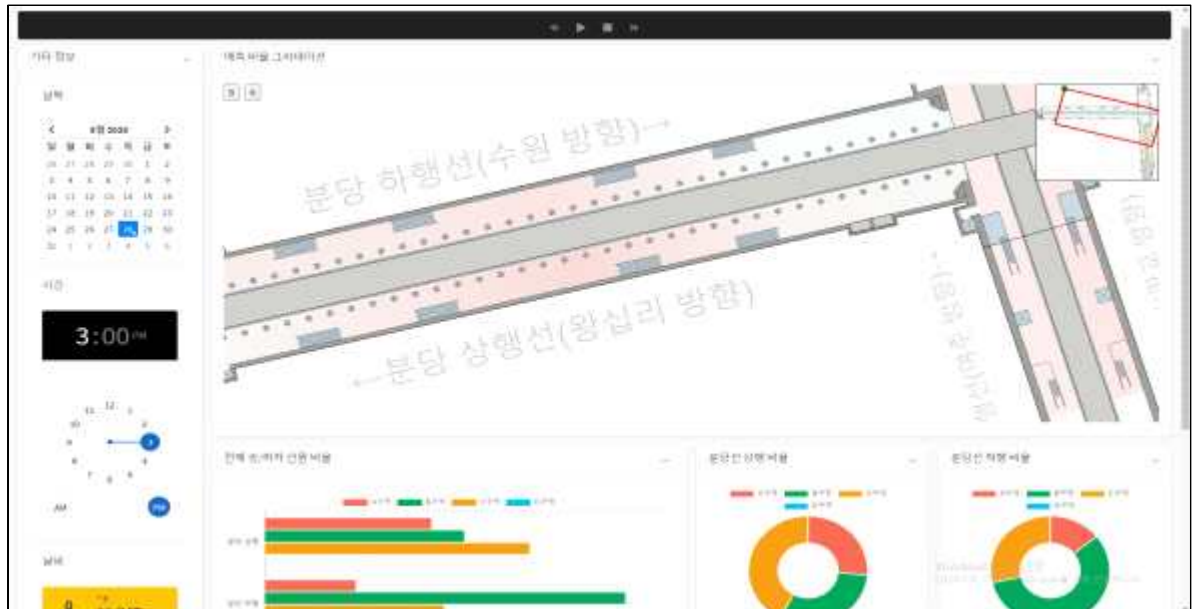


그림 80. 지하공간 밀집 추정 모델 시각화 시스템 화면

시스템은 밀집도 추정 API를 주기적으로 호출하는 연속 모니터링 모드와 날짜 및 시간을 선택하여 특정 시점에 대한 추정데이터를 확인하는 고정 모드, 그리고 도면상의 각 승차장별로 밀집도에 대한 비율을 그래데이션으로 시각화하였다. 화면 하단부에 전체, 승차장별 추정인원 비율에 대한 차트를 표시하고 하단부 좌측에 추정데이터로 사용된 환경정보를 확인할 수 있도록 하였다.



그림 81. 지하공간 밀집 추정 모델 시각화 시스템 하단부

라. 대피훈련 비상관리 기술 개발

(1) 국내외 현황

(가) 해외 현황

①재난 대응 시스템

EOC(Emergency Operations Center; 재난안전대책본부)에서 재난 발생 시 운영하는 시스템으로 E-Team, DisasterLan, 그리고 webEOC 등이 있다. 여기서는 대표적인 webEOC에 대하여 살펴보기로 한다.

webEOC는 ESI 회사 제품으로 NIMS(National Incident Management System) 기능과 ICS(Incident Command Structure), ESF(Emergency Support Function; 우리나라의 13개 협업 기능에 해당) 기능을 지원한다. FEMA의 표준에 따른 각종 표준 보고용 문서 등을 지원하며, 재난 현장과 EOC 사이에 임무 수행과 상황 파악을 위한 기능을 제공하고 있다.

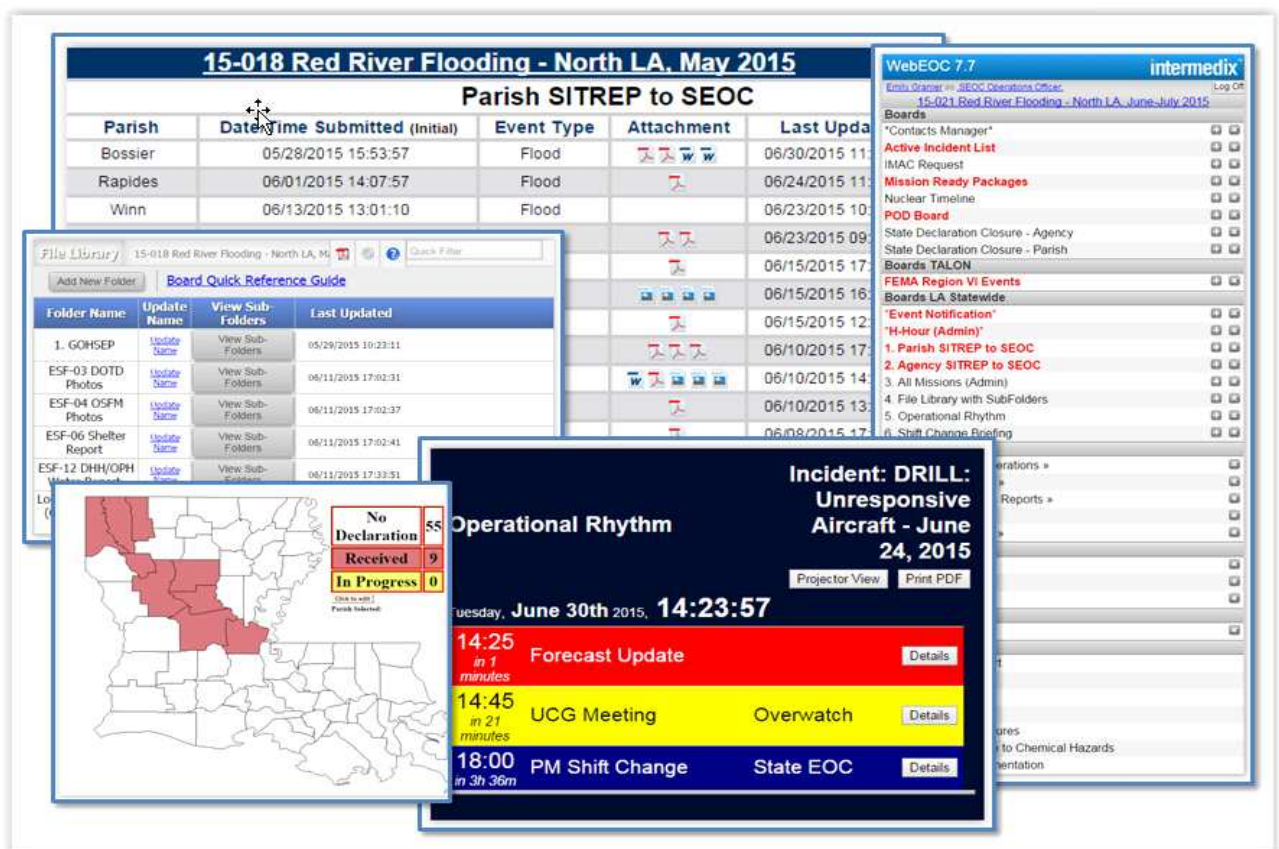


그림 82. webEOC의 Dashboard 형태의 보기

제공하는 기능은 재난 정보, 재난 대응 조직도(담당자 등록, 배정), 임무, 피난처, 사용

자, 재난안전대책본부 양식, ICS 양식, 체크리스트, 연락처, 담당자 사이 채팅방, 메시지, 지도, 기상 정보, 연계 기능(타 사이트), ESF(Emergency Support Functions) 기능(우리나라 13개 협업 기능 같은 유형), 자원 요청/배정 기능이 있다.

②UICDS

통합 지휘 및 의사 결정 지원 시스템인 UICDS(Unified Incident Command and Decision Support)는 범국가적으로 재난의 예방, 대비, 대응, 복구 전체 단계에서 통합적인 재난 관리를 위해 민간 부문과 정부의 관리 시스템간에 정보 공유와 의사 결정을 지원하는 중계 시스템으로 이 시스템을 통하여 NRF(National Response Framework), NIMS, ICS를 실현할 수 있다. 이는 여러 시스템 사이 부분적 데이터를 통하여 상황 인식 기능을 제공한 웹 기반 체계로 다음과 같이 특정 재난에 대하여 관할 지역에서 다양한 기관의 정보를 공유하여 운영하는 그림을 그림 83. 과 같이 도식적으로 보여주고 있다.

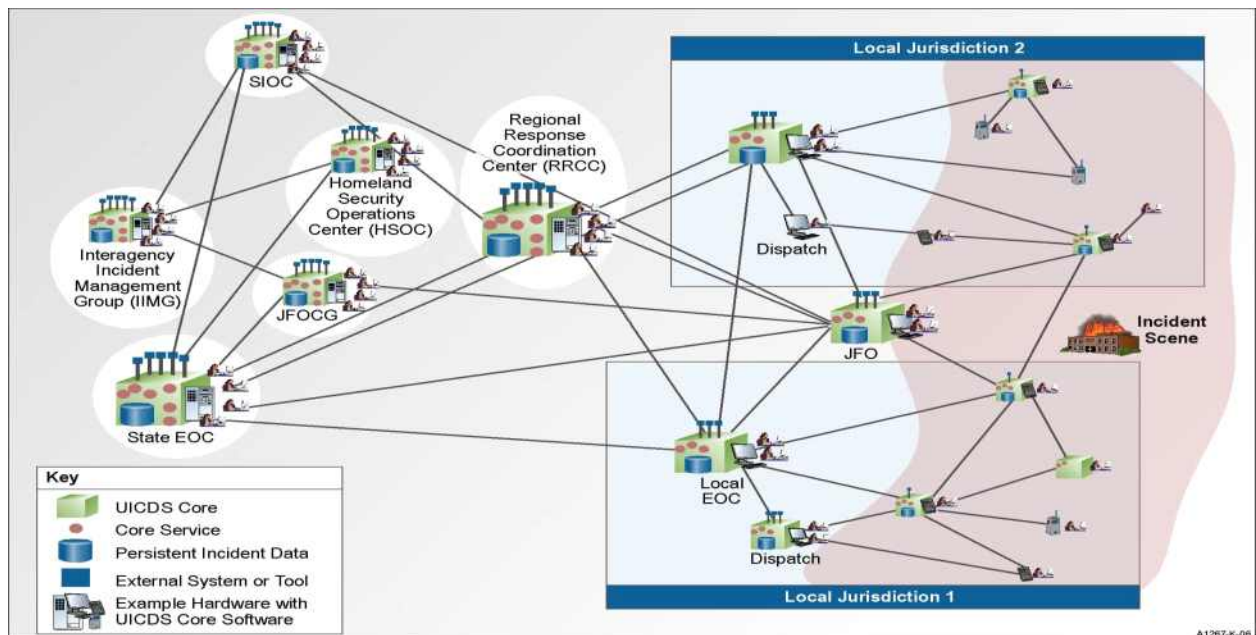


그림 83. UICDS의 상호 연계도

이는 대부분의 재난 관리가 여러 위치에서 발생되고(상황실, 현장본부 등), 여러 기관(소방서, 경찰서, 병원, 시민, 민간영역 등)이 관여하며 이들 기관이 사용하는 시스템이 자신의 목적에 맞게 만들어져 있기 때문에 필요한 재난 관련 정보를 얻기 위하여는 많은 기관의 특정 데이터(A 기관의 자원 정보, B 기관의 영상, C 기관의 센서 데이터, D 기관의 문서, E 기관의 보고서 등)를 필요로 한다. 이러한 상황에 대하여 UICDS는 필요로 하는 데이터를 기관 사이에 공유하는 방법을 제공한다.

그림 81.은 캘리포니아 통합 지진 네트워크와, California OneView Portal 시스템에서 지진 정보(CISN), 비디오 정보, 지도 정보 등을 상호 연계하여 보는 내용을 제공하고 있다.



그림 84. UICDS를 통한 지진 정보 연계 보기

그림 85. 는 Indian Hills 고등학교의 학교 안전에 관한 응용으로 모바일 앱을 통하여 학교 정보를 공유하고(1), 대피 경로 정보를 제공하며(2), 층별 설계도를 보여주고(3), 교통 안전 CCTV를 제공하며(4), 비상 계획서 문서를 제공하고(5), 비상사태 발생시 UICDS를 통하여 학생의 안전을 교사로부터 보고하게 하며(6), 감독 기관에 제공하는(7) 서비스를 보여주고 있다.

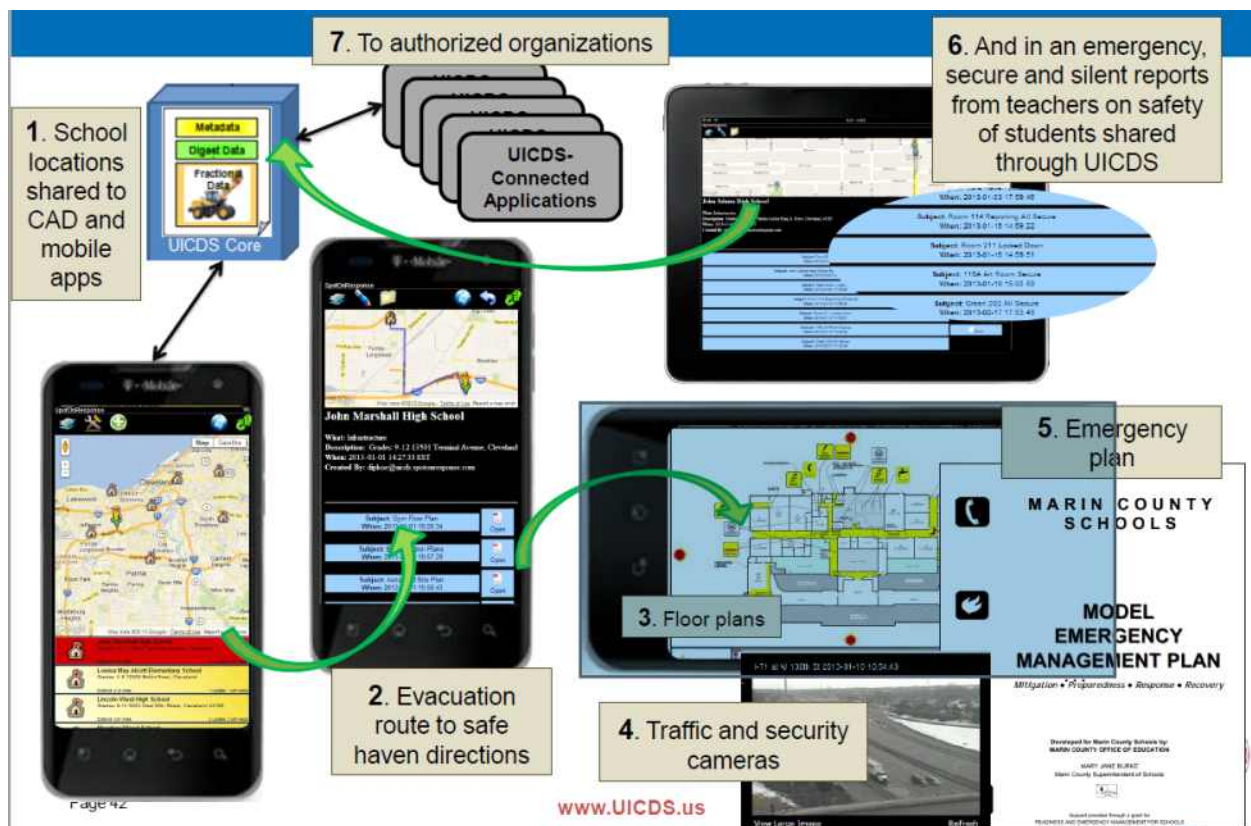


그림 85. UICDS를 통한 정보 공유 사례

(나) 국내 현황

①표준 현장조치 행동 매뉴얼

2016년 9월 행정안전부는 재난관리책임기관별로 보유하고 있는 현장조치 행동 매뉴얼의 세부 사항이 전부 상이하여 이를 운영할 수 있게 하기 위하여, “현장조치 행동매뉴얼 작성 기준 표준안”을 제시하였으며 해당 매뉴얼의 표준 체계 개념은 재난 대응체계 표준 프레임을 기반으로 13개 협업기능 기반 조직도, 4단계의 재난대응 프로세스, 그리고 43개 SOP(Standard Operation Procedure; 임무/표준행동절차) 그룹에 213개 표준 SOP를 제시하고 있다.

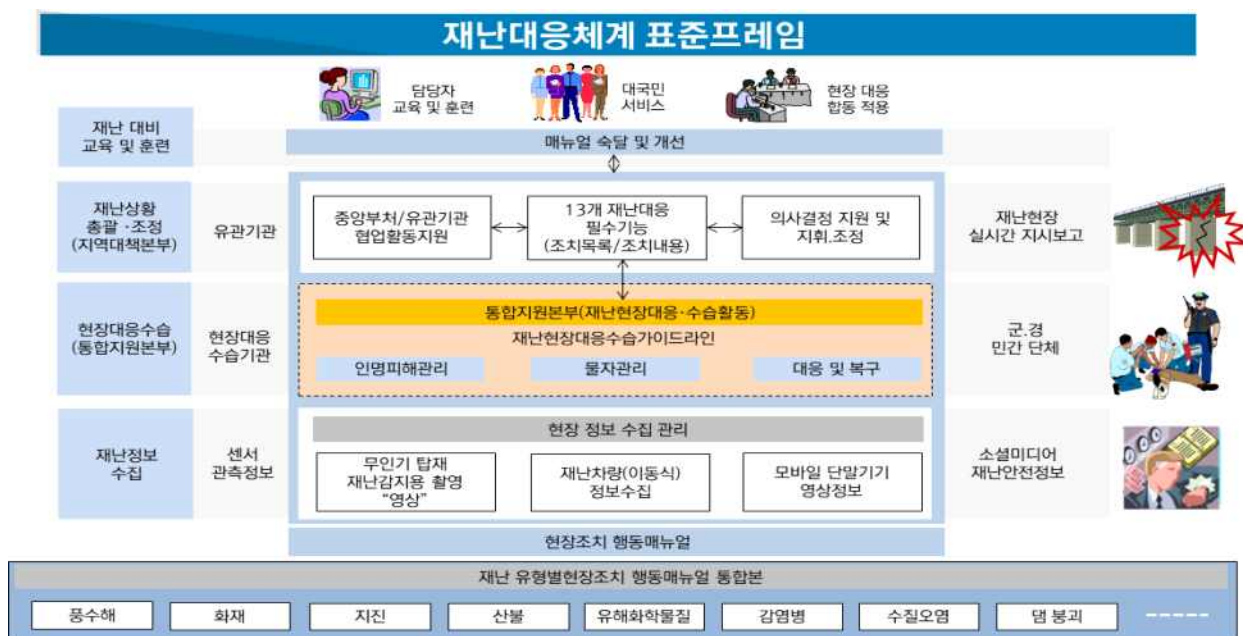


그림 86. 재난 대응 체계 표준 프레임

재난 유형에 따라 현장조치 행동매뉴얼이 만들어져야 하며, 재난 유형은 지속적으로 변화하고 있는 상태이다. 2020년 2월 기준으로 행정안전부에서 제시하는 재난의 유형은 다음 표와 같으며, 재난 유형에 따라 관리 번호를 부여하고 있다. 자연재난 13종, 사회재난 28종 그리고 주요상황 16종을 제시하고 있다.

재난관리책임기관은 표 5. 에서 정의되지 않았지만, 해당 기관에 발생할 수 있는 재난에 대하여 추가로 작성하여 운영할 수 있다.

표 5. 현장조치행동매뉴얼 관리번호

구분	번호	재난/사고의 유형	번호	재난/사고의 유형
자연 재난	01	풍수해	02	지진·지진해일
	03	대형 화산폭발	04	적조
	05	가뭄	06	조수

	07	우주전파재난	08	낙조
	09	산사태	10	낙뢰
	11	한파	12	폭염
	13	소행성·유성체 등 자연과 우주 물체의 추락·충돌		
사회 재난	01	산불	02	유해화학물질 유출사고
	03	대규모 수질오염	04	대규모 해양오염
	05	공동구 재난	06	댐 붕괴
	07	지하철 대형사고	08	고속철도 대형사고
	09	다중밀집시설 대형화재	10	인접국가 방사능 누출사고
	11	해양 선박사고	12	사업장 대규모 인적사고
	13	다중밀집건축물 붕괴 대형사고	14	교정시설 재난 및 사고
	15	가축질병	16	감염병
	17	정보통신	18	금융전산
	19	원전안전	20	전력
	21	원유수급	22	보건의료
	23	식용수	24	육상화물운송
	25	GPS전파혼신	26	우주전파재난
	27	경기장 및 공연장 안전사고	28	미세먼지
주요 상황	01	정부중요시설	02	도로터널 사고
	03	항행안전시설 장애	04	항공기 사고
	05	항공운송 마비	06	가스수급
	07	접경지역 사고	08	내수면 유도선 사고
	09	황사	10	위험물 사고
	11	저수지 붕괴	12	문화재 시설 사고
	13	학교시설 재난 및 사고	14	국방시설 재난 및 사고
	15	인공우주물체의 추락·충돌	16	식품·의약품 안전사고

②국립재난안전연구원의 SBB

국립재난안전연구원은 재난 정보를 통합하여 분석하고, 소셜네트워크 정보를 이용한 양방향 소통을 통해 재난 관리를 수행할 수 있도록 SBB(Smart Big Board)를 공개 플랫폼 기반으로 2014년에 개발하였으며, 2018년 현재 3.0 버전으로 업그레이드 하여 운영 중에 있

다.

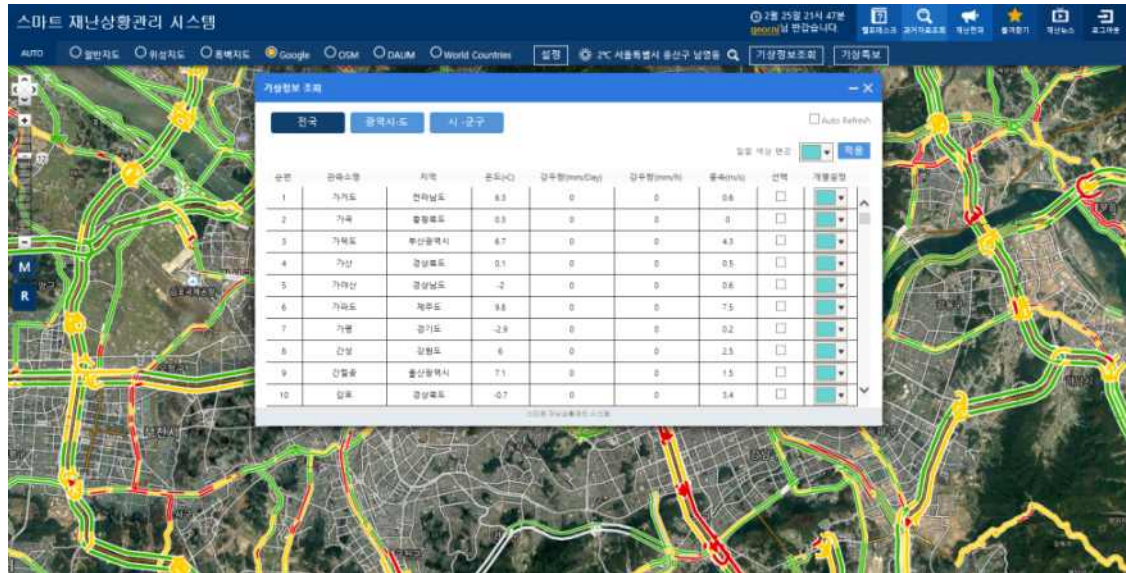


그림 87. SBB(Smart Big Board) 개념도

SBB는 재난 정보의 활용을 위한 기본 프레임워크로 다음과 같이 사용할 수 있게 기능을 제공하고 있다. 이러한 SBB를 기반으로 부산시인 경우 표 6. 과 같은 정보를 연계하여 재난 상황을 파악할 수 있는 기능으로 사용하고 있다.

표 6. SBB 기반 부산광역시에서 운영하는 재난관리시스템

재난관리시스템	운영 내용
풍향·풍속 시스템	전국 해안가의 14개소에 설치하여 풍향·풍속 정보 제공
온천천 예·경보 시스템	부산 온천천 지역 6개소에 온/습도, 수위 정보 제공, 경보 방송 장비 구축
기상관측 시스템	수위, 강우량, 풍향, 풍속 등 정보 제공
침수예측 시스템	상습적인 침수 지역 5개소의 강우량, 수위, CCTV 정보 제공
U-방재 시스템	기상청 기상 관측 정보와 수위, 배수펌프장, 댐을 연동하여 정보 제공
CCTV 통합 감시 시스템	재난 감시 CCTV 모니터링 정보 제공
풍수해 예측 시스템	기상 특보, 태풍 정보에 대한 예측 정보 제공
사면 계측 시스템	돌산공원의 경사면에 설치하여 사면 계측 정보 제공
행정안전부 CCTV	시군구의 재난 감시를 위해 125대 운영
피해현황 시스템	피해 현황에 대한 이력 정보 제공



그림 88. 부산광역시 SBB - 재난감시 CCTV 연계

- 대전광역시

대전광역시는 지령대 시스템, 통합 재난방송시스템, 통합 CCTV 관리 시스템, 제설상황관계 시스템을 연계하여 SBB를 구축하였다.

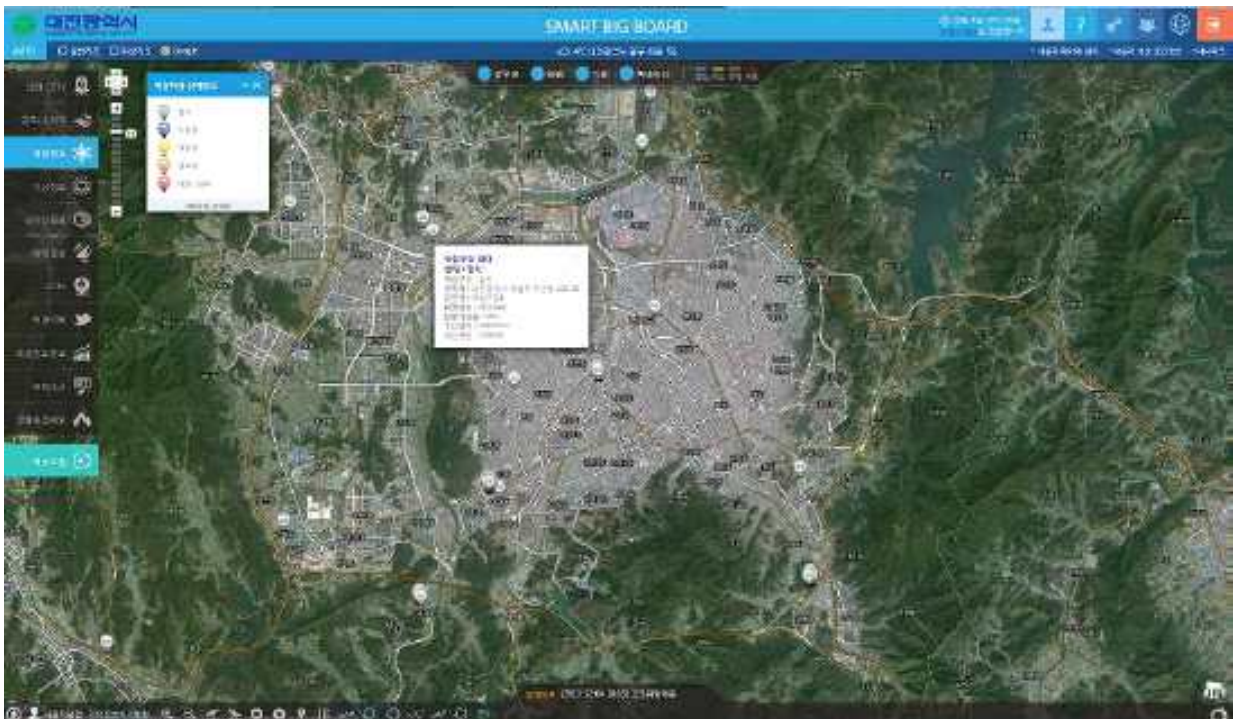


그림 89. 대전광역시 SBB - 제설 상황 정보 연계

- 울산시

울산시는 종합재난상황실 구축 계획에 따라 2017년 10월 완공을 목표로 SBB를 구축하고 있다. SBB와 연계 대상 시스템으로는 자동수위 경보 시스템, 원격 자동우량 경보 시스템, 지진/해일 경보 시스템과 CCTV 통합 감시 시스템 등이 있다.

오픈 환경에서 SBB를 효율적으로 운영하기 위해서는 기존에 표준화 되지 않고 제공되던 많은 재난 안전 데이터에 대한 표준화 작업을 통해 교환, 저장, 활용에 대한 표준체계의 구축이 필요하다. SBB에서 제시하는 데이터 표준은 표 5. 와 같다.

표 7. SBB에서 제시하는 데이터 표준

구분	형식	표준 스킴	
텍스트	Text, Json, XML	공간정보 - WKT, 좌표 - WGS83, EPSG:4326, 문자 부호화 - UTF-8	
데이터	NetCDF, Binary File		
GIS 파일	Shape, KML, ASCII		표출 심볼로지
테이블	DB, XLS		데이터 정보 Key, Type
영상정보		스트리밍 : RTMP, 비디오 : MP4, FLV	

③ 행정안전부의 통합재난안전정보체계



그림 90. 행정안전부 통합재난안전정보시스템 개념 구성도

행정안전부는 범 정부 거버넌스 체계 정립을 위하여 2013년도에 BPR/ISP(3단계)를 실시하고 기존에 구축된 정보 시스템에 대한 운영 실태 분석을 통하여 미비점을 도출하는 한편 신규 업무 영역 확대에 따른 재난 관리 전 과정에 대한 방향 설정을 하여 기존 시스템을 41개 단위 시스템과 8개 플랫폼, 1개 인프라로 재구성하였으며, 이를 기반으로 2014년부

터 2018년까지 통합재난안전정보체계를 구축하였다.

통합재난안전정보시스템 구축은 2014년부터 2018년까지 구축되었으며, 목표 시스템은 그림과 같이 크게 3개의 모듈(재난관리업무포털, 모바일 재난관리 포털, 대국민 재난안전포털(safekorea))로 구성된다.

종합적인 재난상황에 대한 정보 파악을 통해 전자지도(GIS)로 정보를 표출하여 재난 상황에 대한 의사결정을 지원하는 시스템으로 사고 속보, 기상 특보, 선박의 위치 등을 실시간으로 감시하는 기능을 제공하고 있다. 모바일을 활용한 상황 보고와 재난 현장의 상황정보에 대한 공유가 가능하며, 헬기나 지휘차의 위치 정보, CCTV 영상 및 재난 자원 정보를 기반으로 재난 위험 경보 수준과 대응 단계에 따라 자동으로 상황을 전파할 수 있는 기능을 제공하고 있다.

- 종합상황판 서비스

- . 재난지역정보 관리 : 재난지역 피해 현황 및 대응 자원 표출
- . SNS 관리 : SNS 실시간 모니터링, 분석
- . 재난상황정보 관리 : 재난 유형별, 업무별, 영역별 표출
- . 관측정보 관리 : 기상, CCTV, 댐 정보
- . 재난 연계 정보 표출 : 상황보고, 소방차 정보, 모바일 현장 정보
- . 재난 관제 정보 관리 : 현장 조사자 관제 정보, 기상 관제, 재난 표출
- . 재난 영상 정보 관리 : 회의 영상 수신, 현장 영상 추적

- 보고 지원 : 재난 상황 보고, 현장 상황 보고

- 모바일 상황 보고 : 상황 보고, 실시간 영상, 지령 관리, 재난 상황 보고

- 모바일 상황 전파 : 메시지 작성 및 관리

- 상황전파 : 메시지 작성 및 관리. 매체별(FAX, DMB, DITS, CBS) 관리

- 집계 관리 : 상황전파 메시지 집계 등

- 통합 자동 상황전파 : 연계 관리

- 훈련 모드 지원

- 데이터 연계 활용 : 국토교통부, 국립해양조사원, 미래기후변화정보시스템, 지진재해대응시스템, 함정영상정보시스템, 전자팩스시스템

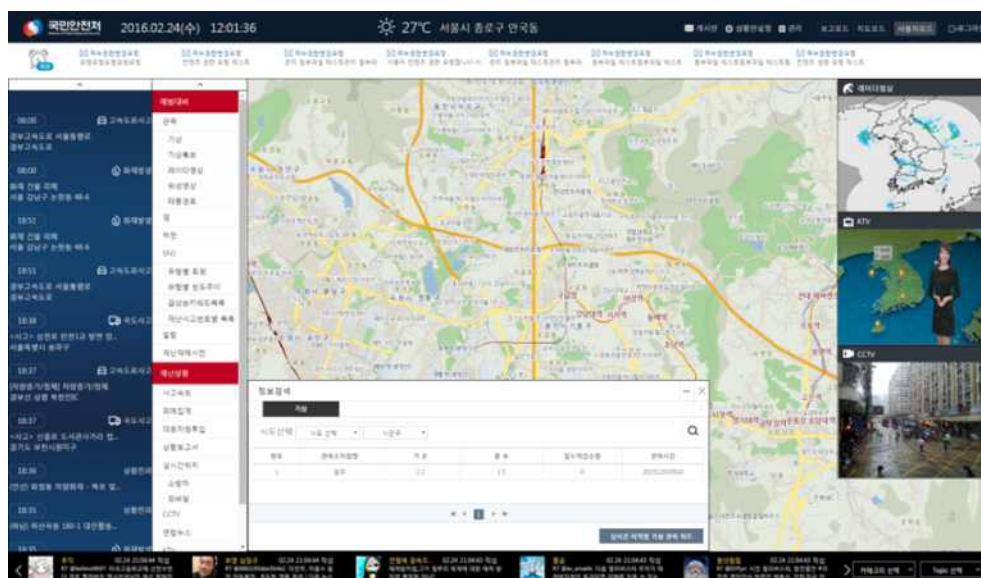


그림 91. 전자지도 기반 상황관리시스템 화면

상황관리시스템은 정보 표출과 상황전파 메시지를 보낼 수 있는 시스템으로서 재난종합상황판에 표출되는 정보는 표 8. 과 같다.

표 8. 상황관리시스템의 표출 정보 리스트

표출 정보	상세 내용
재난 지역 정보	재난 위치 전자지도(GIS) 표출, 시도별 피해 현황, 재난 대응 자원 투입 현황
재난 상황 정보	재난 유형과 업무 및 영역에 대한 재난 상황 정보
재난 관측 정보	기상특보, 기온, 강우량, 풍향, 풍속, 위성 및 레이더 영상, 태풍경로, 과거 태풍
SNS 정보	실시간 알림 감시 정보, 재난 유형별 관련 키워드 정보, 재난 사고 번호 정보
CCTV 영상 정보	행정안전부, 경찰청, 국가교통정보센터, 한국수자원공사, 홍수통제소
재난 관제 정보	기상 관제, 현장 조사자 관제 위치 설정, 재난/재해 관제 정보 표출
재난 연계 정보	상황보고서 정보, 모바일 현장 정보, 사고 지점, 속보, 소방차 정보
상황 전파 메시지	수신처 관리, 대국민통합전파, 자동 전파, 재난 위치 등록, 메시지 등록, 전송
사고 타임 정보	NDMS 재난 정보 관리, 사고 정보 관리, 상황전파 목록조회, 119 소방정보관리
댐 정보	댐 및 하천 수위, 댐 가동 현황
재난 영상 정보	현장 영상, 다자간 회의 영상
보고 지원 정보	재난상황보고 정보, 현장 집계 정보

(2) 철도안전관리체계

(가) 재난사고 유형

철도안전관리체계 기술 기준(국토부고시 제945호)

- 철도비상사태란 열차충돌, 탈선, 화재, 폭발, 자연재해 및 테러 등의 중대한 사고 발생으로 열차운행이 중단되거나 인적 및 물적 피해가 발생하는 상황 또는 위험개소(터널, 교량 등) 내 장시간 열차가 정차하는 상황을 말한다.

철도의 경우 재난(사고) 유형(7종)

- 열차충돌사고, 열차탈선사고, 열차화재사고, 위험물사고, 자연재해, 테러, 위험개소 장시간 정차

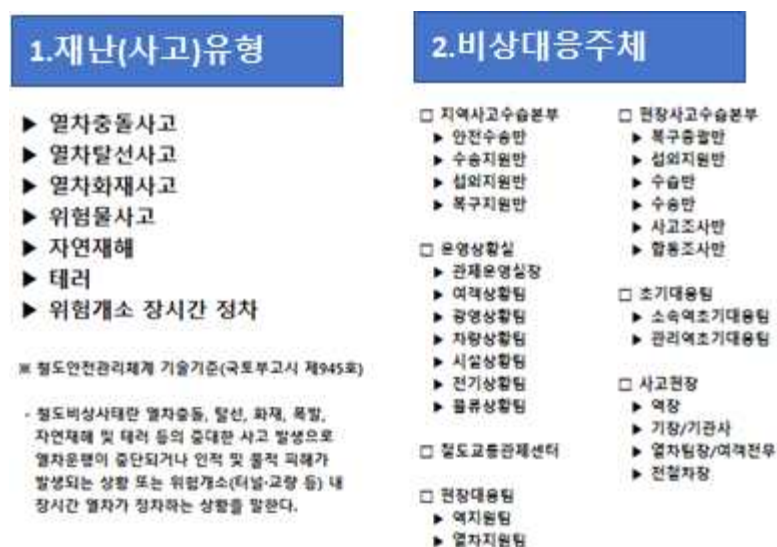


그림 92. 재난(사고)유형 및 비상대응주체

철도 재난(사고) 대응단계 : 3단계

- 1단계(10분 이내) -> 2단계(30분 이내) -> 3단계(1시간 이내)
- 4단계 재난대응단계로 보면 초기대응 -> 비상대응 -> 수습복구 단계임

(나) 현장조치매뉴얼(이매역) 분석

①국립재난안전연구원의 SBB

원인요인 : 시설불량, 휴먼에러, 테러, 풍수해, 지진

사고유형(12종, 이매역의 경우)

- 구분 : 열차 비상사태(7종), 역사 내 비상사태(5종)
- 본 과업의 대상은 역사 내 비상사태 발생시 대피훈련 및 비상관리 기술 개발임

표 9. 역사 및 열차의 비상상태 분류

열차 비상사태(7종)	역사 내 비상상태(5종)
열차충돌 및 탈선	화재 발생
열차화재 및 폭발	지진 발생
위험개소(터널내) 차량고장 정차	독가스 테러 발생
사상사고 발생	폭발물 테러 발생
위험품 누출	정전 발생
건널목 사고	-
동물 접촉	-

②사고 보고 체계



그림 93. 철도사고 보고 체계

- 당직근무자에 대한 보고계통은 통상일근시간 이외의 경우로 함
- 내부 직원에 대한 연락방법은 비상연락망에 따르고, 유관기관은 부록 '관계기관 연락

처'에 의함

- (관제센터장, 지역본부장, 역장) : 지원기관(소방서, 경찰서, 군부대 등) 의료기관, 민간장비업체, 연계수송업체 등

○ 비상연락망

구 분	철도전화	일반전화	구 분	철도전화	일반전화	
본 부 장	924-7701	02-3299-7701	경영인사처장	921-2102	02-3149-2102	
안전환경처장	924-7703	02-3299-7701	영업처장	924-7704	02-3299-7704	
담 직	924-7775	02-3299-7705	승부처장	924-7705	02-3299-7705	
관제센터	일반	923-7151	02-2027-7151	차량처장	924-7776	02-3299-7706
	고속	923-7108	02-2027-7108	시설처장	924-7707	02-3299-7707
기타	상법팀	924-7141	02-960-1661	전기처장	924-7708	02-3299-7708
	신원부	921-2210	02-3149-2206	건축사업소장	924-7710	02-3299-7710
	신원부	923-3755	02-2639-3755	철도사법경찰대	921-2213	02-3149-2213
수서관리역장	924-5211	02-2226-6881	한티역장	924-5227	02-569-9704	
압구정역내역장	924-5560	02-3298-7744	도곡역장	924-5323	02-576-2706	
강남구청역장	924-5570	02-540-7750	태평역장	924-5214	031-751-7788	
선정릉역장	924-5580	02-511-4761	모란역장	924-5215	031-721-4600	
선릉역장	924-5228	02-568-7715	아탑역장	924-5216	031-703-3238	
서현역장	924-5217	031-703-7788				

그림 94. 비상연락망 사례(이매역)

(다) 사고 유형별 조치절차(이매역)

역사 내 재난(지진) 발생시 비상관리 흐름도는 그림 95. 와 같다.

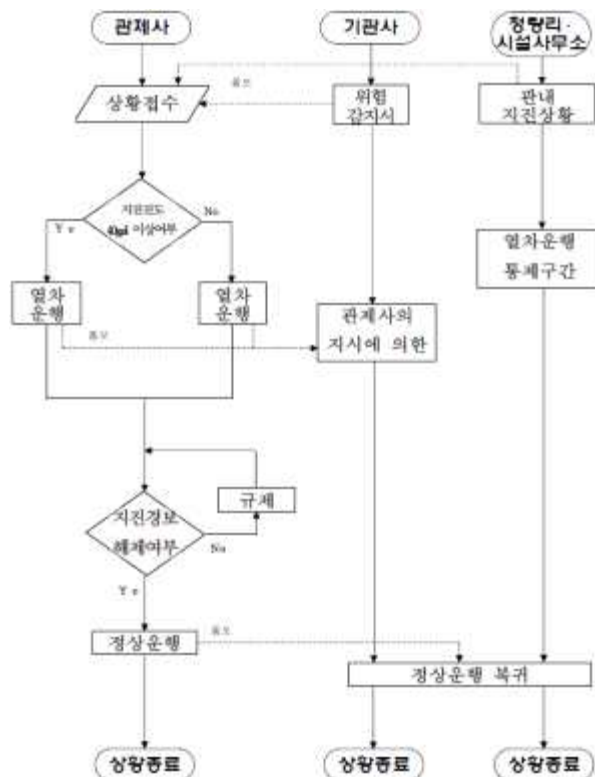


그림 95. 비상연락망 사례(이매역)

역사 내 재난 시 개인별 업무 분장은 그림 96. 과 같다.



그림 96. 재난 시 업무 분장

(3) 현장대응 관리시스템 개발

(가) 목표 시스템 개념 구성도

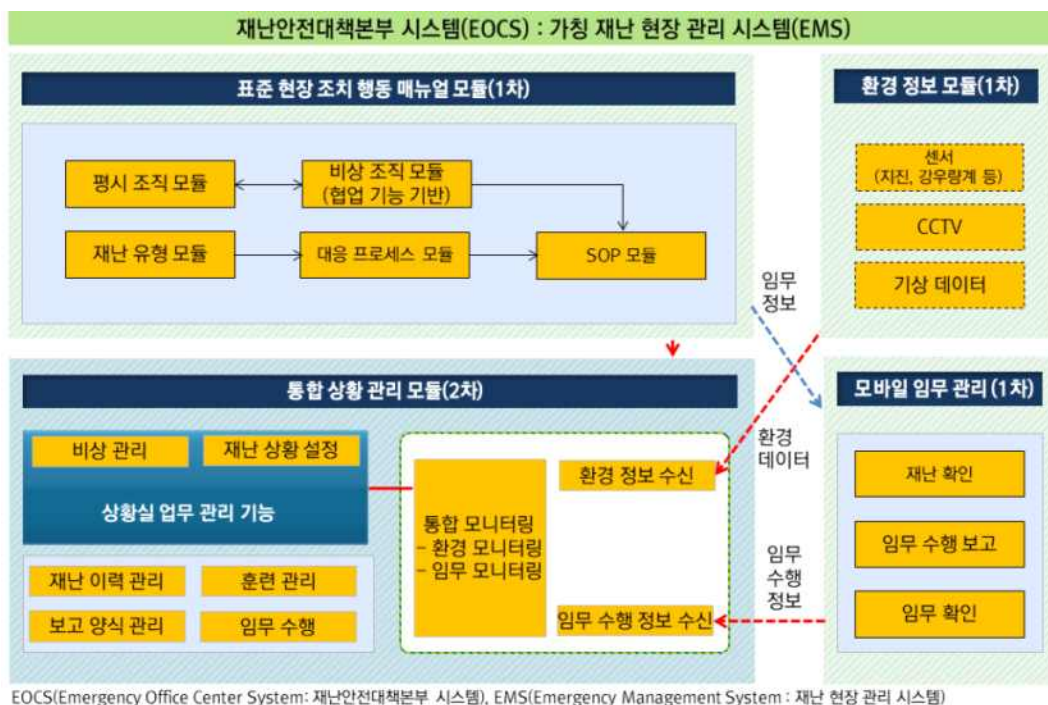


그림 97. 시스템 흐름도

목표 시스템은 다음과 같이 4개의 모듈로 구성되었다.

- 표준 현장조치 행동 매뉴얼 모듈 : 표준 매뉴얼 기반 기능 지원
- 환경 정보 모듈 : 센서 정보 제공 기능
- 모바일 임무 관리 모듈 : 현장에서 임무 수행 확인 및 보고
- 통합 상황 관리 모듈 : 통합 모니터링 및 훈련 관리
 - . 재난대응단계 및 위기경보 수준을 쉽게 인지할 수 있도록 표현
 - . 환경정보는 CCTV 현장 영상과 각 센서의 관측 정보를 차트를 통해 시각화
 - . 현장에서 보고된 이미지와 영상 정보를 통해 임무 수행 내역 정보 표시

(나) 시스템 개발 결과

①재난안전대책본부 구성

재난상황	서비스재난	재난대응단	담당부서
지역사고수습본부-안전수습		안전수습단	안전수습단
지역사고수습본부-수습지원		수습지원단	수습지원단
지역사고수습본부-설비지원		설비지원단	설비지원단
지역사고수습본부-복구지원		복구지원단	복구지원단
환경상황실-환경상황		환경상황실	환경상황실
환경상황실-지역상황반응-사고원인 파악/조사 및 대응책 마련		사고원인 파악/조사 및 대응책 마련	사고원인 파악/조사 및 대응책 마련
환경상황실-지역상황반응-지역별/지역별		지역별/지역별	지역별/지역별
환경상황실-지역상황반응-역사 내/지역별		역사 내/지역별	역사 내/지역별
환경상황실-지역상황반응-습지/습지		습지/습지	습지/습지
환경상황실-환경상황		환경상황실	환경상황실
환경상황실-지역상황반응-고급		고급지원	고급지원
환경상황실-지역상황반응-일반		일반지원	일반지원
환경상황실-지역상황반응-전통		전통지원	전통지원
환경상황실-시설		시설상황실	시설상황실
환경상황실-전기통신		전기상황실	전기상황실
환경상황실-물류		물류상황실	물류상황실
정보고통관리		정보고통관리센터	정보고통관리센터
현장대응팀-대응		대응지원	대응지원
현장대응팀-대응		대응지원	대응지원
현장수습본부-복구		복구지원단	복구지원단
현장수습본부-설비		설비지원단	설비지원단
현장수습본부-수습		수습단	수습단
현장수습본부-수습		수습단	수습단
현장수습본부-사고조사		사고조사단	사고조사단
현장수습본부-활동조사단		활동조사단	활동조사단
조기대응팀-수속/조기		조기대응팀	조기대응팀
조기대응팀-관리/조기		조기대응팀	조기대응팀
책임		책임	책임
기관/기관		기관/기관	기관/기관
협력팀(지역별)		협력팀(지역별)	협력팀(지역별)
협력팀		협력팀	협력팀

그림 98. 재난대응 조직 설정 UI

② 재난(사고) 대응단계 / 위기경보

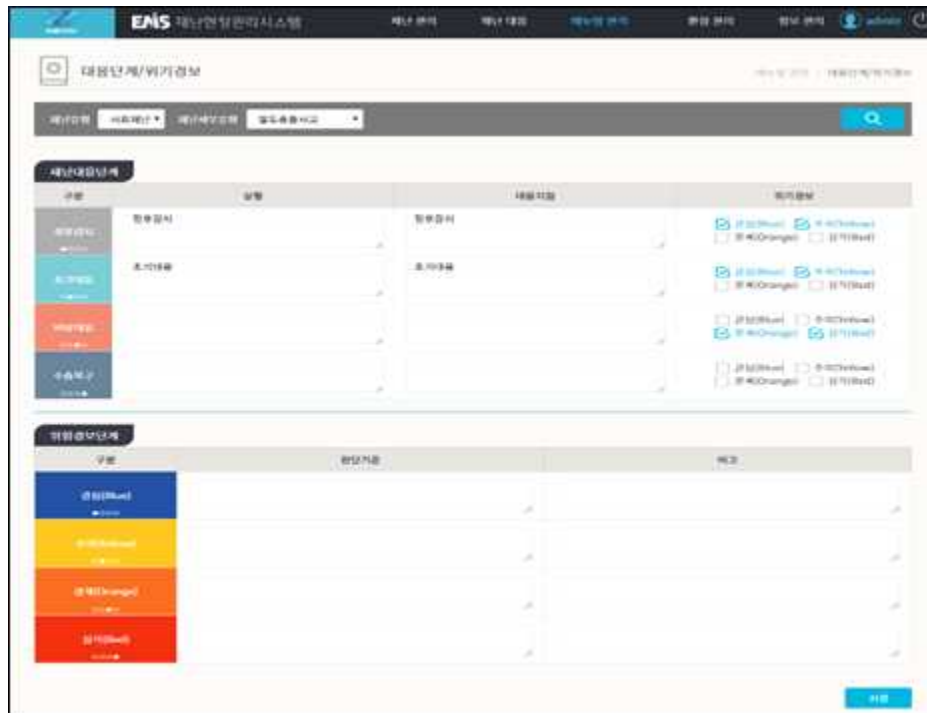


그림 99. 단계별/상황별 대응 설정 UI

③ 재난(사고)대응 프로세스 관리

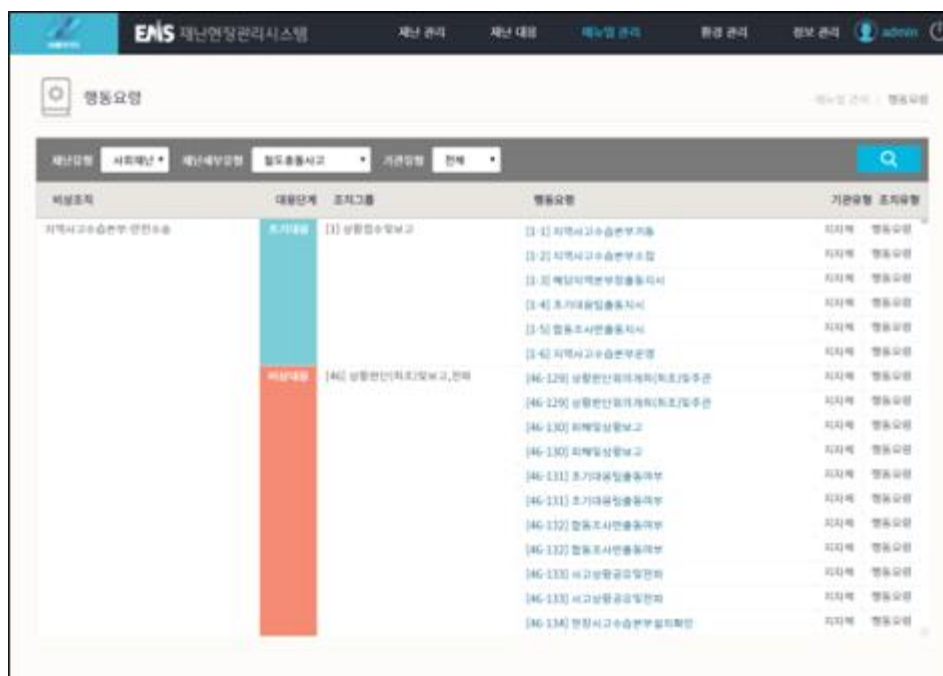


그림 100. 재난대응 프로세스 설정 UI

신기술(NET Mark)인증은 2020.01.02. ~ 03.에 있었던 3자검사를 시작으로 2020.01.13.에 있었던 2020년 1차에 접수하였고, 2020.02.14. 1차 구술 심사, 2020.05.18.~ 19. 2차 3자 인증검사, 2020.05.22. 2차 접수하여 2020.06.18. 2차 심사를 완료하였다.



시험성적서

발주처 명칭 : KRICT

발주처
주관부서 : (주)산업기술개발사업팀
과 : 연구 개발과 / 과장 : 정영준(054-709-4444-4444-4444)

내용사항 : 화학·소재·에너지 분야 산업기술개발사업 결과물 중 화학·소재 분야 고급 연구 결과물 연구

발주처명 : (주) KRICT

내용사항 : (주) KRICT 연구개발사업 결과물
내용물명 : (주) KRICT 연구개발사업 결과물

발주처 명칭 : KRICT

발주처
주관부서 : (주)산업기술개발사업팀
과 : 연구 개발과 / 과장 : 정영준(054-709-4444-4444-4444)

내용사항 : 화학·소재·에너지 분야 산업기술개발사업 결과물 중 화학·소재 분야 고급 연구 결과물 연구

발주처명 : (주) KRICT

내용사항 : (주) KRICT 연구개발사업 결과물
내용물명 : (주) KRICT 연구개발사업 결과물

발주처
명 : KRICT



발주처장
명 : 정영준

본 시험성적서는 본 시험성적서 발행일로부터 1년 동안 유효하며, 그 후의 사용은 무효입니다.

2024.01.01



대구기계부품연구원





[illegible][illegible][illegible]

그림 101. 신기술 신청서 1차, 2차

바. GS인증

소프트웨어산업진흥법 제13조에 의거한 소프트웨어의 품질 인증으로서, ISO/IEC 25023, 25041, 25051 국제표준을 기반으로 기능적합성, 성능효율성, 사용성, 신뢰성, 보안성 등에 대한 시험 수행하며, SW 유형별 테스트케이스를 개발·적용하고, 결함 발견 시 결함리포트 및 보완 계획 제공 후 회귀시험 수행하여, 제품의 품질을 종합적으로 평가한 후 인증심의위원회를 통해 인증 여부를 결정하는 테스트이다.

2020년 8월 3일에 재난현장관리시스템(EMS Lite v1.1)에 대한 1등급 인증을 최종 획득하였다.



그림 102. 소프트웨어품질인증서 (GS 인증)

사. 피난대피유도훈련

시스템이 개발되었어도, 시스템을 적용하고, 운영 상황이 주어지지 않으면 시스템은 큰 발전이 없을 것이다. 이를 위하여 본 과제에서는 20년 1월부터 8월까지 약 8개월간의 실제 시스템 운영을 통하여 피난대피 시스템의 운영에 대한 데이터를 누적하였고, 이 데이터의 누적이 일정 임계사항을 넘었다는 판단하에 위탁기관인 한국철도공사와의 협조하에 본 과제를 통하여 개발된 ‘지하공간 재난상황 인지 IoT 플랫폼 및 재난대응 지능형 긴급대피 경로 유도 플랫폼’을 이용한 피난대피 유도훈련을 계획/시행 하였다.

(1) 기획

최초 기획 의도는 한국철도공사의 연단위로 기획된 안전 훈련에 본 시스템을 적용하여 훈련을 기획하였으나, 훈련 기간중 발생한 COVID-19의 여파로 인하여 한국철도공사측의 연단위 훈련이 무기한 연기됨에 따라 최소 인원으로, 철도공사 훈련과는 별개의 훈련을 진행하게 되었으며, 시스템 기능연동 및 방재상황, 등을 고려하여 최소 인원으로 훈련이 계획되었다.

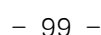
(2) 본 훈련



그림 104. 피난대피유도훈련 계획도

훈련은 2020년 8월 12일 23:00 - 명일 05:30분까지 수행하였으며, 전체적인 훈련 시나

기사 링크는 (<http://www.eachj.co.kr/news/articleView.html?idxno=451>)와 같다.



아. 특허 출원 및 등록

그림 106. 는 본 과제를 통해 특허출원을 실시하여 얻은 출원번호 통지서 이다.

관인생략 출원번호통지서 출원일자 2019.10.19 특기사항 심사청구(유) 공개신청(무) 출원번호 10-2019-0123314 (출원번호 1-1-2019-1034027-06) 출원인명칭 (주)송하이기술혁신연구소(1-2013-020942-3) 대리인성명 김현석(9-1908-000099-7) 발명자성명 임광근 박미연 나용현 박지수 발명의명칭 공기질 지능 측정을 이용한 지능화장기 내외 유동 방법	관인생략 출원번호통지서 출원일자 2019.12.17 특기사항 심사청구(유) 공개신청(무) 출원번호 10-2019-0169311 (출원번호 1-1-2019-1306992-37) 출원인명칭 (주)송하이기술혁신연구소(1-2013-020942-3) 대리인성명 김현석(9-1908-000099-7) 발명자성명 임광근 박미연 나용현 박지수 기종호 발명의명칭 영상정보 처리를 이용한 지능화장기 관련 정보 처리 방법	관인생략 출원번호통지서 출원일자 2019.12.17 특기사항 심사청구(유) 공개신청(무) 출원번호 10-2019-0169311 (출원번호 1-1-2019-1306995-74) 출원인명칭 (주)송하이기술혁신연구소(1-2013-020942-3) 대리인성명 김현석(9-1908-000099-7) 발명자성명 임광근 박미연 나용현 박지수 기종호 발명의명칭 영상정보 전송을 이용한 지능화장기 관련 정보 처리 방법
특허청장	특허청장	특허청장
<< 안내 >> 1. 귀하의 출원된 발명 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 통해 확인하실 수 있습니다. 2. 출원된 발명 수료는 접수일로부터 다름날까지 통발된 납입수수료에 성명, 납부자번호 등을 기재하여 가까운 우체국 또는 은행에 납부하여야 합니다. 3. 납부자번호: 0131(가온카드) - 접수번호 3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고려변호 정보변경(공표)], 변경 신고서를 제출하여야 하며 이후의 각종 통지서를 정상적으로 받을 수 있습니다. 4. 특허(실용신안특허)출원인 명세서 또는 도면의 보정이 필요한 경우, 등록결정 이전 또는 심사시 제출기간 이내에 출원서에 최초로 첨부된 명세서 또는 도면에 기재된 사항의 범위 안에서 보정할 수 있습니다. 5. 외국으로 출원하고자 하는 경우 PCT 제도(특허 출원신안)나 마드리드 제도(상표)를 이용할 수 있습니다. 국내출원일을 외국에서 인정받고자 하는 경우에는 국내출원일로부터 일정한 기간 내에 외국에 출원하여야 우선권을 인정받을 수 있습니다. 6. 제2도 안내 http://www.kipo.go.kr/특허정보/PCT/영문도 7. 중언변언: 직무수행과정에서 개발된 발명을 사용지(기업)가 명확하게 공개하지 않은 경우, 특허법 제83조의2에 따라 심사단계에서 특허가결정되거나 특허법 제(1)조의2에 따라 특허이후의 특허무효사유가 될 수 있습니다. 8. 기타 심사 절차에 관한 사항은 통발된 안내서를 참조하시기 바랍니다.	<< 안내 >> 1. 귀하의 출원된 발명 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 통해 확인하실 수 있습니다. 2. 출원된 발명 수료는 접수일로부터 다름날까지 통발된 납입수수료에 성명, 납부자번호 등을 기재하여 가까운 우체국 또는 은행에 납부하여야 합니다. 3. 납부자번호: 0131(가온카드) - 접수번호 3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고려변호 정보변경(공표)], 변경 신고서를 제출하여야 하며 이후의 각종 통지서를 정상적으로 받을 수 있습니다. 4. 특허(실용신안특허)출원인 명세서 또는 도면의 보정이 필요한 경우, 등록결정 이전 또는 심사시 제출기간 이내에 출원서에 최초로 첨부된 명세서 또는 도면에 기재된 사항의 범위 안에서 보정할 수 있습니다. 5. 외국으로 출원하고자 하는 경우 PCT 제도(특허 출원신안)나 마드리드 제도(상표)를 이용할 수 있습니다. 국내출원일을 외국에서 인정받고자 하는 경우에는 국내출원일로부터 일정한 기간 내에 외국에 출원하여야 우선권을 인정받을 수 있습니다. 6. 제2도 안내 http://www.kipo.go.kr/특허정보/PCT/영문도 7. 중언변언: 직무수행과정에서 개발된 발명을 사용지(기업)가 명확하게 공개하지 않은 경우, 특허법 제83조의2에 따라 심사단계에서 특허가결정되거나 특허법 제(1)조의2에 따라 특허이후의 특허무효사유가 될 수 있습니다. 8. 기타 심사 절차에 관한 사항은 통발된 안내서를 참조하시기 바랍니다.	<< 안내 >> 1. 귀하의 출원된 발명 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 통해 확인하실 수 있습니다. 2. 출원된 발명 수료는 접수일로부터 다름날까지 통발된 납입수수료에 성명, 납부자번호 등을 기재하여 가까운 우체국 또는 은행에 납부하여야 합니다. 3. 납부자번호: 0131(가온카드) - 접수번호 3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고려변호 정보변경(공표)], 변경 신고서를 제출하여야 하며 이후의 각종 통지서를 정상적으로 받을 수 있습니다. 4. 특허(실용신안특허)출원인 명세서 또는 도면의 보정이 필요한 경우, 등록결정 이전 또는 심사시 제출기간 이내에 출원서에 최초로 첨부된 명세서 또는 도면에 기재된 사항의 범위 안에서 보정할 수 있습니다. 5. 외국으로 출원하고자 하는 경우 PCT 제도(특허 출원신안)나 마드리드 제도(상표)를 이용할 수 있습니다. 국내출원일을 외국에서 인정받고자 하는 경우에는 국내출원일로부터 일정한 기간 내에 외국에 출원하여야 우선권을 인정받을 수 있습니다. 6. 제2도 안내 http://www.kipo.go.kr/특허정보/PCT/영문도 7. 중언변언: 직무수행과정에서 개발된 발명을 사용지(기업)가 명확하게 공개하지 않은 경우, 특허법 제83조의2에 따라 심사단계에서 특허가결정되거나 특허법 제(1)조의2에 따라 특허이후의 특허무효사유가 될 수 있습니다. 8. 기타 심사 절차에 관한 사항은 통발된 안내서를 참조하시기 바랍니다.
관인생략 출원번호통지서 출원일자 2019.12.17 특기사항 심사청구(유) 공개신청(무) 출원번호 10-2019-0169311 (출원번호 1-1-2019-1306997-65) 출원인명칭 (주)송하이기술혁신연구소(1-2013-020942-3) 대리인성명 김현석(9-1908-000099-7) 발명자성명 임광근 박미연 나용현 박지수 기종호 발명의명칭 공기질 지능 측정을 이용한 지능화장기 내외 유동 방법	관인생략 출원번호통지서 출원일자 2019.12.17 특기사항 심사청구(유) 공개신청(무) 출원번호 10-2019-0169311 (출원번호 1-1-2019-1306999-56) 출원인명칭 (주)송하이기술혁신연구소(1-2013-020942-3) 대리인성명 김현석(9-1908-000099-7) 발명자성명 임광근 박미연 나용현 박지수 기종호 발명의명칭 영상정보 처리를 이용한 지능화장기 관련 정보 처리 방법	관인생략 출원번호통지서 출원일자 2020.05.07 특기사항 심사청구(유) 공개신청(무) 참조번호(1) 출원번호 10-2020-0054558 (출원번호 1-1-2020-0463293-14) 출원인명칭 (주)송하이기술혁신연구소(1-2013-020942-3) 대리인성명 임광근 박미연 나용현 박지수 기종호 신정호 발명자성명 임광근 박미연 나용현 박지수 기종호 신정호 발명의명칭 Adaptive-Astar 알고리즘을 이용한 최적 경로 산출 기술
특허청장	특허청장	특허청장
<< 안내 >> 1. 귀하의 출원된 발명 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 통해 확인하실 수 있습니다. 2. 출원된 발명 수료는 접수일로부터 다름날까지 통발된 납입수수료에 성명, 납부자번호 등을 기재하여 가까운 우체국 또는 은행에 납부하여야 합니다. 3. 납부자번호: 0131(가온카드) - 접수번호 3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고려변호 정보변경(공표)], 변경 신고서를 제출하여야 하며 이후의 각종 통지서를 정상적으로 받을 수 있습니다. 4. 특허(실용신안특허)출원인 명세서 또는 도면의 보정이 필요한 경우, 등록결정 이전 또는 심사시 제출기간 이내에 출원서에 최초로 첨부된 명세서 또는 도면에 기재된 사항의 범위 안에서 보정할 수 있습니다. 5. 외국으로 출원하고자 하는 경우 PCT 제도(특허 출원신안)나 마드리드 제도(상표)를 이용할 수 있습니다. 국내출원일을 외국에서 인정받고자 하는 경우에는 국내출원일로부터 일정한 기간 내에 외국에 출원하여야 우선권을 인정받을 수 있습니다. 6. 제2도 안내 http://www.kipo.go.kr/특허정보/PCT/영문도 7. 중언변언: 직무수행과정에서 개발된 발명을 사용지(기업)가 명확하게 공개하지 않은 경우, 특허법 제83조의2에 따라 심사단계에서 특허가결정되거나 특허법 제(1)조의2에 따라 특허이후의 특허무효사유가 될 수 있습니다. 8. 기타 심사 절차에 관한 사항은 통발된 안내서를 참조하시기 바랍니다.	<< 안내 >> 1. 귀하의 출원된 발명 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 통해 확인하실 수 있습니다. 2. 출원된 발명 수료는 접수일로부터 다름날까지 통발된 납입수수료에 성명, 납부자번호 등을 기재하여 가까운 우체국 또는 은행에 납부하여야 합니다. 3. 납부자번호: 0131(가온카드) - 접수번호 3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고려변호 정보변경(공표)], 변경 신고서를 제출하여야 하며 이후의 각종 통지서를 정상적으로 받을 수 있습니다. 4. 특허(실용신안특허)출원인 명세서 또는 도면의 보정이 필요한 경우, 등록결정 이전 또는 심사시 제출기간 이내에 출원서에 최초로 첨부된 명세서 또는 도면에 기재된 사항의 범위 안에서 보정할 수 있습니다. 5. 외국으로 출원하고자 하는 경우 PCT 제도(특허 출원신안)나 마드리드 제도(상표)를 이용할 수 있습니다. 국내출원일을 외국에서 인정받고자 하는 경우에는 국내출원일로부터 일정한 기간 내에 외국에 출원하여야 우선권을 인정받을 수 있습니다. 6. 제2도 안내 http://www.kipo.go.kr/특허정보/PCT/영문도 7. 중언변언: 직무수행과정에서 개발된 발명을 사용지(기업)가 명확하게 공개하지 않은 경우, 특허법 제83조의2에 따라 심사단계에서 특허가결정되거나 특허법 제(1)조의2에 따라 특허이후의 특허무효사유가 될 수 있습니다. 8. 기타 심사 절차에 관한 사항은 통발된 안내서를 참조하시기 바랍니다.	<< 안내 >> 1. 귀하의 출원된 발명 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 통해 확인하실 수 있습니다. 2. 출원된 발명 수료는 접수일로부터 다름날까지 통발된 납입수수료에 성명, 납부자번호 등을 기재하여 가까운 우체국 또는 은행에 납부하여야 합니다. 3. 납부자번호: 0131(가온카드) - 접수번호 3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고려변호 정보변경(공표)], 변경 신고서를 제출하여야 하며 이후의 각종 통지서를 정상적으로 받을 수 있습니다. 4. 특허(실용신안특허)출원인 명세서 또는 도면의 보정이 필요한 경우, 등록결정 이전 또는 심사시 제출기간 이내에 출원서에 최초로 첨부된 명세서 또는 도면에 기재된 사항의 범위 안에서 보정할 수 있습니다. 5. 외국으로 출원하고자 하는 경우 PCT 제도(특허 출원신안)나 마드리드 제도(상표)를 이용할 수 있습니다. 국내출원일을 외국에서 인정받고자 하는 경우에는 국내출원일로부터 일정한 기간 내에 외국에 출원하여야 우선권을 인정받을 수 있습니다. 6. 제2도 안내 http://www.kipo.go.kr/특허정보/PCT/영문도 7. 중언변언: 직무수행과정에서 개발된 발명을 사용지(기업)가 명확하게 공개하지 않은 경우, 특허법 제83조의2에 따라 심사단계에서 특허가결정되거나 특허법 제(1)조의2에 따라 특허이후의 특허무효사유가 될 수 있습니다. 8. 기타 심사 절차에 관한 사항은 통발된 안내서를 참조하시기 바랍니다.



그림 106. 특허 출원번호 통지서

그림 107. 은 본 과제를 통해 특허출원을 실시하여 얻은 특허 등록증이다.



그림 107. 특허 등록증

자. 매출액

그림 108. 은 본 과제로 인한 매출 증빙이다.

전자세금계산서					승인번호	20200624-10000000-63497151					
영 증 명 서	발행 번호 (영인필)	214-86-65907		통시인증 번호		영 증 명 서	발행 번호 (영인필)	211-88-77373		통시인증 번호	
	발행 처 (영인필)	마루이비(주)		영 증 명 서			발행 처 (영인필)	스마트코(주)		영 증 명 서	이목수
	사업장 주소	서울특별시 송파구 뚝방로11길 11, 비봉 412호(문정동, 현대차(주)상업센터)			사업장 주소		서울특별시 강남구 테헤란로 112길 14				
	이메일	서비즈		유효	소프트북이개발(유)공		이메일	서비즈		유효	소프트북이개발(유)공
	이메일	me@marat.net					이메일	master@smatcoop.kr			
작성일자		금액가액		세액	합계		부가				
2020-06-24		50,000,000		5,000,000	합계합계						
일	일	금액	금액	금액	금액	단가	금액가액	세액	비고		
06	24	재난현장대응관리시스템 EMS 1.0 및 커스트마이징					50,000,000	5,000,000			
합계금액		총액		부가	대금	합계합계		미결제금 (부가 포함)			
55,000,000											

본 인쇄물은 국세청 홈택스(www.hometax.go.kr)에서 발급 또는 전송 입력된 전자(세금)계산서입니다.
발급사실 확인은 상기 홈페이지의 '조회/발급' 전자세금계산서> 제3자 발급사실 조회'를 이용하시기 바랍니다.

그림 108. 전자세금계산서 재난현장대응관리시스템 EMS 1.0 및 커스트마이징(50,000,000원, 부가세별도)

차. 소프트웨어 등록

그림 109. 은 본 과제의 소프트웨어 등록증이다.

제 C-2020-027892 호

저작권 등록증

1. 저작물의 제호(명칭)	Star_Astar - 승화 이등 경호산출
2. 저작물의 종류	컴퓨터프로그램저작물>응용프로그램
3. 저작자 성명(법인명)	주식회사 승화기술정책연구소 경기도 성남시 수정구 창영로
4. 생년월일(법인등록번호)	110111-4718677
5. 창작연월일	2020년08월14일
6. 공표연월일	2020년08월14일
7. 등록연월일	2020년08월18일
8. 등록사항	저작자 : 주식회사 승화기술정책연구소, 창작 : 2020.08.14, 공표 : 2020.08.14

『저작권법』 제53조에 따라 위와 같이 등록되었음을 증명합니다.

2020년 08월 18일

한국저작권위원회

제 C-2018-027745 호

저작권 등록증

1. 저작물의 제호(명칭)	Star(Monitoring)모니터링
2. 저작물의 종류	컴퓨터프로그램저작물>응용프로그램>산업용 S/W
3. 저작자 성명(법인명)	주식회사 승화기술정책연구소 서울특별시 서초구 백원로16길
4. 생년월일(법인등록번호)	110111-4718677
5. 창작연월일	2018년10월15일
6. 공표연월일	2018년10월15일
7. 등록연월일	2018년10월18일
8. 등록사항	저작자 : 주식회사 승화기술정책연구소, 창작 : 2018.10.15, 공표 : 2018.10.15

『저작권법』 제53조에 따라 위와 같이 등록되었음을 증명합니다.

2018년 10월 18일

한국저작권위원회

제 C-2020-013442 호

저작권 등록증

1. 저작물의 제호(명칭)	재난현장대응관리시스템 v1.1(EMS Lite v1.1)
2. 저작물의 종류	컴퓨터프로그램저작물>응용프로그램
3. 저작자 성명(법인명)	미래아이티 주식회사 서울특별시 송파구 법원로11길
4. 생년월일(법인등록번호)	110111-2041228
5. 창작연월일	2020년01월31일
6. 공표연월일	2020년01월31일
7. 등록연월일	2020년04월20일
8. 등록사항	저작자 : 미래아이티 주식회사, 창작 : 2020.01.31, 공표 : 2020.01.31

『저작권법』 제53조에 따라 위와 같이 등록되었음을 증명합니다.

2020년 04월 20일

한국저작권위원회

제 C-2020-025633 호

저작권 등록증

1. 저작물의 제호(명칭)	지하공간 밀집 예측 시스템
2. 저작물의 종류	컴퓨터프로그램저작물>응용프로그램>과학기술>시뮬레이션 S/W
3. 저작자 성명(법인명)	미래아이티 주식회사 서울특별시 송파구 법원로11길
4. 생년월일(법인등록번호)	110111-2041228
5. 창작연월일	2020년06월25일
6. 공표연월일	-
7. 등록연월일	2020년08월03일
8. 등록사항	저작자 : 미래아이티 주식회사, 창작 : 2020.06.25

『저작권법』 제53조에 따라 위와 같이 등록되었음을 증명합니다.

2020년 08월 19일

한국저작권위원회

그림 109. 소프트웨어 등록증

카. 비즈니스 모델 및 마케팅 전략 수립

그림 110. 은 본 과제의 사업화를 위한 비즈니스 모델 보고서로 보고서 작성 후 전문가의 자문 및 검증을 완료하여, 본 과제로 개발된 개발 제품의 사업화에 활용하고자 한다.

작성자	검수자	주제책임자

비즈니스 모델 수립 보고서
(지하공간 재난상황 인지 IoT 플랫폼 및 재난대응 지원형 긴급 대피 경로 유도 플랫폼 개발)

2020. 6. 30.

1. SWOT 분석(Strengths) 분석 및 전략 수립

가. SWOT 분석 : 기업이 환경 분석과 내부 능력 분석이 결과 종합하여 전략 방향 도출

환경의 기회와 내재적 강점을 활용하여 외부적 환경과 자원을 효과적으로
기업이 처한 (강점)의 상황에 대한 체계적인 분석과 기업이 갖는 능력과 전략 (강점)을 도출하는 역할

강점(S)	약점(W)
• 높은 개발 및 연구 능력 • 높은 시장 점유율 • 소비자 맞춤 제품 및 서비스 제공 • 기존 시스템 분석 가능	• 낮은 자재 생산능력 • 낮은 브랜드 파워 • 부족한 부재
기회(O)	위협(T)
• 디지털 뉴딜 • 인공지능 강화 정책 • 한국형도급사 공동연구 개발 • 시제품제작의 저비용 • 대학입학 지원	• 유동성 부재 • 낮은 브랜드 • 기술 도입기 • 경쟁업체 증가 가능성 • 자금 및 기술 부족

나. SWOT 분석을 통한 전략

구분	강점(S)	약점(W)
강점(S)	IoT 정책(국립연구개발) • 디지털 뉴딜정책에 따라, 공공기관 주요 분야에 우선적 지원 집중 확보 • 다양한 분야의 기술 정책 및 기술사업자 지원	IoT 정책(국립연구개발) • 낮은 자재 생산능력, 부품, 부품 (OEM) 생산 능력 지원 • 높은 브랜드 파워 확보를 위해 공동연구 및 대기업 납품 실적 확보
위협(T)	IoT 정책(국립연구개발) • 디지털 뉴딜에 따른 경쟁업체 증가 가능성 • 기술 고비용, 생산 비용의 증가, 추가 연구 개발을 통한 업그레이드 필요 • 높은 시장점유율 확보를 위해 '높은' 및 '기술' 지원 필요	IoT 정책(국립연구개발) • 낮은 브랜드 파워를 위해 자체 브랜드 확보, 높은 생산량 확보 필요 • 낮은 브랜드 파워 확보를 위해 다양한 납품 실적 및 높은 품질 확보

그림 110. 비즈니스 모델 보고서

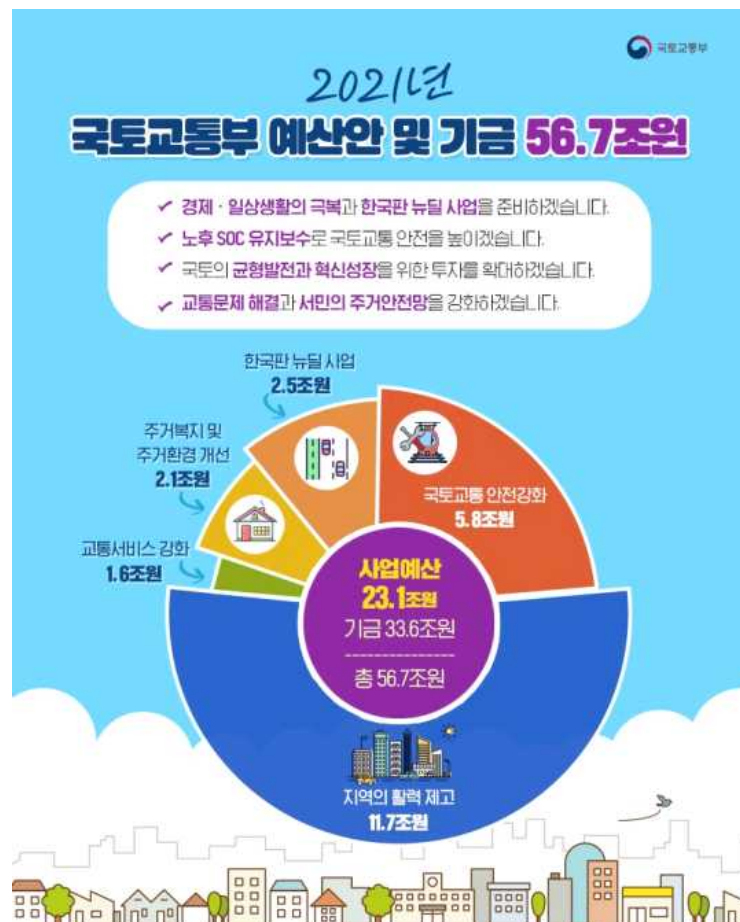
비즈니스 모델 수립 보고서의 주요 내용은 다음과 같다.

1. 외부환경 분석

가. 거시환경 분석(PEST분석) : 정치적, 경제적, 사회문화적, 기술적 요인을 통한 기회요인과 위협요인 도출

★ 거시환경 분석 체크리스트(Political, Economic, Social and Technological analysis)

환경 요인	분석내용	기회요인	위협요인
정치적, 법적 환경요인 (P)	정치이념, 정책노선, 개혁정책, 지원정책, 무역장벽 및 자유화, 조세, 특허, 저작권, 안전, 환경, 규제 또는 규제완화 등	SOC 디지털화 사업	법규 미지정
		국토교통 안전사업 강화 정책	소방설비기준
		장애인 정보 피난설비	
		설치기준 마련 권고	
경제적 환경요인 (E)	국제경기상황, 경기주기, 인플레이션 또는 디플레이 션, 이자율, 환율, 금융/재정 정책, 소비성향, 금융시장의 변화, 임금수준, 에너지 가격 등	인건비 상승	2020년 마이너스 경제성장률
		경제 규모 상승	공공기관 부채비율 상승
			ICT분야 수출/수입 전년 대비 하락
사회, 문화적 환경요인 (S)	인구구조, 라이프스타일, 소비자 행태, 여성의 사회진출, 환경보호, 에너지에 대한 관심, 교육수준, 소비문화, 인터넷 보급률, 노동조합 및 시민단체의 활동 등	무선인터넷 보급률 100%	일자리 감소
		외부활동 증가	
		높은 교육 수준	
기술적 환경요인 (T)	정보기술 및 신기술 발전, 기술혁신 및 확산, 신제품 혁신, 고객관리도구, 전자상거래 기술, 솔루션 개발 기술, 기술개발을 위한 정책적 지원 등	한국판 뉴딜	
		6세대 이동통신	
기타의 환경요인	정치적 법적 환경, 경제적 환경, 사회 문화적 환경, 기술적 환경 이외의 기타 환경요인	대단위 복합단지 계획	
		지하철 연장 계획	



< 2021년 국토교통부 예산안 >

2021년 한국 경제전망



주: 2020년과 2021년 수치는 2020년 4월 기준 필자의 전망치임.
자료: 한국은행, 한국무역협회

< 2021년 한국 경제전망 그래프-한국은행, 한국무역협회 제공 >

□ '19.10월 ICT 수출은 154.9억 달러(Δ23.3%), 수입 93.5억 달러(Δ6.8%)를 기록

- (수출) 반도체·디스플레이·휴대폰 등 3대 주력 품목 감소로 전년 동월대비 23.3% 하락해 '18.10월 이후 12개월 연속 역성장세
- 품목별로는 D-TV, 컴퓨터 및 주변기기가 증가한 반면 반도체·디스플레이·휴대폰 등 주요 수출 품목이 부진했으며 국가별로는 중국(홍콩포함)·미국·EU·베트남·일본 등 주요국이 감소

표 1-1 주요 ICT 품목별 및 국가별 수출

구분 (억 달러, %)	반도체	디스플레이	휴대폰	D-TV	컴퓨터 및 주변기기	전체
전세계	79.5 (Δ32.1)	19.2 (Δ27.0)	13.3 (Δ6.7)	2.1 (105.4)	9.4 (7.2)	154.9 (Δ23.3)
중국(홍콩)	48.9 (Δ34.9)	9.2 (Δ25.7)	4.6 (25.8)	0.1 (98.2)	3.8 (Δ13.6)	73.9 (Δ29.4)
미국	5.8 (Δ35.9)	0.2 (Δ10.2)	2.7 (Δ47.7)	0.1 (Δ24.0)	3.0 (88.4)	17.0 (Δ21.7)
일본	1.1 (Δ16.8)	0.1 (Δ25.6)	0.1 (Δ4.7)	0.1 (Δ4.3)	0.4 (23.2)	3.4 (Δ8.3)
EU	1.6 (Δ18.1)	0.5 (Δ61.3)	1.1 (146.3)	0.4 (117.4)	1.0 (Δ10.7)	9.8 (Δ9.4)
베트남	7.3 (Δ16.6)	8.4 (Δ19.9)	2.9 (26.0)	0.3 (592.2)	0.1 (Δ40.4)	22.9 (Δ11.7)

자료: IITP, KTSPI

< ICT분야 수출 자료-정보통신기획평가원 제공 >

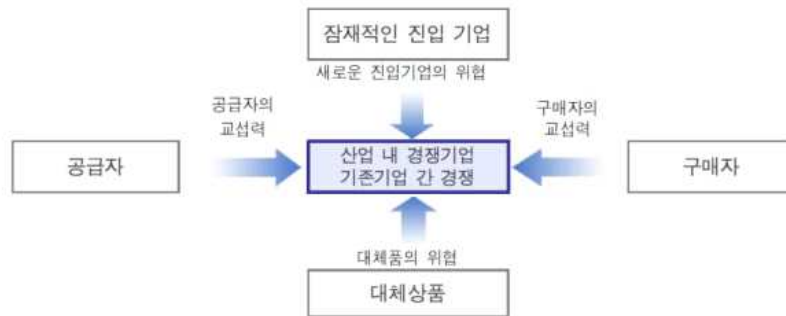
나. 산업환경 분석 : 산업동향, 시장, 기술, 수명주기, 유통채널 등의 분석을 통해 기회요인과 위협요인 도출

★ 산업환경 분석 체크리스트

환경 요인	분석내용	기회요인	위협요인
산업 트렌드	산업자동화 트렌드, 생산 및 소비 트렌드, 산업의 발전 전망, 기술동향, 유통시스템의 변화, 산업정책의 변화 동향	SOC 디지털화	
		국토교통 안전사업 강화 정책	
시장규모 및 성장성	국내 시장 및 해외시장의 규모, 시장의 성장 가능성, 공급자 및 수요자의 현황, 시장의 특성, 소비자의 소비형태 및 기호변화, 시장규모 추이 분석 등	국내 시장 증가	경쟁업체 증가
		공공기관 위주 시장	납품 조건 및 절차
기술 및 원가구조	신기술 및 기술 인프라, 제품 및 기술 수준, 요구제품 및 요구 기술수준, 기술도입 가능성, 제품 및 생산 수준, 요구 제품에 대한 원가 및 이익구조 등	국토교통 안전사업 확대	대량 생산 구조
		제품의 원가 저렴	생산 라인 신규 구축
산업의 수명주기	도입기, 성장기, 성숙기, 쇠퇴기	기존 설비 활용	
유통채널	채널의 유형, 채널의 단계, 구조의 형태, 채널의 주요 동향	도입기	도입기
		공공부문 유통 직접 납품	유통채널 미확보

다. 경쟁환경 분석 : 구매자 교섭력, 공급자 교섭력, 대체재의 위협, 새로운 경쟁자의 위협, 기존 업체들 간의 경쟁

★ 5 forces model (경쟁환경분석)



구분	분석내용	기회요인	위협요인
기존기업 간의 경쟁	산업 성장성, 제품 차별성, 고정비 또는 합몰비, 간헐적인 수요 초과, 브랜드 지명도, 전환 비용, 경쟁의 집중도와 균형, 정보의 복잡성, 경쟁자의 다양성, 후퇴 장벽 등 규모의 경제, 브랜드 지명도, 독점적 제품 차별성, 전환 비용, 소요 자본.	차별성 확보	후발주자
		분산 분석기술	지명도 부족
		영상분석 접목	
새로운 경쟁자의 진입	유통망, 정부시책, 압도적 비용 우위, 독점적 학습, 요구되는 인력 자원, 독점적 저비용 제품 설계비, 예상되는 보복 등	철도공사 공동연구	중소기업
		전환비용 발생	유통망 부족
			대기업 참여 가능성
대체상품의 위협	대체재의 출현 가능성, 전환 비용, 대체재의 성능 대비 가격, 대체재에 대한 구매자의 반응 등	현재 대체상품 없음	출현 가능성
		전환비용 발생	
구매자의 교섭력	구매력, 기업 집중도 대비, 구매자 집중도, 구매자의 수, 기업의 전환 비용 대비, 구매자의 전환 비용, 구매자의 정보, 후방 통합 능력, 대체 제품, 총 구매 가격, 브랜드 지명도, 제품 차별성 등	공공기관 위주 구매자	구매자 수 증가
		구매자 수 증가	
		구매자 맞춤 생산	
공급자의 교섭력	산업내 공급자와 기업의 전환 비용, 대체재의 존재 여부, 공급자 집중도, 공급망, 산업 내 기업의 전방통합의 위협과 관련된 후방 통합의 위협 등	대체재 없음	주문 생산
		자체 생산능력 미비	
		전환 비용 발생	

라. 소비자행동 분석

소비자의 욕구 발생 단계에서부터 실제 구매에 이르는 과정까지 외부 환경에 어떻게 영향을 받는지 분석이 필요

구분	분석내용	기회요인	위협요인
정보처리과정	산업 성장성, 제품 차별성, 고정비 또는 합몰비, 간헐적인 수요 초과, 브랜드 지명도, 전환 비용, 경쟁의 집중도와 균형, 정보의 복잡성, 경쟁자의 다양성, 후퇴 장벽 등 규모의 경제, 브랜드 지명도, 독점적 제품 차별성, 전환 비용, 소요 자본.	4차 산업	낮은 브랜드
		성장 가능성	경쟁자의 부재
구매 의사결정 과정	자본. 유통망, 정부시책, 압도적 비용 우위, 독점적 학습, 요구되는 인력 자원, 독점적 저비용 제품 설계비, 예상되는 보복 등	제품의 독점성	유통망 부재
		4차 산업	낮은 브랜드
		디지털 뉴딜	주문 제작
소비자 행동의 영향요인	대체재의 출현 가능성, 전환 비용, 대체재의 성능 대비 가격, 대체재에 대한 구매자의 반응 등	낮은 대체재	높은 대체재 출현 가능성
		전환 비용 발생	

2. 내부환경 분석

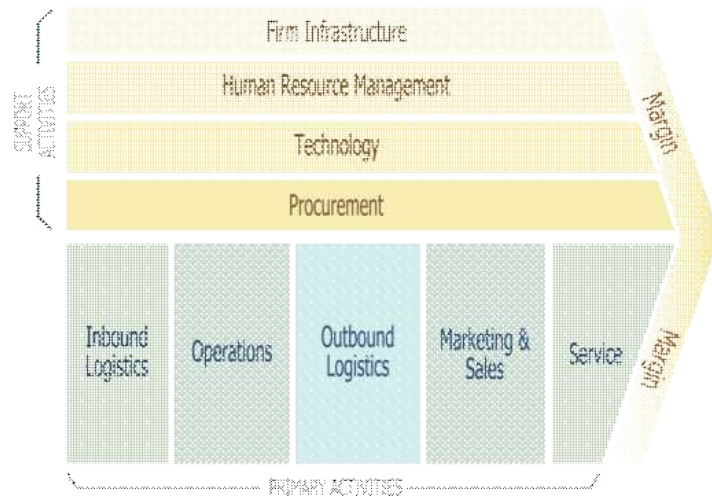
과거 성과와의 비교, 산업 진화 단계에 따른 평가, 경쟁사와의 비교, 산업 내 핵심성공요인과의 비교를 통해 자사의 강점과 약점을 진단하고 경쟁적 위상을 평가

가. 기능별 내부능력 분석

부문	핵심요소	강점	약점
경영전반	경영자의 능력, 기업이미지, 기업문화, 정보시스템 등	자유로운 기업문화	중소기업
		연구자 출신 경영자	연구자 출신 경영자
		인력 교육 투자	정보시스템 부재
R&D/기술	신제품개발, R&D투자 연구개발인력, 기술개발체제, 경험 및 노하우, 기술 차별성, 특허보유정도 등	높은 신제품개발 능력	기술개발체제 확립 부족
		높은 R&D 투자 능력	
		관련 특허 보유	
생산	공장입지, 생산성, 공정혁신, 품질, 설비의 효율성, 원자재 조달, 생산능력, 제품의 다양성 등	높은 품질 유지	낮은 생산능력
		OEM을 통한 높은 생산성	낮은 제품의 다양성
인사/조직	조직체제 및 인사정책, 인력의 질과 숙련도, 교육훈련 투자, 노사관계 등	높은 인력의 질	불완전 조직체제
		높은 교육 투자	
재무	매출 및 수익, 재무구조, 자본 조달 능력, 자금 운용 능력 등	안정적 매출 및 수익	낮은 자본 조달 능력
		안정적 자금 운용	
마케팅	브랜드, 가격, 고객, 서비스, 유통 채널, 촉진활동, 영업조직, 시장점유율	높은 시장점유율	낮은 브랜드
		높은 서비스 품질	단일 고객
글로벌화	수출비중, 해외시장기반, 국제적 전문인력		낮은 수출 비중
			국제적 전문인력 부족

나. 가치사슬분석 : 기업의 전략적 단위 활동을 통해 강점과 약점을 파악

부문	핵심요소	강점	약점
본원적활동 (주요활동)	물류투입, 생산 및 운영, 산출물, 마케팅과 판매, 서비스	산출물 직접 관리	자체 생산 미비
		맞춤 생산	마케팅 부족
		마케팅 자체 가능	
지원활동	구매조달(투입요소의 획득), 기술 및 연구개발, 인적자원관리, 기업의 전반관리(하부구조)	연구개발업	구매조달라인 부족
		적극적 인적 자원 관리	
가치사슬(Value chain) 연계	가치사슬 내의 연계 공급자나 유통경로의 가치사슬과 기업의 가치사슬의 연계 등	기술지원 서비스	생산라인 미확보
		다양한 계통 공급	
		기존 시설 연계	



< 가치사슬(Value chain) 연계 >

3. 스왓분석(SWOT) 분석 및 전략 도출

가. SWOT 분석 : 기업의 환경 분석과 내부 능력 분석의 결과 종합하여 전략 방안 도출

환경의 기회와 내부적 강점을 활용하여 외부적 환경과 기업의 약점을 최소화

기업이 처한 대내외의 상황에 대한 체계적인 분석과 기업이 선택 가능한 전략 대안들을 도출하는 역할

강점(S) • 높은 개발 및 연구 능력 • 높은 시장 점유율 • 소비자 맞춤 제작 및 서비스 제공 • 기존 시스템 연계 가능	약점(W) • 낮은 자체 생산능력 • 낮은 브랜드 평판 • 유통망 부재
기회(O) • 디지털 뉴딜 • 이동통신 강화 정책 • 한국철도공사 공동연구 개발 • 기존제품과의 차별성 • 대체상품 없음	위협(T) • 유통망 부재 • 낮은 브랜드 • 기술 도입기 • 경쟁업체 증가 가능성 • 법규 및 기준 미비

나. SWOT분석을 통한 전략

구분	강점(S)	약점(W)
기회(O)	SO 전략(우선수행과제) • 디지털 뉴딜정책에 따라 공공기관 수요 증가에 우선적 시장 점유 확보 • 다양한 분야의 맞춤 제작 및 기술서비스 제공	WO 전략(우선보완과제) • 낮은 자체 생산능력 보안을 위한 OEM 생산 체계 마련 • 낮은 브랜드 평판 확보를 위해 공공기관 및 대기업 납품 실적 확보
위협(T)	ST 전략(위험해결과제) • 디지털 뉴딜에 따른 경쟁업체 등장 가능성을 고려한 생산 표준화 및 추가 연구 개발을 통한 업그레이드 필요 • 높은 시장점유율 활용을 위한 법규 및 기준 마련 필요	WT 전략(장기보완과제) • 낮은 생산력 해결을 위해 자체 생산력 확보 또는 생산품 표준화 필요 • 낮은 브랜드 평판 해결을 위해 다양한 납품 실적 및 홍보 통로 마련

그림 111. 은 본 과제의 사업화를 위한 마케팅전략 수립 보고서로 보고서 작성 후 전문가의 자문 및 검증을 완료하여, 본 과제로 개발된 개발 제품의 사업화에 활용하고자 한다.

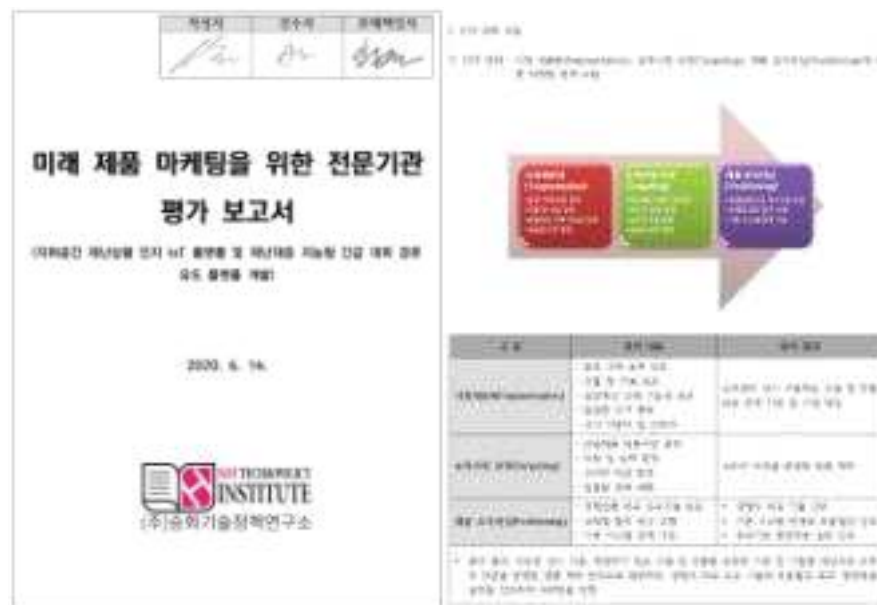


그림 111. 마케팅 전략 보고서

마케팅 전략 수립 보고서의 주요 내용은 다음과 같다.

1. STP 전략 수립

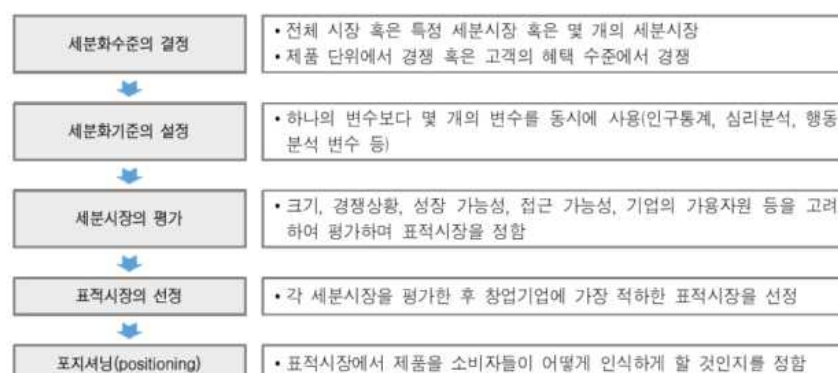
‘STP’는 세분화(segmentation), 목표시장선정(targeting), 포지셔닝(positioning)을 의미한다.

시장 세분화 : 제품에 대한 니즈, 인구통계학적, 심리학적, 지리적 등 일정한 변수에 따라 집단으로 구분하여 전체 시장을 나누는 것이다.

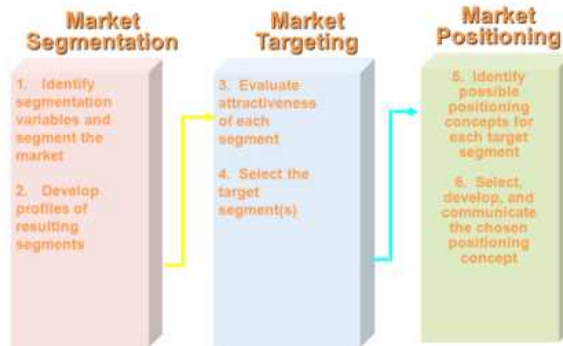
목표시장 선정 : 세분화된 각각의 시장의 매력도를 고려하여 제품에 맞는 시장을 선정하는 것이다.

포지셔닝 : 세분화된 시장을 목표로 설정하여 제품을 어떠한 이미지로 각인 시킬 것인가를 결정하는 활동이다.

★ STP 전략 진행 단계



Steps in Market Segmentation, Targeting, and Positioning

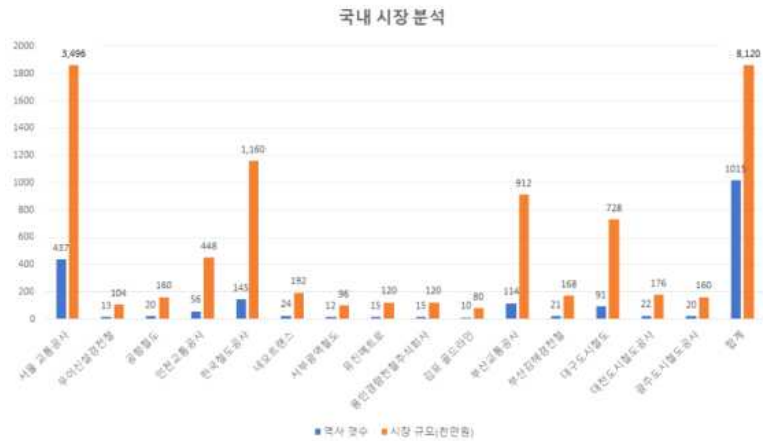


가. 시장세분화(Segmentation)

(1) 시장세분화의 요건

- 측정가능성 : 크기, 구매력, 세분시장의 특성 등 정보의 측정 및 획득이 용이
- 접근가능성 : 효과적으로 접근할 수 있고 전달이 용이
- 시장의 크기 : 충분한 수익을 얻을 수 있을 만큼의 크기
- 차별화 가능성(내부적 동질성&외부적 이질성) : 세분화된 시장에서는 동질적이며, 개별 세분시장은 이질적
- 일관성 및 지속성 : 시장의 특성에 따른 일관성과 일정 기간 지속가능성이 존재

측정가능성	• 시장 크기, 구매력에 대한 정보 획득 용이 - 공공기관
접근가능성	• 시장 접근성 용이 - 공공기관
시장의 크기	• 100 억 이상 크기의 시장 - 공공기관, 대기업
내부적 동질성 외부적 이질성	• 지하철 대상 현장 적용 - 지하철 및 지하철
일관성 및 지속성	• 시장 일관성 존재 - 공공기관, 대기업



(2) 시장 세분화 기준 · 변수

- 지리적 변수 : 국가, 지역, 도시의 크기, 인구밀도, 기후 등
- 인구통계적 변수 : 나이, 성별, 소득, 직업, 교육수준, 종교, 인종, 세대, 국적, 가족 규모 등
- 심리분석적 변수 : 사회적 계층, 생활양식, 개성 등
- 구매 행위적 변수 : 구매 또는 사용상황, 추구하는 편익, 사용량, 상표충성도, 구매자의 상태 등



나. 표적시장 선정(Targeting) : 기업의 목표, 능력과자원, 경쟁적우위 등을 고려하여 선정

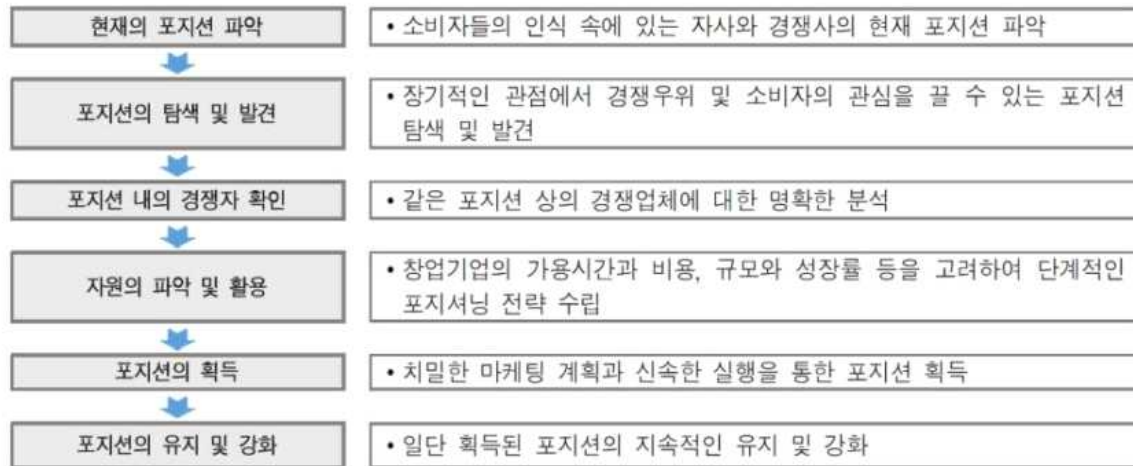
- 중소기업에 적합한 집중화 전략 선정

유형	주요 내용	장점	단점
비차별화 전략	• 세분시장의 차이를 무시하고 단일 마케팅프로그램으로 전체 시장을 공략 • 대량생산, 대량유통, 광고 등으로 비용을 최소화 • 세분화가 덜 진행된 도입기에 사용될 수 있음 • 가격경쟁과 고객 불만족 초래 가능성	• 전체시장에 단일마케팅 • 규모의 경제로 생산 및 재고관리 비용의 절감	• 특징 없는 제품의 제공 • 적용분야가 제한적임 • 경쟁사가 다양한 제품으로 시장에 진출할 경우 위험도가 높음
차별화 전략	• 세분시장마다 차별적인 마케팅프로그램을 사용 • 자원이 풍부한 기업이 사용 가능, 비용이 많이 소요 • 전체 판매량 증가, 소비자만족 증가의 이점	• 두 개 이상 시장에 동시에 진입 • 위험부담 감소 • 많은 소비자 확보로 매출액과 이익 증가	• 고 비용발생 • 자기참식 우려
집중화 전략	• 단일 제품으로 단일 세분시장을 공략 • 자원이나 능력이 한정되어 있을 때 사용가능 • 하나의 세분시장(니치마켓)에서 경쟁력을 가질 수 있음 • 세분시장 내 소비자욕구의 변화, 강력한 경쟁자 진입 등의 경우 위험분산이 되지 않을 가능성이 있음	• 창업기업에 적합 • 소비자의 욕구 파악 • 자원의 집중화 • 시장 안에서의 강력한 위치 점유	• 대기업 경쟁사 진입이 쉬움 • 시장의 불확실성으로 높은 위험의 감수

다. 제품 포지셔닝(Positioning) : 광고, 디자인 등 다양한 마케팅 수단을 통해 소비자의

인식속에 위치하기 위한 노력

(1) 포지셔닝 절차



(2) 포지셔닝의 유형

속성에 의한 포지셔닝	- 경쟁업체 비교 우수 기술 보유 - 분산 분석, 빅데이터 기반 밀집 예측 등
이미지에 의한 포지셔닝	- 한국철도공사 참여 연구 결과물 - 공공기관 납품 신뢰도 및 실적
사용상황에 의한 포지셔닝	- 재난 상황 신속한 피난 대피 능력 - 현장 적용 결과
제품사용자에 의한 포지셔닝	- 소비자 요구 수용 맞춤형 제작 - 시설 및 건물 종류에 따른 변경 가능 - 기존 재난 대응 시스템 연계 - 비용 절감
경쟁제품에 의한 포지셔닝	- 경쟁업체 비교 우수 기술 보유 - 분산 분석, 빅데이터 기반 밀집 예측 등

라. STP분석에 따른 마케팅전략 수립

- 시장 세분화(Segmentation), 표적시장 선정(Targeting), 제품 포지셔닝(Positioning)에 따른 마케팅 전략 수립



구 분	분석 내용	분석 결과
시장세분화(Segmentation)	• 높은 구매 능력 보유 • 건물 및 시설 보유 • 일관적인 구매 가능성 보유 • 밀집한 인구 분포 • 상시 이용자 및 근무자	수도권의 상시 이용하는 시설 및 건물 보유 공공 기관 및 기업 대상
표적시장 선정(Targeting)	• 단일제품 세분시장 공략 • 자원 및 능력 한정 • 소비자 의견 반영 • 집중화 전략 채택	소비자 의견을 반영한 맞춤 제작
제품 포지셔닝(Positioning)	• 경쟁상품 비교 우수기술 보유 • 코레일 참여 연구 수행 • 기존 시스템 연계 가능	• 경쟁사 비교 기술 강조 • 기존 시스템 연계로 비용절감 강조 • 공공기관 현장적용 실적 강조
• 분석 결과, 수도권 상시 이용, 작업자가 있는 시설 및 건물을 보유한 기관 및 기업을 대상으로 소비자 의견을 반영한 맞춤 제작 방식으로 제공하며, 경쟁사 비교 우수 기술과 비용절감 효과, 현장적용 실적을 강조하여 마케팅을 진행		

타. 신규 고용

그림 112. 는 본 과제를 통해 신규로 고용된 신규 인력 채용 확인서이다.

[서식 제5호]

신규 참여연구원 채용 확인서			
□ 과제현황 (단위 : 천원)			
사 업 명	국토교통기술사업화지원		
과 제 명	지하공간 대형 재난예방 기능형 긴급 대피경로 플랫폼 개발		
주관연구기관	한국과학기술정책연구원	주관연구책임자	임종권
당해연차 협약기간	2019.01.01. ~ 2020.08.31.(20개월)		
당해연차 연구개발비	979,000		
참여기업	한국과학기술정책연구원		
□ 신규 참여연구원 인력사항			
성 명	문재훈	정년인력 대상 여부	
과학기술인등록번호	1015-3701	생년월일 (월년일련한 기재)	59.11.27
소속 및 직위	기술연구소 / 소장		
담당업무 / 전공	연구개발 / 환경시스템공학		
주소 / 연락처	서울시 송파구 오금로11길 38 / 010-7433-0370		
참여기간	2020/07/01 ~ 2020/08/31	참 여 률	100%
참기인은 국토교통부에서 시행하는 2019년도 국토교통연구개발사업의 연구과제 수행을 위하여 신규로 채용하였음을 확인합니다.			
2020.07.01			
기관명 :	한국과학기술정책연구원 (인)		
대표자 :	박 미 연 (인)		
주관연구책임자 :	임 종 권 (인)		
국토교통과학기술진흥원장 귀하			
첨부 : 고용계약서(정년인력 고용의 경우 생년월일 정보 추가 또는 확인 가능 서류 추가) 사본 1부.			

[서식 제5호]

신규 참여연구원 채용 확인서			
□ 과제현황 (단위 : 천원)			
사 업 명	국토교통기술사업화지원		
과 제 명	지하공간 대형 재난예방 기능형 긴급 대피경로 플랫폼 개발		
주관연구기관	한국과학기술정책연구원	주관연구책임자	임종권
당해연차 협약기간	2019.01.01. ~ 2020.08.31.(20개월)		
당해연차 연구개발비	979,000		
참여기업	한국과학기술정책연구원		
□ 신규 참여연구원 인력사항			
성 명	이진	정년인력 대상 여부	
과학기술인등록번호	1261-7408	생년월일 (월년일련한 기재)	60.01.05
소속 및 직위	기술연구소 / 부장		
담당업무 / 전공	연구개발지원 / 환경학		
주소 / 연락처	부산시 금정구 중앙대로 1920		
참여기간	2020/07/02 ~ 2020/08/31	참 여 률	100%
참기인은 국토교통부에서 시행하는 2019년도 국토교통연구개발사업의 연구과제 수행을 위하여 신규로 채용하였음을 확인합니다.			
2020.07.02			
기관명 :	한국과학기술정책연구원 (인)		
대표자 :	박 미 연 (인)		
주관연구책임자 :	임 종 권 (인)		
국토교통과학기술진흥원장 귀하			
첨부 : 고용계약서(정년인력 고용의 경우 생년월일 정보 추가 또는 확인 가능 서류 추가) 사본 1부.			

[서식 제5호]

신규 참여연구원 채용 확인서			
□ 과제현황 (단위 : 천원)			
사 업 명	국토교통기술사업화지원		
과 제 명	지하공간 대형 재난예방 기능형 긴급 대피경로 플랫폼 개발		
주관연구기관	한국과학기술정책연구원	주관연구책임자	임종권
당해연차 협약기간	2019.01.01. ~ 2020.08.31.(20개월)		
당해연차 연구개발비	979,000		
참여기업	한국과학기술정책연구원		
□ 신규 참여연구원 인력사항			
성 명	박지민	정년인력 대상 여부	○
과학기술인등록번호	1261-7320	생년월일 (월년일련한 기재)	95.12.16
소속 및 직위	기술연구소 / 연구원		
담당업무 / 전공	연구개발지원 / 영어영문		
주소 / 연락처	경기도 군포시 송파로49번길 15 / 010-3608-2216		
참여기간	2020/07/13 ~ 2020/08/31	참 여 률	100%
참기인은 국토교통부에서 시행하는 2019년도 국토교통연구개발사업의 연구과제 수행을 위하여 신규로 채용하였음을 확인합니다.			
2020.07.13			
기관명 :	한국과학기술정책연구원 (인)		
대표자 :	박 미 연 (인)		
주관연구책임자 :	임 종 권 (인)		
국토교통과학기술진흥원장 귀하			
첨부 : 고용계약서(정년인력 고용의 경우 생년월일 정보 추가 또는 확인 가능 서류 추가) 사본 1부.			

[서식 제5호]

신규 참여연구원 채용 확인서			
□ 과제현황 (단위 : 천원)			
사 업 명	국토교통기술사업화지원사업		
과 제 명	지하공간 재난상황인지 IoT 플랫폼 및 재난대응 기능형 긴급 대피 경로 유도 플랫폼 개발		
주관연구기관	(주)송파기술정책연구원	주관연구책임자	임 종 권
제출기관	메타라이츠(주)	제출기관 연구책임자	박 길 주
당해연차 협약기간	2019. 01. 01. ~ 2020. 08. 31.		
당해연차 연구개발비	1,573,000원		
□ 신규 참여연구원 인력사항			
성 명	노재민	정년인력 대상 여부	○
과학기술인등록번호	1184-1835	생년월일 (월년일련한 기재)	90.02.07
소속 및 직위	메타라이츠(주) / 부장		
담당업무 / 전공	연구개발 / 컴퓨터공학		
주소 / 연락처	경기도 성남시 중원구 대일동 35번길 9-21 / 010-8974-9558		
참여기간	2019. 02. 01. ~ 2020. 08. 31.	참 여 률	100%
참기인은 국토교통부에서 시행하는 2019년도 국토교통연구개발사업의 연구과제 수행을 위하여 신규로 채용하였음을 확인합니다.			
2019. 02. 01.			
기관명 :	메타라이츠 (인)		
대표자 :	송 정 화		
주관연구책임자 :	박 길 주		
국토교통과학기술진흥원장 귀하			
첨부 : 고용계약서(정년인력 고용의 경우 생년월일 정보 추가 또는 확인 가능 서류 추가) 사본 1부.			

그림 112. 신규 인력 채용 확인서

3. 목표 달성도 및 관련 분야 기여도

3-1. 목표

구분	핵심연구성과 (level1)		단위 성과 (level2)		총괄 실적 성과					해당 기관	예산 (백만)	총괄 양적 목표	
					질적 성과지표	목표치	측정방법	검증방법	가중 치			목표치	가중치
총괄	A	피난대피 유도 기술	A-1	피난유도 플 랫 폼 제품화	① IoT 피난대피유 도 플랫폼 개발	피난 대피 유도 플랫폼 시제품 제작	보고서	보고서	5	주관	490.2	시제품: 1건 SW: 2건 특허: 출원 2건 등록: 1건	20
	B	지능형 밀 집예측 기 술	B-1	지하공간 밀집 예 측 모듈	② 밀집 예측 S/W 개발	밀집 예측 S/W	보고서	보고서	5	공동	258	시제품: 1건 SW: 1건 특허: 출원 1건	20
		대피훈련 및 비상관 리 기술	B-2	훈련 및 비상관리	③ 현장 대응 관리 S/W 개발	현장 대응 관리 S/ W	보고서	보고서		232	시제품: 1건 SW: 1건 특허: 출원 1건	20	
	C	사업화	C-1	전략 수립	④ 비즈니스모 델 수립	가치 제안 목표 고객 가치 사슬/조직 전달 방식 수익 흐름에 따른 비즈니스 모델 제시	보고서 작성	전문가 검증	10	주관	170	매출액: 50 백만원 신규고객: 4명	30
			C-2		⑤ 마케 팅 전략 수립	전문기관 평가	보고서 작성	전문가 검증	주관	170	NET인증: 1건 GS인증: 1건	10	
	계									20		1,320.2	특허: 출원 4 건 등록: 1 건 시제품: 3 건 SW: 4 건 매출액: 50 백만원 신규고객: 4 명 NET인증: 1 건 GS인증: 1 건

3-2. 목표 달성여부

목 표	달 성 도(%)	내 용
○ 피난유도플랫폼 제품화	100	IoT모니터링 노드 모듈 제작 피난대피 유도모듈 제작 재난상황인지 피난대피유도 솔루션 개발
○ 지능형 밀집예측 기술	100	재난사고 유형 분석 현장조치메뉴얼 분석 현장대응 관리시스템 개발
○ 훈련 및 비상관리기술	100	데이터 수집 수집데이터 분석 현장수집 데이터 가공 지하공간 밀집도 예측 모델 개발
○ 사업화 전략 수립	95	테스트 베드 구성 NET인증 GS인증
○ 양적성과	100	특허 출원 4건, 등록 1건 시제품 3건 S/W등록 4건 매출액 50백만원 신규고용 4명

3-3. 목표 미달성 시 원인(사유) 및 차후대책(후속연구의 필요성 등)

○ NET 인증

- 연구 기간내 총 2회(2020년 1회, 2회) 신청하여, 서류접수를 완료하였으며, 1회차는 2020.02.13. 1차 심사발표, 2회차는 2020.06.18. 에 1차 심사발표를 실시 하였다.

- 1회차 선정제외 사유

1회차 발표 후 선정제외 사유는 아래 표와 같다.

검토의견	- 신청 기술의 범위가 광범위하여 구체적으로 신청기술 범위의 설정이 필요함 - 신청기관에서 제시한 기술 성능의 우수성 및 신뢰성을 입증할 수 있는 구체적인 공인시험 데이터가 요구됨 - 최적 경로를 증빙할 수 있는 정량적인 평가지표와 데이터가 필요함 - 특허 등록 기술은 가스탐지를 위한 망 구성방안 이며 신청기술과는 관련성이 부족하다고 판단되어 신청 기술과 관련된 특허 내용의 제시가 요구됨 - 시험성적서의 성능지표에 IOT센서 동작여부만 제시되어 있고, 실제 핵심기술인 최적경로대피 경로 생성에 대한 인증자료가 부족함. 핵심기술인 최적경로대피 경로 생성에 대한 관련 특허 자료가 부족하여 이에 대한 보완이 필요함 - 핵심기술이 무엇인가에 대한 질의에 최적 경로 설정기술이 핵심기술이라고 답변하였으나, 이에 대한 설명이 미흡하고 객관적인 타당성과 우수성을 검증할 수 있는 정량적인 데이터가 명확하게 제시되어 있지 않음. - 또한, 제시된 시험성적서는 개발기술에 대한 동작여부만을 시험한 것으로써, 핵심기술에 대한 주요 성능시험 자료가 부족함
-------------	--

- 2회차 신청

2회차 신청 시 1회차의 검토의견을 반영하여 구체적인 신청기술 범위를 산정하고, 정량적인 평가지표와 데이터를 바탕으로 제시한 기술에 대해 KOLAS 등록 시험기관에서 공인인증을 수행하였으며, 신청기술과 관련된 특허를 추가 출원하였다. 상세 보완내용은 아래 표와 같다.

검토의견	보완내용 요약
- 신청 기술의 범위가 광범위하여 구체적으로 신청기술 범위의 설정이 필요함	- 이번 신청하는 신기술은 “재난 시 지하 공간의 최적 대피경로 산출기술”로 범위를 선정하였다.
- 신청기관에서 제시한 기술 성능의 우수성 및 신뢰성을 입증할 수 있는 구체적인 공인시험 데이터가 요구됨	- 정량적인 평가지표와 데이터를 바탕으로 제시한 기술에 대해 KOLAS 등록 시험기관에서 공인인증을 수행하였다. 신청기술과 관련된 특허를 추가 출원하였다.
- 최적 경로를 증빙할 수 있는 정량적인 평가지표와 데이터가 필요함	
- 특허 등록 기술은 가스탐지를 위한 망 구성방안이며 신청기술과는 관련성이 부족하다고 판단되어 신청 기술과 관련된 특허 내용의 제시가 요구됨	
- 시험성적서의 성능지표에 IoT센서 동작여부만 제시되어 있고, 실제 핵심기술인 최적경로대피 경로 생성에 대한 인증자료가 부족함. 핵심기술인 최적경로대피 경로 생성에 대한 관련 특허 자료가 부족하여 이에 대한 보완이 필요함	- 최적 경로 설정 기술에 대해 상세히 기술하였고 본 기술에 대해 KOLAS 등록 시험기관에서 공인인증을 수행하였다.
- 핵심기술이 무엇인가에 대한 질의에 최적 경로 설정기술이 핵심기술이라고 답변하였으나, 이에 대한 설명이 미흡하고 객관적인 타당성과 우수성을 검증할 수 있는 정량적인 데이터가 명확하게 제시되어 있지 않음. 또한, 제시된 시험성적서는 개발기술에 대한 동작여부만을 시험한 것으로서, 핵심기술에 대한 주요 성능시험 자료가 부족함	

- 2회차 선정제외 사유

2회차 발표 후 선정제외 사유는 아래 표와 같다.

검토의견	- 신기술인증은 국내에서 개발된 독창적인 기술로, 기존 기술 대비 경제적, 기술적 파급효과가 큰 기술을 대상으로 하나, 신청기술은 기존 알고리즘과 비교하여 핵심기술(긴급 대피 경로 알고리즘)에 대한 독창성과 우수성을 확인할 수 있는 데이터가 미흡하여 객관적인 판단이 어려움. 신청기술의 독창성과 우수성을 입증할 수 있는 공인기관 시험성적서 등의 정량적인 평가지표 보완이 필요함 - 신청기관은 핵심기술로 Adaptive Astar 알고리즘 기술 적용을 제시하였으나, 알려진 알고리즘을 적용하는 것만으로는 신청기술의 독창성, 우수성등을 설명하기 미흡함. 예를 들어, 지하철역이 번잡하다고 하여 대피경로가 바뀌거나 다양한 것은 아니므로 단순히 복잡한 알고리즘을 적용한다고 하여 안전성을 크게 높이는 기술이라고 보기 어려움. 또한, 해당 알고리즘의 우수성을 입증할 정량적인 데이터가 미흡하여 기존 기술과 비교해서 혁신적으로 개선되었다고 보기 어려움. 신청기술의 우수성을 입증할 수 있는 정량적인 평가지표 보완이 필요함 - 신청기관은 공인기관 시험성적서를 통해 정량적인 데이터를 제시하고 있으나, 현실에서 발생할 수 있는 복잡한 상황을 고려할 때 좀 더 포괄적인시험 데이터가 필요함. 신청 핵심 기술로 제시하고 있는 알고리즘의 경우 외국에서 같은 이름과 유사한 목적으로 발표된 알고리즘이 있어 이에 대한 비교 자료를 통해 차별성과 우수성을 입증할 수 있어야 하나, 자료가 미흡하여 객관적인 판단이 어려움. 정량적인 평가지표 보완을 통해 객관성을 확보해야 함. 또한, 화재시 설치된 센서의 작동이 알고리즘 성능에 중요한 영향을 미칠 수 있어, 기술의 안정성을 위해 이에 대한 고려가 필요함
------	--

- 차후대책(후속연구의 필요성)

- NET 인증을 위해 한국산업기술진흥협회 신기술 인증 홈페이지(<https://www.netmark.or.kr>)에서 2021년도 신기술 심사일정 공개 시 재신청 계획에 있음
- 본 연구성과를 이용하여 신기술 신청 시 연구성과와 부합하는 신기술 범위를 적용하기 위해 본 기술의 핵심인 “최적 대피 경로 산출 기술”을 이용하여 신기술을 인증 받을 계획에 있음
- NET 인증을 위해 부족한 평가지표를 완성하고 객관적 평가지표를 바탕으로 공인인증기관의 공인인증 시험을 의뢰할 예정임
- 또한, 부족한 데이터 확보를 위해 현장뿐만 아니라 회사 내부에 시험환경을 제작하여 대량의 데이터 확보 후 공인인증기관의 공인인증 시험 자료로 사용할 예정임
- 본 과제는 사업화 과제로 본 과제에서 개발된 지하 공간 재난대피 시스템을 이용하여 1개의 역사(이매역)에 시험 설치, 운영 및 테스트를 거쳐 시스템의 성능 및 안정성 검토를 완료함
- 그러나 추후 사업화하여 설치되는 역사는 각 역사의 특성 및 상태를 고려한 공간 그리드 개발이 요구됨
- 또한, 본 시스템의 활발한 사업화가 진행된다면, 각 Part의 대량생산 방법 및 대량생산에 따른 불량 검출과 최소화 방법, 대량 시공 방법 등도 문제가 될 수 있어 고려할 사항 중 하나임

4. 연구개발성과의 활용 계획 등

가. 국내외 사업화 환경변화 분석

(1) 국가연구개발사업 유사연구 검토

(단위 : 억원)

부처명	사업명	과제명	예산	비 고
소방방재청	13년도 자연재해 저감기술개발사업	지능형CCTV재난활용 및 경고·대응기술개발	3.88	
중소기업청	중소기업기술 혁신개발사업	USN기반의 지능형 환경 모니터링 및 재 난 감지 서비스	3.10	
미래창조 과학부	차세대정보·컴퓨팅기술개발	사용자 적응형 MIMO HCI를 위한 지능형 양방향 피드백 인터페이스 연구	1.65	
소방방재청	인적재난안전기술 개발사업	지하 공동구 다중 센서 기반 스마트 DCV(Demand Control Ventilation) 시스 템 개발	1.87	
중소기업청	첫걸음기술개발사업	무선인터넷을 이용한 종합재난 방재 장치 와 그 시스템	0.99	
정보통신부	IT신성장동력 기술개발	재난구조용 웨어러블 인터페이스 기술개발	13.50	
환경부	지역환경 기술개발사업	부산지역 지하철 역사 내 실시간 환경모 니터링 시스템 개발	0.40	

(2) 사업화과제의 차별화 방안

(가) 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 113. 과 같이 공간구조가 복잡할 경우 측정지점에 따라 큰 결과치를 나타내므로 대표적인 모니터링 값을 추출하기가 어려운 문제가 있다.

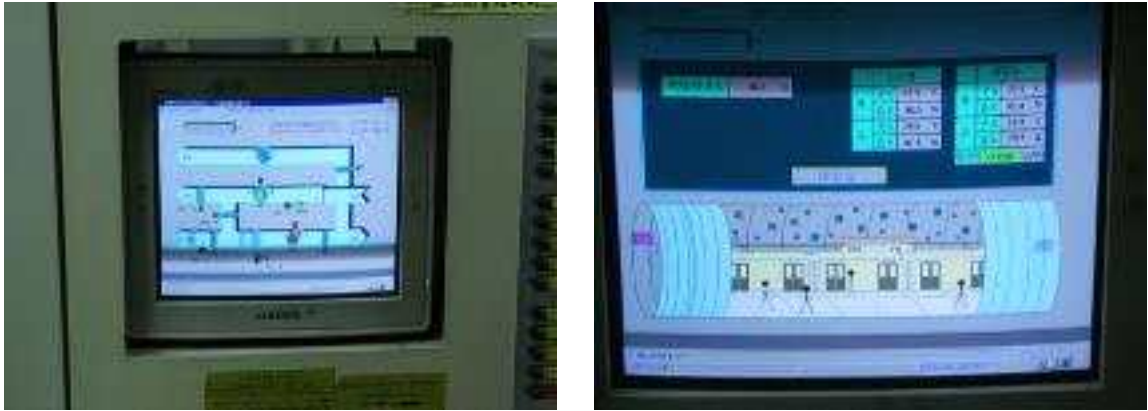


그림 113. 국내 도시철도의 방재모니터링시스템

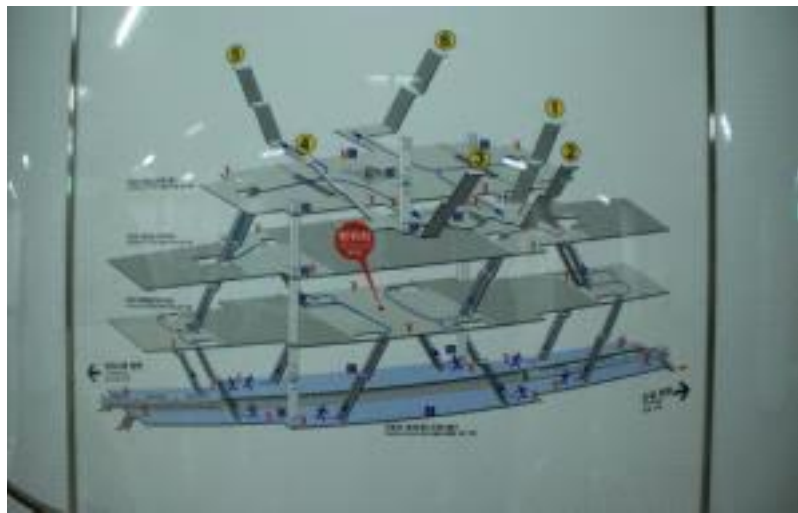


그림 114. 국내 지하철의 복잡한 지하 공간 구조

국내 지하철의 재난예방 및 대처시스템은 그림 115. 에 나타난 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 115. 화재 관련 정보 수신기

[문제점1] 유선 중심의 시스템으로 국부유선 화재감지기 센서를 설치하여 운영하므로 화재 외에 독가스살포, 지진, 생물학적 테러 등의 재난에는 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 미고려된 상황으로 기술 개발을 통한 시장성 확보가 필요한 단계이다.

(나) 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 변형여부 파악 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 수집하면, 중앙 서버에서는 데이터와 각종 파라미터를 바탕으로 재난 발생 시 대응 시나리오 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 서버에 이상이 생겼을 경우, 전체 네트워크가 마비되는 현재의 시스템적 문제를 극복해줄 수 있으며, 주요 내용을 수집하는 선단에서 불필요한 데이터의 필터링을 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 116. 평상시의 시스템 운용 예



그림 117. 재난 발생 시의 시스템 운용 예

(다) 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 118. 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 지역에 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 한다. 또한, 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 동시에 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 있을 것이라고 판단된다.

해외 선진국의 사례를 살펴보면 미국에서는 재난재해정보의 지원에 있어 재해 예방 단계에서 많은 정보를 기반으로 의사결정을 위해 정확한 정보를 제공하는 전국가적 혹은 국제적 네트워크를 구축하려는 노력을 진행해 왔으며 구축·운영되고 있는 방재관련 시스템은 지진이나 홍수 등의 재해발생을 사전에 파악하기 위한 감시 및 예·경보시스템과 재해에 의한 피해예측, 피해액 산정시스템 등을 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. 그러나 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 의사결정지원의 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 수

준에 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 초기 대응이 가장 중요한 핵심 요소라고 할 수 있는데 이에 핵심적이고 필수적인 초기대응 매뉴얼들을 시스템화 하여 적극적으로 기능화 할 수 있도록 하고 신속한 재난대응체계를 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.

(라) 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	

(마) 국내외 시장분석 및 사업화 전망

위기상황 감지 및 의사 전달 기술 개발을 위해 로봇, USN/RFID 태그, 유비쿼터스, 센싱, 메시지 송수신 장치, 노드모듈, 사용자 단말기, 디지털 사이니지 등 여러 가지 매체를 활용한 중계기기로써 역할을 하고 있으나 그 기능이 전달이나 감지정도의 수준이어서 중앙관

리시스템과의 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 재난 대응 시스템으로 사용하기에 모호하다고 판단된다.

현재 국내에 현장조치 행동매뉴얼이 시설물별로 다양하게 존재하지만 지하시설의 경우 그 시나리오가 많아 어떤 것을 취사선택해야 할지 절차가 수립되지 않아서 의사결정에 어려움을 겪고 있다. 이는 행동매뉴얼을 적시 적소에 상용할 수 있도록 시스템이 마련되어야 할 것으로 관리자뿐만 아니라 사물을 통한 초기대응 역시 필요한 것으로 판단된다. 국내외 관련 기술 특허 동향 결과를 이용하여 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 전력선기반(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 재난 관련 메시지(SMS) 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, 응급처치 요령 제공)를 한다는 점이 유사 • 차이점: 대화형 의사소통 불가능	• 데드 레코닝(Dead Reckoning) 위치 파악 가능함
5	고효율 다위처리 프로토콜을 위한 현장센서노드장치	• 무선을 사용한 중계기(허브) 간의 대화형 네트워크로 존재 • 차이점 : 센서 자체적으로 의사결정 지원이 이루어지지 않고, 센서는 존재하지만 디스플레이 및 s op가 미흡하여 통합시스템이라고 하기 어려움	• RFID(무선태그인식 기술) 업그레이드 • 가능

no.	지식재산권명	차이점	비고(활용 고려)
6	스마트형 소방 방재시스템	- 화재 탐지설비를 통하여 의사결정을 하며, 중계 기기는 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 가능한 점이 유사 - 차이점 : 중계기기 자체적으로 의사결정 지원이 불가능하여 초기대응이 어려움	

(바) 예상 시장규모

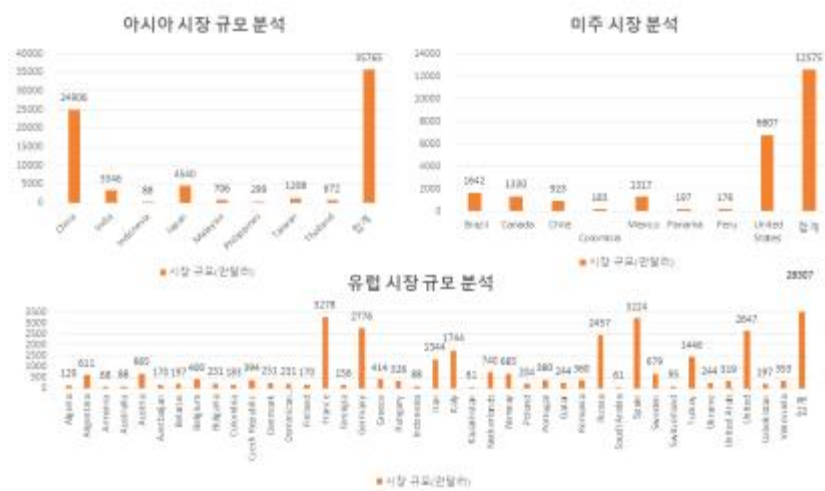
재난이라 함은 국민의 생명·신체 및 재산과 국가에 피해를 주거나 줄 수 있는 것으로 태풍, 홍수, 호우, 폭우, 폭설, 가뭄, 지진 등 자연현상으로 발생하는 재해, 화재, 붕괴, 폭발, 교통사고, 환경오염사고, 테러 등 이와 유사한 사고로 정의 한다.

본 과제의 개발 시스템은 수 많은 재난 중 지하 시설물, 다중이용시설 등에서 발생할 수 있는 화재, 테러 등의 재난을 중심으로 개발 되었으며, 추후 다른 재난에 대해서도 대응 가능한 시스템으로 확장 가능 하다.

기존 역사내에서 운영되는 화재대응시스템의 화재감지는 화재감지 센서가 화재를 감지하고 RS-485통신을 이용하여 중앙관제센터에서 화재의 유무를 인식하고 있다.

본 과제의 개발시스템은 기본적으로 RS-485통신을 지원하며, 현재 운영되고 있는 지하철 역사나 대형 건물의 화재시스템과 연동이 가능하다. 단, 지하철은 국가보안시설로 본 과제의 개발시스템으로 연동하여 테스트를 진행하지 못한 점이 있으나, RS-485는 통신 프로토콜 표준으로 현재 역사내 시스템과 연동하는데 큰 무리가 없을 것으로 예상된다.

현재의 수도권 광역전철역사는 24개통 30개 노선, 680개역사로서 현재 폐사의 예상 1개 역사당 구성 비용은 약 1.2억원 이다. 관련하여 첫해에는 약 5%, 이후 연당 2배수로 계산 하였으며, 해외의 경우 국내 매출의 약 7할로 계산하였다.



구 분	전년도	해당 연도	다음 연도
국내(억원)	20.4	40.8	81.6
해외(백만\$)	1.75	3.5	7.0

(사) 종합결론

S STRENGTHS ✓ 방재분야 전문인력 확보 및 국가연구 실적 다수 확보 ✓ 참여기업의 유사연구분야 참여 실적 보유 ✓ 구매가능기관 참여기관으로 참여	W WEAKNESSES ✓ 지하공간 재난 대응 관련 지침 전무 ✓ 국내 최초 개발에 따른 자료 확보의 어려움 발생 가능 ✓ 정부 주관기관(재난) 연계 방안 부재
O OPPORTUNITIES ✓ 현재 사회 사회 이슈분야 해결안 제시 ✓ 국내 최초 M2M 방식 방재분야 적용 사례 ✓ 미래 신성장 동력 선진기술 확보 ✓ 선진 방재 기술 확보 가능	T THREATS ✓ 독자적 지식재산권 확보의 어려움 ✓ 타기관 유사기술 개발에 따른 중복가능 내재(현재 중복과제 없음)

SO : 방재 및 IT 분야 다수 보유를 통해 현재 국내 재난에 의해 발생하는 피해를 최소화하고 미래 선진기술을 국내기술력으로 확보 한다.

ST : 재난 대응 절차 및 프로세스 확보는 어려우나, 방재 및 재난 연구에 다수 실적을 가진 인력 및 네트워크를 동원하여 재난관련 전문가 및 재난 기관에 자문을 구한다.

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

WT : 빠른 기술력 확보로 프로세스를 구현하여 시범 적용 문제점 극복

(사) 사업화 추진 상황

○ 삼성전자

- 본 시스템 관련하여 홍보를 위해 '2019 소방방재 기술 산업전'에 참석하여 관심을 보인 삼성전자의 Global EHS센터와 컨택함

- 현재 올해 10월 중순에 예정되어 있는 삼성전자 '환경안전혁신대회'에 출품 하기 위한 업무 진행 중에 있음

- 삼성전자의 '환경안전혁신대회'는 삼성 전계역사 환경안전 임원, 직원이 관람하며, 삼성 해외사업장 환경안전 임원, 직원 및 삼성 협력사 임직원이 관람하는 대회임

- 올해는 COVID-19로 인해 비대면으로 실시하며, 내년에도 출품 요청을 받음

- 현재 삼성전자 Global EHS센터 측과 함께 혁신대회 출품 및 삼성디지털시티(약 4,000평)에 도입관련하여, 견적서 발신 등의 프로세스 진행중에 있음.



○ 국토교통부, 철도공사

- 한국판 디지털 뉴딜 사업으로 본 과제를 철도공사에서 기획재정부에 제출하여 자율과제로 선정되었음

주 의

1. 이 보고서는 국토교통부에서 시행한 국토교통기술사업화지원사업의 연구보고서입니다.
2. 이 보고서 내용을 발표하는 때에는 반드시 국토교통부에서 시행한 국토교통기술사업화지원사업의 연구결과임을 밝혀야 합니다.
3. 국가과학기술 기밀 유지에 필요한 내용은 대외적으로 발표 또는 공개하여서는 안 됩니다.