성과점검기준표

(M2M기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크 시스템 개발)

구분	핵심연구성과 (level1)		단위성과물 (level2)		최종 성과점검기준									
					성과검증 기준									
					질적성과지표		목표수준		측정방법	증빙	전문기관 검토의견 (인정/불인정/일부미흡)	해당기관	가중치 (세부별)	달성평점 (평점 :1~5)	점수 (100점 만점)
							목표치	달성치							
3차년도	A	SOP기반 재난 대응 프로세스 및 재난전조 대응 M2M 알고리즘 S/W	A-1	S/W 고도화	①	논문 1편	국내저널 투고	1건/100%	게재 확인	[A-1-1]논문게재예정증명서		㈜승화기술 정책연구소	10		
					②	재난전조 및 대피경로 안내 SOP S/W 등록	저작권 등록	8월 2일 등록	저작권 등록증	[A-1-2] 저작권 등록증		㈜승화기술 정책연구소	5		0.00
	B	스마트N키퍼 개발	B-1	스마트 N키퍼 고도화	①	센서 노드 통신 데이터 수집 및 분석 모듈	재난 데이터 상황인지 가능 90점 이상		3인 이상 전문가 평가	[B-1-1] 장기계측보고서		㈜승화기술 정책연구소	10		
					②	SOP 및 시나리오 통합 대피경로 안내 모듈	90점 이상		3인 이상 전문가 평가	[증빙001] 전문가 평가 보고서		㈜승화기술 정책연구소	20		
			B-2	통합보드 시제품 개발	③	M2M 통합 N키퍼 보드 시제품 개발	특허출원 1건		특허출원	-		㈜승화기술 정책연구소	20		
					④	M2M 통합보드 대피 경로 안내용 보조패널의 방향기호 인지/시인성	90점 이상		3인 이상 전문가 평가	[증빙001] 전문가 평가 보고서		㈜승화기술 정책연구소	20		
	C	스마트N키퍼 테스트베드 운영 고도화	C-1	S/W, H/W 개발 운용	①	테스트베드 운용을 통한 시제품 테스트 : 유사재난 상황별 대응 능력 검증	시스템 연동 95% 이상		3인 이상 전문가 평가	[증빙001] 전문가 평가 보고서		㈜승화기술 정책연구소	5		
					②	M2M 웹기반 관리시스템	90점 이상		3인 이상 전문가 평가	[증빙001] 전문가 평가 보고서		㈜승화기술 정책연구소	10		
													100.0	-	0

양적 달성 정도	목 표			목표 건수	달성 건수	증빙	인정 건수	달성율 (%)	가중치(%)	점수 (100점 만점)
	사업연계지표	1	특허출원/등록	1	0	-		0	50	#DIV/0!
		2	국내논문	2	1	[B-1-1]논문게재예정증명서		0	50	0
		3	시제품	1	1					0
		4	S/W 등록	1	1	[B-1-2] 저작권 등록증				
									100.0	#DIV/0!

시스템 입력값	질적달성도	1
	양적달성도	#DIV/0!

논문 게재(예정) 증명서

제 KSDS-5-012 호

목 차	내 용
논 문 명	지하철 재난 전조 예측 모델 개발 Development of predictive models for subway disaster forecasting
논문저자	박미연[1], Mi-Yun Park, 박완순[2], Wan-Soon Park, 이정훈[3], Jeonghun Lee, 권세곤[4], Se-Gon Kwon
게재권호	한국방재안전학회논문집 Vol.10, No.2
편집인쇄	한 림 원 (주)
출판예정일	2017 년 9 월 30 일경

본 학회는 상기와 같이 학회 회원의 제출논문을 2017년 9월 30일경 논문게재 출판 예정임을 확인합니다.

2017 년 8 월 14 일

사단법인 한국방재안전학회

회장 김 재 권 (직인)

제 C-2017-018040 호

저작권 등록증

1. 저작물의 제호(명칭)	재난전조 및 대피경로 안내 SOP(표준행동요령) 프로그램
2. 저작물의 종류	컴퓨터프로그램저작물>시스템프로그램>미들웨어
3. 저작자 성명(법인명)	주식회사 승화기술정책연구소 서울특별시 서초구 매헌로16길
4. 생년월일(법인등록번호)	110111-4718677
5. 창작연월일	2017년07월14일
6. 공표연월일	2017년07월14일
7. 등록연월일	2017년08월02일
8. 등록사항	저작자 : 주식회사 승화기술정책연구소, 창작 : 2017.07.14, 공표 : 2017.07.14

『저작권법』 제53조에 따라 위와 같이 등록되었음을 증명합니다.

2017년 08월 02일

한국저작권위원회

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

재난 대응 M2M 시스템
무선 센서 노드 장기계측 보고서



M2M 기반 지하공간 (지하철) 재난대응
대화형 스마트 네트워크 시스템 개발
M2M-based Underground(subway) Disaster Response
interactive Smart Network System

목 차

제1장 센서노드 네트워크(IoT) 개요 ····· 1

제1절 재난전조 대응 M2M 네트워크의 개요 ····· 1
1. M2M 네트워크 구성 ····· 1
2. M2M 네트워크 계층도 ····· 1
2. 재난 전조 대응 N키퍼 모듈 네트워크 구성 ····· 3
2.1 개요 ····· 3
2.2 주요 서비스 내용 ····· 3
2.3 개발 환경 및 운용 환경 ····· 4
2.4. 용어집 ····· 5

제2절 재난 전조 대응 M2M 웹 어플리케이션 개요 ····· 6
1. M2M 웹 어플리케이션 구성 ····· 7
1.1. 데이터 통신 / 센서노드 관리자 웹 어플리케이션 (m2mserver.js) 구성도 ····· 7
1.2 패키지 구성 ····· 7
1.3 데이터구조 및 클래스 ····· 8
1.4 의존성 패키지 구성도 ····· 9
1.4.1 통신 모듈 (serialport) ····· 9
1.4.2 Redis 모듈 ····· 10
1.4.3. Express 미들웨어 API ····· 12
1.4.4 MQTT 서버/클라이언트 ····· 14
1.4.5 Socket.io 모듈 ····· 16

제3절 무선 센서노드의 설치위치 및 장기계측 데이터 ····· 18
1. 무선 센서노드의 설치위치 산정에 대한 근거 ····· 18
1.1 HCN : 사이안화 수소 ····· 18
1.2 센서노드의 설치간격 ····· 19
1.3 화재 확산에 대한 고려 ····· 19
2. 센서노드 시각동기화 API 개발 및 업데이트 ····· 20
3. 재난 전조 센서노드 장기계측 결과 ····· 33
3.1 장기데이터 시간이력 ····· 33
3.2 센서노드 날짜별 설치위치별 장기 계측 데이터 ····· 34
3.2.1 2017년 5월 29일 ····· 34
3.2.2 2017년 6월 7일 ····· 37
3.2.3 2017년 6월 14일 ····· 43
3.2.4 2017년 6월 23일 ····· 49
3.2.5 2017년 7월 1일 ····· 55
3.3 화재테스트 및 데이터 ····· 67
3.3.1 개포동역 테스트베드 화재 실험 ····· 67
3.3.2 화재테스트 결과 ····· 68
3.3.3 화재 실험 센서노드 (SHTPI001) 데이터 ····· 69
3.3.4 평상시 센서노드 (SHTPI013) 데이터 ····· 69
3.3.5 화재 실험 센서노드 (SHTPI001) 데이터의 평시 데이터와 비교 ····· 70

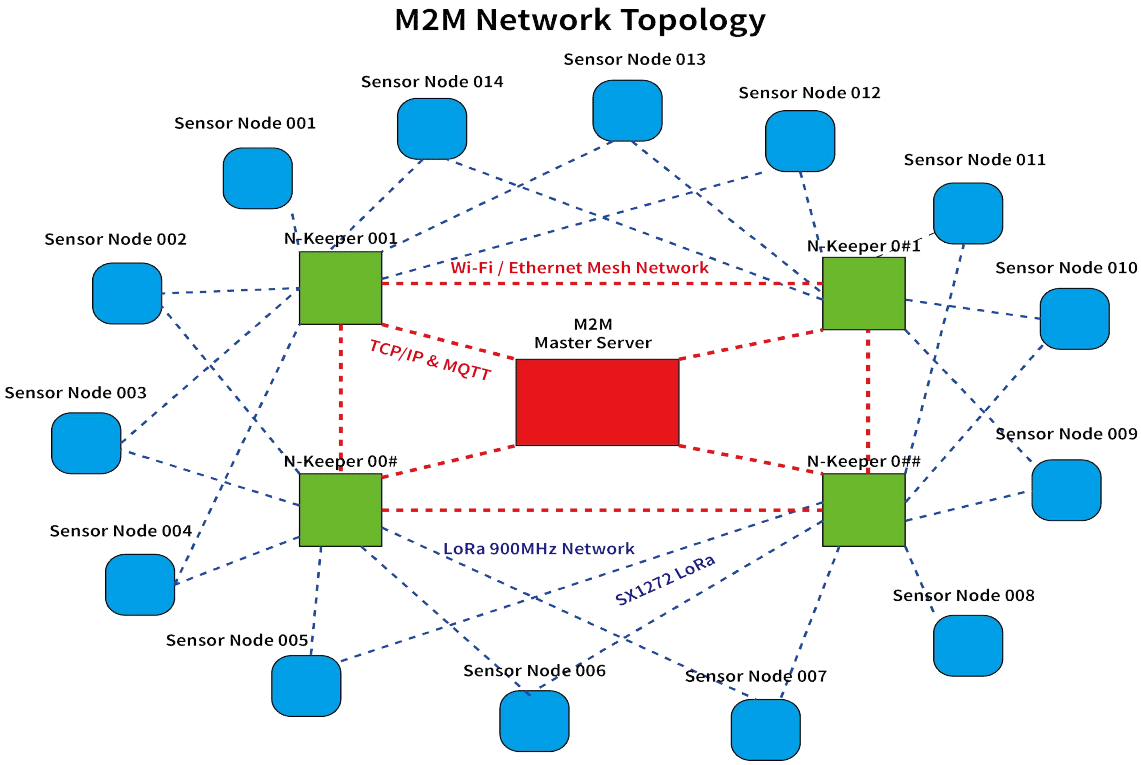
부록 A. 서버API의 데이터 통신을 통한 데이터 분석 코드 (MATLAB) ····· 71
부록 B. 기록된 RAW 데이터 분석을 위한 코드 (MATLAB) 72
부록 C. 장기 데이터 계측 API 로그 (sample) ····· 76

제1장 센서노드 네트워크(IoT) 개요

제1절 재난전조 대응 M2M 네트워크의 개요

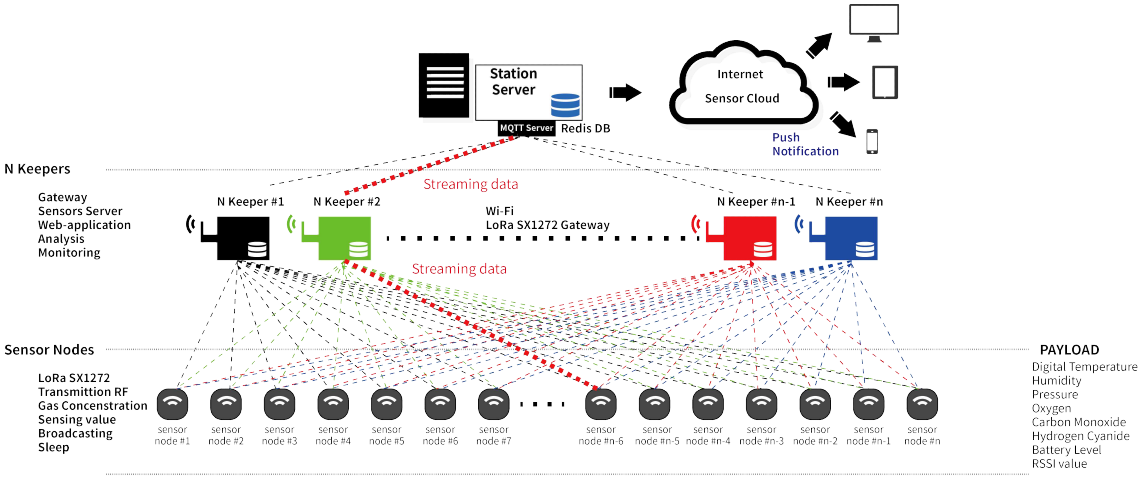
1. M2M 네트워크 구성

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



2. M2M 네트워크 계층도

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



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

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

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

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

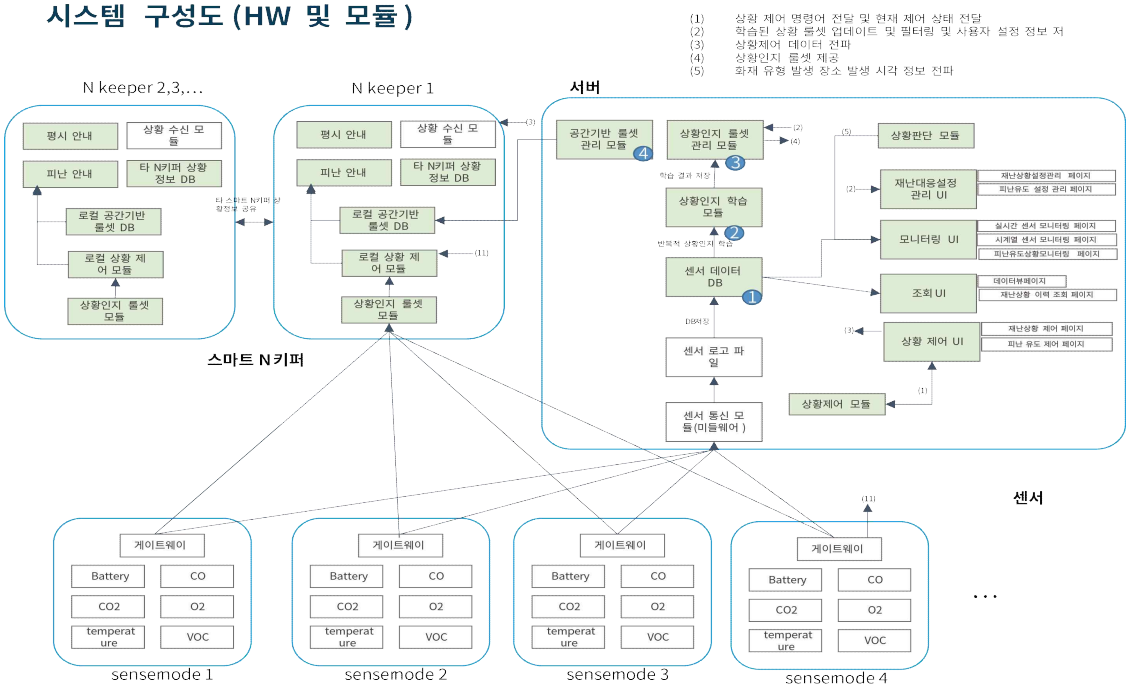
- 마스터 서버(Station server) 레이어 : N키퍼에서 MQTT로 전송한 데이터를 마스터서버의 DB에 저장한다. 최초의 데이터는 Redis에 저장되며 누적된 콜드데이터는 mongodb로 이전한다. 마스터 서버는 M2M LTE 라우터를 통해 인터넷과 연결되며, 본사 사무실 서버에 구축된 센서클라우드로 데이터를 송신할 예정이다. 이를 통해 역무실 담당자나 관리자는 마스터 서버를 통해 데이터를 모니터링 할 수 있으며, Push 알림이나 SMS등 다양한 서비스를 제공할 수 있다.

2. 재난 전조 대응 N키퍼 모듈 네트워크 구성

2.1 개요

- 시스템은 구성도에서 우상단은 마스터의 기능을 나타내며, 좌상단은 키퍼의 기능을 나타낸다.
- 복수의 스마트N키퍼는 TCP/IP로 연결되고 센서노드는 LoRa네트워크로 연결된 센서 네트워크들이 각 키퍼 및 마스터의 센서 데이터 수용모듈을 통해서 각 머신에 저장이 된다.
- 저장된 데이터는 JSON 형식으로 각 머신에서 제공이 된다.
- 키퍼 및 마스터는 각 SW가 설치된 머신의 로컬호스트를 통해서 JSON형식으로 센싱 데이터를 제공받아 자체DB에 저장을 하며 사용을 한다.
- 키퍼 및 마스터는 Ajax로 통신을 하며 필요한 정보를 교환한다.

시스템 구성도 (HW 및 모듈)



2.2 주요 서비스 내용

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

1) 마스터

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

- 상황인지 예측 결과가 이상이나 재난을 나타내게 되면 시스템은 관리자에게 이 정보를 표출하고 관리자의 대응을 요청한다. 관리자는 일정한 시간 이내에 시스템의 요청에 따라서 상황을 인지한 후 정상, 이상, 재난의3가지 상태를 선택하게 된다.
- 정상으로 판단된 경우, 시스템은 정상모드로 돌아가며, 이전 상황을 로그에 기록하고, 이 상황을 룰셋으로 저장한 후 동일한 조건에서 상황을 발생시키지 않도록 한다.
- 이상이나 재난으로 관리자가 판단한 경우, 시스템은 이상 및 재난 상황발생을 수행하며 SOP에 의해서 이후 재난 대응 프로세스를 진행한다.
- 이상 및 재난 상황 종료 후 시스템은 해당 정보를 로그로 기록하여 추후 분석에 사용할 수 있도록 한다.

2) 키퍼

- 키퍼는 평상 시에는 센서 미들웨어의 정보를 실시간으로 모니터링하면서 상황인지를 수행하고, 이상 및 재난 발생 판단 시 서버와 연동하여 상황인지의 정확성을 검토한다.
- 서버에서 동일한 상황인지를 한 경우, 서버의 다음 대응을 기다리며, 서버에서 응답이 없는 경우(연결이 되지 않는 경우)에 일정 시간을 대기한 후에 자율적으로 상황전파 및 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.

2.3. 개발 환경 및 운용환경

1) 개발환경

- 모든 개발은 윈도우 7 및 우분투 16.04의 환경에서 파이썬 3.5.3 및 장고 1.6.4의 환경에서 개발되었다.
- 서비스는 웹으로 표출되며, 프론트엔드(웹페이지)는 우분투 16.04 환경의 파이어폭스을 대상으로 개발하였다.

2) 운용환경

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

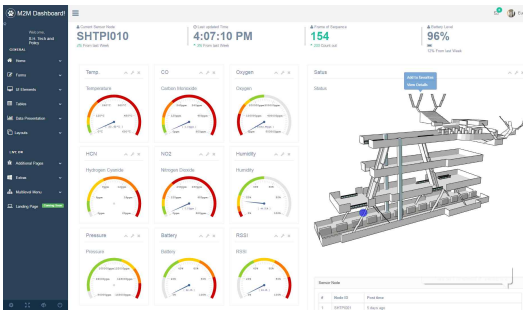
2.4. 용어집

- 본 시스템 및 보고서에서 사용하는 주요 용어는 다음과 같다.

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

제2절 재난 전조 대응 M2M 웹 어플리케이션 개요

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



(a) 스트리밍 게이지 차트 / 위치 알림



(b) 센서노드별 단기 시간이력 조회



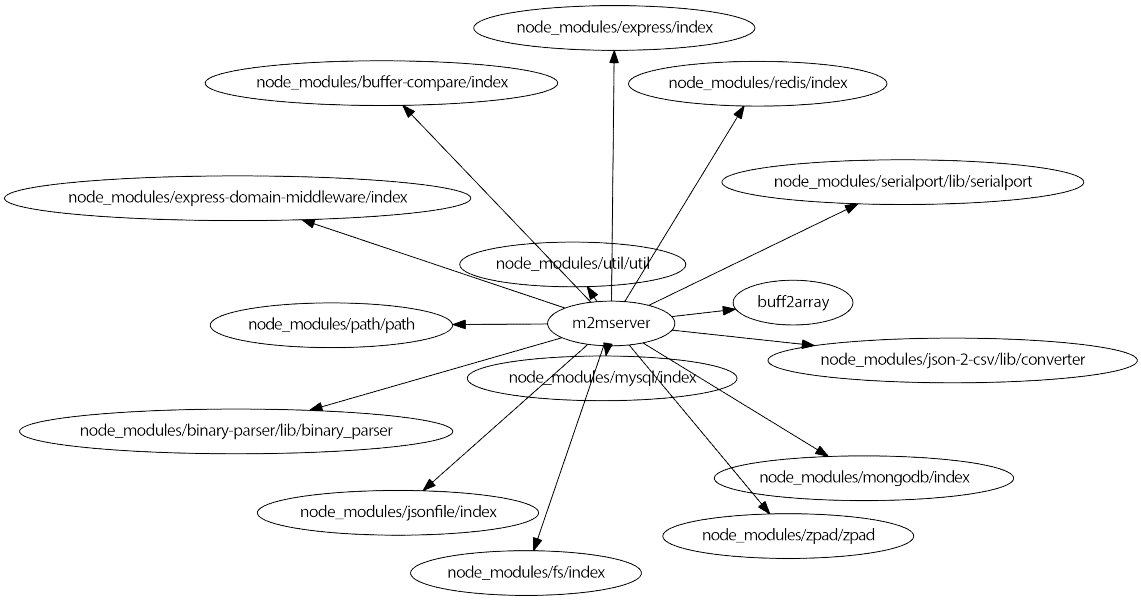
(c) 센서노드별 장기 시간이력 조회



(d) 센서노드 업데이트 상태

1. M2M 웹 어플리케이션 구성

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



1.2 패키지 구성

패키지 및 의존성 관리 파일	**package.json**

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

```
    "redis": "^2.6.0-1",
    "serialport": "^2.1.2",
    "socket.io": "^1.4.5",
    "url": "^0.11.0",
    "util": "^0.10.3",
    "zpad": "^0.5.0"
  },
  "devDependencies": {
    "grunt": "^1.0.1",
    "grunt-contrib-concat": "^1.0.1",
    "grunt-contrib-jshint": "^1.0.0",
    "grunt-contrib-uglify": "^1.0.1"
  }
}
```

1.3 데이터구조 및 클래스

Data Structure & Classes

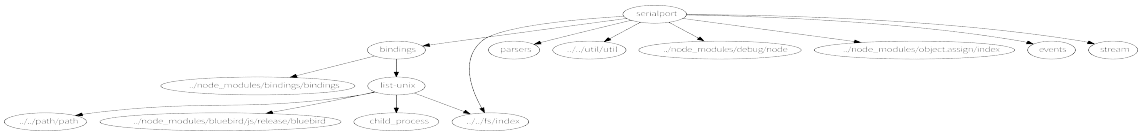
+SensorName.k : string = TimeStampUnix
+SensorTag.k : string = SENSORTST
+SensorUnit.k : string = EpochTime
+SensorValue.k : variable = TimeStamp
+TimeStamp : variable = undefined
+Value : variable = undefined
+buffersize : number = 128
+client : variable = undefined
+count : number = 0
+database : string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

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

1.4 의존성 패키지 구성도

1.4.1 통신 모듈 (serialport)

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



패키지 및 의존성 관리 파일	**package.json**

```
{
  "name": "serialport",
  "version": "2.1.2",
  "description": "Node.js package to access serial ports. Welcome your robotic
javascript overlords. Better yet, program them!",
  "author": {
    "name": "Chris Williams",
    "email": "voodootikigod@gmail.com",
    "url": "http://www.voodootikigod.com"
  },
  "binary": {
    "module_name": "serialport",
    "module_path": "build/{configuration}/",
    "          h          o          s          t          "          :
"https://github.com/voodootikigod/node-serialport/releases/download/2.1.2"
  },
  "main": "./lib/serialport",
  "repository": {
    "type": "git",
    "url": "git://github.com/voodootikigod/node-serialport.git"
  },
  "keywords": [
    "serialport",
    "nodebots",
    "johnny-five"
  ],
  "maintainers": [
    {
      "name": "jjrosent",
      "email": "jakerosenthal@gmail.com"
    },
    {
      "name": "reconbot",
      "email": "wizard@roborooter.com"
    },
    {
      "name": "voodootikigod",
      "email": "voodootikigod@gmail.com"
    }
  ],
  "dependencies": {
    "bindings": "1.2.1",
    "bluebird": "^3.3.5",
    "debug": "^2.1.1",
    "nan": "~2.2.1",
    "node-pre-gyp": "^0.6.26",
    "node-pre-gyp-github": "^1.1.0",
    "object.assign": "^4.0.3",
    "optimist": "~0.6.1",
    "sf": "0.1.7"
  },
  "devDependencies": {
    "chai": "^3.5.0",
    "chai-subset": "^1.2.2",
    "eslint-config-standard": "^5.1.0",
    "eslint-plugin-promise": "^1.1.0",
    "eslint-plugin-standard": "^1.3.2",
```

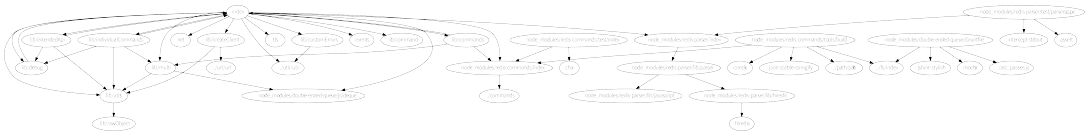
```
    "grunt": "0.4.5",
    "grunt-cli": "1.1.0",
    "grunt-mocha-test": "*",
    "gruntify-eslint": "^2.0.0",
    "mocha": "^2.4.5",
    "sandboxed-module": "^2.0.3",
    "sinon": "^1.17.3",
    "sinon-chai": "^2.8.0"
  },
  "engines": {
    "node": ">= 0.10.0"
  },
  "bin": {
    "serialportlist": "./bin/serialportList.js",
    "serialportterm": "./bin/serialportTerminal.js"
  },
  "license": "MIT",
  "scripts": {
    "install": "node-pre-gyp install --fallback-to-build",
    "rebuild": "npm rebuild && node-pre-gyp rebuild",
    "stress": "mocha --no-timeouts test/arduinoTest/stress.js",
    "test": "grunt --verbose"
  },
  "gypfile": true,
  "gitHead": "ac82f855e5a107ae8535fac9270236a9d06754ae",
  "bugs": {
    "url": "https://github.com/voodootikigod/node-serialport/issues"
  },
  "homepage": "https://github.com/voodootikigod/node-serialport#readme",
  "bundleDependencies": [
    "node-pre-gyp",
    "node-pre-gyp-github"
  ],
  "_id": "serialport@2.1.2",
  "_shasum": "493af176ac59043e7da5f2d7978fa30d1a8cb353",
  "_from": "serialport@>=2.1.2 <3.0.0",
  "_npmVersion": "3.8.6",
  "_nodeVersion": "5.11.0",
  "_npmUser": {
    "name": "reconbot",
    "email": "wizard@roborooter.com"
  },
  "dist": {
    "shasum": "493af176ac59043e7da5f2d7978fa30d1a8cb353",
    "tarball": "https://registry.npmjs.org/serialport/-/serialport-2.1.2.tgz"
  },
  "_npmOperationalInternal": {
    "host": "packages-12-west.internal.npmjs.com",
    "tmp": "tmp/serialport-2.1.2.tgz_1461557022672_0.7833052794449031"
  },
  "directories": {},
  "_resolved": "https://registry.npmjs.org/serialport/-/serialport-2.1.2.tgz",
  "readme": "ERROR: No README data found!"
}
```


1.4.2 Redis 모듈

- 레디스(Redis)는 오픈 소스이며 네트워크화, 인메모리 데이터베이스, 선택적 영구성 키−값 데이터 저장소이다. 레디스의 개발은 2013년 5월부터 Pivotal의 지원을 받고 있으며[1], 그 전에는 VMWare의 지원을 받았다.[2][3] DB−Engines.com의 월간 랭킹에 따르면, 레디스는 가장 인기 있는 키−값 저장소이다.[4]
- 성능은 memcached에 버금가면서 다양한 데이터 구조체를 지원함으로써 Message Queue, Shared memory, Remote directory 용도로 사용할 수 있으며, 이런 이유로 Instagram, Line 메신저 서비스, StackOverFlow등 여러 소셜 서비스에 널리 사용되고 있다. 본 과제에서는 실시간 데이터 표시와 응급

신호의 처리 및 빠른 상황 처리를 위해 적용하였다.

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



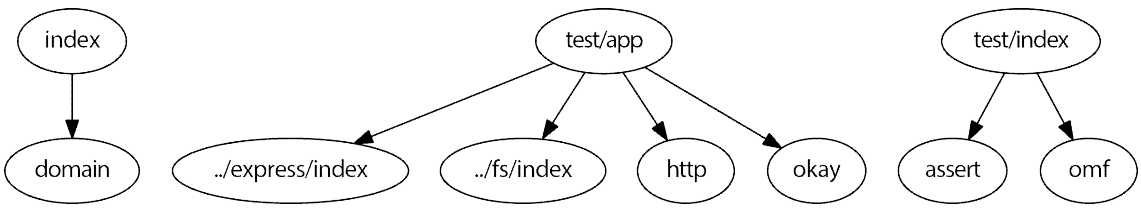
패키지 및 의존성 관리 파일	**package.json**

```
{
  "name": "redis",
  "version": "2.6.0-2",
  "description": "Redis client library",
  "keywords": [
    "database",
    "redis",
    "transaction",
    "pipelining",
    "performance",
    "queue",
    "nodejs",
    "pubsub",
    "backpressure"
  ],
  "author": {
    "name": "Matt Ranney",
    "email": "mjr@ranney.com"
  },
  "license": "MIT",
  "main": "./index.js",
  "scripts": {
    "coveralls": "nyc report --reporter=text-lcov | coveralls",
    "coverage": "nyc report --reporter=html",
    "benchmark": "node benchmarks/multi_bench.js",
    "test": "nyc --cache mocha ./test/*.js ./test/commands/*.js --timeout=8000",
    "posttest": "eslint . --fix"
  },
  "dependencies": {
    "double-ended-queue": "^2.1.0-0",
    "redis-commands": "^1.2.0",
    "redis-parser": "^1.3.0"
  },
  "engines": {
    "node": ">=0.10.0"
  },
  "devDependencies": {
    "bluebird": "^3.0.2",
    "coveralls": "^2.11.2",
    "intercept-stdout": "~0.1.2",
    "eslint": "^2.5.0",
    "metrics": "^0.1.9",
    "mocha": "^2.3.2",
    "nyc": "^6.0.0",
    "tcp-port-used": "^0.1.2",
    "uuid": "^2.0.1",
    "win-spawn": "^2.0.0"
  },
  "repository": {
    "type": "git",
    "url": "git://github.com/NodeRedis/node_redis.git"
  },
  "bugs": {
    "url": "https://github.com/NodeRedis/node_redis/issues"
  },
  "homepage": "https://github.com/NodeRedis/node_redis",
  "directories": {
    "example": "examples",
    "test": "test"
  },
  "gitHead": "a1755b91fb7ace72f79e3a42aeaf79374cd6023c",
  "_id": "redis@2.6.0-2",
  "_shasum": "808493eb168711bd36afdb1ac1ced22d19554779",
```

```
  "_from": "redis@>=2.6.0-1 <3.0.0",
  "_npmVersion": "2.15.1",
  "_nodeVersion": "4.4.3",
  "_npmUser": {
    "name": "bridgear",
    "email": "ruben@bridgewater.de"
  },
  "dist": {
    "shasum": "808493eb168711bd36afdb1ac1ced22d19554779",
    "tarball": "https://registry.npmjs.org/redis/-/redis-2.6.0-2.tgz"
  },
  "maintainers": [
    {
      "name": "bcoe",
      "email": "ben@npmjs.com"
    },
    {
      "name": "bridgear",
      "email": "ruben@bridgewater.de"
    },
    {
      "name": "bryce",
      "email": "bryce@ravenwall.com"
    },
    {
      "name": "mjr",
      "email": "mjr@ranney.com"
    }
  ],
  "_npmOperationalInternal": {
    "host": "packages-12-west.internal.npmjs.com",
    "tmp": "tmp/redis-2.6.0-2.tgz_1461897861582_0.2723832947667688"
  },
  "_resolved": "https://registry.npmjs.org/redis/-/redis-2.6.0-2.tgz",
  "readme": "ERROR: No README data found!"
}
```


1.4.3. Express 미들웨어 API

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



- 패키지 및 의존성 관리 파일 — **package.json**


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

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

```
  },
  "_from": "express-domain-middleware@>=0.1.0 <0.2.0",
  "_npmVersion": "1.2.17",
  "_npmUser": {
    "name": "brianc",
    "email": "brian.m.carlson@gmail.com"
  },
  "maintainers": [
    {
      "name": "brianc",
      "email": "brian.m.carlson@gmail.com"
    }
  ],
  "_shasum": "36731b7c1901284fbf4fb5a62b0e7b0457d8e8c5",
  "_resolved": "https://registry.npmjs.org/express-domain-middleware/-/express-domain-middleware-0.1.0.tgz",
  "bugs": {
    "url": "https://github.com/brianc/node-domain-middleware/issues"
  },
  "homepage": "https://github.com/brianc/node-domain-middleware#readme"
}
```

1.4.4 MQTT 서버/클라이언트

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

패키지 및 의존성 관리 파일	**package.json**

```
{
  "name": "mqtt",
  "description": "A library for the MQTT protocol",
  "version": "1.10.0",
  "contributors": [
    {
      "name": "Adam Rudd",
      "email": "adamvrr@gmail.com"
    },
    {
      "name": "Matteo Collina",
      "email": "matteo.collina@gmail.com",
      "url": "https://github.com/mcollina"
    }
  ],
  "keywords": [
    "mqtt",
    "publish/subscribe",
    "publish",
    "subscribe"
  ],
  "license": "MIT",
  "repository": {
    "type": "git",
    "url": "git://github.com/mqttjs/MQTT.js.git"
  },
  "main": "mqtt.js",
```

```
  "scripts": {
    "test": "mocha",
    "pretest": "npm run check-style",
    "check-style:eslint": "eslint mqtt.js lib test test/browser test/helpers",
    "check-style:jshint": "jshint mqtt.js lib test test/browser test/helpers",
    "check-style:jscs": "jscs mqtt.js lib test test/browser test/helpers",
    "check-style": "npm run check-style:jshint && npm run check-style:jscs && npm run
check-style:eslint",
    "browser-test":   "zuul   --server   test/browser/server.js   --local   --open
test/browser/test.js",
    "sauce-test":   "zuul   --server   test/browser/server.js   --tunnel   ngrok   --
test/browser/test.js"
  },
  "pre-commit": [
    "test"
  ],
  "bin": {
    "mqtt_pub": "./bin/pub.js",
    "mqtt_sub": "./bin/sub.js",
    "mqtt": "./mqtt.js"
  },
  "engines": {
    "node": ">=0.8.6"
  },
  "browser": {
    "./mqtt.js": "./lib/connect/index.js",
    "fs": false,
    "tls": false,
    "net": false
  },
  "dependencies": {
    "commist": "^1.0.0",
    "concat-stream": "^1.4.7",
    "end-of-stream": "^1.1.0",
    "help-me": "^0.1.0",
    "inherits": "^2.0.1",
    "minimist": "^1.1.0",
    "mqtt-connection": "^2.0.0",
    "mqtt-packet": "^3.2.0",
    "pump": "^1.0.1",
    "readable-stream": "~1.0.2",
    "reinterval": "^1.0.1",
    "split2": "^2.0.1",
    "websocket-stream": "^3.0.1",
    "xtend": "^4.0.0"
  },
  "devDependencies": {
    "browserify": "^11.0.1",
    "eslint": "^1.3.1",
    "jscs": "^2.1.1",
    "jshint": "2.7.0",
    "mocha": "*",
    "pre-commit": "1.1.1",
    "should": "*",
    "sinon": "~1.10.0",
    "through2": "^0.6.3",
    "uglify": "^0.1.1",
    "ws": "^0.8.0",
    "zuul": "^3.4.0",
    "zuul-ngrok": "github:gnandretta/zuul-ngrok#upgrade-ngrok"
  },
  "gitHead": "a62351b440e693a73ab0b4f00aa79ec2c7ea2b6f",
  "bugs": {
    "url": "https://github.com/mqttjs/MQTT.js/issues"
  },
  "homepage": "https://github.com/mqttjs/MQTT.js#readme",
  "_id": "mqtt@1.10.0",
  "_shasum": "0e82ea9bb3cde6b385cae0c2c728a6921d338c7c",
  "_from": "mqtt@>=1.8.0 <2.0.0",
  "_npmVersion": "2.15.0",
  "_nodeVersion": "4.4.2",
  "_npmUser": {
    "name": "matteo.collina",
    "email": "hello@matteocollina.com"
  },
  "dist": {
    "shasum": "0e82ea9bb3cde6b385cae0c2c728a6921d338c7c",
    "tarball": "https://registry.npmjs.org/mqtt/-/mqtt-1.10.0.tgz"
  },
  "maintainers": [
    {
```

```
      "name": "adamvr",
      "email": "adam.rudd@uqconnect.edu.au"
    },
    {
      "name": "matteo.collina",
      "email": "hello@matteocollina.com"
    }
  ],
  "_npmOperationalInternal": {
    "host": "packages-16-east.internal.npmjs.com",
    "tmp": "tmp/mqtt-1.10.0.tgz_1461768142810_0.8647426271345466"
  },
  "directories": {},
  "_resolved": "https://registry.npmjs.org/mqtt/-/mqtt-1.10.0.tgz"
}
```

1.4.5 Socket.io 모듈

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

```
```

패키지 및 의존성 관리 파일 | **package.json**

```
{
  "name": "socket.io",
  "version": "1.4.6",
  "description": "node.js realtime framework server",
  "keywords": [
    "realtime",
    "framework",
    "websocket",
    "tcp",
    "events",
    "socket",
    "io"
  ],
  "main": "./lib/index",
  "license": "MIT",
  "repository": {
    "type": "git",
    "url": "git://github.com/Automattic/socket.io.git"
  },
  "scripts": {
    "test": "mocha --reporter dot --slow 200ms --bail"
  },
  "dependencies": {
    "engine.io": "1.6.9",
    "socket.io-parser": "2.2.6",
    "socket.io-client": "1.4.6",
    "socket.io-adapter": "0.4.0",
    "has-binary": "0.1.7",
    "debug": "2.2.0"
  },
  "devDependencies": {
```

```
    "expect.js": "0.3.1",
    "istanbul": "0.4.1",
    "mocha": "2.3.4",
    "superagent": "1.6.1",
    "supertest": "1.1.0",
    "zuul-ngrok": "3.2.0"
  },
  "contributors": [
    {
      "name": "Guillermo Rauch",
      "email": "rauchg@gmail.com"
    },
    {
      "name": "Arnout Kazemier",
      "email": "info@3rd-eden.com"
    },
    {
      "name": "Vladimir Dronnikov",
      "email": "dronnikov@gmail.com"
    },
    {
      "name": "Einar Otto Stangvik",
      "email": "einaros@gmail.com"
    }
  ],
  "gitHead": "dcca01f5a42b49e6fe68abefaabe5291e834c5d0",
  "bugs": {
    "url": "https://github.com/Automattic/socket.io/issues"
  },
  "homepage": "https://github.com/Automattic/socket.io#readme",
  "_id": "socket.io@1.4.6",
  "_shasum": "0dddc2cb8fb9b66fc928604f13f6aa910254cc1c",
  "_from": "socket.io@>=1.4.5 <2.0.0",
  "_npmVersion": "2.15.0",
  "_nodeVersion": "4.4.2",
  "_npmUser": {
    "name": "rauchg",
    "email": "rauchg@gmail.com"
  },
  "dist": {
    "shasum": "0dddc2cb8fb9b66fc928604f13f6aa910254cc1c",
    "tarball": "https://registry.npmjs.org/socket.io/-/socket.io-1.4.6.tgz"
  },
  "maintainers": [
    {
      "name": "rauchg",
      "email": "rauchg@gmail.com"
    }
  ],
  "_npmOperationalInternal": {
    "host": "packages-16-east.internal.npmjs.com",
    "tmp": "tmp/socket.io-1.4.6.tgz_1462247882801_0.6891660823021084"
  },
  "directories": {},
  "_resolved": "https://registry.npmjs.org/socket.io/-/socket.io-1.4.6.tgz"
}
```

제3절 무선 센서노드의 설치위치 및 장기계측 데이터

1. 무선 센서노드의 설치위치 산정에 대한 근거

1.1 HCN : 사이안화 수소

- 사이안화 수소(hydrogen cyanide) 혹은 청화수소(靑化水素), 청산(靑酸, prussic acid)은 화학식이 HCN인 화합물이다. 물, 에테르, 에탄올 등에는 반드시 녹으며, 거기서도 수용액은 별명이 사이안화 수소산 또는 청산이다. 무색인 맹독성 화합물이며, 휘발성이 있다. 치사량은 2g이며, 불에도 잘 탄다. 실온보다 약간 높은 26°C에서 끓는다. 쓴맛이며 아몬드향이 나지만 유전적 소인으로 냄새를 맡지 못하는 사람도 있다. 사이안화 수소는 약산이며 수용액에서 사이안화 음이온(CN-)을 내놓으며 부분적으로 이온화된다.

• 일반적인 성질	
• IUPAC 이름	• Formonitrile
• 화학식	• HCN
• CAS 번호	• 74-90-8
• PubChem	• 768
• ChemSpider	• 748

• 물리적 성질	
• 분자량	• 27.0253 g/mol
• 녹는점	• 259.75 K (−13.4 °C, 7.88 °F)
• 끓는점	• 298.75 K (25.6 °C, 78.08 °F)
• 밀도	• 0.687 g/cm3

1.2 센서노드의 설치간격

- 본 연구에서는 사인화수소의 화학테러를 중심으로 확산속도를 고려하고 확산방향을 고려하여 센서노드간 설치간격을 계산하였다.
- 확산(diffusion)은 밀도 차이나 농도 차이에 의해 물질을 이루고 있는 입자들이 농도(밀도)가 높은 쪽에서 낮은 쪽으로 액체나 기체 속으로 분자가 퍼져 나가는 현상이다. Graham의 법칙에 의해 가스의 분자량을 통해 확산속도를 구할 수 있다.

- $\frac{v_A}{v_B} = \sqrt{\frac{M_B}{M_A}} = \sqrt{\frac{d_B}{d_A}}$
- 여기서 v_A, v_B는 확산속도, M_A, M_B는 각각 기체의 분자량 d_A, d_B는 기체의 밀도이다.

- 평상시의 공기의 밀도는 1기압에서 $\rho = \frac{1.293}{1+0.00367t}(kg/m^3)$을 가지고 있다. 사이안화 수소는 평온 1기압에서 $687 kg/m^3$의 밀도를 가지며 공기의 밀도 ($1.255 kg/m^3$)보다 크다. 사이안화수소의 분자량은 27.0253 g/mol이고 공기의 평균 분자량은 28.966 g/mol이다. 바람의 영향이 없을 때 사이안화수소의 공기 중 확산속도는 대략 1.035m/s로 계산할 수 있으며, 센서노드들의 슬립 시간과 센싱시간 그리고 전송까지 걸리는 주기를 포함하면 5초정도 소요된다. 따라서 본 연구에서는 사인화수소의 확산되어 검출되기까지를 고려하여 5m간격으로 설치하도록 한다.

1.3 화재 확산에 대한 고려

- 1995 사린(sarin) 가스 사고
- · 일시 : 1995. 3. 20. 08:09～10:01
- · 장소 : 도쿄 지하철 히비야, 마루노우치, 지요다 등 16개 역
- · 원인 : 16개의 지하철 역구내와 일본 중심에 있는 지하철 3개 노선 열차 5대에 신경독성가스가 설치되어 약 1,000명의 승객을 실은 지하철의 속에서 공포의 독성가스가 살포됨.
- · 피해상황 : 5,510명 (사망 12명, 중상 53명, 경상 5448명)
- · 성상 및 유해성 [CH3(C3H7O)FPO]
- – 무색, 무취의 맹독성, 시안화칼륨(청산가리)의 500배 독성
- – 피부, 눈으로 흡수

- – 증상 : 동공수축, 근육마비, 질식
- – 치사량 :체중 1kg당 0.01mg흡입시

2. 센서노드 시각동기화 API 개발 및 업데이트

- 본 연구에서 구성한 네트워크는 N키퍼와 센서노드간의 스타토폴로지 형식으로 통신한다. 따라서 도달 가능한 센서노드의 결과는 각 N키퍼에 의해 중복될 수 있다. 이러한 중복데이터 처리를 위하여 센서노드의 시각을 N키퍼와 일치시켜 고유화 시켜 Key값과 Score로 처리하게 된다. 가장먼저 도달한 N키퍼의 값을 기준으로 중복된 데이터를 소거하여 역무실 마스터PC와 본사 서버에 저장하게 된다. 이를 위하여 센서노드 API를 구성하였다.
- 코드 원문은 아래와 같으며, 시각동기화를 위하여 N키퍼에 페이로드로 요청을 보내고 타임아웃으로 대기한다. N키퍼는 페이로드를 받을 경우 시각 데이터를 ATCOMMAND로 센서노드로 송신하게 된다.

```
#include <WaspSensorGas_Pro.h>
#include <WaspSX1272.h> // #define SX1272_debug_mode 0 or 1
#include <WaspFrame.h>  //#define GAS_DEBUG 0 or 2

//////////////////////////////////////////
void LoRa_Setup();
//////////////////////////////////////////
int8_t e;
char nodeID[] = "SHTPI005";
int addr = 5;
uint8_t meshlium_address = 1;
int lora_error;
//////////////////////////////////////////
Gas O2(SOCKET_A);
Gas CO(SOCKET_B);
Gas NO2(SOCKET_C);
Gas HCN(SOCKET_F);

float CO_value;                 // Stores the concentration level in ppm
float O2_value;                 // Stores the concentration level in ppm
float NO2_value;     // Stores the concentration level in ppm
float HCN_value;     // Stores the concentration level in ppm
float Temp_value;
float Humi_value;
float Pres_value;

// Define variable to store Epoch time
unsigned long rtc;
unsigned long epoch;

// Define variable for UTC timestamps
timestamp_t    time;
char init_RTC[100];

//Random Backoff  3~8sec
```

```
#define DELAY_MIN 3000
#define DELAY_MAX 8000


//////////////////////////////////////////
char* sleepTime = "00:00:00:05";                    // RTC_ABSOLUTE, RTC_ALM1_MODE4, every 10 seconds in
1 minute
char* sleep_WarmUp_Time = "00:00:02:00";// RTC_OFFSET, RTC_ALM1_MODE1, after 2 minutes

void setup()
{

        // init USB port
        USB.ON();
        RTC.ON();

        // Setting time [yy:mm:dd:dow:hh:mm:ss]
        RTC.setTime("15:01:01:05:00:00:00");  //RTC Setting for Comparison with Gateway RTC value
        delay(100);

//      USB.println(F("No Sensor Reading, Test for Recived packet from LoRa Gateway, 20161213!!"));
//      PWR.deepSleep("00:00:03:00", RTC_OFFSET, RTC_ALM1_MODE1, ALL_ON);

        frame.setID(nodeID);

        LoRa_Setup(); //first configure

        sx1272.setRTCfromLoRaGateway_SH(addr);  // Node TimeSync from LoRaGateway

        delay(100);

        CO.ON();
        NO2.ON();
        HCN.ON();
        O2.ON();

        USB.println(F("Sensor Wram up, wait for 3minutes, Using deepsleep mode to reduce the battery
consumption"));
        PWR.deepSleep("00:00:03:00", RTC_OFFSET, RTC_ALM1_MODE1, ALL_ON);

        CO.OFF();
        NO2.OFF();
        HCN.OFF();
        O2.OFF();

}

void loop()
{
        USB.ON();
        RTC.ON();
        delay(800);
        LoRa_Setup();   //configuring in Loop


         ////////////////////////////////
         // 1. Get SNR
         ////////////////////////////////
```

```
            e = sx1272.getSNR();

            // check status
            if( e == 0 )
            {
              USB.print(F("Getting SNR ₩t₩t--> OK. "));
              USB.print(F("SNR current value is: "));
              USB.println(sx1272._SNR);
            }
            else
            {
              USB.println(F("Getting SNR --> ERROR"));
            }

            ////////////////////////////////
            // 2. Get channel RSSI
            ////////////////////////////////
            e = sx1272.getRSSI();

            // check status
            if( e == 0 )
            {
              USB.print(F("Getting RSSI ₩t₩t--> OK. "));
              USB.print(F("RSSI current value is: "));
              USB.println(sx1272._RSSI);

            }
            else
            {
              USB.println(F("Getting RSSI --> ERROR"));
            }

            ////////////////////////////////
            // 3. Get last packet received RSSI
            ////////////////////////////////
            e = sx1272.getRSSIpacket();

            // check status
            if( e == 0 )
            {
              USB.print(F("Getting RSSI packet ₩t--> OK. "));
              USB.print(F("Last packet RSSI value is: "));
              USB.println(sx1272._RSSIpacket);
            }
            else
            {
              USB.println(F("Getting RSSI packet --> ERROR"));
            }

          USB.println(F("₩n======================================="));
          //USB.println(F("₩n Phase to packet receive, Wating for packet during about 80 Second ₩n"));
//        USB.println(F("₩n Phase to Time Sync from LoRa Gateway, Wating for packet during about 80 Second ₩n"));

//        sx1272.setRTCfromLoRaGateway_SH(addr);

//        LoRa_Send();    //Packet Send
//        Epochtime();  //compre with epoch & RTC
```

```
    Sensor_Reading();
        LoRa_Send();

        USB.println(F("<<<< Going to Sleep...>>>>"));
        PWR.deepSleep(sleepTime, RTC_OFFSET, RTC_ALM1_MODE1, ALL_OFF);     //Sleep mode
        USB.println(F("<<<< Wake Up! >>>>"));
}


void LoRa_Setup()
{
        USB.println(F("----------------------------------------"));
        USB.println(F("Setting configuration:"));

        // init SX1272 module
        sx1272.ON();

        // Select channel
        lora_error = sx1272.setChannel(CH_08_900);
        USB.print(F("Setting Channel CH_08_900.₩t state "));
        USB.println(lora_error);

        /*
        // Set packet length
        lora_error = sx1272.setPacketLength(83);
        USB.print(F("Setting Payload Length to '83'.₩t state "));
        USB.println(lora_error);
        */

        // Set Header mode: ON
        lora_error = sx1272.setHeaderON();
        USB.print(F("Setting Header ON.₩t₩t state "));
        USB.println(lora_error);

        // Set Mode
        lora_error = sx1272.setMode(1);
        USB.print(F("Setting Mode '1'.₩t₩t state "));
        USB.println(lora_error);

        // Select output power (Max, High or Low)
        lora_error = sx1272.setPower('M');
        USB.print(F("Setting Power to 'M'.₩t₩t state "));
        USB.println(lora_error);

        // Set CRC mode: ON
        lora_error = sx1272.setCRC_ON();
        USB.print(F("Setting CRC ON.₩t₩t₩t state "));
        USB.println(lora_error);

        // Set Node Address: from 2 to 255                                      //////Address Setting
        lora_error = sx1272.setNodeAddress(addr);
//      USB.print(F("Setting Node Address to '%d'.₩t state ",addr));
        USB.printf("Setting Node Address to '%d'.₩t state ",addr);
        USB.println(lora_error);
        USB.println();
        delay(1000);

        USB.println(F("Setting DONE!!"));
```

```
        USB.println(F("---------------------------------------"));
}


uint8_t setRTCfromLoRaGateway1(int addr_)
{
        uint8_t length = 0;
        byte status;
        byte status_rx;
        char buffer[100];
        char timestamp[30];
        int year, yearH;
        int month;
        int date;
        int hour;
        int minute;
        int second;
        uint8_t meshlium_address = 1;
        //float ts[]="";
        int ts1=addr_;
//      char* ts1[];

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

        //ts1 = addr;
//      USB.printf(addr);


//    strcat(ts,ts1)
//      strcpy(ts1,"TS#%d",addr);
//      ts1[10]=ts+*addr;

//      USB.printf(ts);
//      USB.printf(addr);

        // Creating frame to send
        frame.createFrame(BINARY);
        frame.decrementSequence();
        frame.setFrameType(155);
        frame.addSensor(SENSOR_GMT,ts1);

        // Printing frame
        #if (SX1272_debug_mode >= 0)    //init >1
            //USB.println(F("packet sent OK"));
                frame.showFrame();
        #endif

        // Sending packet before ending a timeout


        status = sx1272.sendPacketTimeoutACKRetries( meshlium_address, frame.buffer, frame.length );

        // check tx status
        if( status == 0 )
        {
                #if (SX1272_debug_mode >= 0)    //init >1
```

```
                        USB.println(F("packet sent OK"));
                #endif
        }
        else
        {
                #if (SX1272_debug_mode >= 0)
                        USB.println(F("packet sent ERROR"));
                #endif
                return 1;
        }

        // wait to receive packet with Timestamp for several seconds
        USB.println(F("Wiat for packet receive to 100000 milli second"));

        status_rx = sx1272.receivePacketTimeoutACK(1000);

        USB.printf("receive packet status : '%d'.₩t   ",status_rx);


        // check rx status
        if( status_rx == 0 )
        {
                #if (SX1272_debug_mode >= 0)    //init >1
                USB.println(F("Show packet received OK : "));
                sx1272.showReceivedPacket();
                #endif

                // get payload length
                length = sx1272.packet_received.length - OFFSET_PAYLOADLENGTH;

                // check if valid length then copy buffer
                if( length < sizeof(buffer) )
                {
                        // copy payload
                        memcpy( buffer, sx1272.packet_received.data, length);

                        #if (SX1272_debug_mode >= 0) //init >1
                                USB.print(F("₩nbuffer:"));
                                USB.println(buffer);
                        #endif
                }
                else
                {
                        #if (SX1272_debug_mode >= 0)
                                USB.println(F("ERROR: buffer not enough for length"));
                                USB.print(F("length:"));
                                USB.println(length, DEC);
                        #endif
                        return 1;
                }

                // get all data fields: "YYYYMMDDHHMMSS"
                sscanf( buffer,
                                "%2u%2u%2u%2u%2u%2u%2u",
                                &yearH,
                                &year,
                                &month,
                                &date,
```

```
                                        &hour,
                                        &minute,
                                        &second );


                    unsigned long offsetTime = (unsigned long)(sx1272.timeOnAir(25) +
sx1272.timeOnAir(frame.length))/1000;
                    unsigned long actualTime =  RTC.getEpochTime(year,

                                month,

                                date,

                                hour,

                                minute,

                                second);

                    unsigned long timeToSet = actualTime + offsetTime + 3; // add offset

                    timestamp_t tm;

                    // get time elements from epoch time
                    RTC.breakTimeAbsolute( timeToSet, &tm );

                    // create sentence to set Time
                    snprintf(   timestamp,
                                                sizeof(timestamp),
                                                "%02u:%02u:%02u:%02u:%02u:%02u:%02u",
                                                tm.year,
                                                tm.month,
                                                tm.date,
                                                RTC.dow(tm.year, tm.month, tm.date),
                                                tm.hour,
                                                tm.minute,
                                                tm.second );

                    #if (SX1272_debug_mode >= 0)    //init >1
                                USB.print(F("₩ntimestamp:"));
                                USB.println(timestamp);
                    #endif

                    RTC.ON();
                    RTC.setTime(timestamp);

                    #if (SX1272_debug_mode >= 0)    //init >1
                                USB.print(F("RTC time:"));
                                USB.println(RTC.getTime());
                    #endif
          }
          else
          {
                    #if (SX1272_debug_mode >= 0)  //init >1
                                USB.print(F("Receiving packet TIMEOUT, state. Set Sync retries!!₩n"));    //return to
request packet Phase

                                USB.printf("receive packet status : '%d'.₩t  ",status_rx);
```

```
                #endif
                //return 1;

                ////////////////////////////////////////
                /////// continued until receive packet////////
                ////////////////////////////////////////
                int count1=0;
                while(1){
                        if( status_rx != 0)
                        {

                        USB.printf("₩nReset Count :  %d.₩n",count1);
                        count1++;

                                status  =  sx1272.sendPacketTimeoutACKRetries(  meshlium_address,
frame.buffer,  frame.length );

        // check tx status
        if( status == 0 )
        {
                #if (SX1272_debug_mode >= 0)    //init >1
                        USB.println(F("packet sent OK"));
                #endif
        }
        else
        {
                #if (SX1272_debug_mode >= 0)
                        USB.println(F("packet sent ERROR"));
                #endif
                return 1;
        }

        // wait to receive packet with Timestamp for several seconds
        USB.println(F("Wiat for packet receive to 100000 milli second"));

        status_rx = sx1272.receivePacketTimeoutACK(10000);

        USB.printf("receive packet status : '%d'.₩t  ",status_rx);


        // check rx status
        if( status_rx == 0 )
        {
                #if (SX1272_debug_mode >= 0)    //init >1
                USB.println(F("Show packet received OK : "));
                sx1272.showReceivedPacket();
                #endif

                // get payload length
                length = sx1272.packet_received.length - OFFSET_PAYLOADLENGTH;

                // check if valid length then copy buffer
                if( length < sizeof(buffer) )
                {
                        // copy payload
                        memcpy( buffer, sx1272.packet_received.data, length);
```

```
                #if (SX1272_debug_mode >= 0) //init >1
                        USB.print(F("₩nbuffer:"));
                        USB.println(buffer);
                #endif
        }
        else
        {
                #if (SX1272_debug_mode >= 0)
                        USB.println(F("ERROR: buffer not enough for length"));
                        USB.print(F("length:"));
                        USB.println(length, DEC);
                #endif
                return 1;
        }

        // get all data fields: "YYYYMMDDHHMMSS"
        sscanf( buffer,
                        "%2u%2u%2u%2u%2u%2u%2u",
                        &yearH,
                        &year,
                        &month,
                        &date,
                        &hour,
                        &minute,
                        &second );


        unsigned long offsetTime = (unsigned long)(sx1272.timeOnAir(25) +
sx1272.timeOnAir(frame.length))/1000;
        unsigned long actualTime =  RTC.getEpochTime(year,

                month,

                date,

                hour,

                minute,

                second);

        unsigned long timeToSet = actualTime + offsetTime + 3; // add offset

        timestamp_t tm;

        // get time elements from epoch time
        RTC.breakTimeAbsolute( timeToSet, &tm );

        // create sentence to set Time
        snprintf(     timestamp,
                                sizeof(timestamp),
                                "%02u:%02u:%02u:%02u:%02u:%02u:%02u",
                                tm.year,
                                tm.month,
                                tm.date,
                                RTC.dow(tm.year, tm.month, tm.date),
                                tm.hour,
                                tm.minute,
```

```
                                                tm.second  );

                        #if (SX1272_debug_mode >= 0)    //init >1
                                USB.print(F("₩ntimestamp:"));
                                USB.println(timestamp);
                        #endif

                        RTC.ON();
                        RTC.setTime(timestamp);

                        #if (SX1272_debug_mode >= 0)    //init >1
                                USB.print(F("RTC time:"));
                                USB.println(RTC.getTime());
                        #endif
                        break;

                }
                        else
                {
                        #if (SX1272_debug_mode >= 0)  //init >1
                                USB.print(F("Receiving packet TIMEOUT, state. Set Sync retries!!₩n"));    //return to
request packet Phase

                                USB.printf("receive packet status : '%d'.₩t  ",status_rx);

                        #endif
                        //return 1;
                //return(status_rx);
                                }
                        //return(status_rx);
                                                }

                        }
                        }

                return 0;
}

void Sensor_Reading()
{
        CO.ON();
        HCN.ON();
        O2.ON();
        NO2.ON();
        delay(1000);

//      USB.println(F("Sensor Wram up, wait for 2minutes, Using deepsleep mode to reduce the battery
consumption"));
        PWR.deepSleep("00:00:01:00", RTC_OFFSET, RTC_ALM1_MODE1, ALL_ON);


        //NO2.autoGain();
        //CO.autoGain();
        //O2.autoGain();
        //HCN.autoGain();

        CO_value = CO.getConc();
        O2_value = O2.getConc();
```

```
        HCN_value = HCN.getConc();
        NO2_value = NO2.getConc();   //MCP3421_ULTRA_HIGH_RES

        Temp_value = CO.getTemp();
        Humi_value = CO.getHumidity();
        Pres_value = CO.getPressure();

        USB.println(F("***************  Gas Sonsor  ********************"));
        USB.print(F("CO concentration: "));
        USB.print(CO_value);
        USB.println(F(" ppm"));
        USB.print(F("NO2 concentration: "));
        USB.print(NO2_value);
        USB.println(F(" ppm"));
        USB.print(F("O2 concentration: "));
        USB.print(O2_value);
        USB.println(F(" ppm"));
        USB.print(F("HCN concentration: "));
        USB.print(HCN_value);
        USB.println(F(" ppm"));

        USB.print(F("Temperature: "));
        USB.print(Temp_value);
        USB.println(F(" Celsius degrees"));
        USB.print(F("RH: "));
        USB.print(Humi_value);
        USB.println(F(" %"));
        USB.print(F("Pressure: "));
        USB.print(Pres_value);
        USB.println(F(" Pa"));

        CO.OFF();
        NO2.OFF();
        HCN.OFF();
        O2.OFF();

}

void Epochtime()
{

        ////////////////////////////////////////////////////
        //          1. Compare with RTC and eopch time
        ////////////////////////////////////////////////////


        // Get Epoch time from RTC values

                epoch = RTC.getEpochTime();
                USB.print(F("Epoch Time from RTC: "));
                USB.println( epoch );
                delay(100);

  ////////////////////////////////////////////////////
  //  2. Break Epoch time into UTC Time
  ////////////////////////////////////////////////////

  // Break Epoch time into UTC time
```

```
  RTC.breakTimeAbsolute( epoch, &time );

  USB.print(F("Break Time from epoch time:"));
  USB.print( time.year, DEC );     USB.print(F("/"));
  USB.print( time.month, DEC );    USB.print(F("/"));
  USB.print( time.date, DEC );     USB.print(F(","));
  USB.print( time.hour, DEC );     USB.print(F(":"));
  USB.print( time.minute, DEC );   USB.print(F(":"));
  USB.print( time.second, DEC );
  USB.println();
  delay(100);


 /*       rtc = unsigned long(RTC.getTime());
                  USB.print(F("RTC Time : "));
                  USB.println(RTC.getTime());
                  delay(100);

  ////////////////////////////////////////////////////
  //  3. Break seconds time into days/hours/minutes/seconds
  ////////////////////////////////////////////////////

  // Break Time from epoch subtraction
  RTC.breakTimeOffset(rtc, &time );

  USB.print(F("Break Time from RTC time:"));
  USB.print( time.year, DEC );     USB.print(F("/"));
  USB.print( time.month, DEC );    USB.print(F("/"));
  USB.print( time.date, DEC );     USB.print(F(","));
  USB.print( time.hour, DEC );     USB.print(F(":"));
  USB.print( time.minute, DEC );   USB.print(F(":"));
  USB.print( time.second, DEC );
  USB.println();
  delay(100);
*/
          USB.print(F("RTC Time : "));
          USB.println(RTC.getTime());

  USB.println(F("-----------------------------------"));
  delay(100);

}

void LoRa_Send()
{

           // Creat Frame
           frame.createFrame(BINARY);
           frame.addSensor(SENSOR_BAT,PWR.getBatteryLevel());
           frame.addSensor(SENSOR_RSSI, sx1272._RSSI);
      frame.addSensor(SENSOR_TST, RTC.getEpochTime());
           frame.addSensor(SENSOR_GP_O2, O2_value);
           frame.addSensor(SENSOR_GP_TC, Temp_value);
           frame.addSensor(SENSOR_GP_HUM, Humi_value);
           frame.addSensor(SENSOR_GP_PRES, Pres_value);
           frame.addSensor(SENSOR_GP_CO, CO_value);
           frame.addSensor(SENSOR_GP_NO2, NO2_value);
           frame.addSensor(SENSOR_GP_HCN, HCN_value);
```

```
            //frame.addSensor(SENSOR_RSSI, sx1272._RSSIpacket);
              frame.showFrame();


          // Sending packet before ending a timeout
          lora_error = sx1272.sendPacketTimeoutACK( meshlium_address, frame.buffer, frame.length );

          // if ACK was received check signal strength
          if( lora_error == 0 )
          {
                    USB.println(F("Packet sent OK!!"));
                    //Utils.setLED(LED1, LED_ON);delay(1000);Utils.setLED(LED1, LED_OFF);
                    Utils.blinkLEDs(200);
          }
          else
          {
               //Random backoff function call on 3 times(in Delay_Min to Delay_Max)
               int count=0;
                    for(count=0;count<2;count++)
                              {
                              RandomWait();
                              USB.printf("Retries count : %d₩n",count);
                              }


                    USB.println(F("Error sending the packet"));
                    USB.print(F("state: "));
                    USB.println(lora_error, DEC);
                    Utils.setLED(LED0, LED_ON);delay(1000);Utils.setLED(LED0, LED_OFF);
          }
         USB.println(RTC.getEpochTime());

}

void RandomWait()
{

             unsigned long randomDelay = 0;
             unsigned long epoch = 0;

             epoch = RTC.getEpochTime();
             srandom(epoch);
             randomDelay = rand()%(DELAY_MAX-DELAY_MIN)+DELAY_MIN;
             delay(randomDelay);

          // Sending packet before ending a timeout
          lora_error = sx1272.sendPacketTimeout( meshlium_address, frame.buffer, frame.length );

          // if ACK was received check signal strength
          if( lora_error == 0 )
          {
                    USB.println(F("Packet sent OK!!"));
                    //Utils.setLED(LED1, LED_ON);delay(1000);Utils.setLED(LED1, LED_OFF);
                    Utils.blinkLEDs(200);
          }

}
```

3. 재난 전조 센서노드 장기계측 결과

3.1 장기데이터 시간이력

- 테스트베드 개포동역에 설치된 센서노드는 상시전원이 연결되어 있지 않기 때문에 완충상태에서 센서노드를 설치하고 배터리 방전시까지 장기데이터를 계측하였다.
- 계측한 데이터는 평시데이터로 무빙에버리징을 통한 핵밀도추정(Kernel density estimation, KDE)으로 평시데이터의 통계치를 계측하였다.
- 통계학에서, kernel density estimation(또는 Parzen window method, Emanuel Parzen의 이름을 딴)은 확률변수(random variable)의 확률밀도함수(probability density function, pdf)를 추정하는 비모수적(non-parametric) 방법이다. 한 예로, 모집단 표본에 대해 몇몇 자료가 주어졌을때, kernel density estimation은 외삽법(extrapolation)에 의한 전체의 모집단 추정을 가능케 한다.
- 또한 Parzen window는 Blackman-Tukey procedure와 같은 procedure에서 lag window로써 신호처리(signal processing)에 사용된다.
- Histogram은 Kernel density estimate로부터 수집된 점 표본들의 모임이라고 생각될 수 있다.
- 정의

만약 x1, x2, ..., xN ~ f이 서로 독립적이며 동일한 분포를 따르는(independent and identically-distributed) 확률변수들의 표본이라면, 확률밀도함수의 kernel density approximation은

$$\widehat{f_h}(x) = \frac{1}{N_h}\sum_{i=1}^{N}\left(\frac{x-x_i}{h}\right)$$

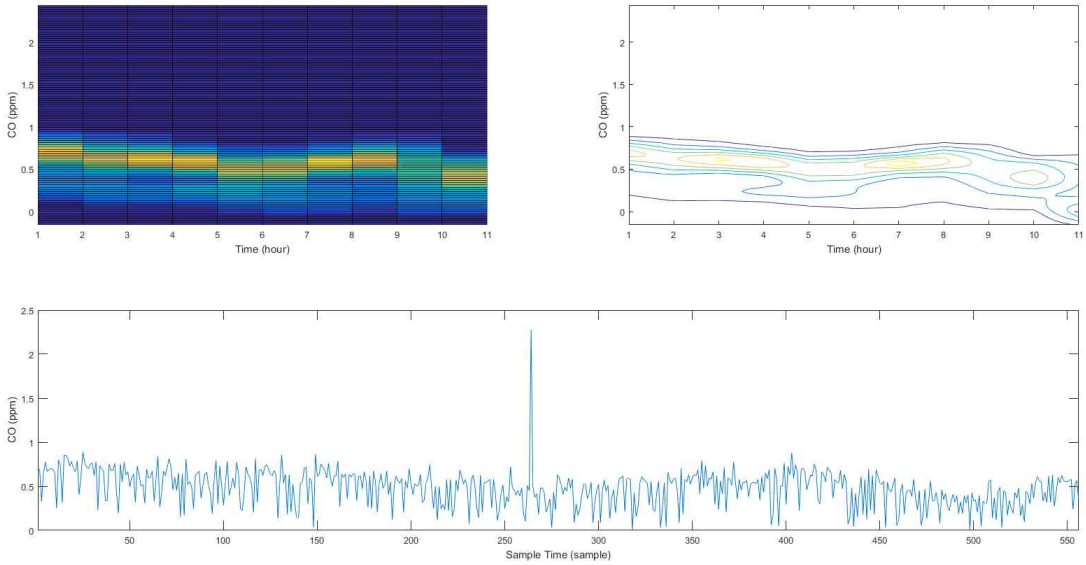
이다. 이때, K는 어떠한 kernel이고, h는 banwidth라 불리는 평활모수(smoothing parameter)이다. 종종 K는 평균이 0이고 분산이 1인 표준정규분포를 취한다. 따라서, 분산은 모수 h를 통해 간접적으로 제어된다. :

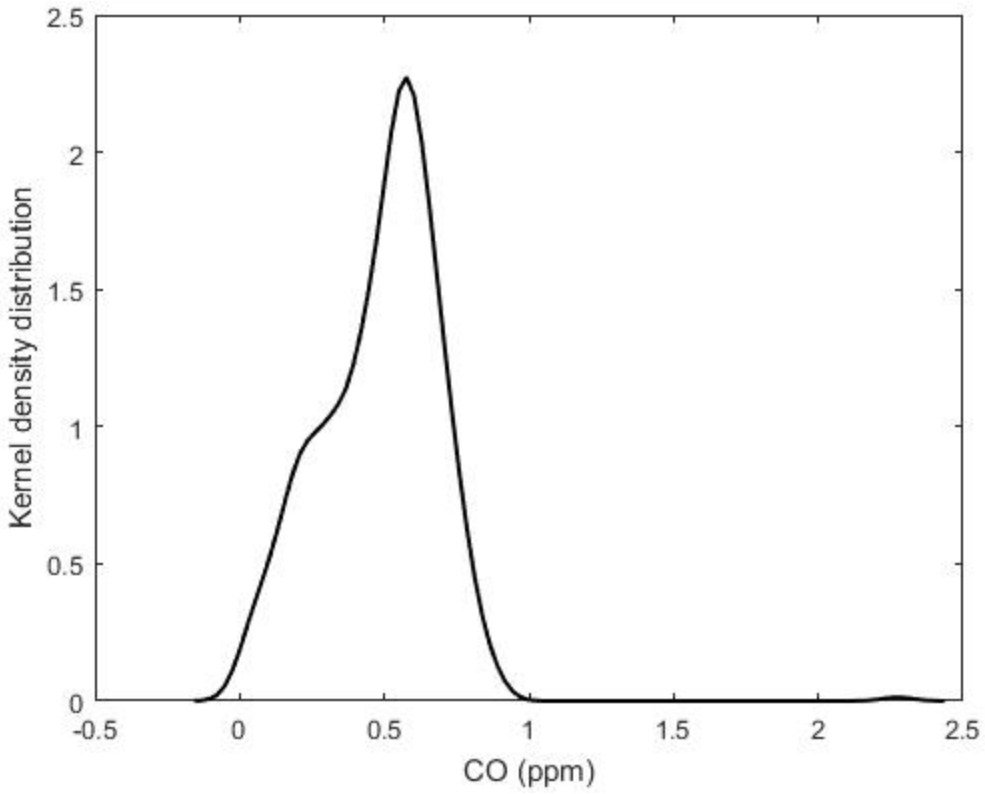
$$K\left(\frac{x-x_i}{h}\right) = \frac{1}{\sqrt{2\pi}}e^{-\frac{(x-x_i)^2}{2h^2}}$$

3.2 센서노드 날짜별 설치위치별 장기 계측 데이터

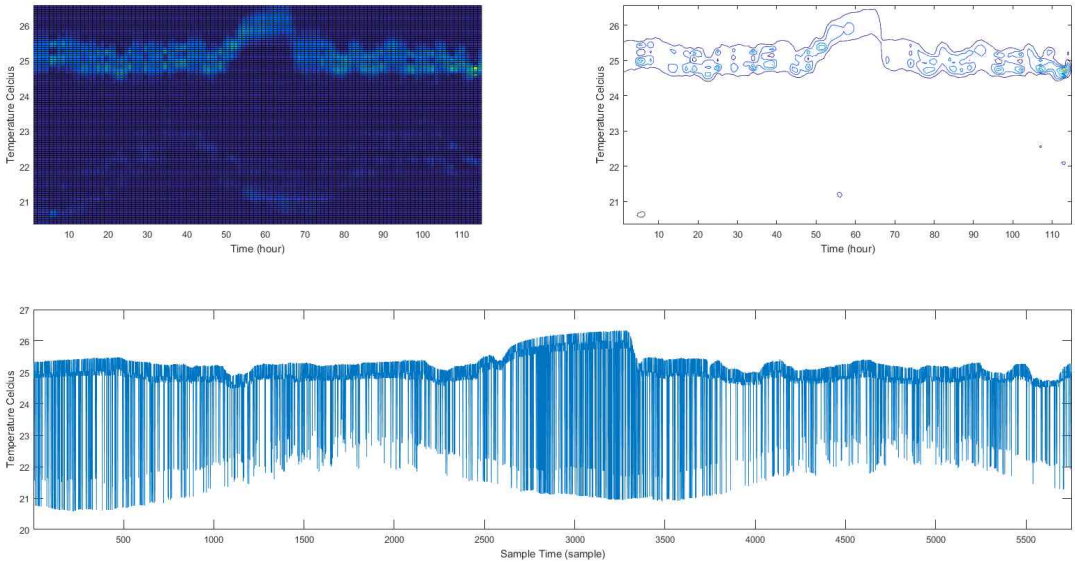
3.2.1 2017년 5월 29일

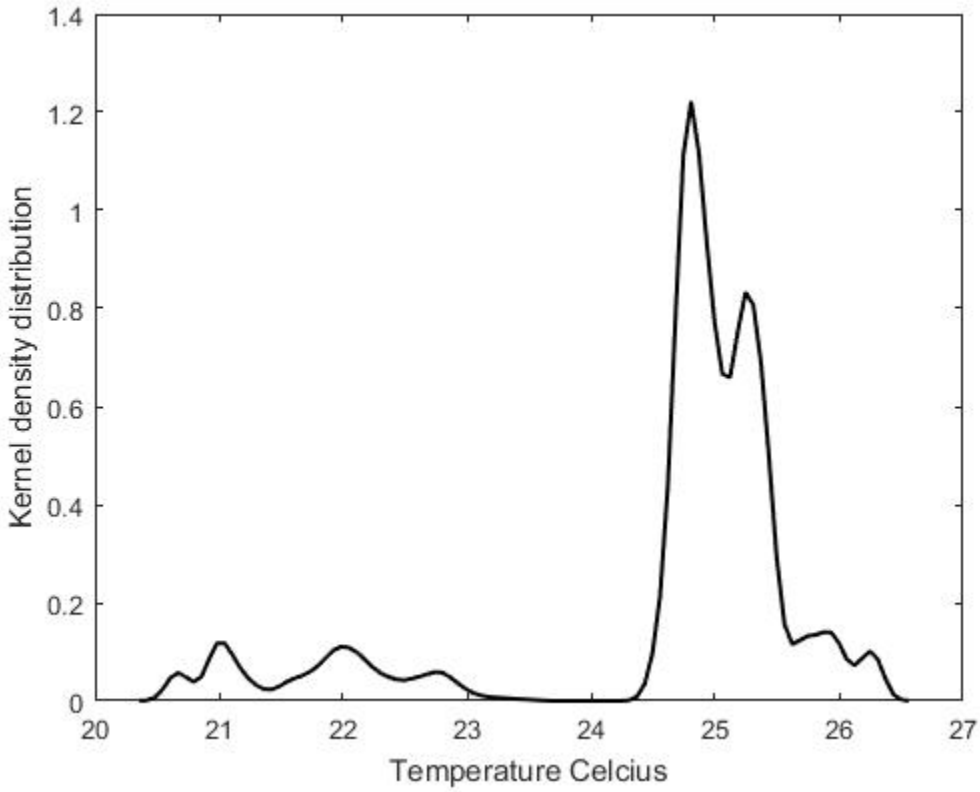
- 센서노드 SHTPI010의 CO값의 KDE추정 및 시간이력 데이터



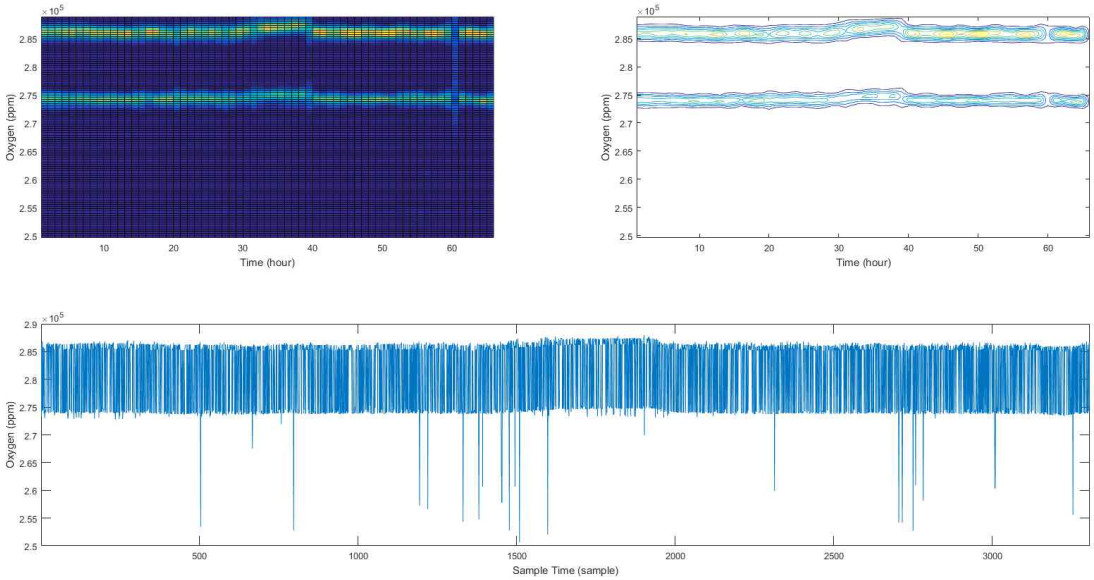


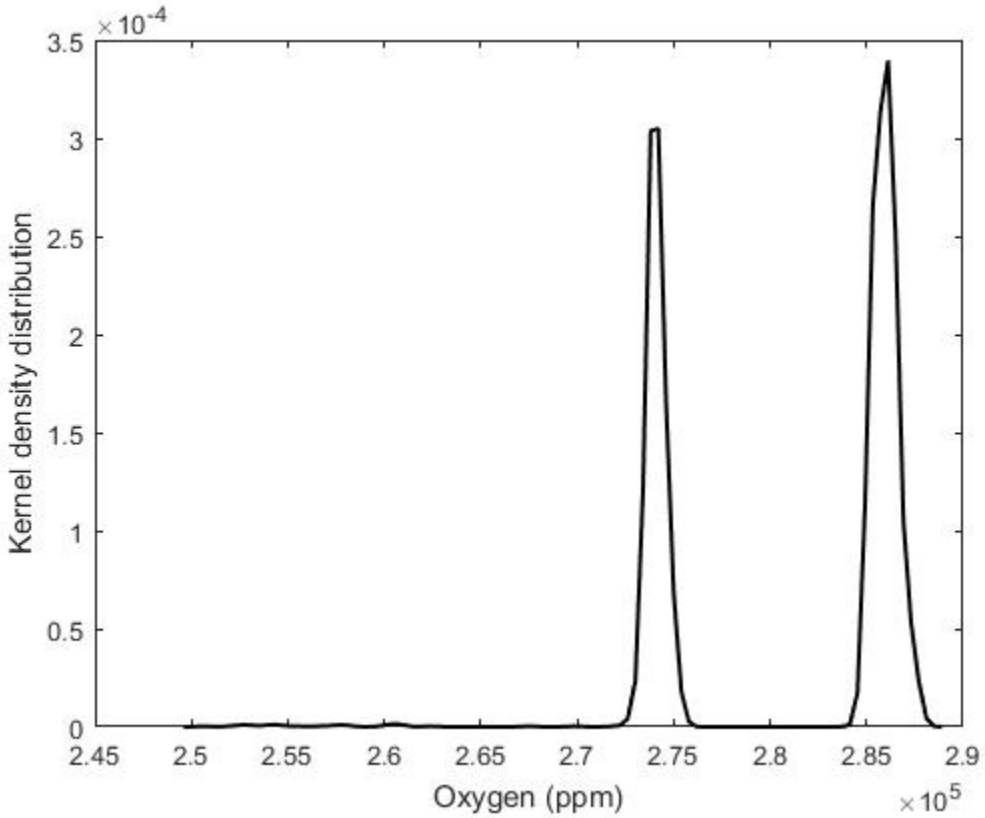
- 센서노드 SHTPI010의 온도값의 KDE추정 및 시간이력 데이터





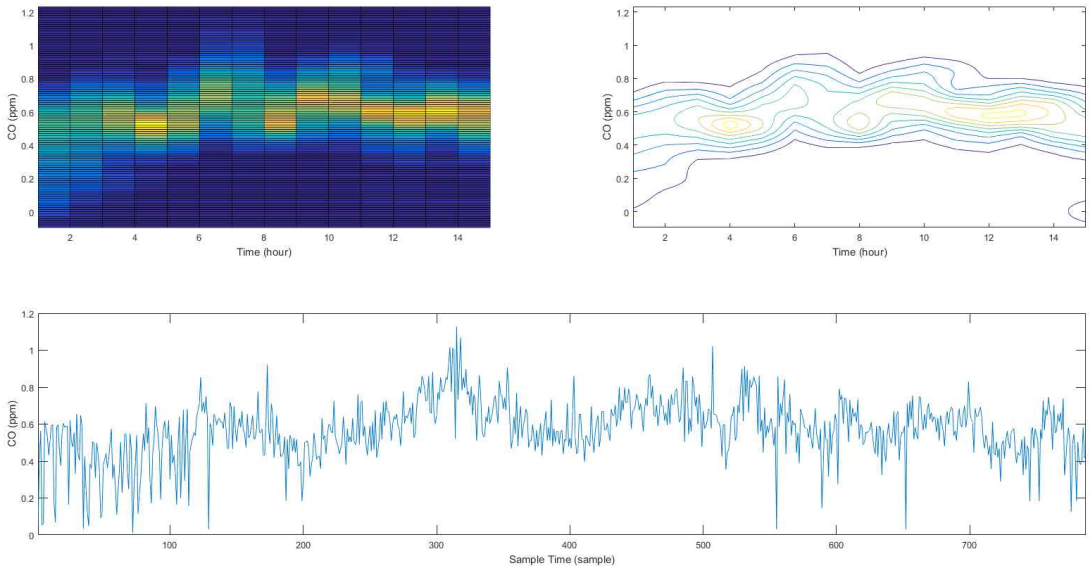
- 센서노드 SHTPI010의 산소농도값의 KDE추정 및 시간이력 데이터

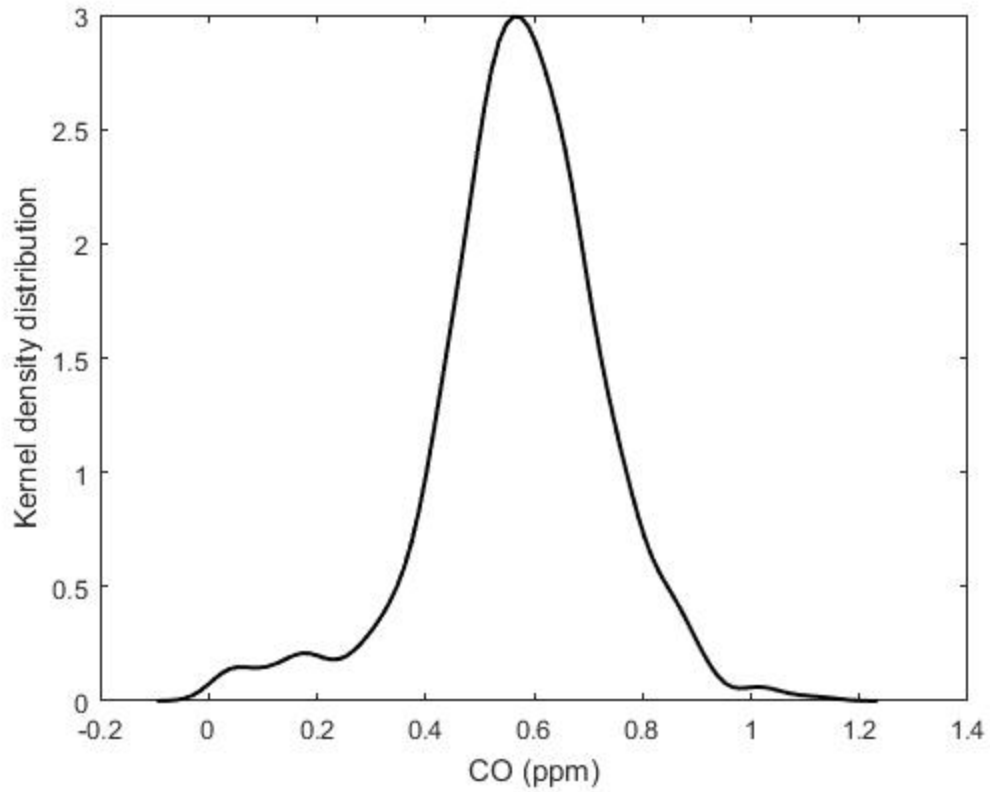




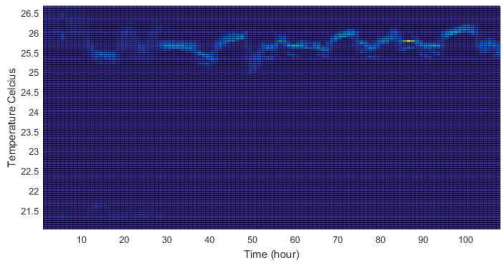
3.2.2 2017년 6월 7일

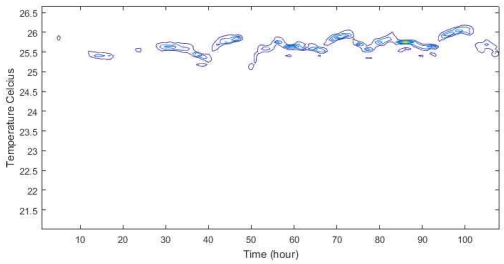
- 센서노드 SHTPI011의 CO농도값의 KDE추정 및 시간이력 데이터

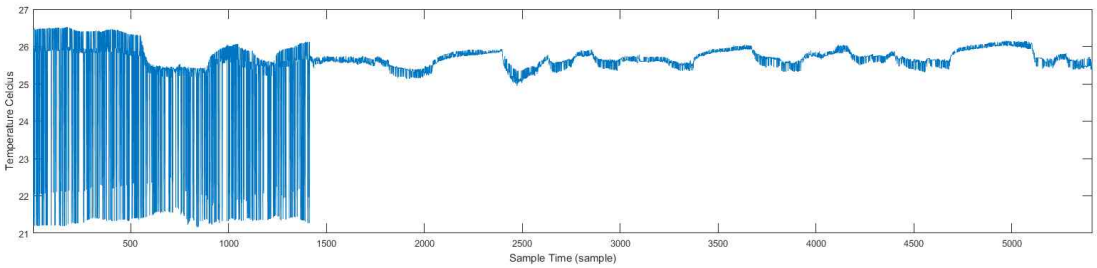


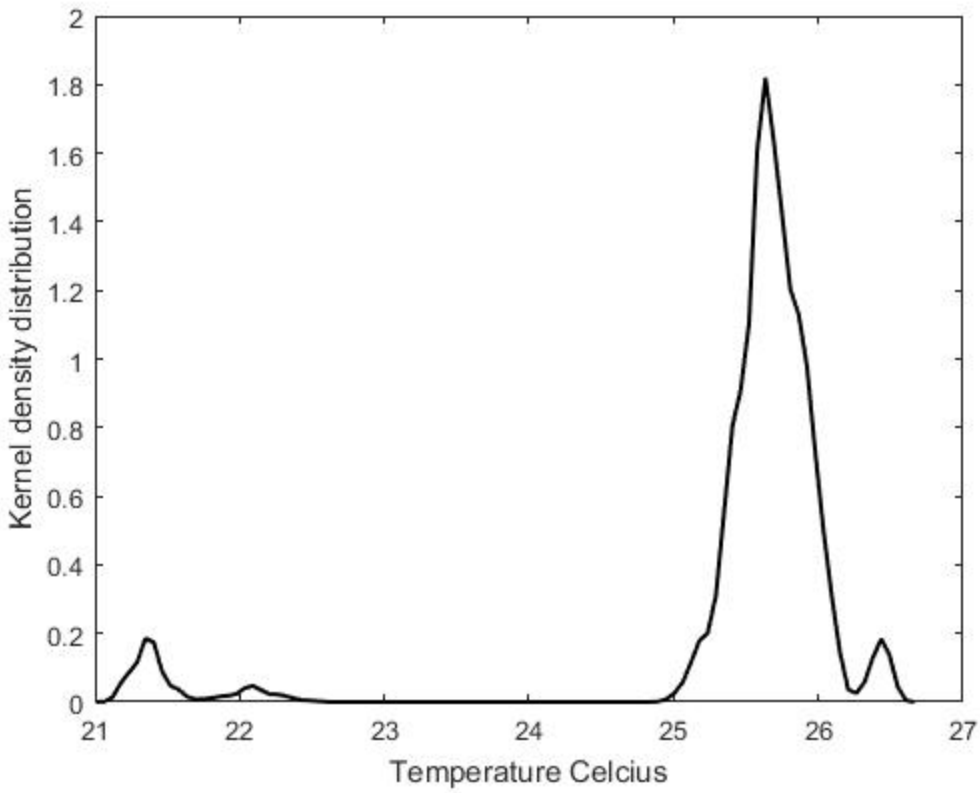


- 센서노드 SHTPI011의 온도값의 KDE추정 및 시간이력 데이터

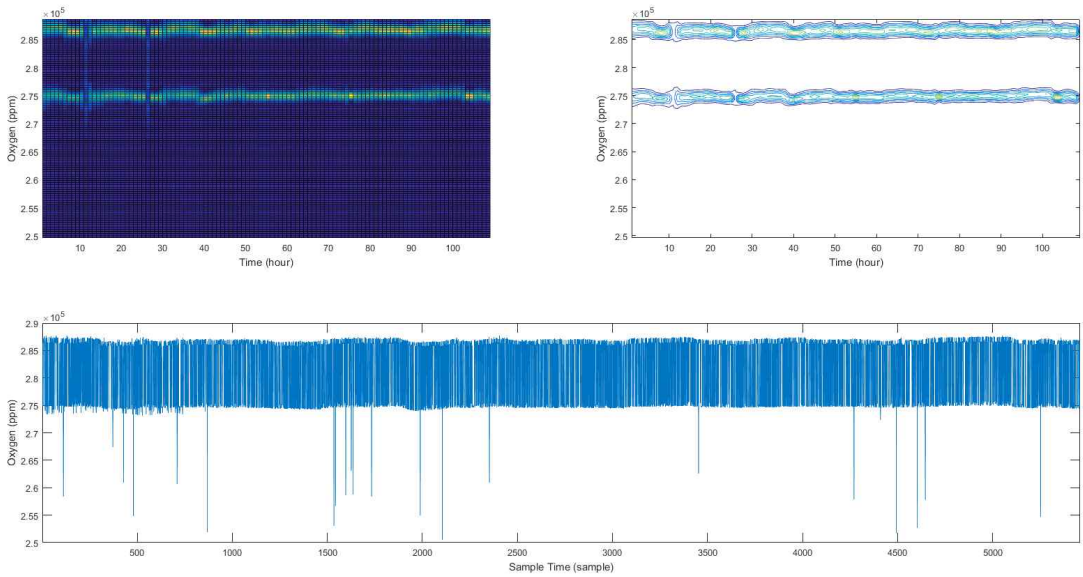


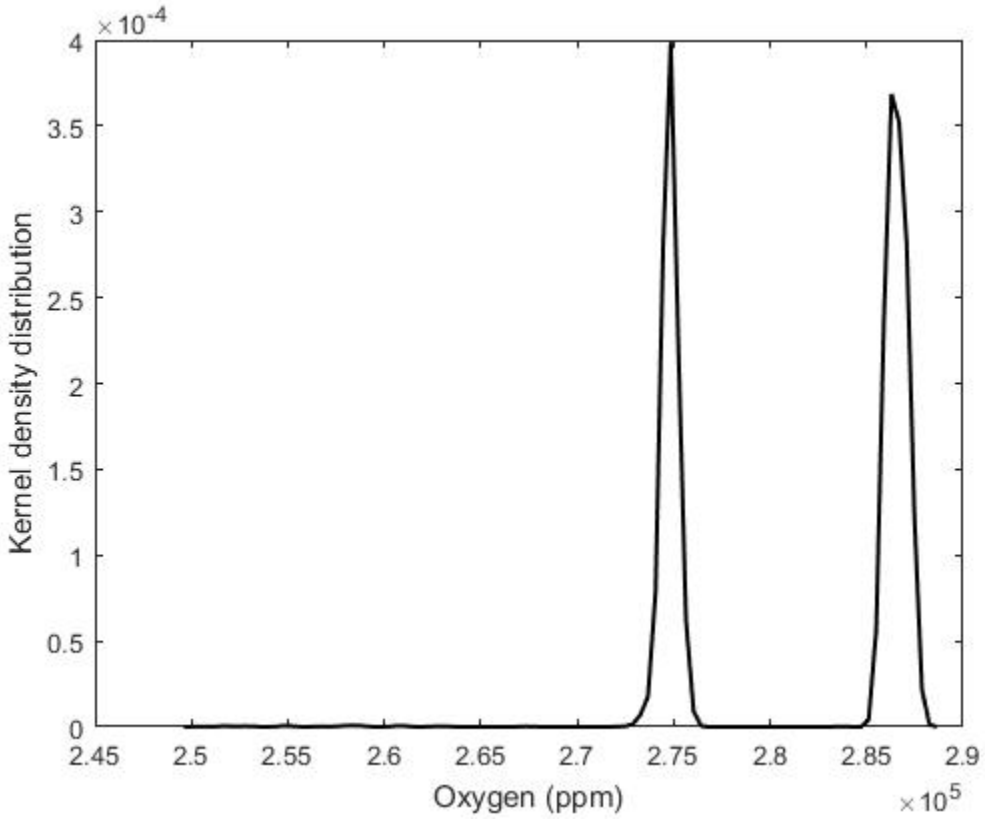




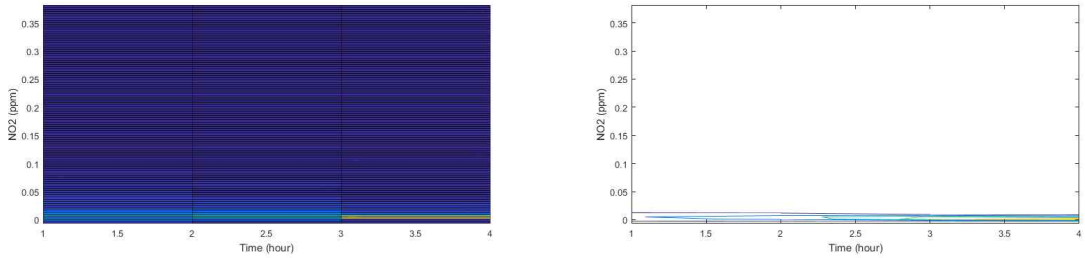


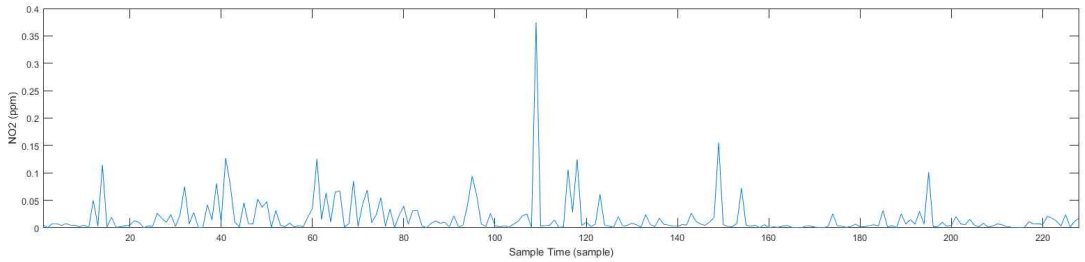
- 센서노드 SHTPI011의 산소농도값의 KDE추정 및 시간이력 데이터

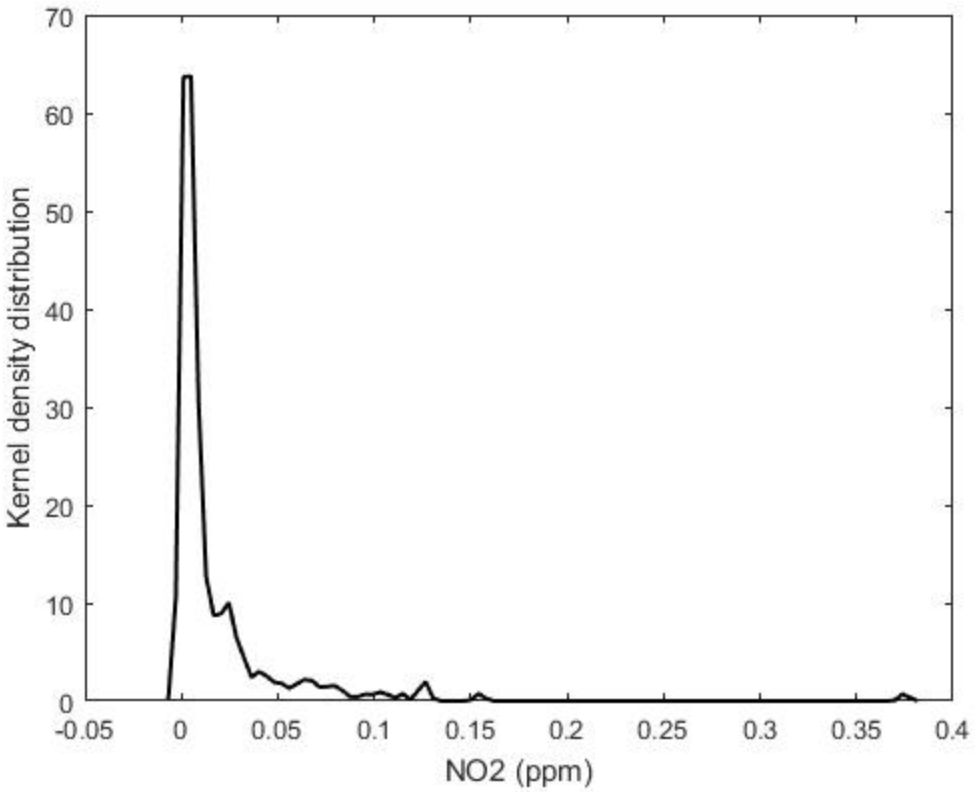




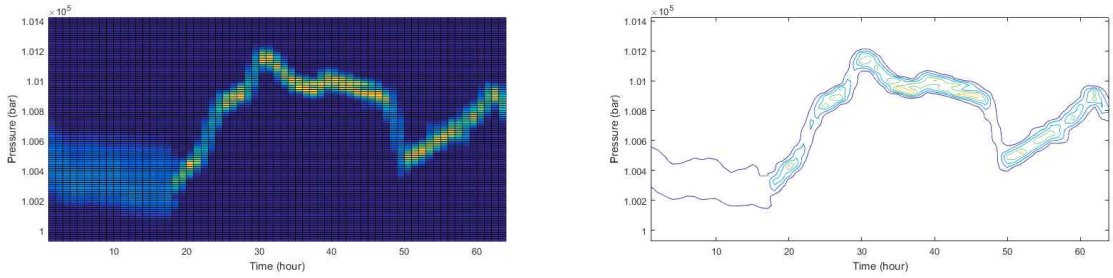
- 센서노드 SHTPI011의 이산화질소값의 KDE추정 및 시간이력 데이터

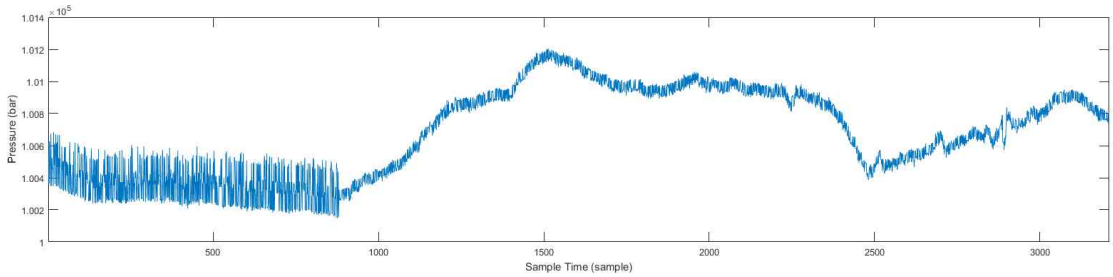


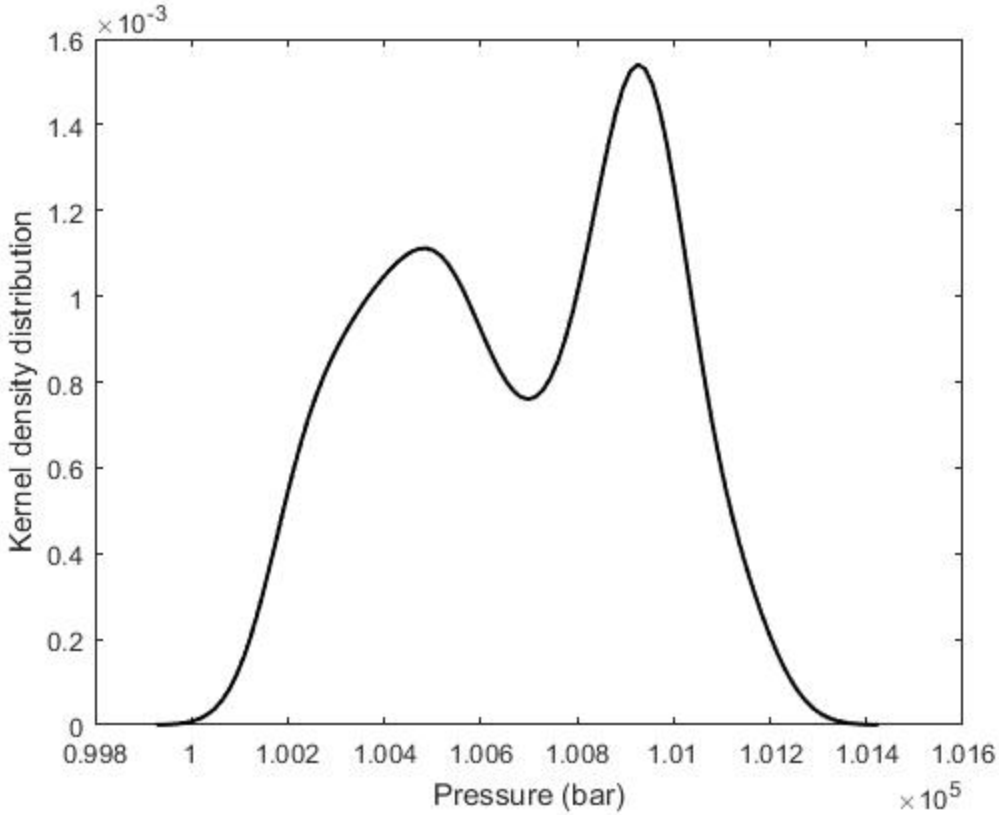




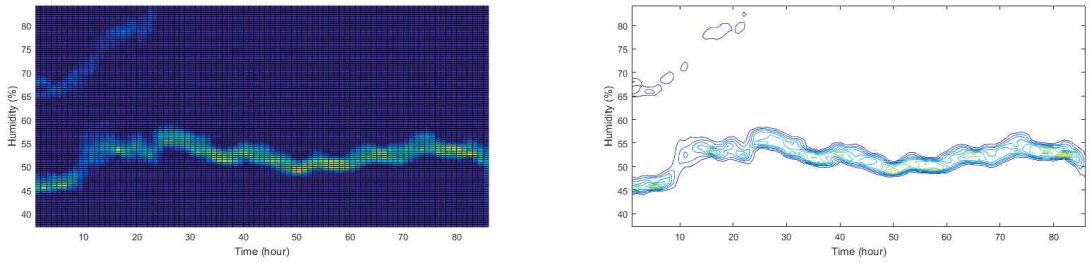
- 센서노드 SHTPI011의 기압값의 KDE추정 및 시간이력 데이터

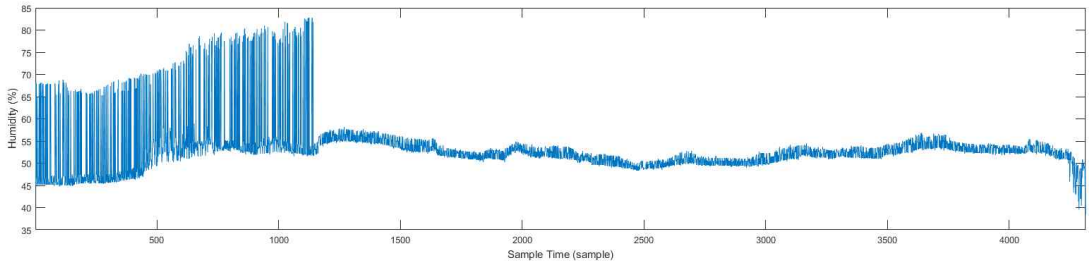


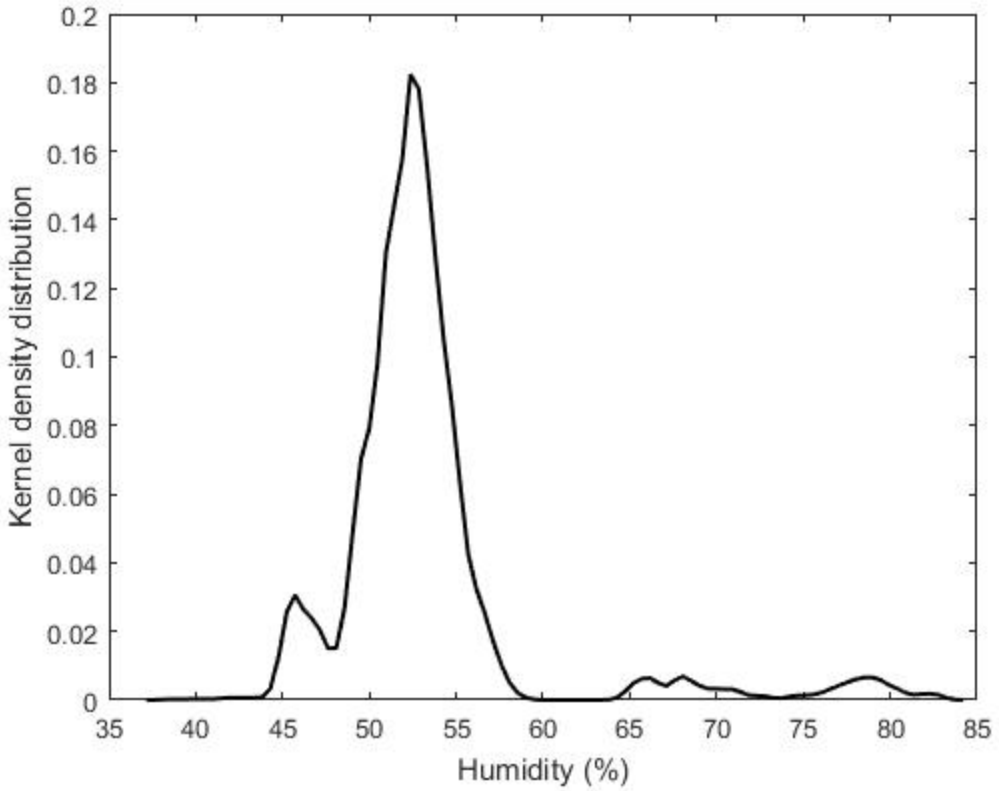




- 센서노드 SHTPI011의 습도값의 KDE추정 및 시간이력 데이터

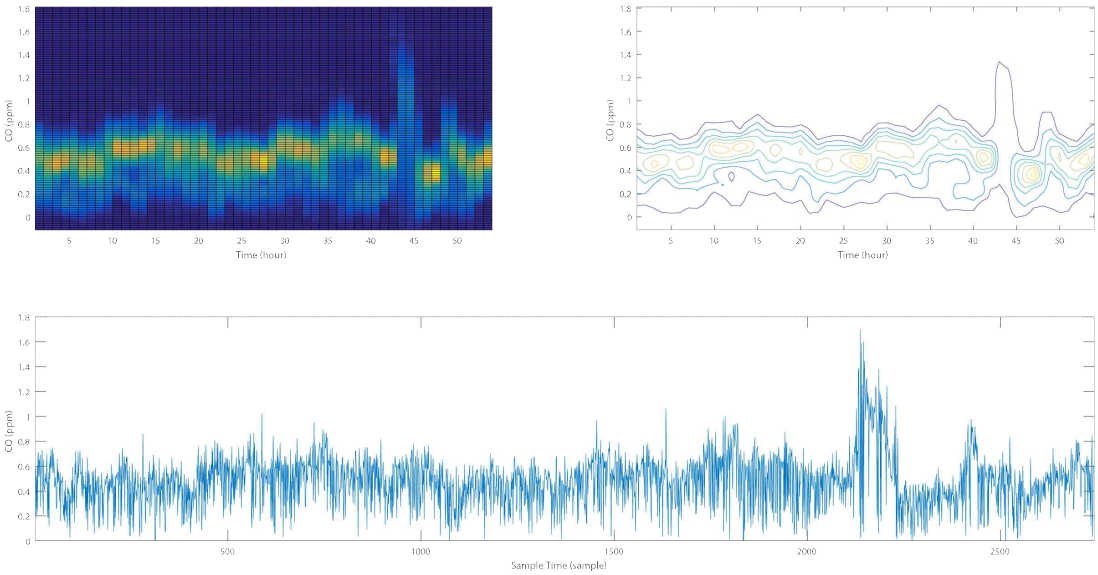


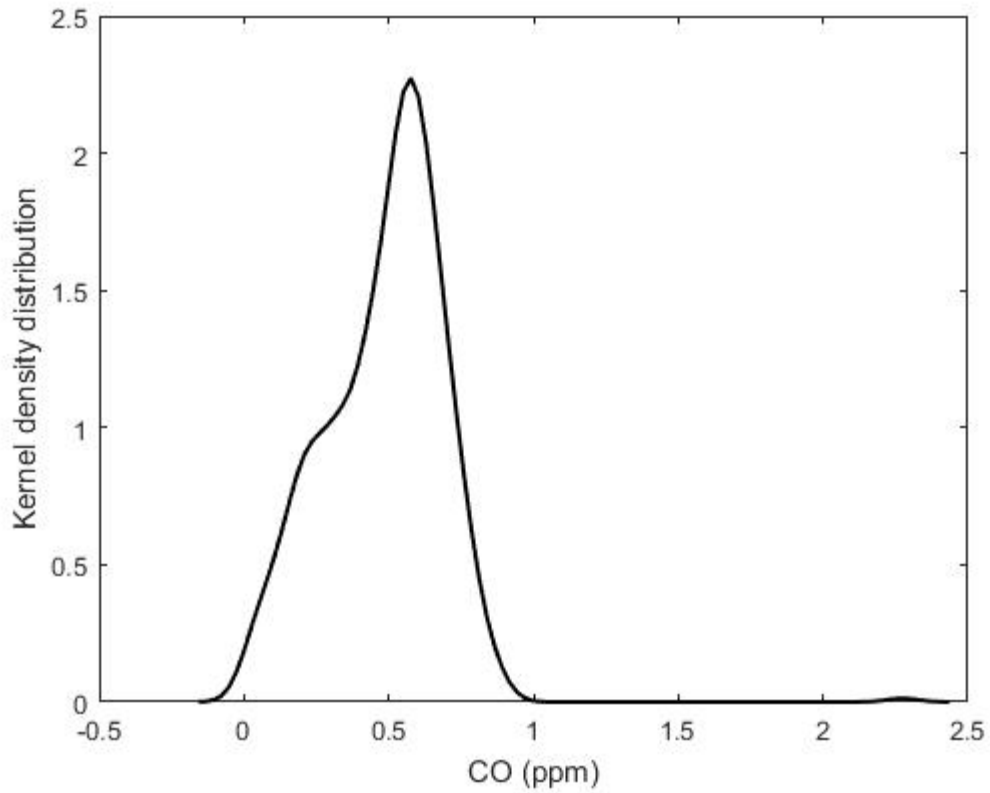




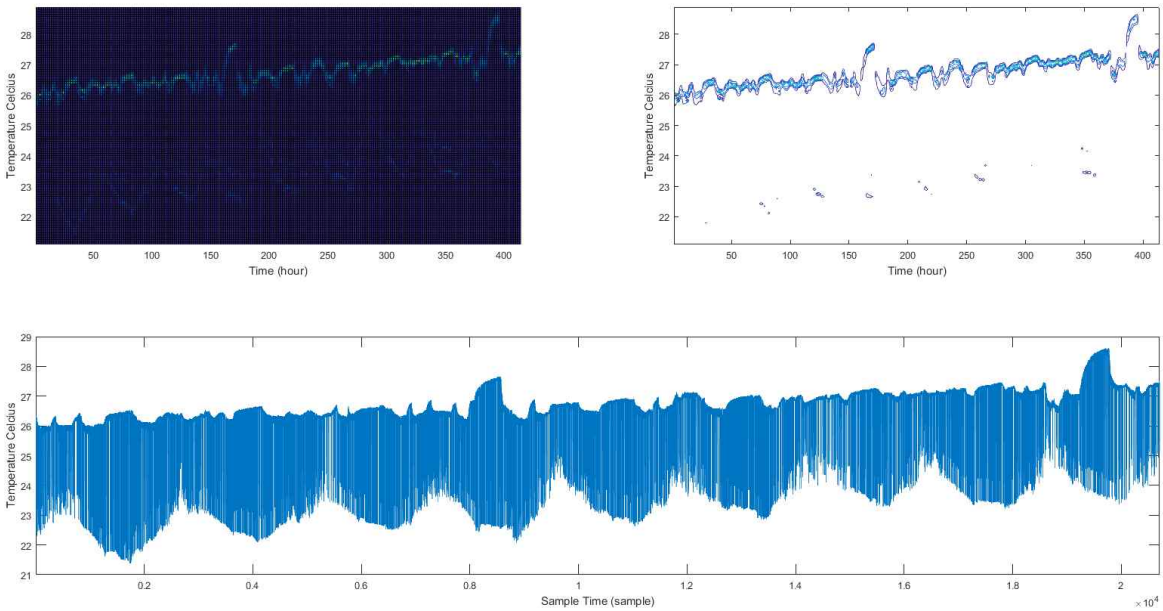
3.2.3 2017년 6월 14일

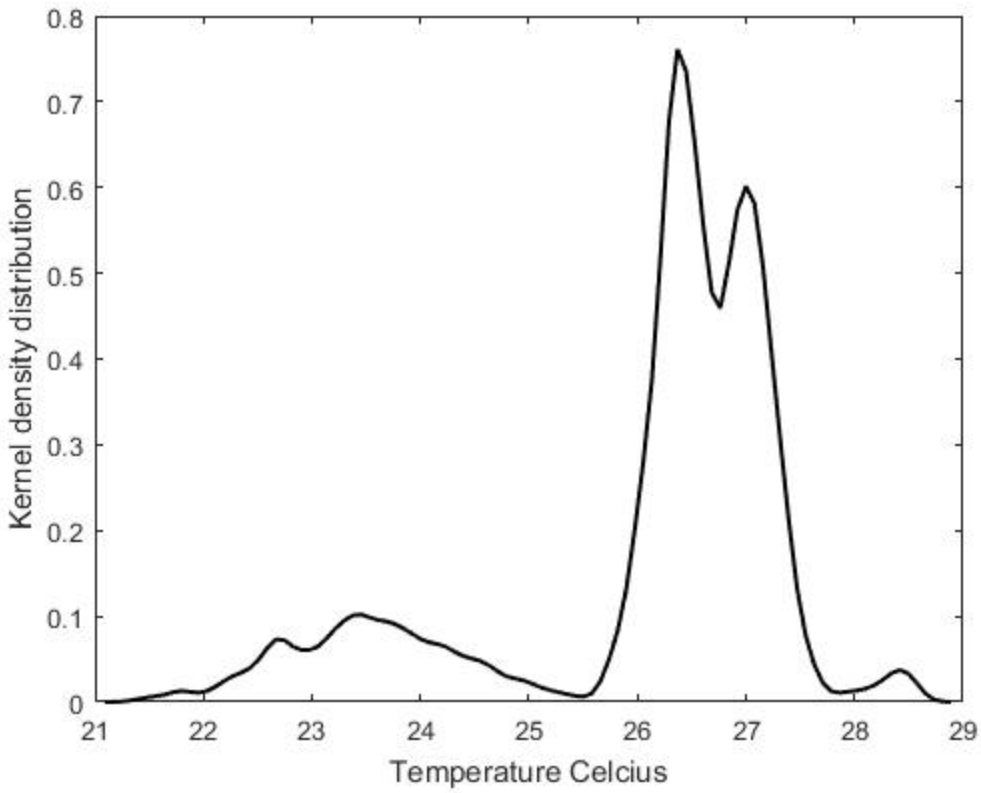
- 센서노드 SHTPI011의 CO값의 KDE추정 및 시간이력 데이터



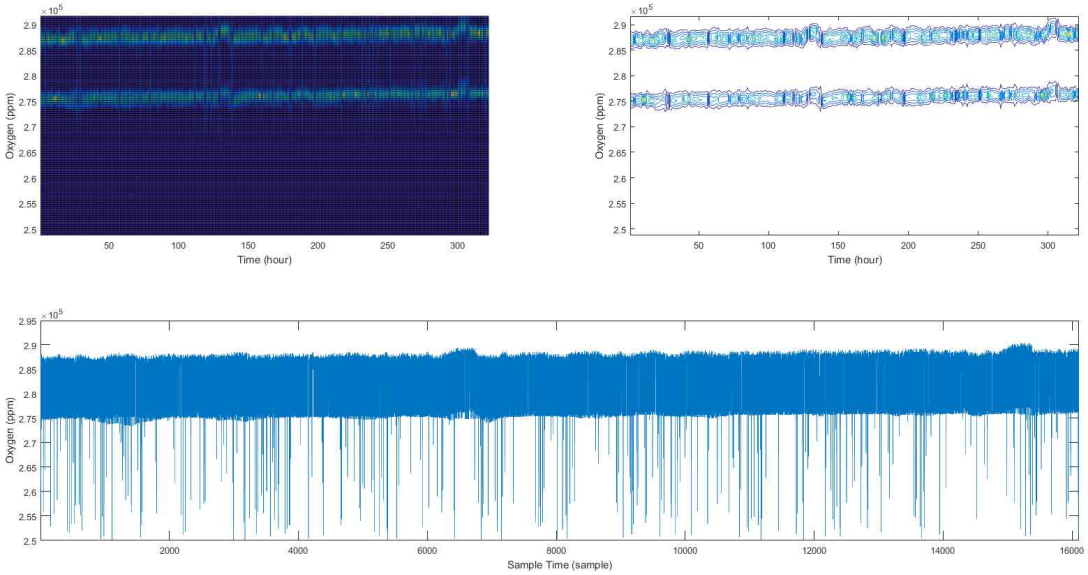


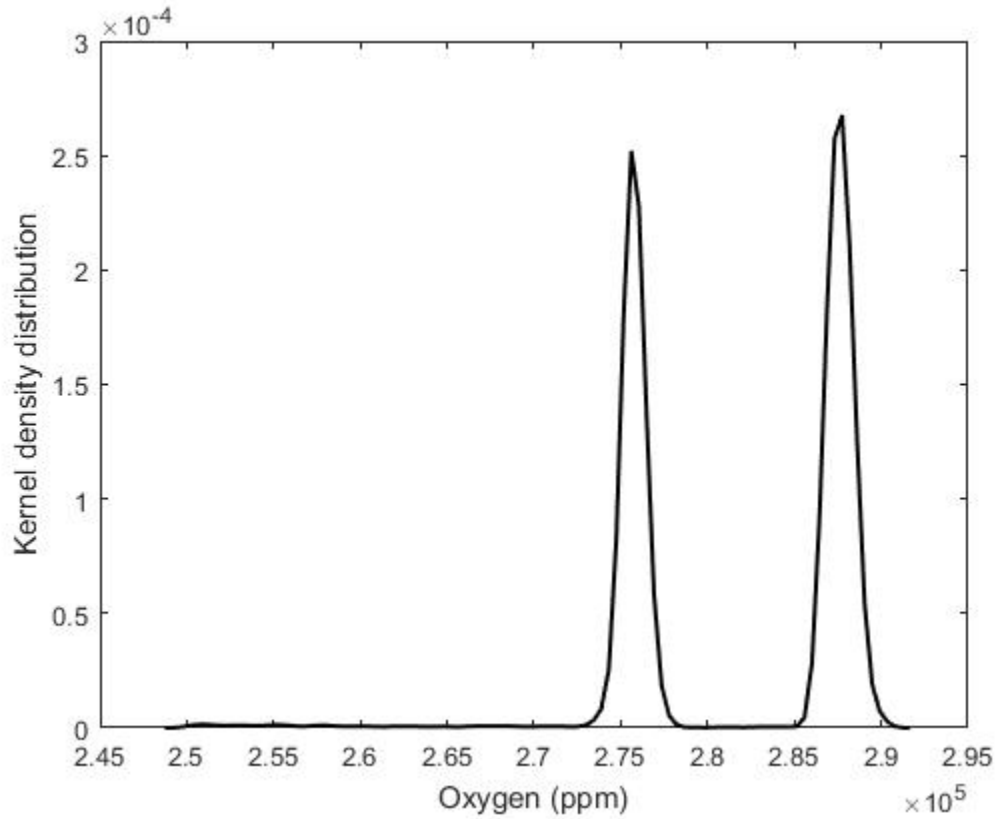
- 센서노드 SHTPI011의 온도값의 KDE추정 및 시간이력 데이터



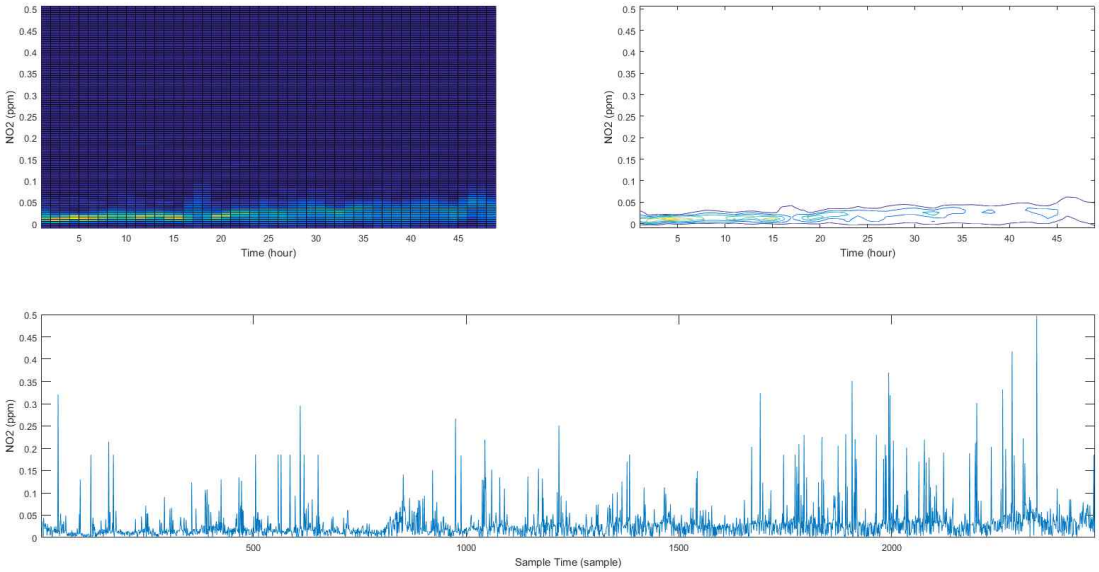


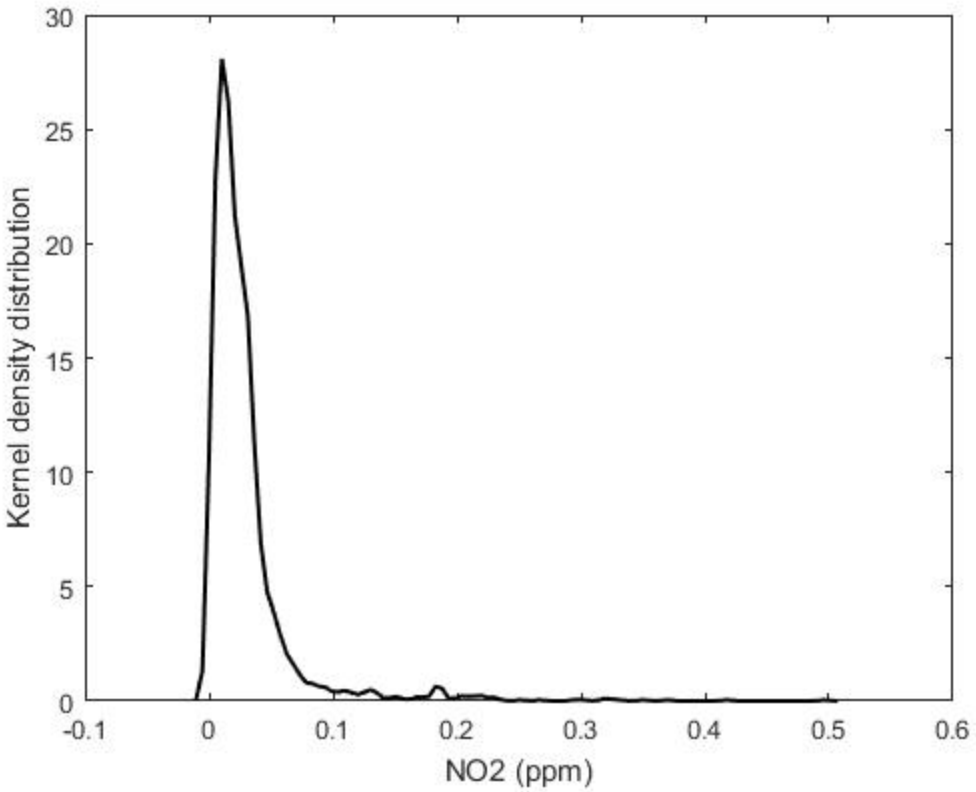
- 센서노드 SHTPI011의 산소농도값의 KDE추정 및 시간이력 데이터



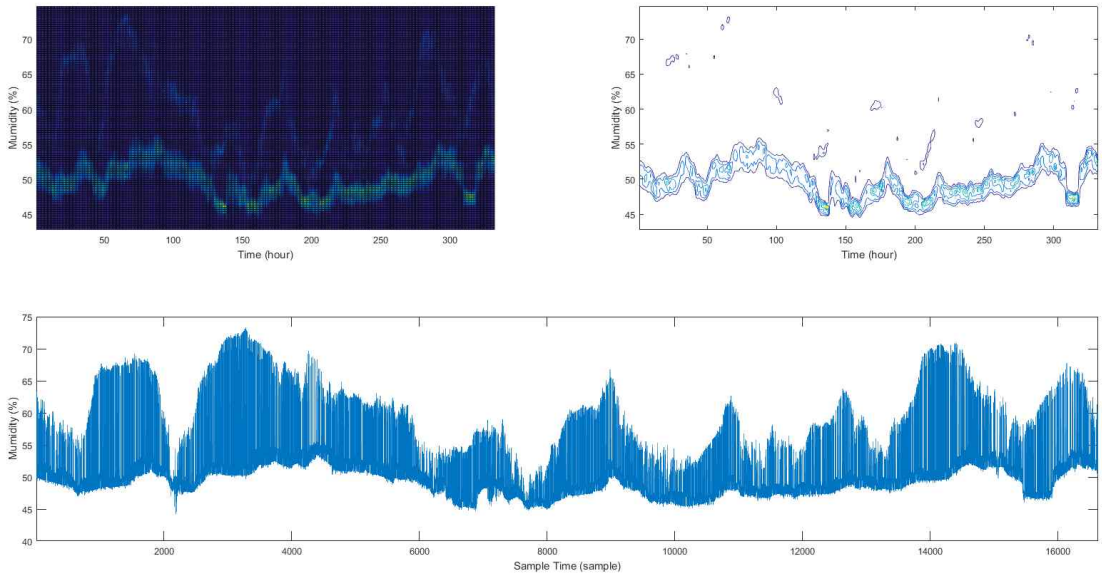


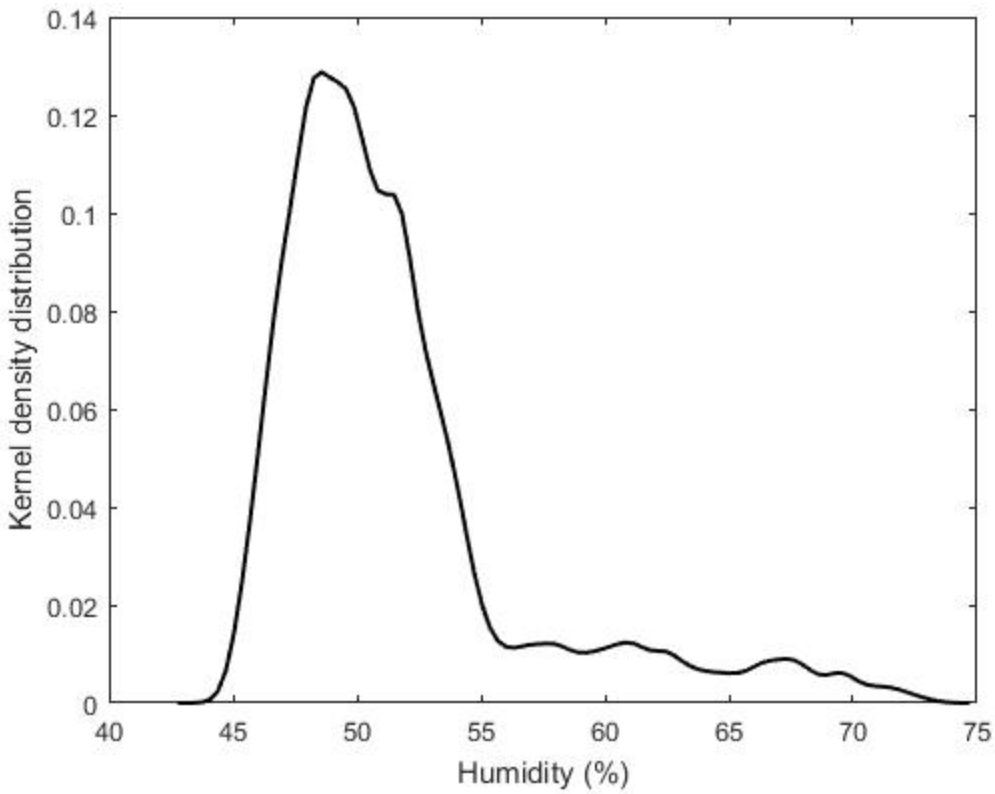
- 센서노드 SHTPI011의 이산화질소 농도값의 KDE추정 및 시간이력 데이터



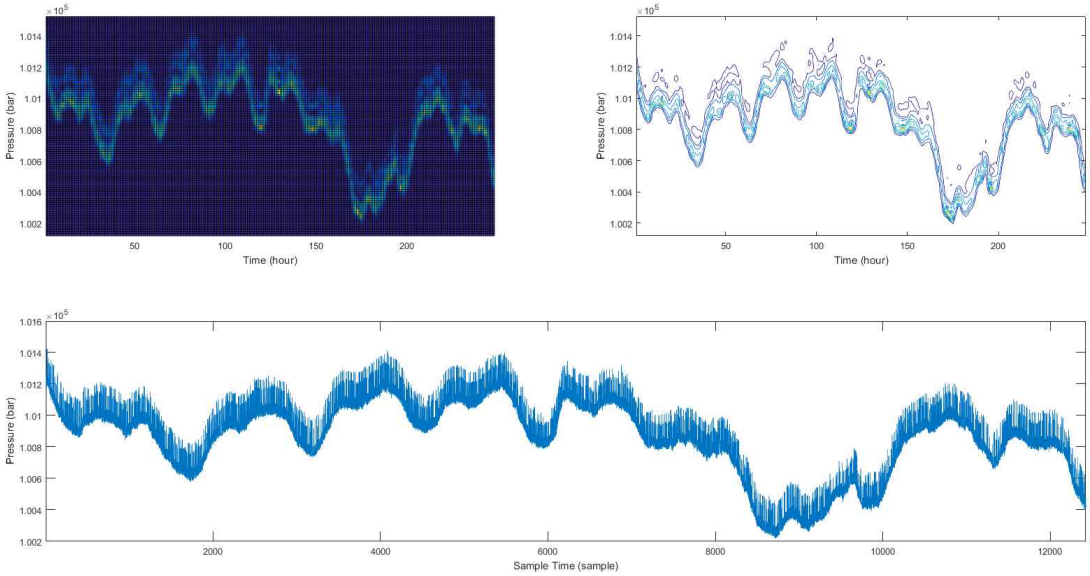


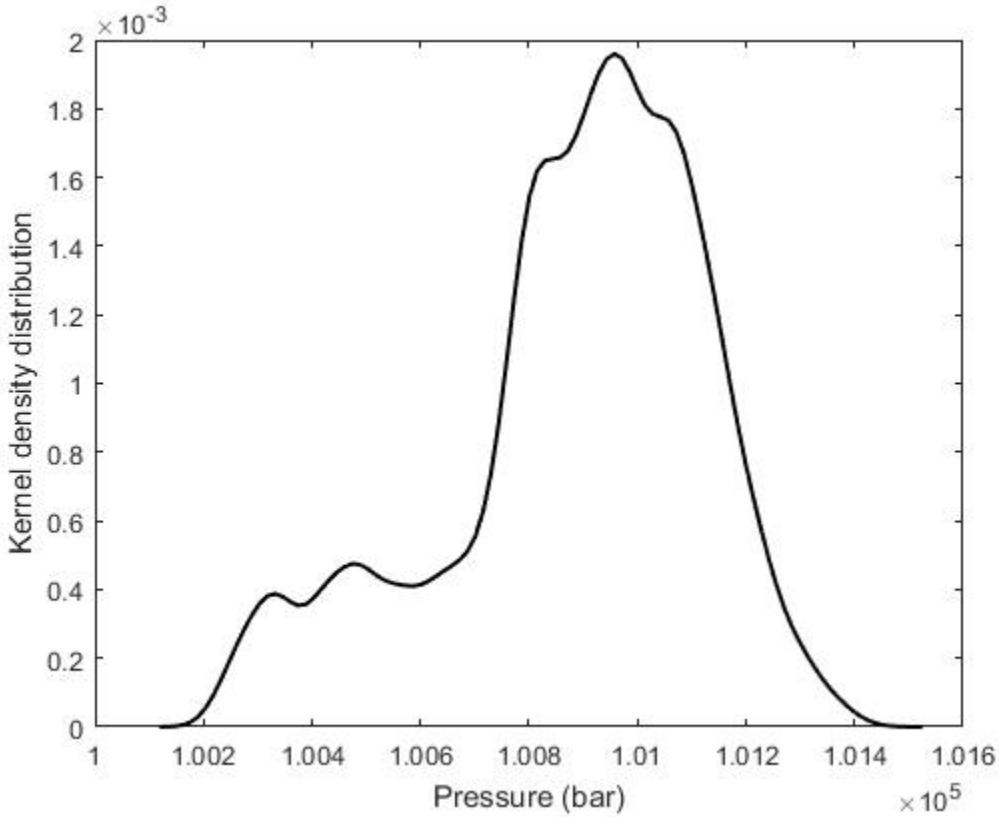
- 센서노드 SHTPI011의 습도값의 KDE추정 및 시간이력 데이터





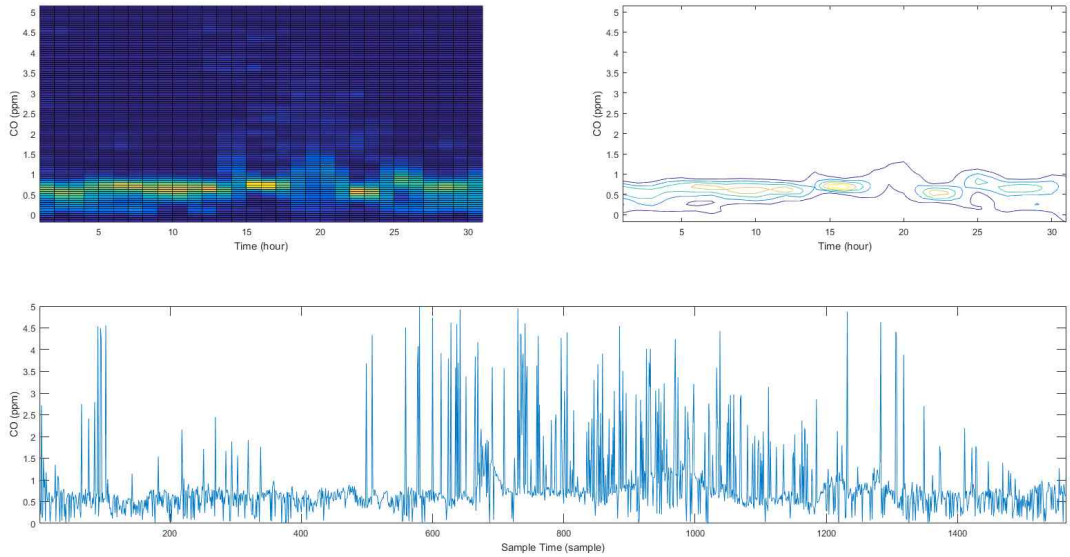
- 센서노드 SHTPI011의 압력값의 KDE추정 및 시간이력 데이터

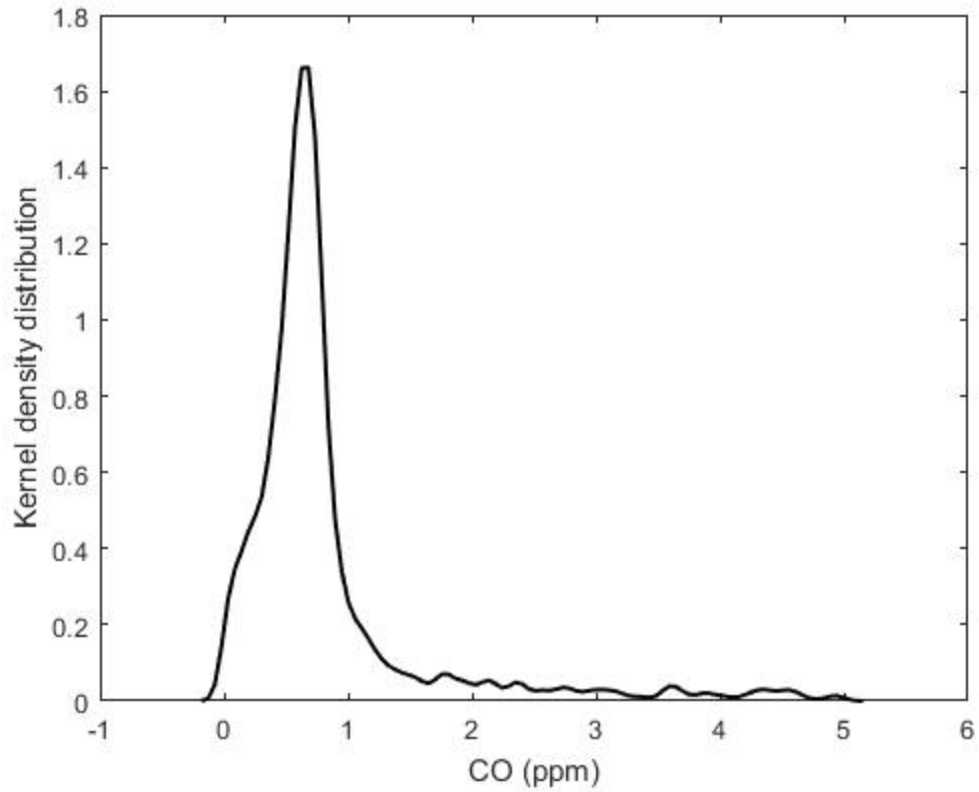




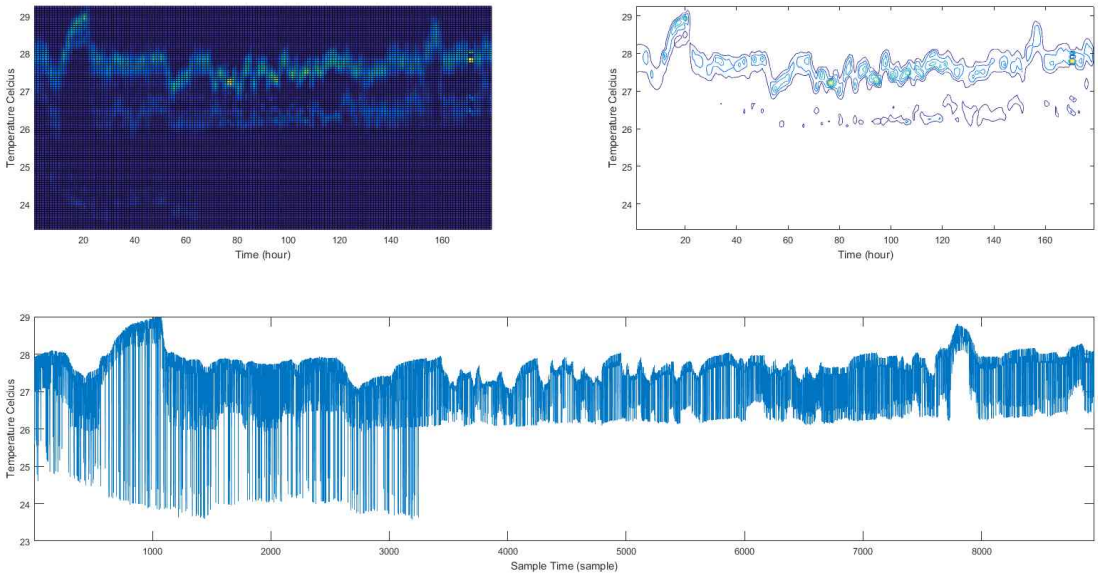
3.2.4 2017년 6월 23일

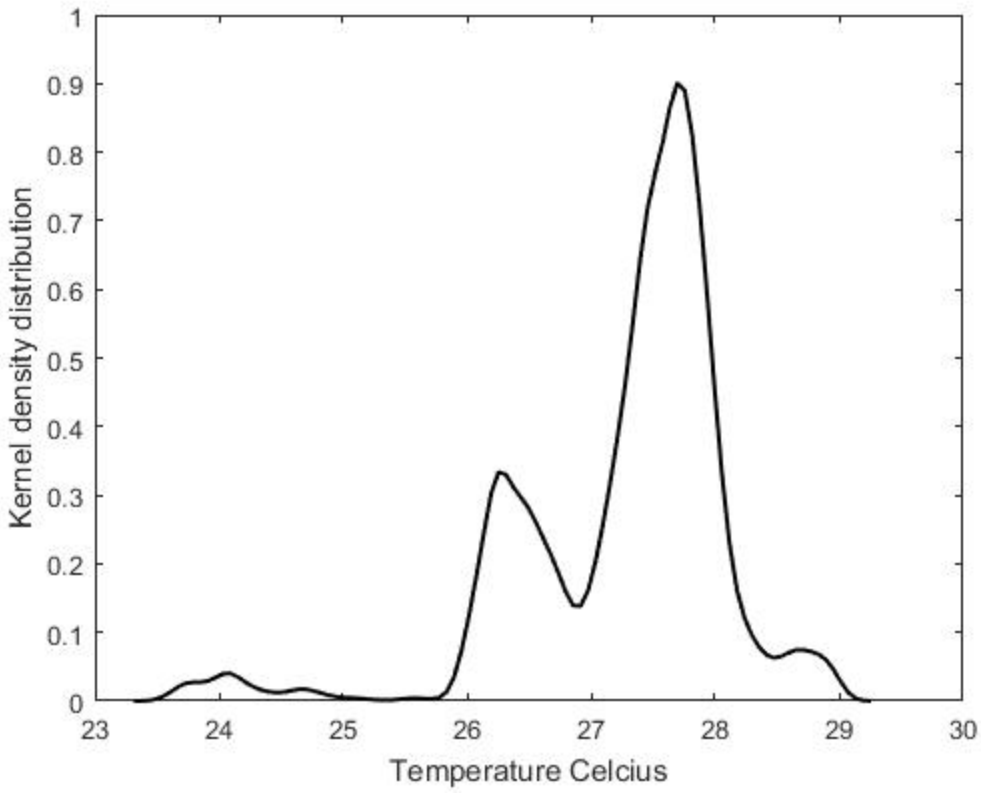
- 센서노드 SHTPI007의 CO값의 KDE추정 및 시간이력 데이터



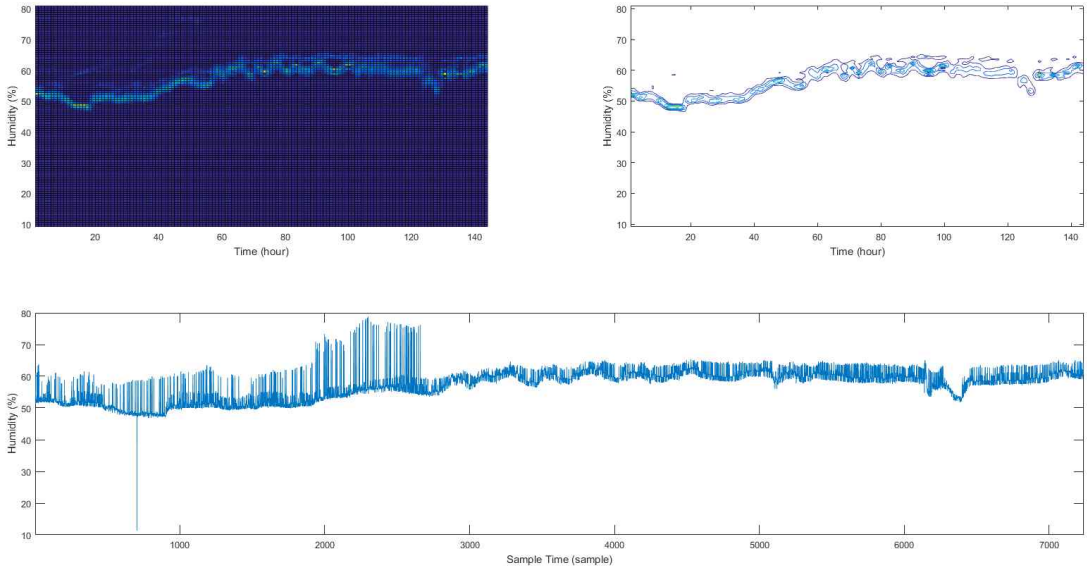


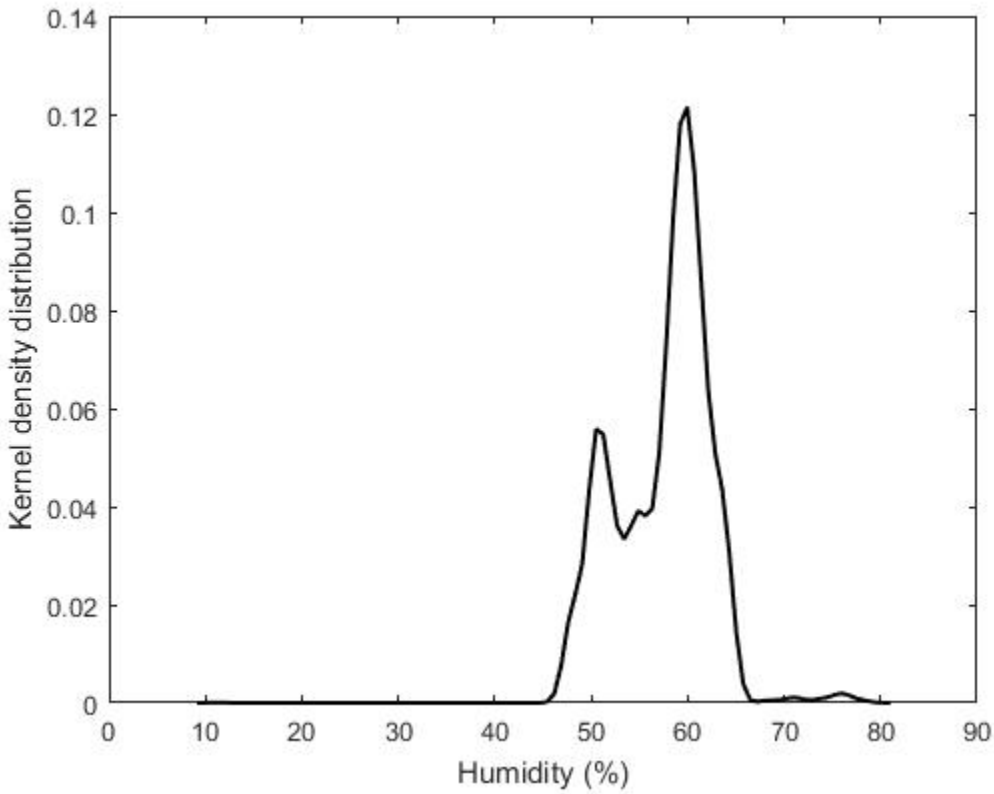
- 센서노드 SHTPI007의 온도값의 KDE추정 및 시간이력 데이터



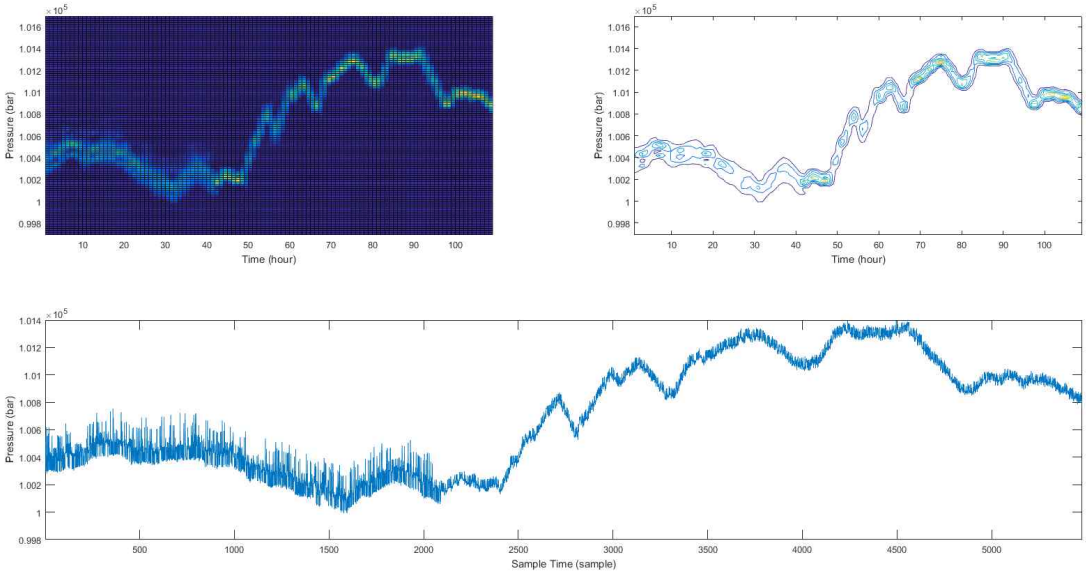


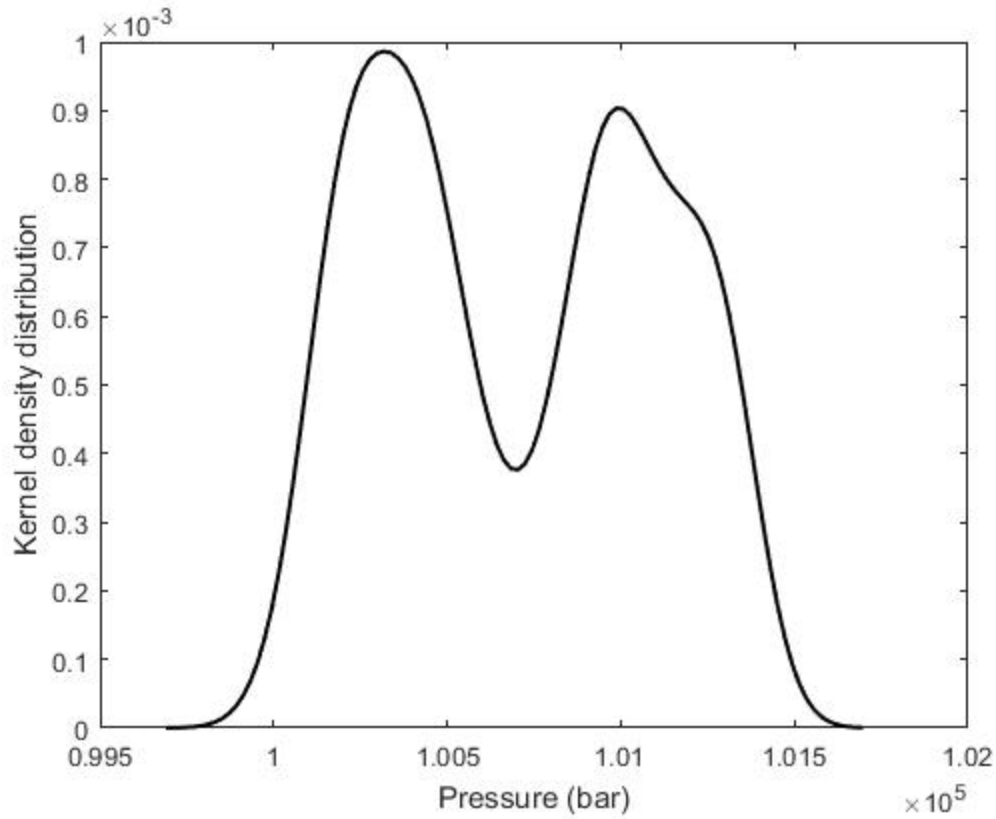
- 센서노드 SHTPI007의 습도값의 KDE추정 및 시간이력 데이터



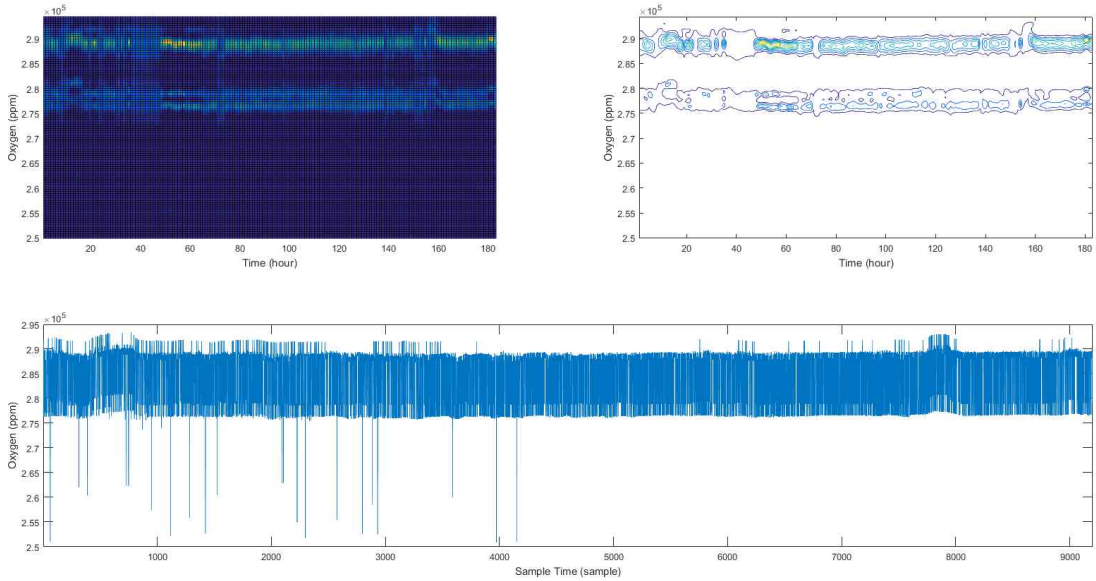


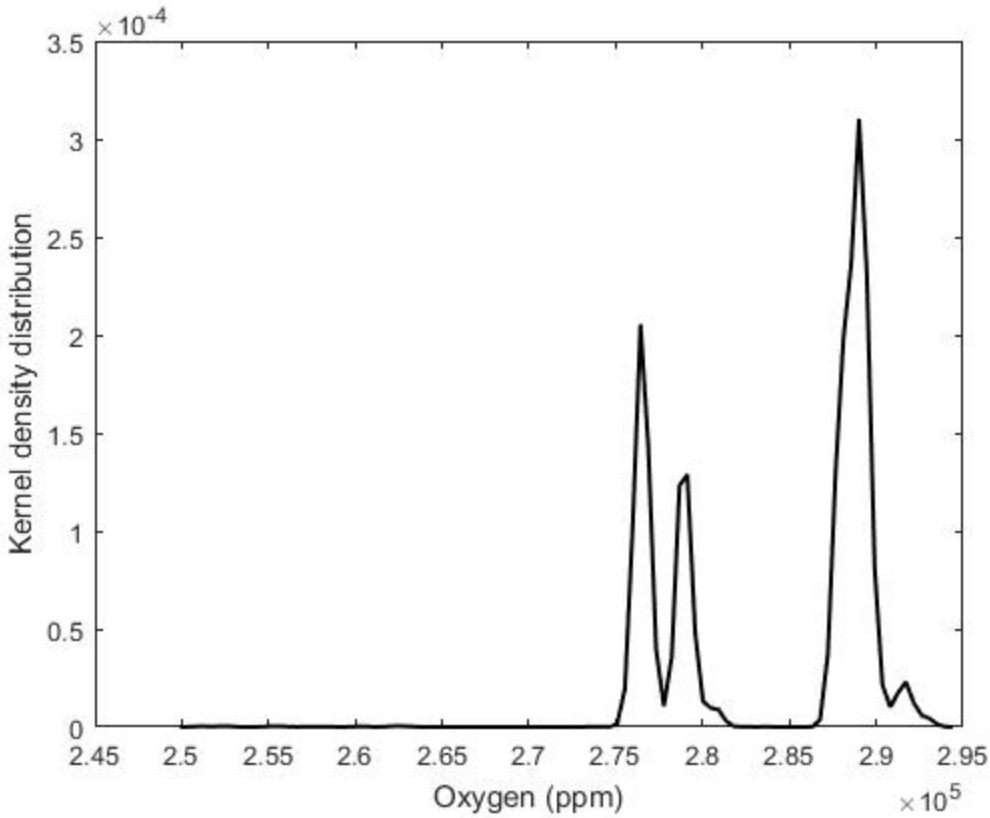
- 센서노드 SHTPI007의 압력값의 KDE추정 및 시간이력 데이터



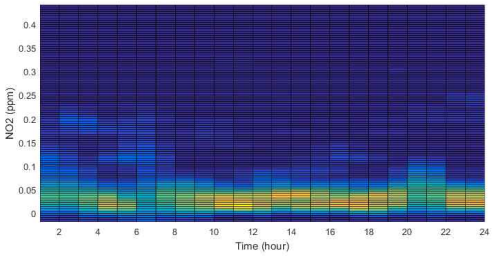


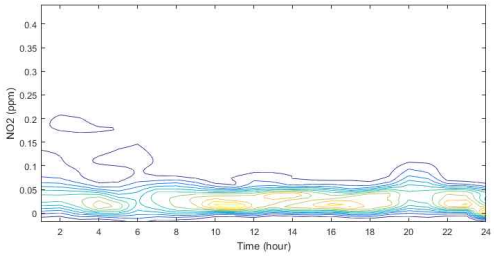
- 센서노드 SHTPI007의 산소 농도값의 KDE추정 및 시간이력 데이터

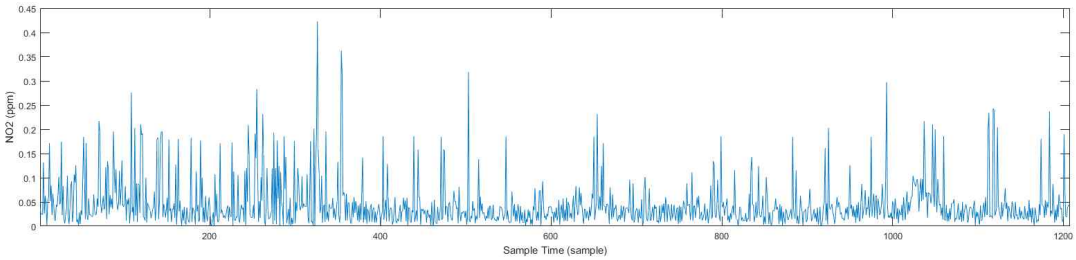


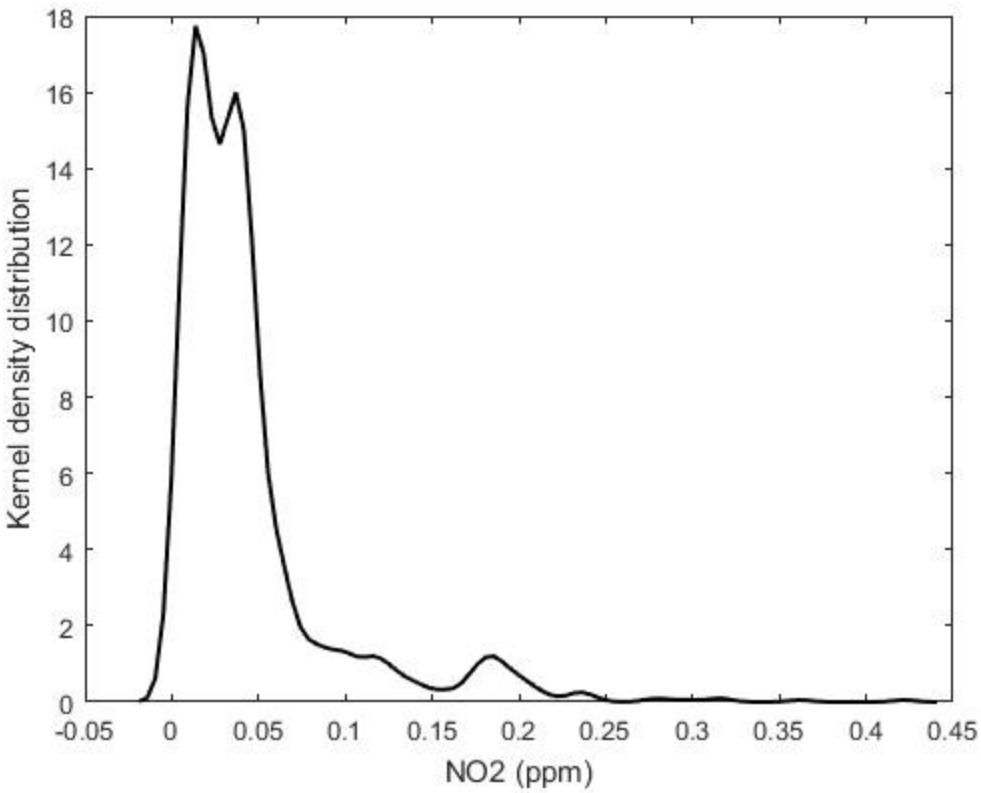


- 센서노드 SHTPI007의 이산화질소 농도값의 KDE추정 및 시간이력 데이터



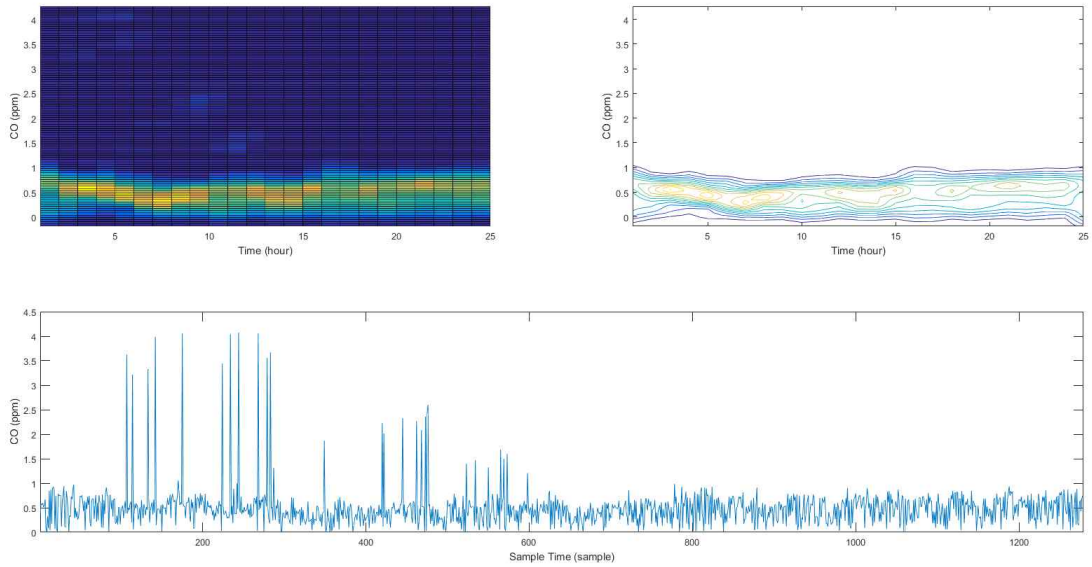


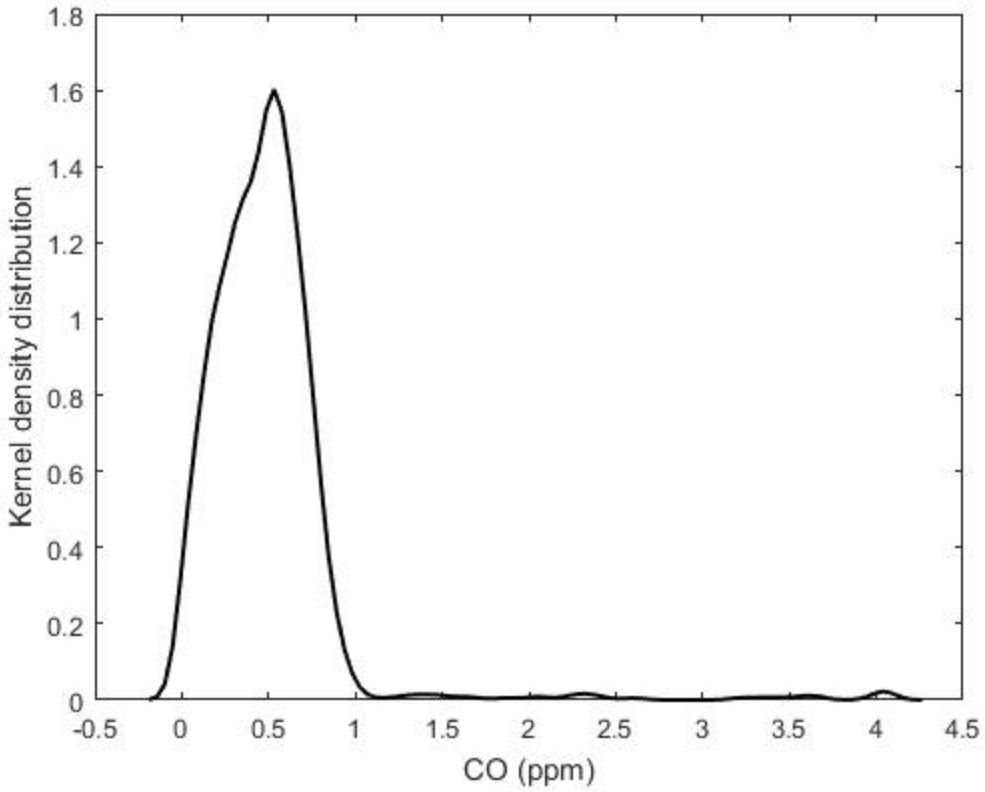




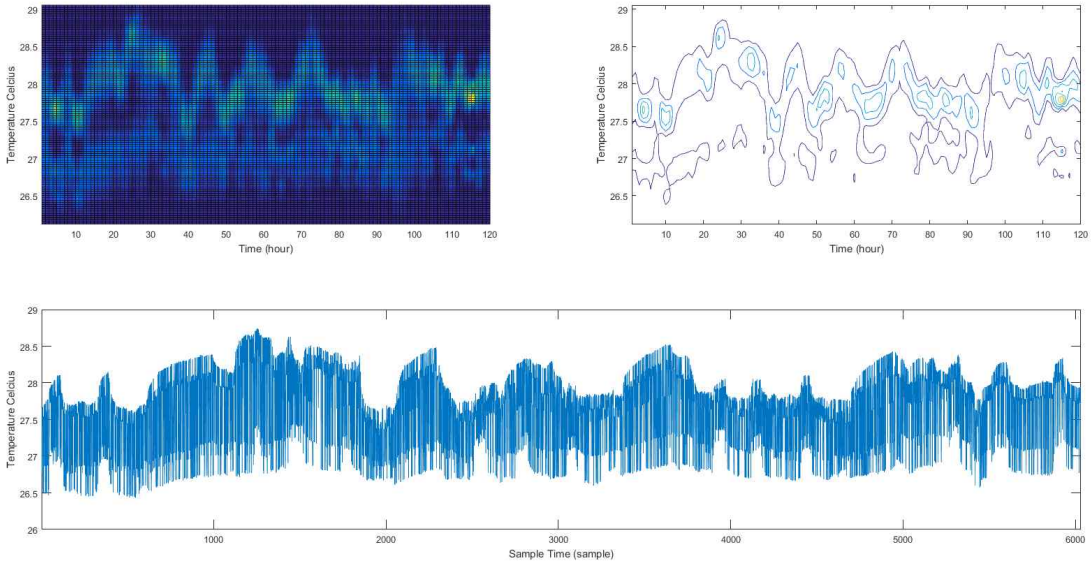
3.2.5 2017년 7월 1일

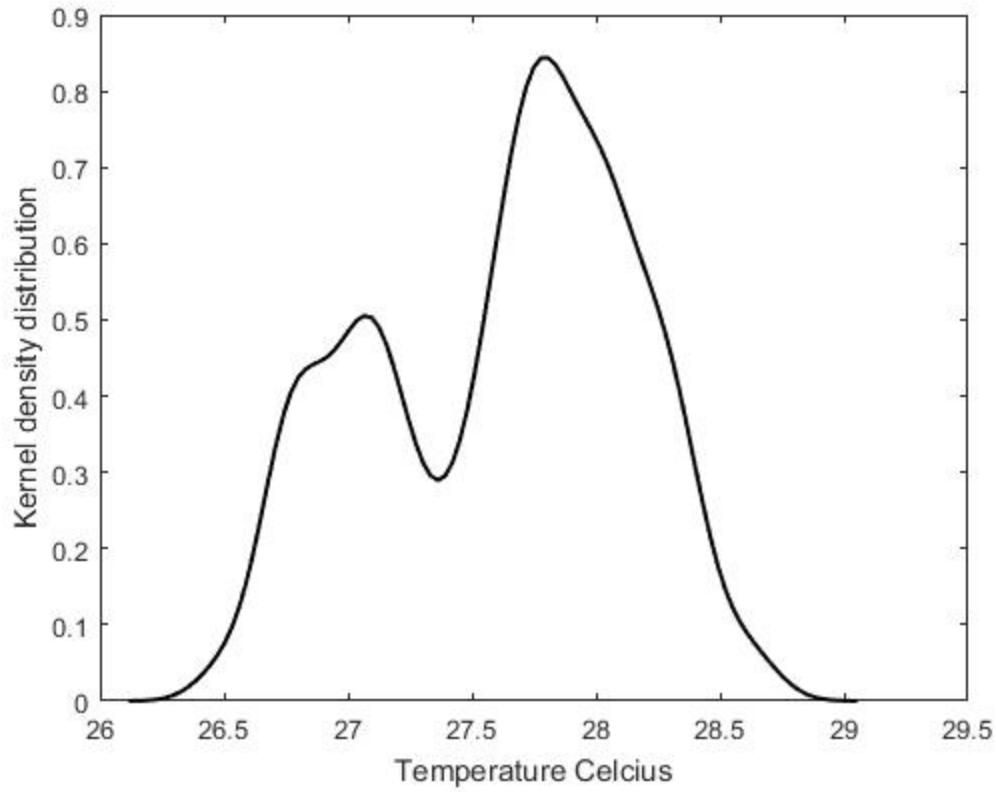
- 센서노드 SHTPI004의 CO값의 KDE추정 및 시간이력 데이터



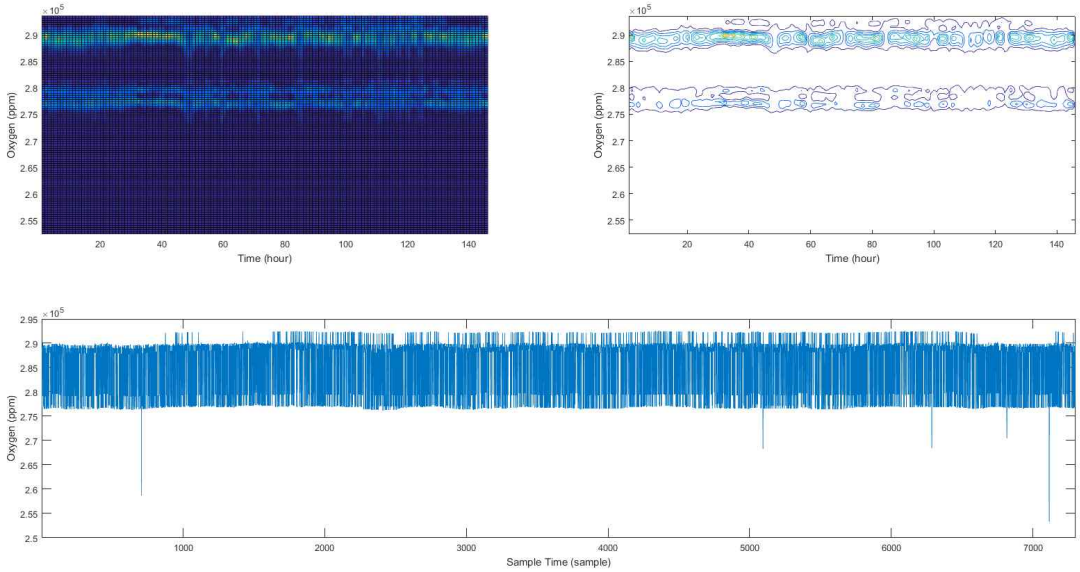


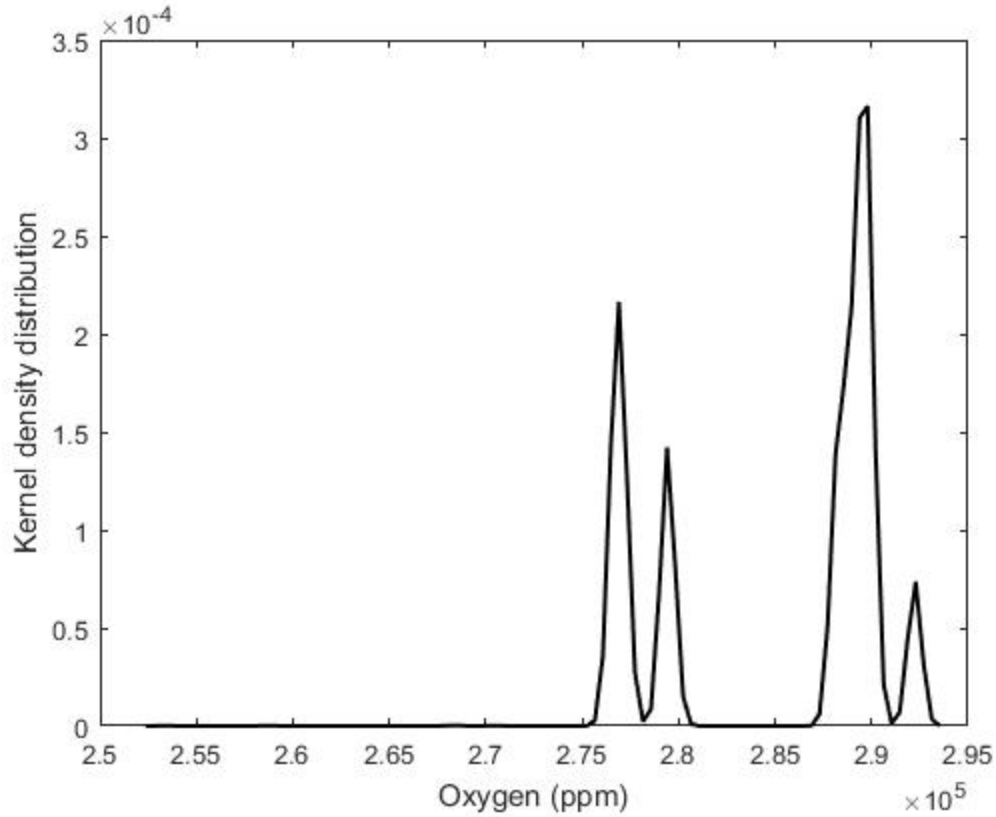
- 센서노드 SHTPI004의 온도값의 KDE추정 및 시간이력 데이터



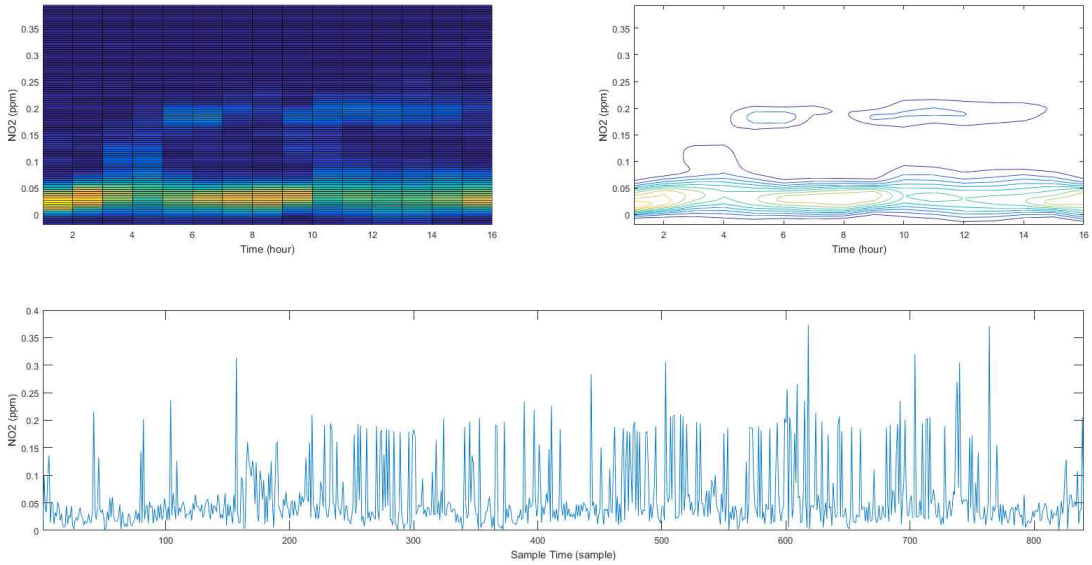


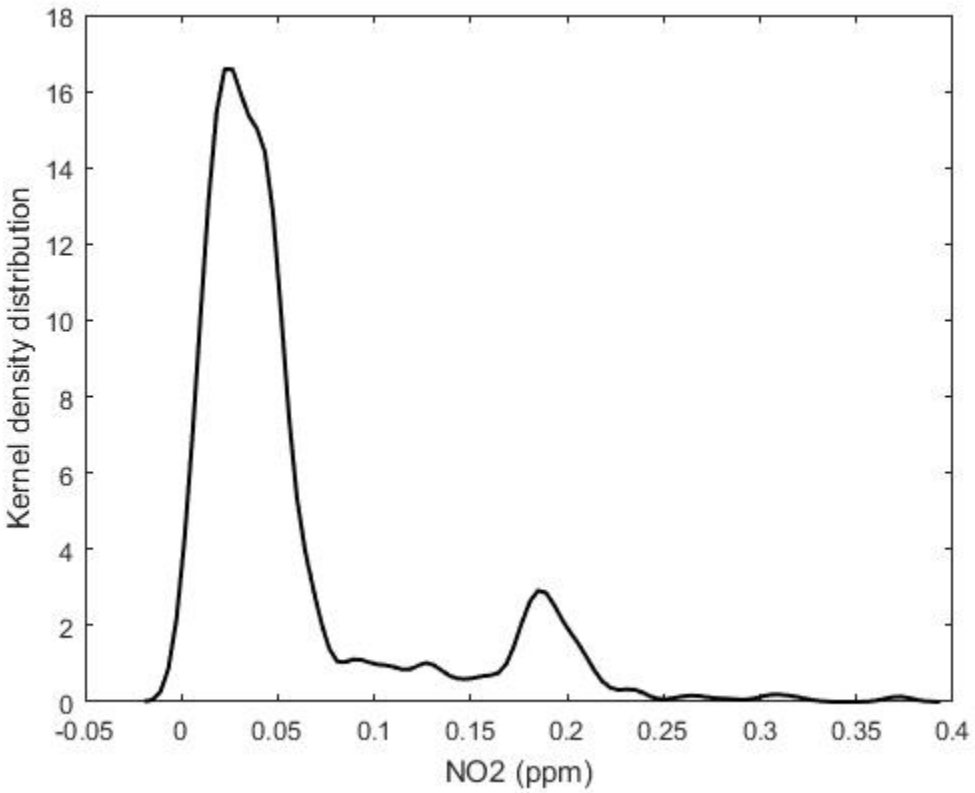
- 센서노드 SHTPI004의 산소 농도값의 KDE추정 및 시간이력 데이터



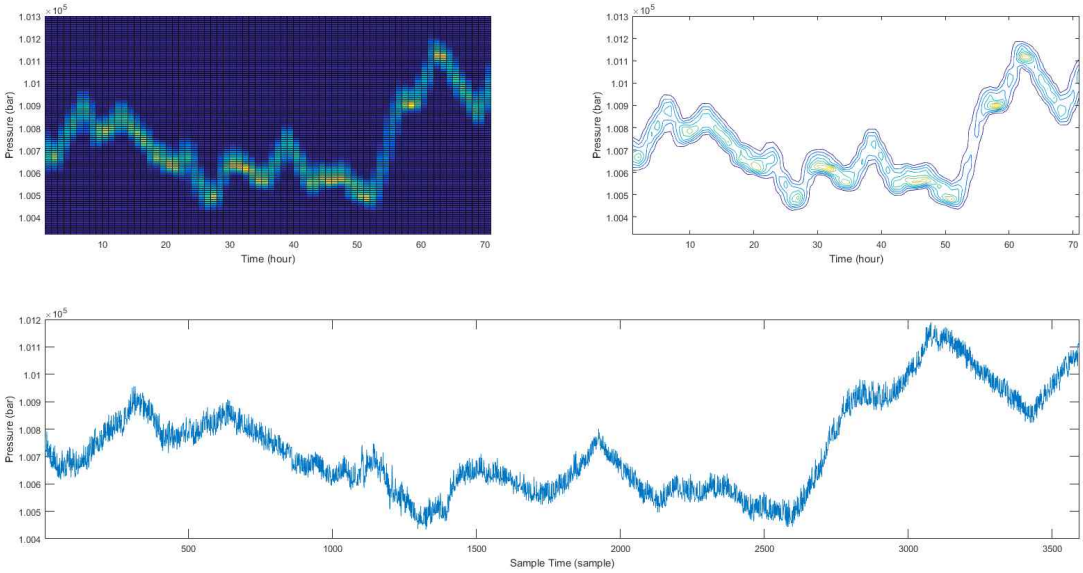


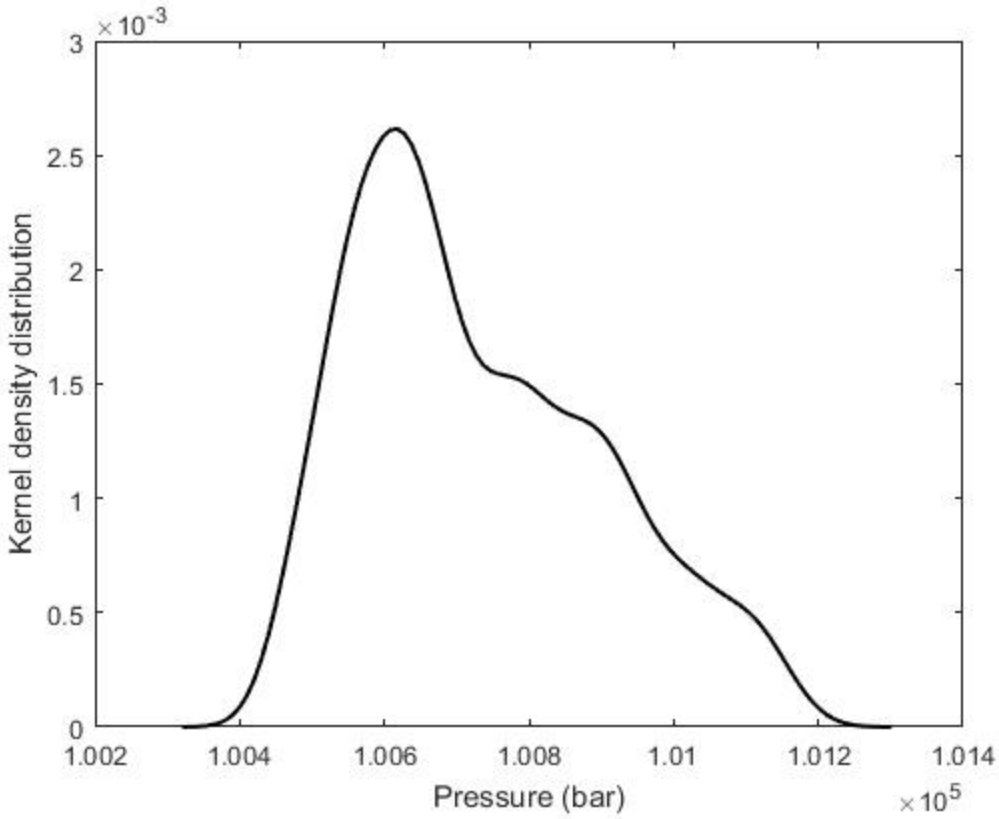
- 센서노드 SHTPI004의 이산화질소 농도값의 KDE추정 및 시간이력 데이터



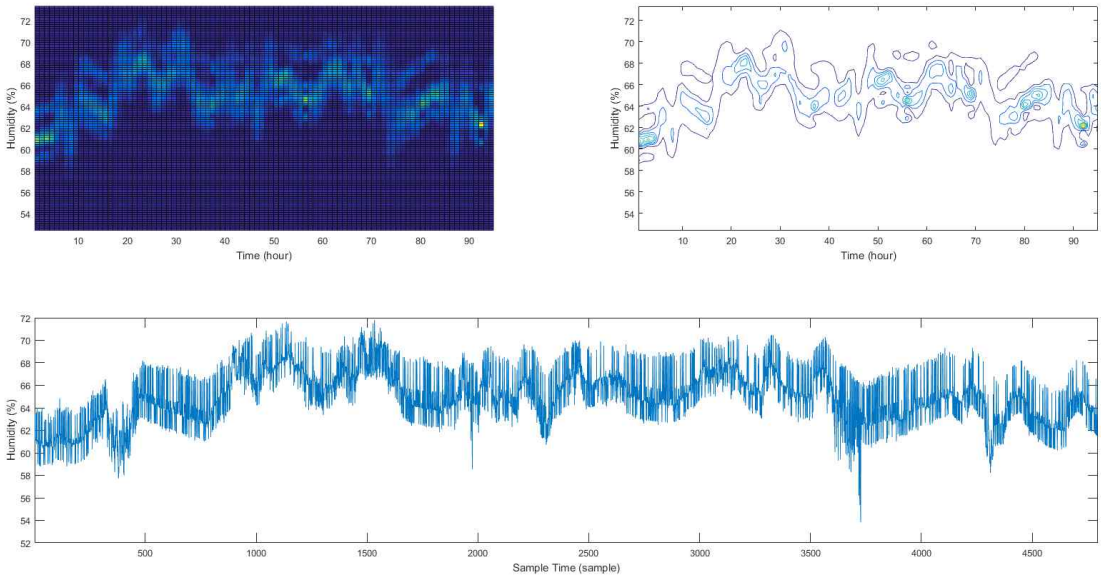


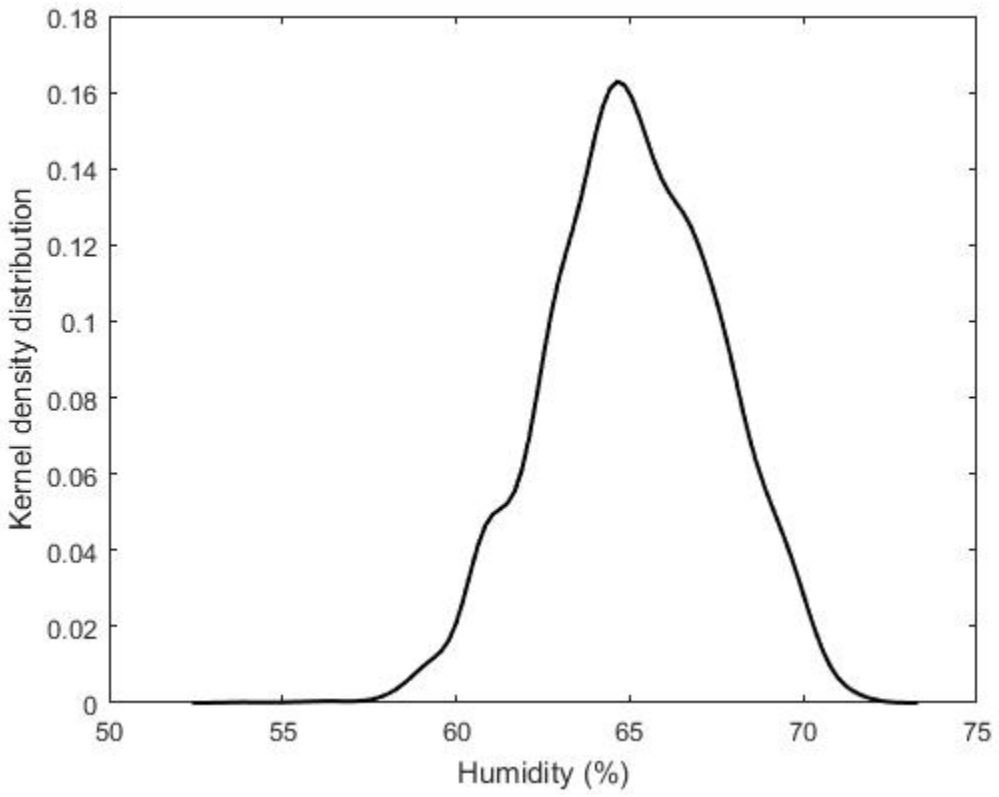
- 센서노드 SHTPI004의 대기압의 KDE추정 및 시간이력 데이터





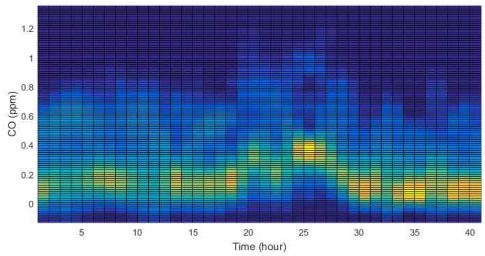
- 센서노드 SHTPI004의 습도값의 KDE추정 및 시간이력 데이터

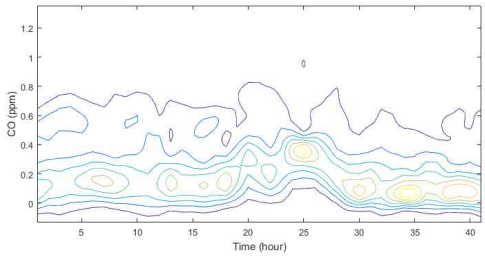


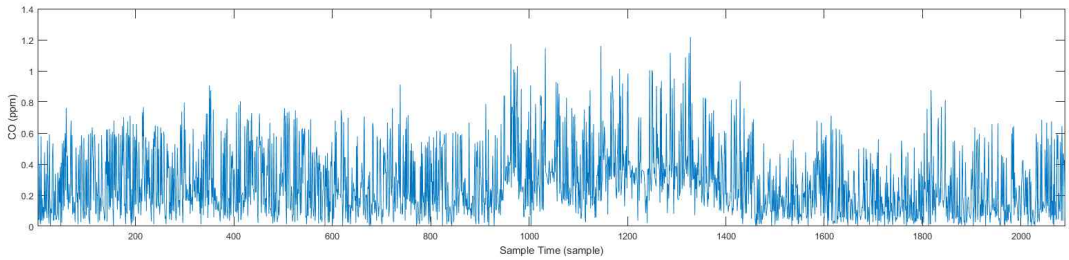


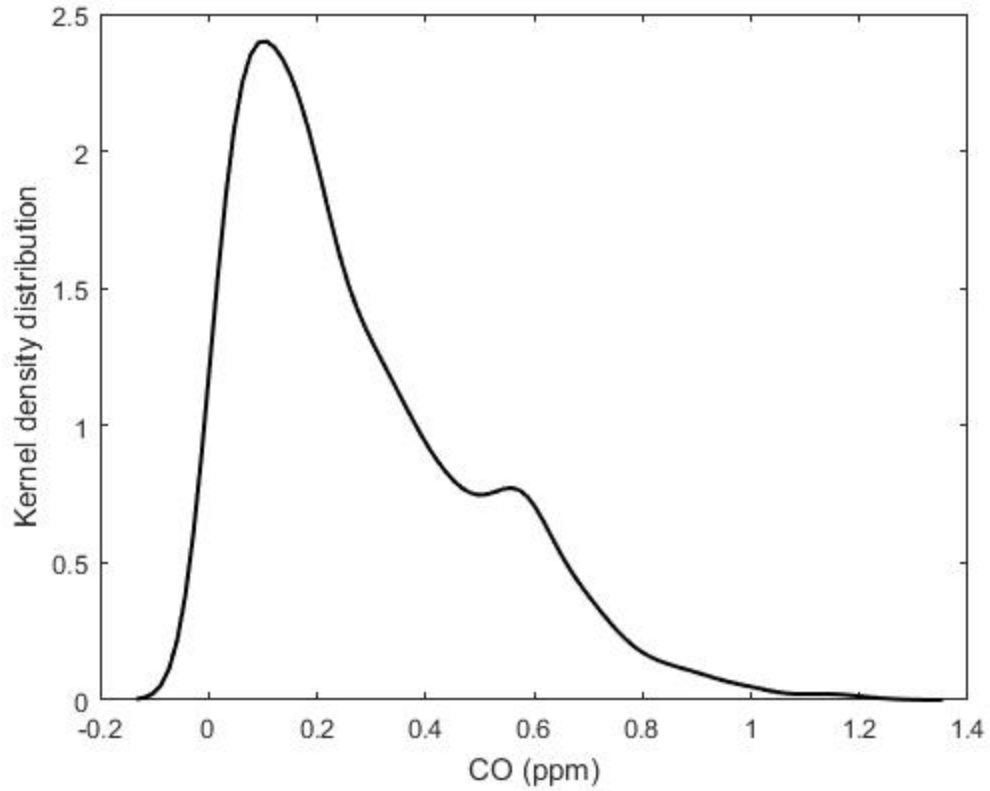
3.2.6 2017년 7월 21일

- 센서노드 SHTPI004의 CO값의 KDE추정 및 시간이력 데이터

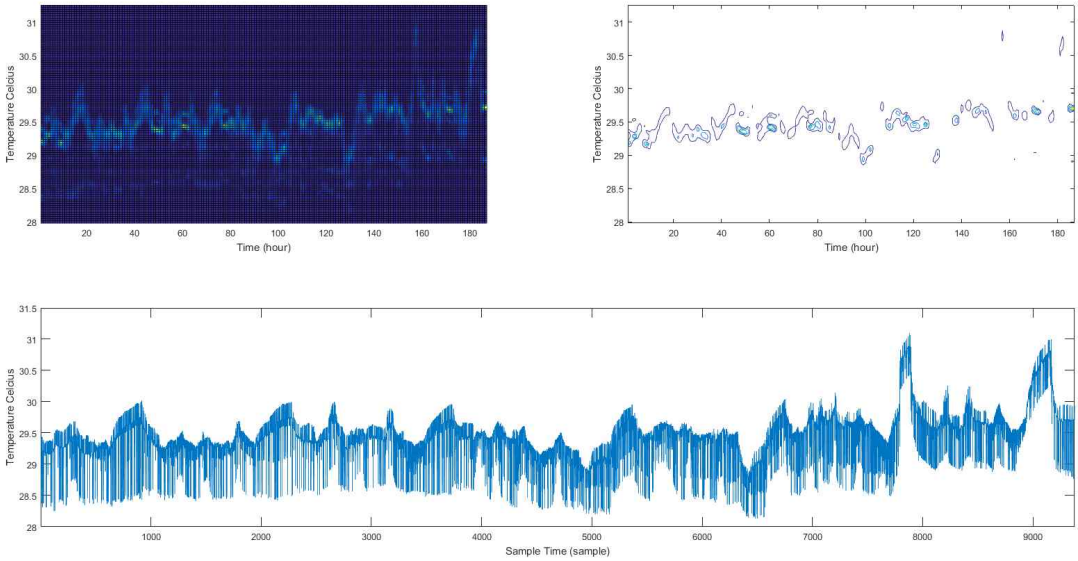


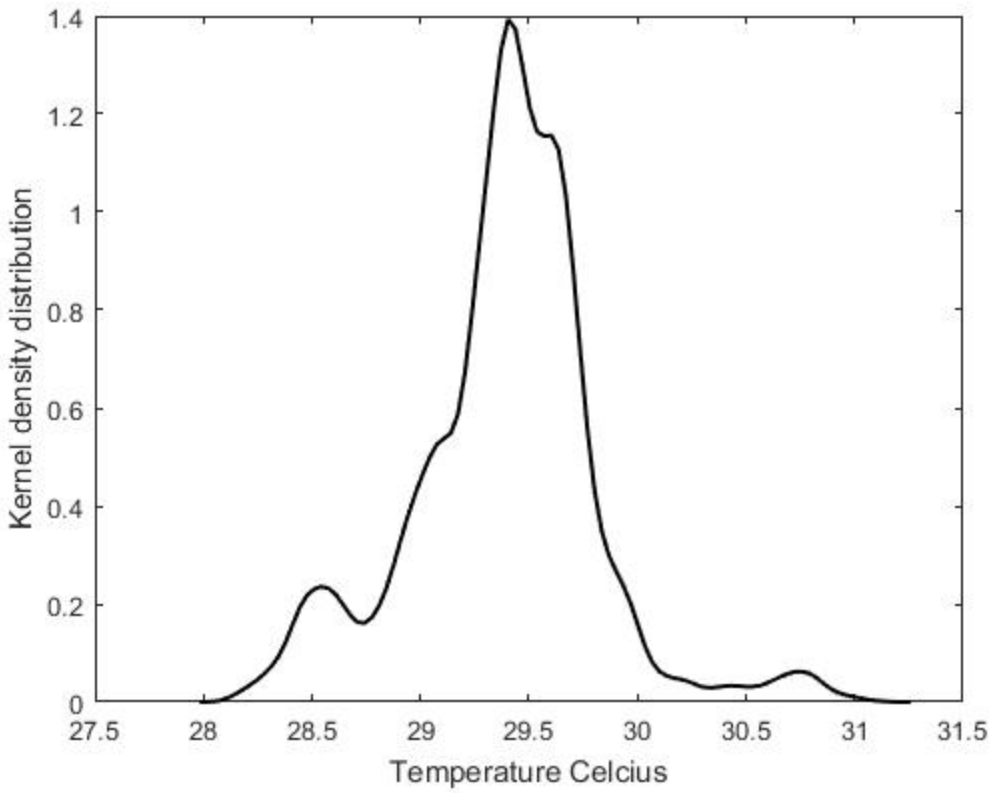




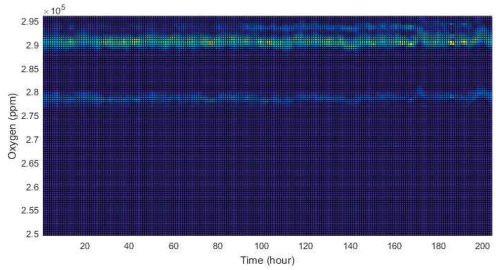


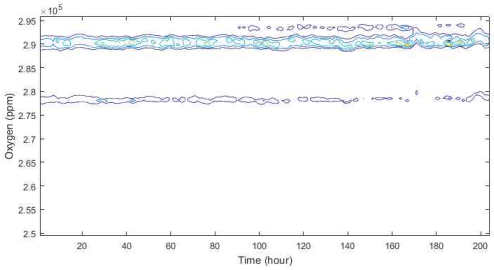
- 센서노드 SHTPI004의 온도값의 KDE추정 및 시간이력 데이터

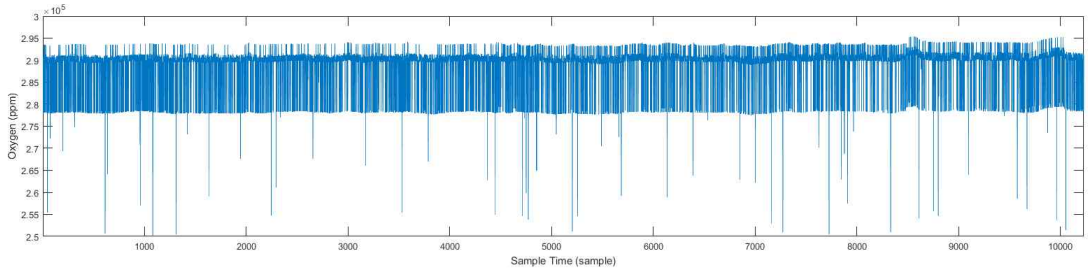


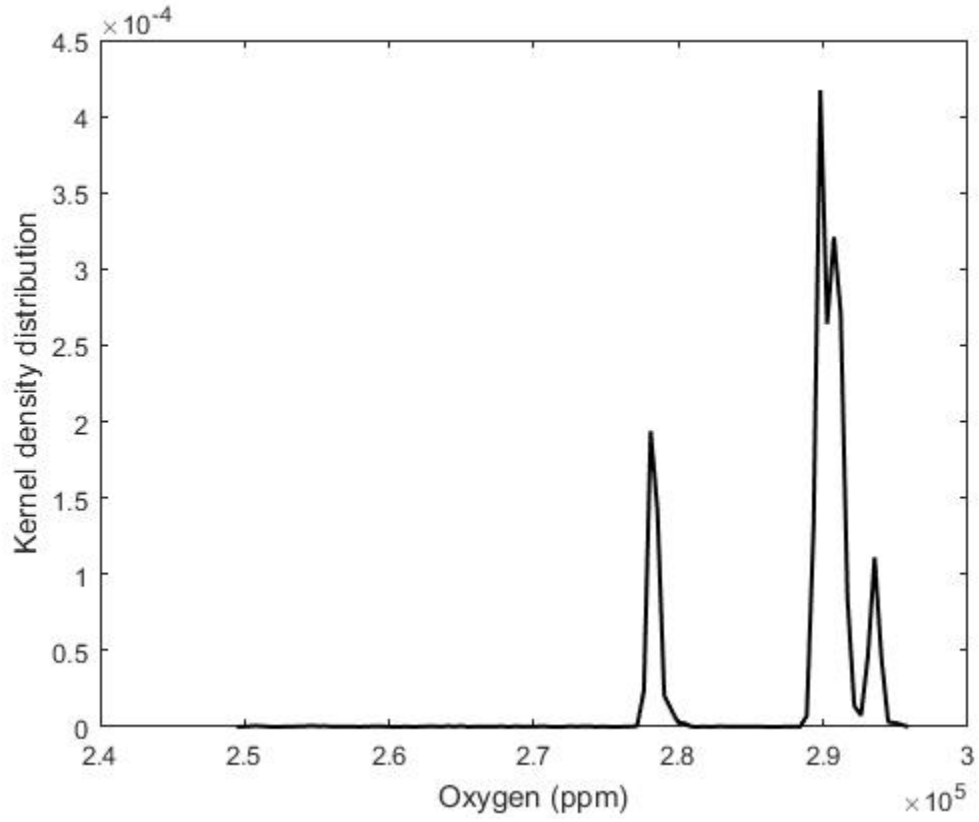


- 센서노드 SHTPI004의 산소 농도값의 KDE추정 및 시간이력 데이터

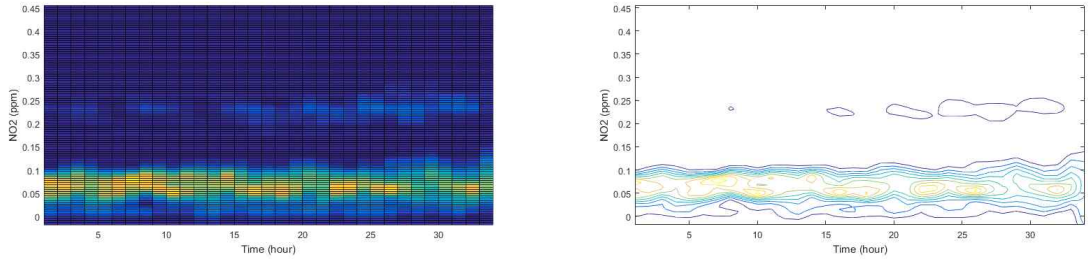


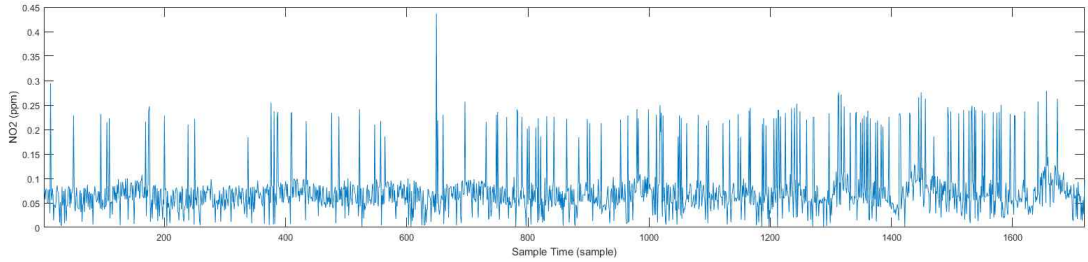


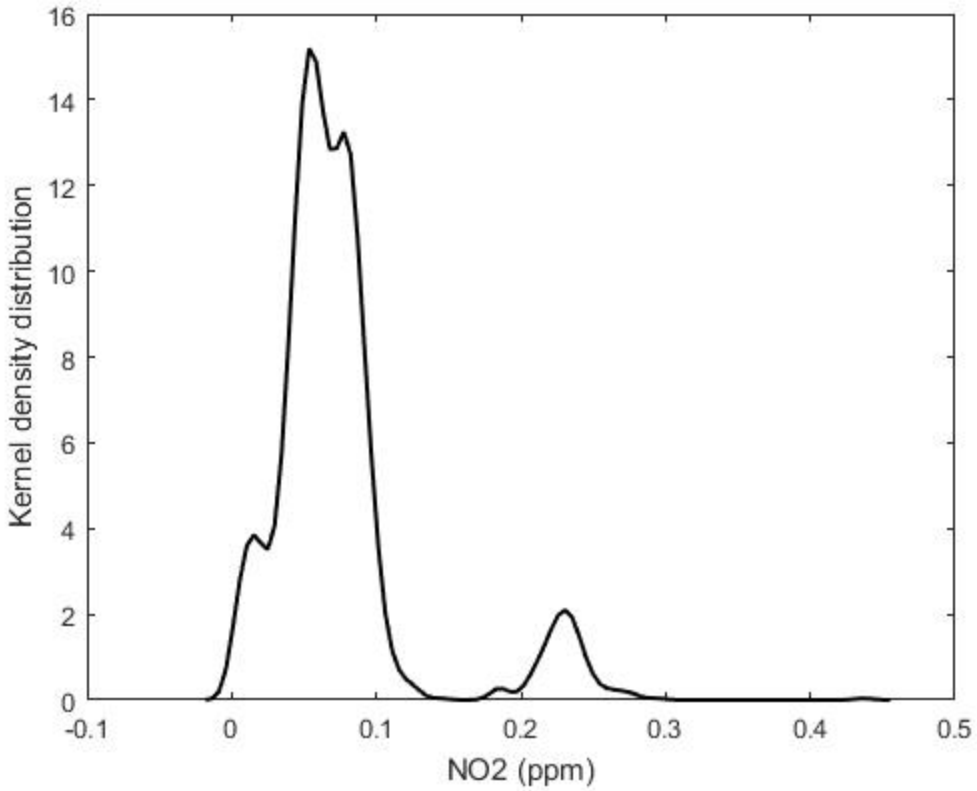




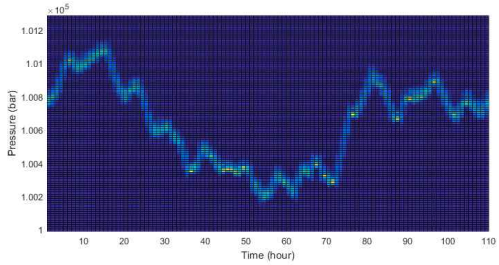
- 센서노드 SHTPI004의 이산화질소 농도값의 KDE추정 및 시간이력 데이터

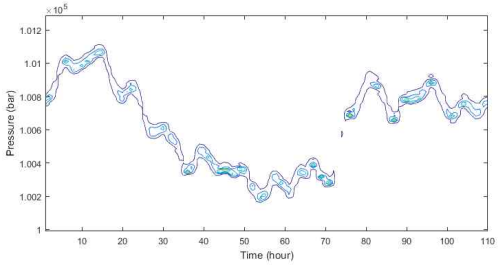


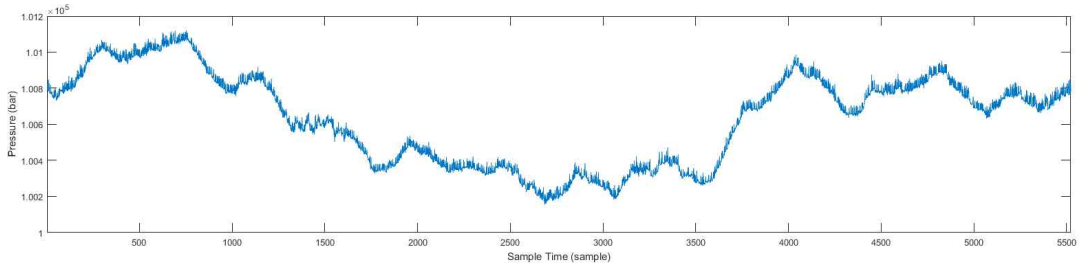


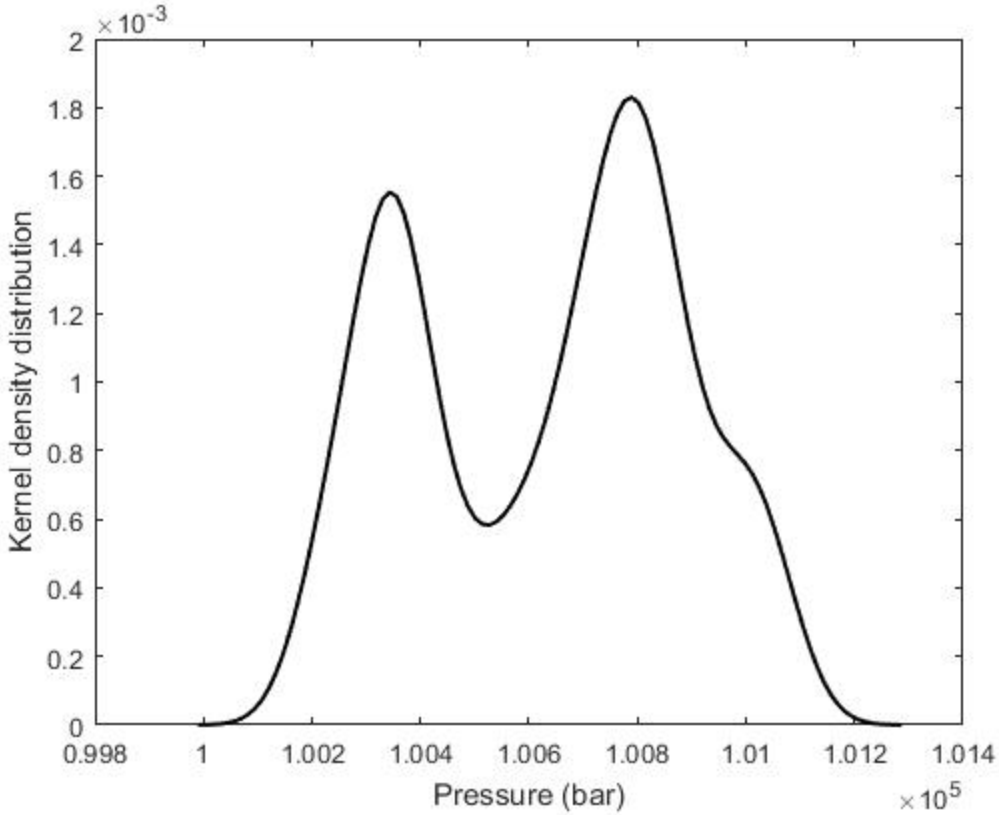


- 센서노드 SHTPI004의 대기압의 KDE추정 및 시간이력 데이터

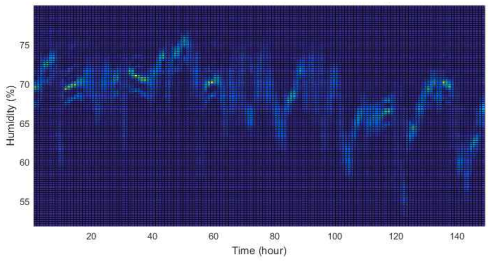


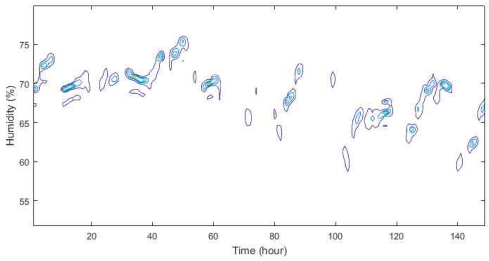


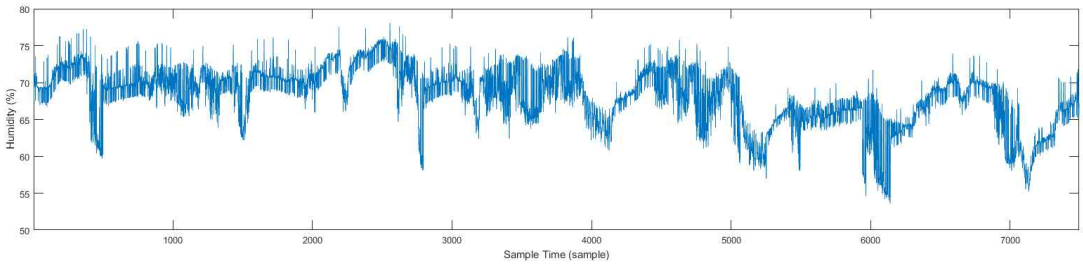


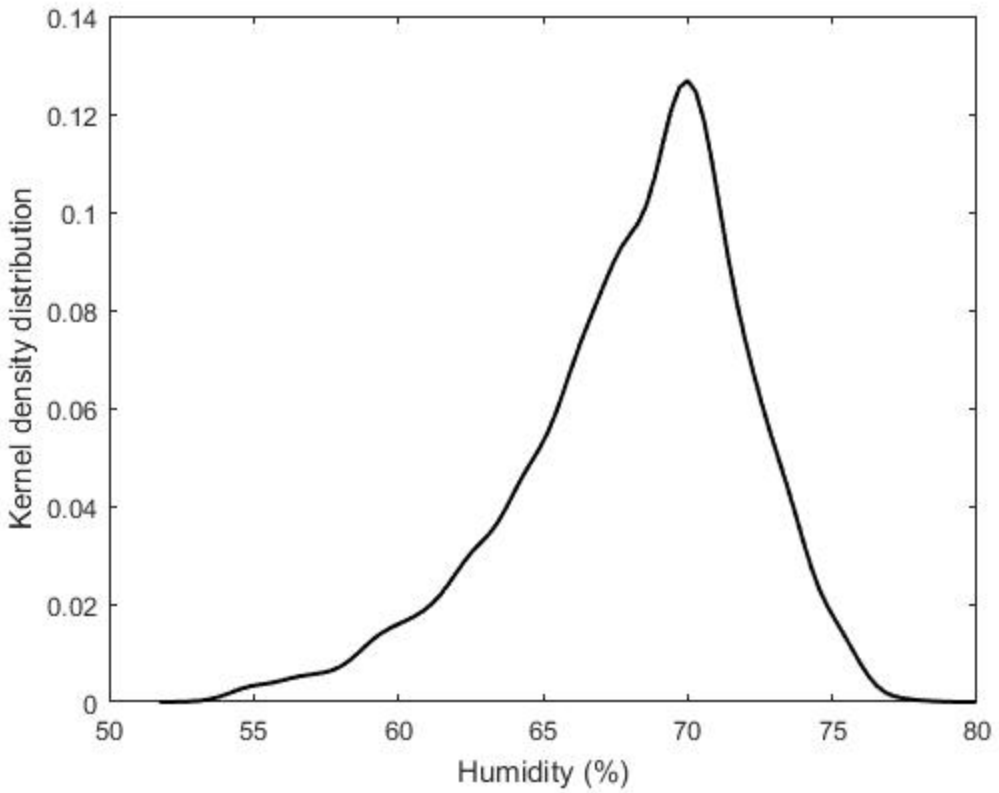


- 센서노드 SHTPI004의 습도값의 KDE추정 및 시간이력 데이터





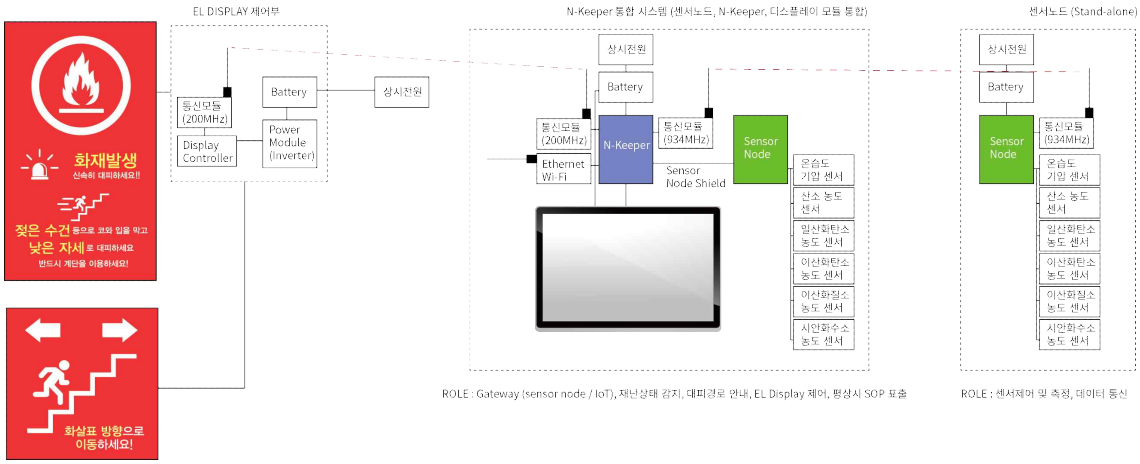




3.3 화재테스트 및 데이터

3.3.1 개포동역 테스트베드 화재 실험

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



– 중형 EL 디스플레이 패널



– 원기둥 부착형 EL 디스플레이 패널



- 화재 대피 안내문



- 화학 테러 대피 안내문

– 비상구형 및 테이프형 EL 디스플레이 설치



- 비상구형 EL 디스플레이 패널



- 계단 하부 테이프형 방향지시 EL 디스플레이 패널

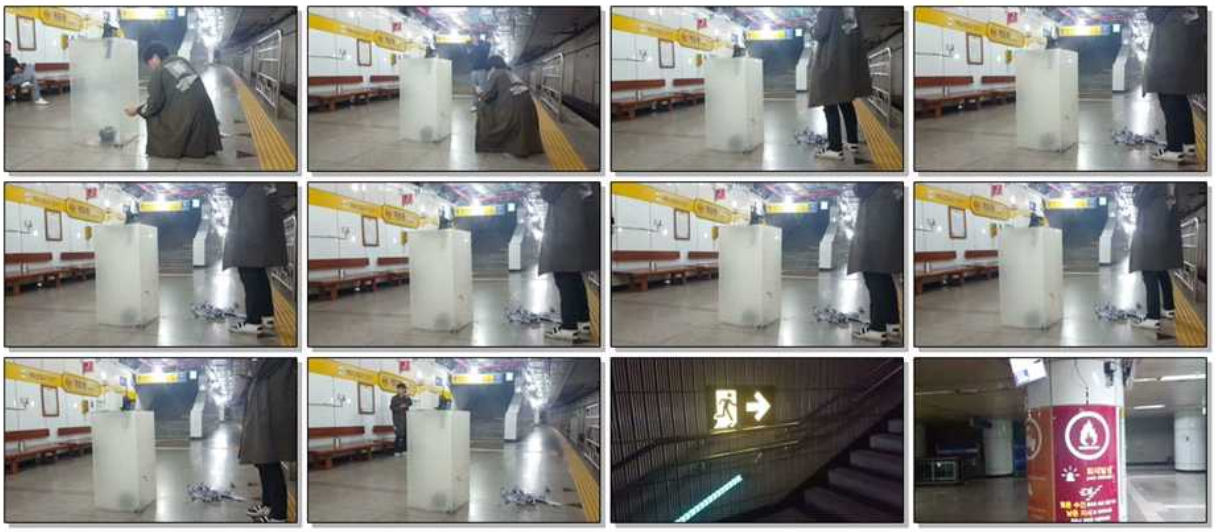
3.3.2 화재테스트 결과

– 화재테스트는 2017년 4월 21일 02:33 영업 종료후 개포동역에서 실시하였다.

– 센서노드에서 화재를 발생시키고 알고리즘 및 룰셋이 작동하여 1분 30초만에 화재인식이 되었다.

– 역무실 응답 대기 30초 후 재난모드로 자동전환하여 대피경로 안내를 표시하

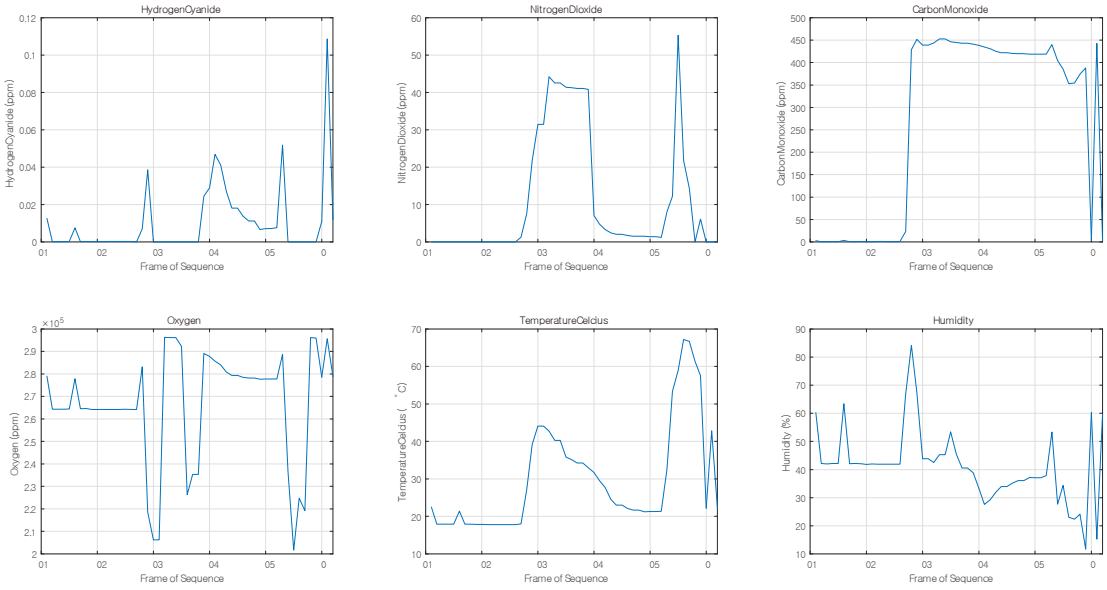
였다.



[화재테스트 영상 촬영 스냅샷]

3.3.3 화재 실험 센서노드 (SHTPI001) 데이터

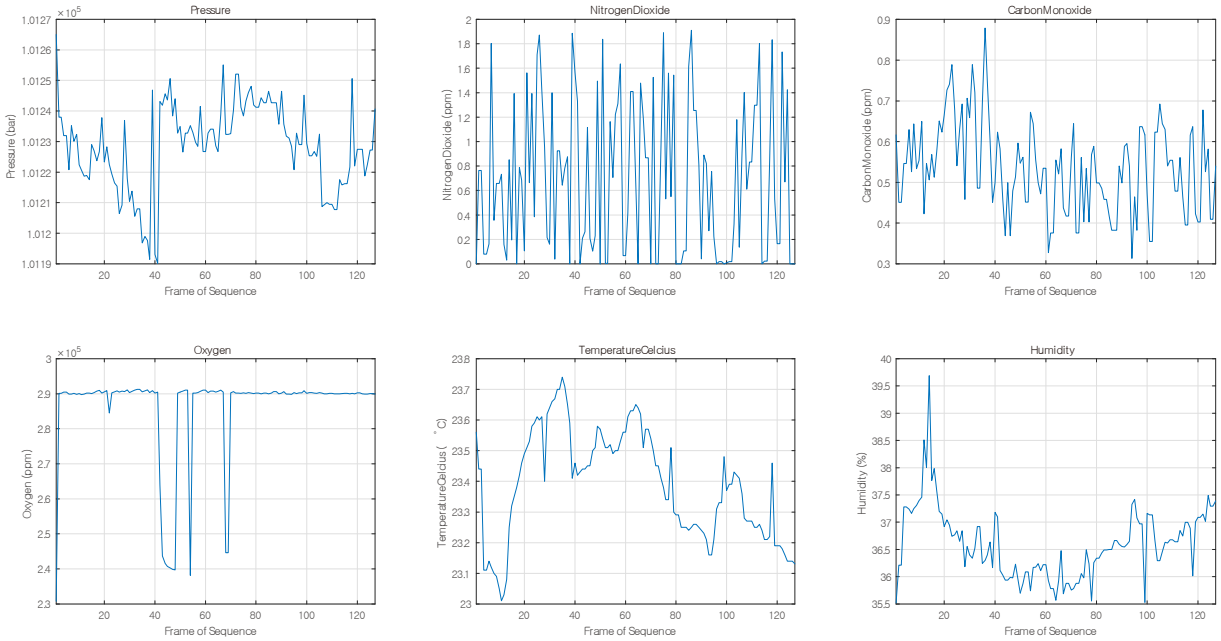
화재 발생시, HCN은 약간 증가하는 양상을 보이나, 이산화질소(NO_2)와 일산화탄소(CO)는 급격히 증가함을 알 수 있다. 또한 산소농도는 초기에 급격히 떨어지며, 온도는 섭씨 70도 가까이 증가한다.



[센서노드 SHTPI001 화재테스트 결과]

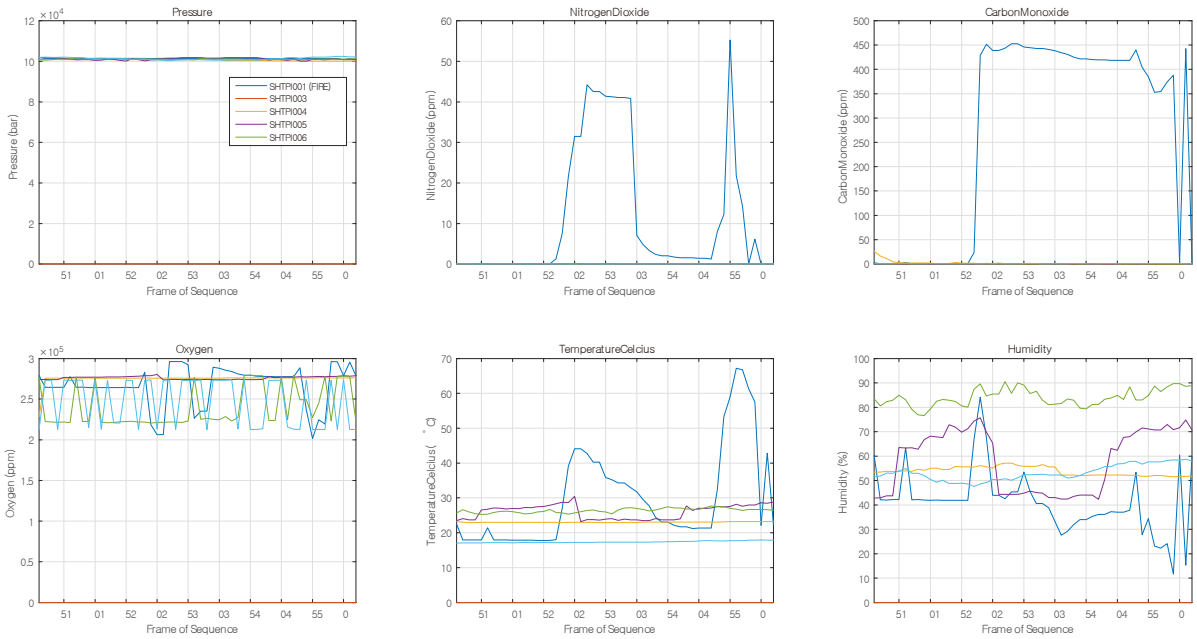
3.3.4 평상시 센서노드 (SHTPI013) 데이터

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



3.3.5 화재 실험 센서노드 (SHTPI001) 데이터의 평시 데이터와 비교

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



[센서노드 SHTPI001 화재 감지된 센서와 평시 모드의 센서데이터 비교]

부록

A. 서버API의 데이터 통신을 통한 데이터 분석 코드 (MATLAB)

```
clear all; clc; close all;

time_unix = 1339116554872; % example time
time_reference = datenum('1970', 'yyyy');
time_matlab = time_reference + time_unix / 8.64e7;
time_matlab_string = datestr(time_matlab, 'yyyymmdd HH:MM:SS.FFF');
n=100;
figure
for ii=[7]
data =
webread(['http://223.171.32.11/data?sensorNode=SHTPI00',num2str(ii),'&nStart=',num2str(-n),'&nStrop=-1']);
for kk=1:n
    cm(kk) = data(kk).CarbonMonoxide;
    oxygen(kk) = data(kk).Oxygen;
    temp(kk) = data(kk).TemperatureCelcius;
    nd(kk) = data(kk).NitrogenDioxide;
    times(kk) = data(kk).TimeStamp_Unix_Epoch_;
    mtime(kk) = time_reference + times(kk) / 8.64e7;
    %try
     %hcn(kk) = data(kk).HydrogenCyanide;
    %catch me
    pressure(kk) = str2num(data(kk).Pressure);
    %end
    hum(kk) = data(kk).Humidity;

end
vcm(:,ii)=cm;
voxygen(:,ii)=oxygen;
vtemp(:,ii)=temp;
vnd(:,ii)=nd;
vpres(:,ii)=pressure;
vhum(:,ii)=hum;
end


%try
%ax(1)=subplot(231);plot(hcn),title('HydrogenCyanide'),xlabel('Frame of Sequence'),ylabel('HydrogenCyanide (ppm)'),grid on
%catch me
ax(1)=subplot(231);plot(pressure),title('Pressure'),xlabel('Frame of Sequence'),ylabel('Pressure (bar)'),grid on
%legend('SHTPI001 (FIRE)','SHTPI003','SHTPI004','SHTPI005','SHTPI006')
%end
ax(2)=subplot(232);plot(nd),title('NitrogenDioxide'),xlabel('Frame of Sequence'),ylabel('NitrogenDioxide (ppm)'),grid on
ax(3)=subplot(233);plot(cm),title('CarbonMonoxide'),xlabel('Frame of Sequence'),ylabel('CarbonMonoxide (ppm)'),grid on
ax(4)=subplot(234);plot(oxygen),title('Oxygen'),xlabel('Frame of Sequence'),ylabel('Oxygen (ppm)'),grid on
ax(5)=subplot(235);plot(temp),title('TemperatureCelcius'),xlabel('Frame of Sequence'),ylabel('TemperatureCelcius (^₩circ C)'),grid on
ax(6)=subplot(236);plot(hum),title('Humidity'),xlabel('Frame of Sequence'),ylabel('Humidity (%)'),grid on
```

```
linkaxes(ax,'x');
xlim([1 n])
```

B. 기록된 RAW 데이터 분석을 위한 코드 (MATLAB)

```
clear all; clc; close all;

filename = 'rawstream_20170731';
fid=fopen(filename,'r');

kk=0;
while 1
    kk=kk+1;
    temp = fread(fid,1,'uint8');
    if isempty(temp)
        break;
    end
    if feof(fid)
        break;
    end
    data(kk)=temp;
end
fclose(fid);

ll=1;
for ii=1:length(data)
    if (data(ii)==60&&data(ii+1)==61&&data(ii+2)==62)
        pos(ll)=ii;
        ll=ll+1;
    end
end
CO=[];
NO2=[];
O2=[];
HCN=[];
TC=[];
HUM=[];
PRES=[];
wasp='          ';
for mm=1:length(pos)-1
    try
    frame=data(pos(mm):pos(mm+1)-1);
    delim=char(frame(1:3));
    framtype=frame(4);
    nob = frame(5);
    serialID = typecast(uint8(frame(6:9)),'uint32');
    waspID = char(frame(10:17));
    separator = frame(18);
    fos = frame(19);
    ind=20;
    k=1;
    disp(waspID)

    if all(waspID=='SHTPI010')
```

```
while (ind<=length(frame)-5)
    disp(ind)
    sID = frame(ind);
    if sID==123
        timestamp = typecast(uint8(frame(ind+1:ind+4)),'uint32');
        ind=ind+4;
        disp(['TimeStamp Epoch : ', num2str(timestamp)])
    end
    if sID==129
        GPCO=typecast(uint8(frame(ind+1:ind+4)),'single');
        ind=ind+4;
        CO=[CO,GPCO];
        disp(['GP_CO : ', num2str(GPCO)])
    end
    if sID==134
        GPHCN=typecast(uint8(frame(ind+1:ind+4)),'single');
        ind=ind+4;
        HCN=[HCN,GPHCN];
        disp(['GP_HCN : ', num2str(GPHCN)])
    end
    if sID==137
        GPNO2=typecast(uint8(frame(ind+1:ind+4)),'single');
        ind=ind+4;
        NO2=[NO2,GPNO2];
        disp(['GP_NO2 : ', num2str(GPNO2)])
    end
    if sID==138
        GPO2=typecast(uint8(frame(ind+1:ind+4)),'single');
        ind=ind+4;
        O2=[O2,GPO2];
        disp(['GO_O2 : ', num2str(GPO2)])
    end
    if sID==144
        GPTC=typecast(uint8(frame(ind+1:ind+4)),'single');
        ind=ind+4;
        TC=[TC,GPTC];
        disp(['GP_TC: ' , num2str(GPTC)])
    end
    if sID==146
        GPHUM=typecast(uint8(frame(ind+1:ind+4)),'single');
        ind=ind+4;
        HUM=[HUM,GPHUM];
        disp(['GP_Hum : ', num2str(GPHUM)])
    end
    if sID==147
        GPPRES=typecast(uint8(frame(ind+1:ind+4)),'single');
        ind=ind+4;
        PRES=[PRES,GPPRES];
        disp(['GP_Pressure : ', num2str(GPPRES)])
    end
    if sID==52
        BAT=frame(ind+1);
        ind=ind+1;
        disp(['Battery : ', num2str(BAT)])
    end
    ind=ind+1;

end
```

```
        end

        catch me
            disp('error')
        end
    end
    %%
    close all
    tCO=CO(CO<=5);
    tCO=tCO(tCO>=0.00001);
    tNO2=NO2(NO2<=5);
    tNO2=tNO2(tNO2>=0.00001);
    tTC=TC(TC<=1e2);
    tTC=tTC(tTC>=20);
    figure,plot(tCO)
    tO2=O2(O2<=1e7);
    tO2=tO2(tO2>=2.5e5);
    figure,plot(tO2)
    tPRES=PRES(PRES<=1e6);
    tPRES=tPRES(tPRES>=1000);
    tHUM=HUM(HUM<=1e6);
    tHUM=tHUM(tHUM>=1);



    [fCO1,xiCO1,bw] = ksdensity(tCO);
    figure,plot(xiCO1,fCO1,'-k','LineWidth',1.5),xlabel('CO (ppm)'),ylabel('Kernel density distribution')
    COBuf=buffer(tCO,100,50);
    [r,c]=size(COBuf);
    for kk=1:c
        [fCO(:,kk),xiCO(:,kk)]=ksdensity(COBuf(:,kk),xiCO1);
    end
    figure,ax1(1)=subplot(221);surf(1:c-1,xiCO1,fCO(:,2:end)),xlabel('Time (hour)'),ylabel('CO (ppm)'),zlabel('kernel
    density');
    view(2)
    ax1(2)=subplot(222);contour(1:c-1,xiCO1,fCO(:,2:end)),xlabel('Time (hour)'),ylabel('CO (ppm)');
    linkaxes(ax1);
    xlim([1 c-1]),ylim([xiCO1(1) xiCO1(end)])
    subplot(2,2,[3 4]),plot(tCO),xlim([1 length(tCO)]),xlabel('Sample Time (sample)'),ylabel('CO (ppm)')

    [fTC1,xiTC1,bw] = ksdensity(tTC);
    figure,plot(xiTC1,fTC1,'-k','LineWidth',1.5),xlabel('Temperature Celcius'),ylabel('Kernel density distribution')
    TCBuf=buffer(tTC,100,50);
    [r,c]=size(TCBuf);
    for kk=1:c
        [fTC(:,kk),xiTC(:,kk)]=ksdensity(TCBuf(:,kk),xiTC1);
    end
    figure,ax1(1)=subplot(221);surf(1:c-1,xiTC1,fTC(:,2:end)),xlabel('Time (hour)'),ylabel('Temperature
    Celcius'),zlabel('kernel density');
    view(2)
    ax1(2)=subplot(222);contour(1:c-1,xiTC1,fTC(:,2:end)),xlabel('Time (hour)'),ylabel('Temperature Celcius');
    linkaxes(ax1);
    xlim([1 c-1]),ylim([xiTC1(1) xiTC1(end)])
    subplot(2,2,[3 4]),plot(tTC),xlim([1 length(tTC)]),xlabel('Sample Time (sample)'),ylabel('Temperature Celcius')

    [fO21,xiO21,bw] = ksdensity(tO2);
    figure,plot(xiO21,fO21,'-k','LineWidth',1.5),xlabel('Oxygen (ppm)'),ylabel('Kernel density distribution')
    O2Buf=buffer(tO2,100,50);
```

```
[r,c]=size(O2Buf);
for kk=1:c
    [fO2(:,kk),xiO2(:,kk)]=ksdensity(O2Buf(:,kk),xiO21);
end
figure,ax1(1)=subplot(221);surf(1:c-1,xiO21,fO2(:,2:end)),xlabel('Time (hour)'),ylabel('Oxygen (ppm)'),zlabel('kernel
density');
view(2)
ax1(2)=subplot(222);contour(1:c-1,xiO21,fO2(:,2:end)),xlabel('Time (hour)'),ylabel('Oxygen (ppm)');
linkaxes(ax1);
xlim([1 c-1]),ylim([xiO21(1) xiO21(end)])
subplot(2,2,[3 4]),plot(tO2),xlim([1 length(tO2)]),xlabel('Sample Time (sample)'),ylabel('Oxygen (ppm)')

[fNO21,xiNO21,bw] = ksdensity(tNO2);
figure,plot(xiNO21,fNO21,'-k','LineWidth',1.5),xlabel('NO2 (ppm)'),ylabel('Kernel density distribution')
NO2Buf=buffer(tNO2,100,50);
[r,c]=size(NO2Buf);
for kk=1:c
    [fNO2(:,kk),xiNO2(:,kk)]=ksdensity(NO2Buf(:,kk),xiNO21);
end
figure,ax1(1)=subplot(221);surf(1:c-1,xiNO21,fNO2(:,2:end)),xlabel('Time (hour)'),ylabel('NO2 (ppm)'),zlabel('kernel
density');
view(2)
ax1(2)=subplot(222);contour(1:c-1,xiNO21,fNO2(:,2:end)),xlabel('Time (hour)'),ylabel('NO2 (ppm)');
linkaxes(ax1);
xlim([1 c-1]),ylim([xiNO21(1) xiNO21(end)])
subplot(2,2,[3 4]),plot(tNO2),xlim([1 length(tNO2)]),xlabel('Sample Time (sample)'),ylabel('NO2 (ppm)')

[fPRES1,xiPRES1,bw] = ksdensity(tPRES);
figure,plot(xiPRES1,fPRES1,'-k','LineWidth',1.5),xlabel('Pressure (bar)'),ylabel('Kernel density distribution')
PRESBuf=buffer(tPRES,100,50);
[r,c]=size(PRESBuf);
for kk=1:c
    [fPRES(:,kk),xiPRES(:,kk)]=ksdensity(PRESBuf(:,kk),xiPRES1);
end
figure,ax1(1)=subplot(221);surf(1:c-1,xiPRES1,fPRES(:,2:end)),xlabel('Time (hour)'),ylabel('Pressure
(bar)'),zlabel('kernel density');
view(2)
ax1(2)=subplot(222);contour(1:c-1,xiPRES1,fPRES(:,2:end)),xlabel('Time (hour)'),ylabel('Pressure (bar)');
linkaxes(ax1);
xlim([1 c-1]),ylim([xiPRES1(1) xiPRES1(end)])
subplot(2,2,[3 4]),plot(tPRES),xlim([1 length(tPRES)]),xlabel('Sample Time (sample)'),ylabel('Pressure (bar)')

[fHUM1,xiHUM1,bw] = ksdensity(tHUM);
figure,plot(xiHUM1,fHUM1,'-k','LineWidth',1.5),xlabel('Humidity (%)'),ylabel('Kernel density distribution')
HUMBuf=buffer(tHUM,100,50);
[r,c]=size(HUMBuf);
for kk=1:c
    [fHUM(:,kk),xiHUM(:,kk)]=ksdensity(HUMBuf(:,kk),xiHUM1);
end
figure,ax1(1)=subplot(221);surf(1:c-1,xiHUM1,fHUM(:,2:end)),xlabel('Time (hour)'),ylabel('Humidity
(%)'),zlabel('kernel density');
view(2)
ax1(2)=subplot(222);contour(1:c-1,xiHUM1,fHUM(:,2:end)),xlabel('Time (hour)'),ylabel('Humidity (%)');
linkaxes(ax1);
xlim([1 c-1]),ylim([xiHUM1(1) xiHUM1(end)])
subplot(2,2,[3 4]),plot(tHUM),xlim([1 length(tHUM)]),xlabel('Sample Time (sample)'),ylabel('Humidity (%)')
```

C. 장기 데이터 계측 API 로그 (sample)

```
indicator : <=>
saving binary stream : rawstream_20170106
1
Express server listening on port 80
## serial open
============================================================
Raw                                                     data                                                    :
52,60,61,62,0,59,16,89,12,24,83,72,84,80,73,48,49,50,35,10,52,100,54,119,0,123,33,212,70,88,138,53,88,133,72,1
44,92,143,162,65,146,0,32,250,65,147,235,248,199,71,129,225,164,176,65,137,0,0,0,0,134,119,185,201,58,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 10
Serial ID : 403462416
Waspmote ID : SHTPI012
## Battery
SENSOR_BAT : 100 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481036833 Epoch Time
## Oxygen
SENSOR_GP_O2 : 273089.65625 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.31999969482422 ºC
## Humidity
SENSOR_GP_HUM : 31.265625 %RH
## Pressure
SENSOR_GP_PRES : 102385.8359375 Pa
## Carbon Monoxide
SENSOR_GP_CO : 22.080507278442383 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.001539035583846271 ppm
sensor:SHTPI012 1481036833 1481036833
true
============================================================
Raw                                                     data                                                    :
10,60,61,62,0,39,209,103,12,24,83,72,84,80,73,48,49,55,35,11,52,97,54,119,0,123,175,150,65,57,144,246,40,100,
65,146,0,193,33,66,147,50,113,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 11
Serial ID : 403466193
Waspmote ID : SHTPI017
## Battery
SENSOR_BAT : 97 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960599727 Epoch Time
## Temperature Celcius
```

```
SENSOR_GP_TC : 14.260000228881836 ºC
## Humidity
SENSOR_GP_HUM : 40.4384765625 %RH
## Pressure
SENSOR_GP_PRES : 102626.390625 Pa
sensor:SHTPI017 960599727 960599727
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,59,195,130,12,24,83,72,84,80,73,48,48,49,35,10,52,96,54,119,0,123,64,212,70,88,138,99,44,135,72,
144,102,102,164,65,146,0,6,250,65,147,114,17,200,71,129,238,188,225,65,137,0,0,0,0,134,130,33,25,59,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 10
Serial ID : 403473091
Waspmote ID : SHTPI001
## Battery
SENSOR_BAT : 96 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481036864 Epoch Time
## Oxygen
SENSOR_GP_O2 : 276835.09375 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.549999237060547 ºC
## Humidity
SENSOR_GP_HUM : 31.2529296875 %RH
## Pressure
SENSOR_GP_PRES : 102434.890625 Pa
## Carbon Monoxide
SENSOR_GP_CO : 28.21725082397461 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.002336591947823763 ppm
sensor:SHTPI001 1481036864 1481036864
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,59,195,130,12,24,83,72,84,80,73,48,48,49,35,10,52,96,54,119,0,123,64,212,70,88,138,99,44,135,72,
144,102,102,164,65,146,0,6,250,65,147,114,17,200,71,129,238,188,225,65,137,0,0,0,0,134,130,33,25,59,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 10
Serial ID : 403473091
Waspmote ID : SHTPI001
## Battery
SENSOR_BAT : 96 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481036864 Epoch Time
## Oxygen
SENSOR_GP_O2 : 276835.09375 ppm
## Temperature Celcius
```

```
SENSOR_GP_TC : 20.549999237060547 ºC
## Humidity
SENSOR_GP_HUM : 31.2529296875 %RH
## Pressure
SENSOR_GP_PRES : 102434.890625 Pa
## Carbon Monoxide
SENSOR_GP_CO : 28.21725082397461 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.002336591947823763 ppm
sensor:SHTPI001 1481036864 1481036864
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,54,65,252,10,24,83,72,84,80,73,48,48,53,35,15,52,98,54,119,0,123,108,212,70,88,138,189,218,132,
72,144,164,112,165,65,146,0,46,249,65,147,218,54,200,71,129,68,226,65,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 15
Serial ID : 403373121
Waspmote ID : SHTPI005
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481036908 Epoch Time
## Oxygen
SENSOR_GP_O2 : 272085.90625 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.68000030517578 ºC
## Humidity
SENSOR_GP_HUM : 31.1474609375 %RH
## Pressure
SENSOR_GP_PRES : 102509.703125 Pa
## Carbon Monoxide
SENSOR_GP_CO : 48.47096252441406 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI005 1481036908 1481036908
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,54,84,2,11,24,83,72,84,80,73,48,48,55,35,15,52,95,54,119,0,123,108,212,70,88,138,193,23,138,72,1
44,0,0,162,65,146,0,206,247,65,147,77,27,200,71,129,231,158,20,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 15
Serial ID : 403374676
Waspmote ID : SHTPI007
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
```

```
SENSOR_TST : 1481036908 Epoch Time
## Oxygen
SENSOR_GP_O2 : 282814.03125 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.25 ºC
## Humidity
SENSOR_GP_HUM : 30.9755859375 %RH
## Pressure
SENSOR_GP_PRES : 102454.6015625 Pa
## Carbon Monoxide
SENSOR_GP_CO : 37.15517807006836 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI007 1481036908 1481036908
true
================================================================
Raw                                                 data                                                :
10,60,61,62,0,39,209,103,12,24,83,72,84,80,73,48,49,55,35,14,52,97,54,119,0,123,254,150,65,57,144,246,40,100,
65,146,0,154,32,66,147,132,114,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 14
Serial ID : 403466193
Waspmote ID : SHTPI017
## Battery
SENSOR_BAT : 97 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960599806 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.260000228881836 ºC
## Humidity
SENSOR_GP_HUM : 40.150390625 %RH
## Pressure
SENSOR_GP_PRES : 102629.03125 Pa
sensor:SHTPI017 960599806 960599806
true
================================================================
Raw                                                 data                                                :
10,60,61,62,0,39,209,103,12,24,83,72,84,80,73,48,49,55,35,15,52,97,54,119,0,123,33,151,65,57,144,246,40,100,6
5,146,0,44,32,66,147,100,115,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 15
Serial ID : 403466193
Waspmote ID : SHTPI017
## Battery
SENSOR_BAT : 97 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960599841 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.260000228881836 ºC
## Humidity
```

```
SENSOR_GP_HUM : 40.04296875 %RH
## Pressure
SENSOR_GP_PRES : 102630.78125 Pa
sensor:SHTPI017 960599841 960599841
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,39,189,251,10,24,83,72,84,80,73,48,49,54,35,14,52,95,54,119,0,123,23,133,65,57,144,82,184,106,6
5,146,0,181,29,66,147,123,44,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 14
Serial ID : 403372989
Waspmote ID : SHTPI016
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960595223 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.670000076293945 ºC
## Humidity
SENSOR_GP_HUM : 39.4267578125 %RH
## Pressure
SENSOR_GP_PRES : 102488.9609375 Pa
sensor:SHTPI016 960595223 960595223
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,59,16,89,12,24,83,72,84,80,73,48,49,50,35,11,52,100,54,119,0,123,207,212,70,88,138,202,91,133,7
2,144,205,204,162,65,146,0,252,248,65,147,190,253,199,71,129,77,113,179,65,137,0,0,0,0,134,220,175,231,58,13,
10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 11
Serial ID : 403462416
Waspmote ID : SHTPI012
## Battery
SENSOR_BAT : 100 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037007 Epoch Time
## Oxygen
SENSOR_GP_O2 : 273118.3125 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.350000381469727 ºC
## Humidity
SENSOR_GP_HUM : 31.123046875 %RH
## Pressure
SENSOR_GP_PRES : 102395.484375 Pa
## Carbon Monoxide
SENSOR_GP_CO : 22.430322647094727 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
```

```
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.0017676311545073986 ppm
sensor:SHTPI012 1481037007 1481037007
true
============================================================
Raw                                                     data                                                    :
10,60,61,62,0,54,105,254,10,24,83,72,84,80,73,48,49,52,35,16,52,98,54,119,0,123,209,212,70,88,138,145,130,137
,72,144,72,225,162,65,146,0,90,252,65,147,188,17,200,71,129,142,133,119,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 16
Serial ID : 403373673
Waspmote ID : SHTPI014
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037009 Epoch Time
## Oxygen
SENSOR_GP_O2 : 281620.53125 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.360000610351562 ºC
## Humidity
SENSOR_GP_HUM : 31.5439453125 %RH
## Pressure
SENSOR_GP_PRES : 102435.46875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 61.88042449951172 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI014 1481037009 1481037009
true
============================================================
Raw                                                     data                                                    :
10,60,61,62,0,54,65,252,10,24,83,72,84,80,73,48,48,53,35,16,52,98,54,119,0,123,209,212,70,88,138,125,223,132,
72,144,143,194,165,65,146,0,230,247,65,147,214,61,200,71,129,94,101,59,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 16
Serial ID : 403373121
Waspmote ID : SHTPI005
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037009 Epoch Time
## Oxygen
SENSOR_GP_O2 : 272123.90625 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.719999313354492 ºC
## Humidity
SENSOR_GP_HUM : 30.9873046875 %RH
## Pressure
SENSOR_GP_PRES : 102523.671875 Pa
```

```
## Carbon Monoxide
SENSOR_GP_CO : 46.84899139404297 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI005 1481037009 1481037009
true
============================================================
Raw                                                 data                                                :
10,60,61,62,0,59,195,130,12,24,83,72,84,80,73,48,48,49,35,11,52,96,54,119,0,123,226,212,70,88,138,5,48,135,72,
144,215,163,164,65,146,0,252,251,65,147,170,14,200,71,129,101,72,221,65,137,0,0,0,0,134,168,78,39,59,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 11
Serial ID : 403473091
Waspmote ID : SHTPI001
## Battery
SENSOR_BAT : 96 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037026 Epoch Time
## Oxygen
SENSOR_GP_O2 : 276864.15625 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.579999923706055 ºC
## Humidity
SENSOR_GP_HUM : 31.498046875 %RH
## Pressure
SENSOR_GP_PRES : 102429.328125 Pa
## Carbon Monoxide
SENSOR_GP_CO : 27.660348892211914 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.0025529060512781143 ppm
sensor:SHTPI001 1481037026 1481037026
true
============================================================
Raw                                                 data                                                :
10,60,61,62,0,59,195,130,12,24,83,72,84,80,73,48,48,49,35,11,52,96,54,119,0,123,226,212,70,88,138,5,48,135,72,
144,215,163,164,65,146,0,252,251,65,147,170,14,200,71,129,101,72,221,65,137,0,0,0,0,134,168,78,39,59,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 11
Serial ID : 403473091
Waspmote ID : SHTPI001
## Battery
SENSOR_BAT : 96 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037026 Epoch Time
## Oxygen
SENSOR_GP_O2 : 276864.15625 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.579999923706055 ºC
```

```
## Humidity
SENSOR_GP_HUM : 31.498046875 %RH
## Pressure
SENSOR_GP_PRES : 102429.328125 Pa
## Carbon Monoxide
SENSOR_GP_CO : 27.660348892211914 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.0025529060512781143 ppm
sensor:SHTPI001 1481037026 1481037026
true
===============================================================
Raw                                                 data                                                :
10,60,61,62,0,54,171,132,12,24,83,72,84,80,73,48,49,49,35,11,52,98,54,119,0,123,18,213,70,88,138,32,38,137,72,
144,92,143,164,65,146,0,128,250,65,147,18,43,200,71,129,9,207,98,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 11
Serial ID : 403473579
Waspmote ID : SHTPI011
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037074 Epoch Time
## Oxygen
SENSOR_GP_O2 : 280881 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.56999969482422 ºC
## Humidity
SENSOR_GP_HUM : 31.3125 %RH
## Pressure
SENSOR_GP_PRES : 102486.140625 Pa
## Carbon Monoxide
SENSOR_GP_CO : 56.70218276977539 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI011 1481037074 1481037074
true
===============================================================
Raw                                                 data                                                :
10,60,61,62,0,39,209,103,12,24,83,72,84,80,73,48,49,55,35,20,52,97,54,119,0,123,156,151,65,57,144,31,133,99,6
5,146,0,185,31,66,147,154,124,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 20
Serial ID : 403466193
Waspmote ID : SHTPI017
## Battery
SENSOR_BAT : 97 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960599964 Epoch Time
```

```
## Temperature Celcius
SENSOR_GP_TC : 14.220000267028809 ºC
## Humidity
SENSOR_GP_HUM : 39.9306640625 %RH
## Pressure
SENSOR_GP_PRES : 102649.203125 Pa
sensor:SHTPI017 960599964 960599964
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,54,105,254,10,24,83,72,84,80,73,48,49,52,35,17,52,98,54,119,0,123,54,213,70,88,138,7,133,137,72,
144,184,30,163,65,146,0,162,250,65,147,108,19,200,71,129,80,252,113,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 17
Serial ID : 403373673
Waspmote ID : SHTPI014
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037110 Epoch Time
## Oxygen
SENSOR_GP_O2 : 281640.21875 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.389999389648438 ºC
## Humidity
SENSOR_GP_HUM : 31.3291015625 %RH
## Pressure
SENSOR_GP_PRES : 102438.84375 Pa
## Carbon Monoxide
SENSOR_GP_CO : 60.49639892578125 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI014 1481037110 1481037110
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,54,65,252,10,24,83,72,84,80,73,48,48,53,35,17,52,98,54,119,0,123,54,213,70,88,138,49,226,132,72,
144,0,0,166,65,146,0,254,245,65,147,244,59,200,71,129,125,111,53,73,8,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 17
Serial ID : 403373121
Waspmote ID : SHTPI005
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037110 Epoch Time
## Oxygen
SENSOR_GP_O2 : 272145.53125 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.75 ºC
```

```
## Humidity
SENSOR_GP_HUM : 30.7490234375 %RH
## Pressure
SENSOR_GP_PRES : 102519.90625 Pa
## Carbon Monoxide
SENSOR_GP_CO : 743159.8125 ppm
sensor:SHTPI005 1481037110 1481037110
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,54,84,2,11,24,83,72,84,80,73,48,48,55,35,17,52,95,54,119,0,123,54,213,70,88,138,178,28,138,72,14
4,102,102,162,65,146,0,158,246,65,147,128,28,200,71,129,34,57,12,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 17
Serial ID : 403374676
Waspmote ID : SHTPI007
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037110 Epoch Time
## Oxygen
SENSOR_GP_O2 : 282853.5625 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.299999237060547 ºC
## Humidity
SENSOR_GP_HUM : 30.8271484375 %RH
## Pressure
SENSOR_GP_PRES : 102457 Pa
## Carbon Monoxide
SENSOR_GP_CO : 35.05579376220703 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI007 1481037110 1481037110
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,59,16,89,12,24,83,72,84,80,73,48,49,50,35,12,52,100,54,119,0,123,86,213,70,88,138,213,94,133,72,
144,61,10,163,65,146,0,28,248,65,147,249,251,199,71,129,54,139,158,65,137,0,0,0,0,134,90,50,228,58,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 12
Serial ID : 403462416
Waspmote ID : SHTPI012
## Battery
SENSOR_BAT : 100 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037142 Epoch Time
## Oxygen
SENSOR_GP_O2 : 273142.65625 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.3799991607666 ºC
```

```
## Humidity
SENSOR_GP_HUM : 31.013671875 %RH
## Pressure
SENSOR_GP_PRES : 102391.9453125 Pa
## Carbon Monoxide
SENSOR_GP_CO : 19.817974090576172 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.0017410025466233492 ppm
sensor:SHTPI012 1481037142 1481037142
true
=============================================================
Raw                                                 data                                                 :
10,60,61,62,0,39,189,251,10,24,83,72,84,80,73,48,49,54,35,21,52,95,54,119,0,123,216,133,65,57,144,61,10,107,6
5,146,0,205,29,66,147,212,35,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 21
Serial ID : 403372989
Waspmote ID : SHTPI016
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960595416 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.6899995803833 ºC
## Humidity
SENSOR_GP_HUM : 39.4501953125 %RH
## Pressure
SENSOR_GP_PRES : 102471.65625 Pa
sensor:SHTPI016 960595416 960595416
true
=============================================================
Raw                                                 data                                                 :
10,60,61,62,0,39,209,103,12,24,83,72,84,80,73,48,49,55,35,23,52,97,54,119,0,123,235,151,65,57,144,51,51,99,65,
146,0,12,34,66,147,157,106,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 23
Serial ID : 403466193
Waspmote ID : SHTPI017
## Battery
SENSOR_BAT : 97 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960600043 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.199999809265137 ºC
## Humidity
SENSOR_GP_HUM : 40.51171875 %RH
## Pressure
SENSOR_GP_PRES : 102613.2265625 Pa
```

```
sensor:SHTPI017 960600043 960600043
true
=============================================================
Raw                                                 data                                                :
10,60,61,62,0,39,189,251,10,24,83,72,84,80,73,48,49,54,35,22,52,95,54,119,0,123,238,133,65,57,144,61,10,107,6
5,146,0,204,29,66,147,152,33,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 22
Serial ID : 403372989
Waspmote ID : SHTPI016
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960595438 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.6899995803833 ºC
## Humidity
SENSOR_GP_HUM : 39.44921875 %RH
## Pressure
SENSOR_GP_PRES : 102467.1875 Pa
sensor:SHTPI016 960595438 960595438
true
=============================================================
Raw                                                 data                                                :
10,60,61,62,0,59,195,130,12,24,83,72,84,80,73,48,48,49,35,12,52,96,54,119,0,123,137,213,70,88,138,206,46,135,
72,144,92,143,164,65,146,0,170,250,65,147,182,14,200,71,129,111,185,218,65,137,0,0,0,0,134,47,253,22,59,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 12
Serial ID : 403473091
Waspmote ID : SHTPI001
## Battery
SENSOR_BAT : 96 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037193 Epoch Time
## Oxygen
SENSOR_GP_O2 : 276854.4375 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.56999969482422 ºC
## Humidity
SENSOR_GP_HUM : 31.3330078125 %RH
## Pressure
SENSOR_GP_PRES : 102429.421875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 27.340543746948242 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.0023039092775434256 ppm
sensor:SHTPI001 1481037193 1481037193
true
```

```
=============================================================
Raw                                                     data                                                    :
10,60,61,62,0,54,84,2,11,24,83,72,84,80,73,48,48,55,35,18,52,95,54,119,0,123,155,213,70,88,138,239,29,138,72,1
44,236,81,162,65,146,0,234,245,65,147,166,27,200,71,129,165,182,7,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 18
Serial ID : 403374676
Waspmote ID : SHTPI007
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037211 Epoch Time
## Oxygen
SENSOR_GP_O2 : 282863.46875 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.290000915527344 ºC
## Humidity
SENSOR_GP_HUM : 30.7392578125 %RH
## Pressure
SENSOR_GP_PRES : 102455.296875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 33.92836380004883 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI007 1481037211 1481037211
true
=============================================================
Raw                                                     data                                                    :
10,60,61,62,0,54,171,132,12,24,83,72,84,80,73,48,49,49,35,12,52,98,54,119,0,123,151,213,70,88,138,172,35,137,
72,144,225,122,164,65,146,0,64,248,65,147,118,35,200,71,129,172,157,82,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 12
Serial ID : 403473579
Waspmote ID : SHTPI011
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037207 Epoch Time
## Oxygen
SENSOR_GP_O2 : 280861.375 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.559999465942383 ºC
## Humidity
SENSOR_GP_HUM : 31.03125 %RH
## Pressure
SENSOR_GP_PRES : 102470.921875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 52.65397644042969 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
```

```
sensor:SHTPI011 1481037207 1481037207
true
============================================================
Raw                                                     data                                                    :
10,60,61,62,0,59,16,89,12,24,83,72,84,80,73,48,49,50,35,13,52,100,54,119,0,123,221,213,70,88,138,41,95,133,72,
144,195,245,162,65,146,0,214,247,65,147,54,5,200,71,129,106,65,146,65,137,0,0,0,0,134,41,121,222,58,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 13
Serial ID : 403462416
Waspmote ID : SHTPI012
## Battery
SENSOR_BAT : 100 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037277 Epoch Time
## Oxygen
SENSOR_GP_O2 : 273145.28125 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.3700008392334 ºC
## Humidity
SENSOR_GP_HUM : 30.9794921875 %RH
## Pressure
SENSOR_GP_PRES : 102410.421875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 18.281940460205078 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.001697336439974606 ppm
sensor:SHTPI012 1481037277 1481037277
true
============================================================
Raw                                                     data                                                    :
10,60,61,62,0,39,189,251,10,24,83,72,84,80,73,48,49,54,35,27,52,95,54,119,0,123,92,134,65,57,144,82,184,106,6
5,146,0,60,29,66,147,198,51,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 27
Serial ID : 403372989
Waspmote ID : SHTPI016
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960595548 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.670000076293945 ºC
## Humidity
SENSOR_GP_HUM : 39.30859375 %RH
## Pressure
SENSOR_GP_PRES : 102503.546875 Pa
sensor:SHTPI016 960595548 960595548
true
```

```
================================================================
Raw                                                     data    :
10,60,61,62,0,39,209,103,12,24,83,72,84,80,73,48,49,55,35,29,52,97,54,119,0,123,111,152,65,57,144,20,174,99,6
5,146,0,25,34,66,147,234,125,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 29
Serial ID : 403466193
Waspmote ID : SHTPI017
## Battery
SENSOR_BAT : 97 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960600175 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.229999542236328 ºC
## Humidity
SENSOR_GP_HUM : 40.5244140625 %RH
## Pressure
SENSOR_GP_PRES : 102651.828125 Pa
sensor:SHTPI017 960600175 960600175
true
================================================================
Raw                                                     data    :
10,60,61,62,0,54,105,254,10,24,83,72,84,80,73,48,49,52,35,19,52,98,54,119,0,123,0,214,70,88,138,145,130,137,7
2,144,195,245,162,65,146,0,0,252,65,147,150,18,200,71,129,231,30,103,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 19
Serial ID : 403373673
Waspmote ID : SHTPI014
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037312 Epoch Time
## Oxygen
SENSOR_GP_O2 : 281620.53125 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.3700008392334 ºC
## Humidity
SENSOR_GP_HUM : 31.5 %RH
## Pressure
SENSOR_GP_PRES : 102437.171875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 57.78017807006836 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI014 1481037312 1481037312
true
================================================================
Raw                                                     data    :
10,60,61,62,0,54,65,252,10,24,83,72,84,80,73,48,48,53,35,19,52,98,54,119,0,123,0,214,70,88,138,96,227,132,72,1
44,0,0,166,65,146,0,166,247,65,147,15,67,200,71,129,230,6,42,66,137,0,0,0,0,13,10
```

```
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 19
Serial ID : 403373121
Waspmote ID : SHTPI005
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037312 Epoch Time
## Oxygen
SENSOR_GP_O2 : 272155 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.75 ºC
## Humidity
SENSOR_GP_HUM : 30.9560546875 %RH
## Pressure
SENSOR_GP_PRES : 102534.1171875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 42.506736755371094 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI005 1481037312 1481037312
true
=============================================================
Raw                                                 data                                                :
10,60,61,62,0,54,84,2,11,24,83,72,84,80,73,48,48,55,35,19,52,95,54,119,0,123,0,214,70,88,138,118,27,138,72,144
,236,81,162,65,146,0,130,246,65,147,50,31,200,71,129,4,29,4,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 19
Serial ID : 403374676
Waspmote ID : SHTPI007
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037312 Epoch Time
## Oxygen
SENSOR_GP_O2 : 282843.6875 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.290000915527344 ºC
## Humidity
SENSOR_GP_HUM : 30.8134765625 %RH
## Pressure
SENSOR_GP_PRES : 102462.390625 Pa
## Carbon Monoxide
SENSOR_GP_CO : 33.02833557128906 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI007 1481037312 1481037312
true
=============================================================
Raw                                                 data                                                :
```

```
10,60,61,62,0,59,195,130,12,24,83,72,84,80,73,48,48,49,35,13,52,96,54,119,0,123,16,214,70,88,138,47,44,135,72,
144,236,81,164,65,146,0,24,251,65,147,179,19,200,71,129,63,187,193,65,137,0,0,0,0,134,248,156,20,59,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 13
Serial ID : 403473091
Waspmote ID : SHTPI001
## Battery
SENSOR_BAT : 96 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037328 Epoch Time
## Oxygen
SENSOR_GP_O2 : 276833.46875 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.540000915527344 ºC
## Humidity
SENSOR_GP_HUM : 31.38671875 %RH
## Pressure
SENSOR_GP_PRES : 102439.3984375 Pa
## Carbon Monoxide
SENSOR_GP_CO : 24.216428756713867 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.002267656847834587 ppm
sensor:SHTPI001 1481037328 1481037328
true
============================================================
Raw                                                 data                                                :
10,60,61,62,0,54,171,132,12,24,83,72,84,80,73,48,49,49,35,13,52,98,54,119,0,123,57,214,70,88,138,56,33,137,72,
144,236,81,164,65,146,0,150,248,65,147,16,35,200,71,129,24,65,89,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 13
Serial ID : 403473579
Waspmote ID : SHTPI011
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037369 Epoch Time
## Oxygen
SENSOR_GP_O2 : 280841.75 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.540000915527344 ºC
## Humidity
SENSOR_GP_HUM : 31.0732421875 %RH
## Pressure
SENSOR_GP_PRES : 102470.125 Pa
## Carbon Monoxide
SENSOR_GP_CO : 54.313568115234375 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
```

```
sensor:SHTPI011 1481037369 1481037369
true
============================================================
Raw                                                 data                                                :
10,60,61,62,0,59,16,89,12,24,83,72,84,80,73,48,49,50,35,14,52,100,54,119,0,123,100,214,70,88,138,123,95,133,7
2,144,195,245,162,65,146,0,174,246,65,147,188,8,200,71,129,28,133,137,65,137,0,0,0,0,134,203,122,226,58,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 14
Serial ID : 403462416
Waspmote ID : SHTPI012
## Battery
SENSOR_BAT : 100 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037412 Epoch Time
## Oxygen
SENSOR_GP_O2 : 273147.84375 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.3700008392334 ºC
## Humidity
SENSOR_GP_HUM : 30.8349609375 %RH
## Pressure
SENSOR_GP_PRES : 102417.46875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 17.18999481201172 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.0017279026797041297 ppm
sensor:SHTPI012 1481037412 1481037412
true
============================================================
Raw                                                 data                                                :
10,60,61,62,0,39,189,251,10,24,83,72,84,80,73,48,49,54,35,32,52,95,54,119,0,123,225,134,65,57,144,61,10,107,6
5,146,0,224,31,66,147,85,60,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 32
Serial ID : 403372989
Waspmote ID : SHTPI016
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960595681 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.6899995803833 ºC
## Humidity
SENSOR_GP_HUM : 39.96875 %RH
## Pressure
SENSOR_GP_PRES : 102520.6640625 Pa
sensor:SHTPI016 960595681 960595681
true
```

```
================================================================
Raw                                                    data                                                    :
10,60,61,62,0,54,65,252,10,24,83,72,84,80,73,48,48,53,35,20,52,98,54,119,0,123,101,214,70,88,138,96,227,132,7
2,144,0,0,166,65,146,0,42,246,65,147,118,67,200,71,129,95,1,37,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 20
Serial ID : 403373121
Waspmote ID : SHTPI005
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037413 Epoch Time
## Oxygen
SENSOR_GP_O2 : 272155 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.75 ºC
## Humidity
SENSOR_GP_HUM : 30.7705078125 %RH
## Pressure
SENSOR_GP_PRES : 102534.921875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 41.251338958740234 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI005 1481037413 1481037413
true
================================================================
Raw                                                    data                                                    :
10,60,61,62,0,39,209,103,12,24,83,72,84,80,73,48,49,55,35,37,52,97,54,119,0,123,57,153,65,57,144,225,122,100,
65,146,0,255,36,66,147,62,109,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 37
Serial ID : 403466193
Waspmote ID : SHTPI017
## Battery
SENSOR_BAT : 97 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960600377 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.279999732971191 ºC
## Humidity
SENSOR_GP_HUM : 41.2490234375 %RH
## Pressure
SENSOR_GP_PRES : 102618.484375 Pa
sensor:SHTPI017 960600377 960600377
true
================================================================
Raw                                                    data                                                    :
10,60,61,62,0,54,65,252,10,24,83,72,84,80,73,48,48,53,35,21,52,98,54,119,0,123,202,214,70,88,138,49,226,132,7
2,144,133,235,165,65,146,0,132,246,65,147,157,61,200,71,129,132,231,31,66,137,0,0,0,0,13,10
```

```
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 21
Serial ID : 403373121
Waspmote ID : SHTPI005
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037514 Epoch Time
## Oxygen
SENSOR_GP_O2 : 272145.53125 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.739999771118164 ºC
## Humidity
SENSOR_GP_HUM : 30.814453125 %RH
## Pressure
SENSOR_GP_PRES : 102523.2265625 Pa
## Carbon Monoxide
SENSOR_GP_CO : 39.97608947753906 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI005 1481037514 1481037514
true
=============================================================
Raw                                                    data                                                    :
10,60,61,62,0,59,195,130,12,24,83,72,84,80,73,48,48,49,35,14,52,96,54,119,0,123,189,214,70,88,138,25,41,135,7
2,144,246,40,164,65,146,0,252,250,65,147,223,14,200,71,129,146,79,199,65,137,0,0,0,0,134,190,23,9,59,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 14
Serial ID : 403473091
Waspmote ID : SHTPI001
## Battery
SENSOR_BAT : 96 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037501 Epoch Time
## Oxygen
SENSOR_GP_O2 : 276808.78125 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.520000457763672 ºC
## Humidity
SENSOR_GP_HUM : 31.373046875 %RH
## Pressure
SENSOR_GP_PRES : 102429.7421875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 24.91385269165039 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.0020918692462146282 ppm
sensor:SHTPI001 1481037501 1481037501
true
```

```
================================================================
Raw                                                     data    :
10,60,61,62,0,54,171,132,12,24,83,72,84,80,73,48,49,49,35,14,52,98,54,119,0,123,223,214,70,88,138,195,30,137,
72,144,246,40,164,65,146,0,88,249,65,147,132,40,200,71,129,46,232,89,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 14
Serial ID : 403473579
Waspmote ID : SHTPI011
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037535 Epoch Time
## Oxygen
SENSOR_GP_O2 : 280822.09375 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.520000457763672 ºC
## Humidity
SENSOR_GP_HUM : 31.16796875 %RH
## Pressure
SENSOR_GP_PRES : 102481.03125 Pa
## Carbon Monoxide
SENSOR_GP_CO : 54.47673797607422 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI011 1481037535 1481037535
true
================================================================
Raw                                                     data    :
10,60,61,62,0,39,209,103,12,24,83,72,84,80,73,48,49,55,35,40,52,97,54,119,0,123,136,153,65,57,144,205,204,100
,65,146,0,218,34,66,147,25,121,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 40
Serial ID : 403466193
Waspmote ID : SHTPI017
## Battery
SENSOR_BAT : 97 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960600456 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.300000190734863 ºC
## Humidity
SENSOR_GP_HUM : 40.712890625 %RH
## Pressure
SENSOR_GP_PRES : 102642.1953125 Pa
sensor:SHTPI017 960600456 960600456
true
================================================================
Raw                                                     data    :
10,60,61,62,0,59,16,89,12,24,83,72,84,80,73,48,49,50,35,15,52,100,54,119,0,123,235,214,70,88,138,123,95,133,7
2,144,195,245,162,65,146,0,56,247,65,147,64,1,200,71,129,10,89,131,65,137,0,0,0,0,134,184,48,224,58,13,10
```

```
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 15
Serial ID : 403462416
Waspmote ID : SHTPI012
## Battery
SENSOR_BAT : 100 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037547 Epoch Time
## Oxygen
SENSOR_GP_O2 : 273147.84375 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.3700008392334 ºC
## Humidity
SENSOR_GP_HUM : 30.90234375 %RH
## Pressure
SENSOR_GP_PRES : 102402.5 Pa
## Carbon Monoxide
SENSOR_GP_CO : 16.418476104736328 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.0017104363068938255 ppm
sensor:SHTPI012 1481037547 1481037547
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,54,65,252,10,24,83,72,84,80,73,48,48,53,35,22,52,98,54,119,0,123,47,215,70,88,138,133,226,132,7
2,144,0,0,166,65,146,0,100,247,65,147,128,63,200,71,129,146,133,27,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 22
Serial ID : 403373121
Waspmote ID : SHTPI005
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037615 Epoch Time
## Oxygen
SENSOR_GP_O2 : 272148.15625 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.75 ºC
## Humidity
SENSOR_GP_HUM : 30.923828125 %RH
## Pressure
SENSOR_GP_PRES : 102527 Pa
## Carbon Monoxide
SENSOR_GP_CO : 38.88043975830078 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI005 1481037615 1481037615
true
```

```
=================================================================
Raw                                                    data                                                    :
10,60,61,62,0,54,84,2,11,24,83,72,84,80,73,48,48,55,35,22,52,95,54,119,0,123,47,215,70,88,138,58,26,138,72,144
,246,40,162,65,146,0,78,247,65,147,132,32,200,71,129,36,143,241,65,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 22
Serial ID : 403374676
Waspmote ID : SHTPI007
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037615 Epoch Time
## Oxygen
SENSOR_GP_O2 : 282833.8125 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.270000457763672 ºC
## Humidity
SENSOR_GP_HUM : 30.9130859375 %RH
## Pressure
SENSOR_GP_PRES : 102465.03125 Pa
## Carbon Monoxide
SENSOR_GP_CO : 30.19489288330078 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI007 1481037615 1481037615
true
=================================================================
Raw                                                    data                                                    :
10,60,61,62,0,54,171,132,12,24,83,72,84,80,73,48,49,49,35,15,52,98,54,119,0,123,100,215,70,88,138,31,32,137,7
2,144,246,40,164,65,146,0,188,250,65,147,186,38,200,71,129,50,145,65,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 15
Serial ID : 403473579
Waspmote ID : SHTPI011
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037668 Epoch Time
## Oxygen
SENSOR_GP_O2 : 280832.96875 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.520000457763672 ºC
## Humidity
SENSOR_GP_HUM : 31.341796875 %RH
## Pressure
SENSOR_GP_PRES : 102477.453125 Pa
## Carbon Monoxide
SENSOR_GP_CO : 48.39179229736328 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
```

```
sensor:SHTPI011 1481037668 1481037668
true
============================================================
Raw                                                     data                                                    :
10,60,61,62,0,59,195,130,12,24,83,72,84,80,73,48,48,49,35,15,52,96,54,119,0,123,98,215,70,88,138,76,43,135,72,
144,113,61,164,65,146,0,156,251,65,147,104,18,200,71,129,118,60,192,65,137,0,0,0,0,134,102,156,13,59,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
Separator : 35
Frame Sequence : 15
Serial ID : 403473091
Waspmote ID : SHTPI001
## Battery
SENSOR_BAT : 96 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037666 Epoch Time
## Oxygen
SENSOR_GP_O2 : 276826.375 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.530000686645508 ºC
## Humidity
SENSOR_GP_HUM : 31.451171875 %RH
## Pressure
SENSOR_GP_PRES : 102436.8125 Pa
## Carbon Monoxide
SENSOR_GP_CO : 24.029521942138672 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.0021608113311231136 ppm
sensor:SHTPI001 1481037666 1481037666
true
============================================================
Raw                                                     data                                                    :
10,60,61,62,0,39,189,251,10,24,83,72,84,80,73,48,49,54,35,43,52,95,54,119,0,123,240,135,65,57,144,41,92,107,6
5,146,0,69,33,66,147,100,49,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 43
Serial ID : 403372989
Waspmote ID : SHTPI016
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960595952 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.710000038146973 ºC
## Humidity
SENSOR_GP_HUM : 40.3173828125 %RH
## Pressure
SENSOR_GP_PRES : 102498.78125 Pa
sensor:SHTPI016 960595952 960595952
true
```

```
==================================================================
Raw                                                 data                                                 :
10,60,61,62,0,39,209,103,12,24,83,72,84,80,73,48,49,55,35,45,52,97,54,119,0,123,3,154,65,57,144,174,71,101,65,
146,0,68,34,66,147,15,125,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 45
Serial ID : 403466193
Waspmote ID : SHTPI017
## Battery
SENSOR_BAT : 97 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960600579 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.329999923706055 ºC
## Humidity
SENSOR_GP_HUM : 40.56640625 %RH
## Pressure
SENSOR_GP_PRES : 102650.1171875 Pa
sensor:SHTPI017 960600579 960600579
true
==================================================================
Raw                                                 data                                                 :
10,60,61,62,0,39,189,251,10,24,83,72,84,80,73,48,49,54,35,44,52,95,54,119,0,123,6,136,65,57,144,41,92,107,65,1
46,0,69,34,66,147,134,67,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 44
Serial ID : 403372989
Waspmote ID : SHTPI016
## Battery
SENSOR_BAT : 95 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960595974 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.710000038146973 ºC
## Humidity
SENSOR_GP_HUM : 40.5673828125 %RH
## Pressure
SENSOR_GP_PRES : 102535.046875 Pa
sensor:SHTPI016 960595974 960595974
true
==================================================================
Raw                                                 data                                                 :
10,60,61,62,0,54,105,254,10,24,83,72,84,80,73,48,49,52,35,23,52,98,54,119,0,123,148,215,70,88,138,117,129,137
,72,144,72,225,162,65,146,0,64,252,65,147,32,30,200,71,129,203,128,82,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 23
```

```
Serial ID : 403373673
Waspmote ID : SHTPI014
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037716 Epoch Time
## Oxygen
SENSOR_GP_O2 : 281611.65625 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.360000610351562 ºC
## Humidity
SENSOR_GP_HUM : 31.53125 %RH
## Pressure
SENSOR_GP_PRES : 102460.25 Pa
## Carbon Monoxide
SENSOR_GP_CO : 52.62577438354492 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI014 1481037716 1481037716
true
============================================================
Raw                                                 data                                                :
10,60,61,62,0,54,65,252,10,24,83,72,84,80,73,48,48,53,35,23,52,98,54,119,0,123,148,215,70,88,138,141,229,132,
72,144,123,20,166,65,146,0,170,247,65,147,176,74,200,71,129,249,221,21,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 23
Serial ID : 403373121
Waspmote ID : SHTPI005
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037716 Epoch Time
## Oxygen
SENSOR_GP_O2 : 272172.40625 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.760000228881836 ºC
## Humidity
SENSOR_GP_HUM : 30.9580078125 %RH
## Pressure
SENSOR_GP_PRES : 102549.375 Pa
## Carbon Monoxide
SENSOR_GP_CO : 37.46677017211914 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI005 1481037716 1481037716
true
============================================================
Raw                                                 data                                                :
10,60,61,62,0,59,16,89,12,24,83,72,84,80,73,48,49,50,35,16,52,100,54,119,0,123,152,215,70,88,138,174,96,133,7
2,144,61,10,163,65,146,0,72,248,65,147,231,14,200,71,129,84,222,138,65,137,0,0,0,0,134,52,185,236,58,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
```

```
Separator : 35
Frame Sequence : 16
Serial ID : 403462416
Waspmote ID : SHTPI012
## Battery
SENSOR_BAT : 100 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037720 Epoch Time
## Oxygen
SENSOR_GP_O2 : 273157.4375 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.3799991607666 ºC
## Humidity
SENSOR_GP_HUM : 31.03515625 %RH
## Pressure
SENSOR_GP_PRES : 102429.8046875 Pa
## Carbon Monoxide
SENSOR_GP_CO : 17.358558654785156 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.0018060565926134586 ppm
sensor:SHTPI012 1481037720 1481037720
true
=============================================================
Raw                                                 data                                                 :
10,60,61,62,0,39,209,103,12,24,83,72,84,80,73,48,49,55,35,48,52,97,54,119,0,123,69,154,65,57,144,184,30,101,6
5,146,0,222,38,66,147,203,119,200,71,13,10
length : 47
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 39
Separator : 35
Frame Sequence : 48
Serial ID : 403466193
Waspmote ID : SHTPI017
## Battery
SENSOR_BAT : 97 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 960600645 Epoch Time
## Temperature Celcius
SENSOR_GP_TC : 14.319999694824219 ºC
## Humidity
SENSOR_GP_HUM : 41.716796875 %RH
## Pressure
SENSOR_GP_PRES : 102639.5859375 Pa
sensor:SHTPI017 960600645 960600645
true
=============================================================
Raw                                                 data                                                 :
10,60,61,62,0,54,171,132,12,24,83,72,84,80,73,48,49,49,35,16,52,98,54,119,0,123,233,215,70,88,138,171,29,137,
72,144,0,0,164,65,146,0,122,248,65,147,186,34,200,71,129,183,46,51,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 16
```

```
Serial ID : 403473579
Waspmote ID : SHTPI011
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037801 Epoch Time
## Oxygen
SENSOR_GP_O2 : 280813.34375 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.5 ºC
## Humidity
SENSOR_GP_HUM : 31.0595703125 %RH
## Pressure
SENSOR_GP_PRES : 102469.453125 Pa
## Carbon Monoxide
SENSOR_GP_CO : 44.79561996459961 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI011 1481037801 1481037801
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,54,65,252,10,24,83,72,84,80,73,48,48,53,35,24,52,98,54,119,0,123,249,215,70,88,138,92,228,132,7
2,144,123,20,166,65,146,0,100,245,65,147,146,56,200,71,129,50,89,17,66,137,0,0,0,0,13,10
length : 62
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 54
Separator : 35
Frame Sequence : 24
Serial ID : 403373121
Waspmote ID : SHTPI005
## Battery
SENSOR_BAT : 98 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037817 Epoch Time
## Oxygen
SENSOR_GP_O2 : 272162.875 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.760000228881836 ºC
## Humidity
SENSOR_GP_HUM : 30.673828125 %RH
## Pressure
SENSOR_GP_PRES : 102513.140625 Pa
## Carbon Monoxide
SENSOR_GP_CO : 36.33710479736328 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
sensor:SHTPI005 1481037817 1481037817
true
============================================================
Raw                                                 data                                                 :
10,60,61,62,0,59,195,130,12,24,83,72,84,80,73,48,48,49,35,16,52,96,54,119,0,123,233,215,70,88,138,23,42,135,7
2,144,246,40,164,65,146,0,26,250,65,147,162,12,200,71,129,255,202,171,65,137,0,0,0,0,134,109,253,12,59,13,10
length : 67
Start Delimiter : <=>
Frame Type : 0
Number Of Bytes : 59
```

```
Separator : 35
Frame Sequence : 16
Serial ID : 403473091
Waspmote ID : SHTPI001
## Battery
SENSOR_BAT : 96 %
## TimeStamp (Unix/Epoch)
SENSOR_TST : 1481037801 Epoch Time
## Oxygen
SENSOR_GP_O2 : 276816.71875 ppm
## Temperature Celcius
SENSOR_GP_TC : 20.520000457763672 ºC
## Humidity
SENSOR_GP_HUM : 31.2626953125 %RH
## Pressure
SENSOR_GP_PRES : 102425.265625 Pa
## Carbon Monoxide
SENSOR_GP_CO : 21.474119186401367 ppm
## Nitrogen Dioxide
SENSOR_GP_NO2 : 0 ppm
## Hydrogen Cyanide
SENSOR_GP_HCN : 0.002151335822418332 ppm
sensor:SHTPI001 1481037801 1481037801
true
```

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

재난 대응 M2M 시스템
전문가 평가 보고서



M2M 기반 지하공간 (지하철) 재난대응
대화형 스마트 네트워크 시스템 개발
M2M-based Underground(subway) Disaster Response
interactive Smart Network System

I 연구목표

1.1 연구목표 및 내용

□ M2M 기반 지하시설(지하철) 재난대응 프로세스(SOP) 개발

- 승객 정보 제공용 비주얼 E-book 제작
  - 기존 재난유형, 안전시설, 재난대응 분석 및 정리
  - 국가화재안전기준 및 NFPA 참조 및 국내외 사례조사
  - 일러스트레이션 위주로 제작하여 정보 전달 효율성 제고
- 국내외 방재전문가 참여 SOP 개발 진행
  - 국내 지하철/건축/토목 분야 방재전문가 참여 SOP 개발
  - 스웨덴 Lund Univ. Fire Safety 학과 방재전문가 기술자문
  - M2M 기반 재난 유형별 대응 전략 수립 및 적정성 평가
  - M2M 기반 재난 대응 시스템의 차별화를 위한 기존 재난대응 행동매뉴얼 및 방재기술에 대한 개선사항 도출

□ SOP기반 재난 대응 프로세스 및 재난전조대응 M2M 알고리즘 개발

- 방재스마트N키퍼 상황인지 및 의사결정 알고리즘 개발
  - 스마트N키퍼를 활용한 재난대응 시스템 관련 논문 작성
- 방재스마트N키퍼 상황인지 정보공유모델 개발 및 구현
  - 위험도 수준 그래프에서 각종 센서 기여도 확인기능 구현
  - 상황인지, 의사결정, 정보공유 기능 내장 S/W α버젼 제작

□ 스마트N키퍼 개발

- 무선 센서링 노드 성능 개선
  - 사전보정된 저전력, 선형응답의 전기화학식 센서를 사용하여 화재, 가스 탐지 정밀도 향상
  - OTA를 이용한 오작동 센서 복원 기능, 센서 수명 연장을 위한 슬립/논슬립 기능 구현

• 배터리 탈부착, 노드 리셋이 용이한 전용 케이스 제작
• IoT 기반 지진계, 수위계 운용 타당성 검토

- 스마트 네트워크 관리 시스템 구성을 위한 통신 기능 개발
  • 스마트N키퍼 통신모듈 β버젼 제작
- 재난상황 발생 이후 신속한 SOP 전개기능 개발
  • 재난상황별 자동알람 기능을 통한 휴먼에러 최소화
  • 스마트N키퍼 SOP모듈 β버젼 제작
- 지하철 승객 피난경로 분석 및 대피 유도 기능 개발
  • 스마트N키퍼 피난경로 분석/대피 유도모듈 β버젼 제작
- 센서 정보 수집 및 효율적인 포워딩 기능 개발
  • 스마트N키퍼 관리모듈 β버젼 제작

□ 스마트N키퍼 테스트베드 대상 확대

- 지하3층 이상 규모의 지하철역 선정 테스트베드 운영
  • 무선 센서링 노드는 벽체 중간부에 설치하여 테스트
  • 유사화재상황에서의 스마트N키퍼 작동 테스트
- 대화형 스마트 네트워크 관리시스템 α버젼 개발
  • REST API를 이용한 웹기반 관리시스템 개발
  • 기존 재난대응시스템 데이터 메쉬업 가능여부 조사
  • 테스트베드들 통합운영을 통한 스마트N키퍼 기능 향상






Ⅱ 설계기준 시방서, 지침에 제안

□ 추후 본 과제의 사업화를 위하여 현재 소방법으로 제정되어 의무적으로 설치해야 하는 소방방재장비를 대신하여 스마트N키퍼의 설치를 의무화하기 위한 관련법 법령을 찾아 기준에 맞게 제안하고자 함.

2.1 재난 및 안전관리 기본법

제4장 재난의 예방

제26조 (재난관리책임기관의 장의 재난예방조치)

① 재난관리책임기관의 장은 소관 관리대상업무의 분야에서 재난의 발생을 사전에 방지하기 위하여 다음 각 호의 조치를 하여야 한다. <개정 2007.1.26.>

1. 재난에 대응할 조직의 구성 및 정비
2. 재난의 예측과 정보전달체계의 구축
3. 재난발생에 대비한 교육·훈련과 재난관리예방에 관한 홍보
4. 재난발생의 위험이 높은 분야에 대한 안전관리체계의 구축 및 안전관리규정의 제정
5. 재난발생의 위험이 높거나 재난예방을 위하여 계속적으로 관리할 필요가 있다고 인정되는 시설 및 지역(이하 "특정관리대상시설등"이라 한다)의 지정·관리 및 정비
6. 제35조의 규정에 의한 물자 및 자재의 비축, 재난방지시설의 정비와 장비 및 인력의 지정
7. 제25조의2의 규정에 따라 지정된 국가기반시설의 관리

제5장 응급대책

제37조 (응급조치)

① 제50조제2항의 규정에 의한 시·도긴급구조통제단 및 시·군·구긴급구조통제단의 단장(이하 "지역통제단장"이라 한다)과 시장·군수·구청장은 재난이 발생할 우려가 있거나 재난이 발생한 때에는 즉시 관계법령이나 시·도 또는 시·군·구의 안전관리계획이 정하는 바에 의하여 수방(수방)·진화·구조 및 구난(구난) 그 밖에 재난의 발생을 예방하거나 피해를 경감하기 위하여 필요한 다음 각호의 응급조치를 실시하여야 한다. 다만, 지역통제단장의 경우에는 제2호중 진화에 관한 응급조치와 제4호 및 제6호의 응급조치에 한한다.

1. **경보의 발령 또는 전달이나 피난의 권고 또는 지시**
2. 진화·수방·지진방재 그 밖의 응급조치와 구호
3. 피해시설의 응급복구 및 방역과 방범 그 밖의 질서의 유지
4. **긴급수송 및 구조 수단의 확보**
5. 급수수단의 확보, 긴급피난처 및 구호품의 확보
6. 현장지휘통신체계의 확보
7. 그 밖에 재난의 발생을 예방하거나 경감하기 위하여 필요한 사항

② 시·군·구의 관할구역안에 소재하는 재난관리책임기관의 장은 시장·군수·구청장 또는 지역통제단장의 요청이 있는 때에는 관계법령 또는 시·군·구안전관리계획이 정하는 바에 의하여 시장·군수·구청장 또는 지역통제단장의 지휘 또는 조정하에 그 소관업무에 관계되는 응급조치를 실시하거나 시장·군수·구청장 또는 지역통제단장이 실시하는 응급조치에 협력하여야 한다.

2.2 소방시설 설치 · 유지 및 안전관리에 관한 법률

제21조의2(피난계획의 수립 및 시행)

① 제20조제2항에 따른 **소방안전관리대상물의 관계인은 그 장소에 근무하거나 거주 또는 출입하는 사람들이 화재가 발생한 경우에 안전하게 피난할 수 있도록 피난계획을 수립하여 시행하여야 한다.**

② 제1항의 **피난계획에는 그 특정소방대상물의 구조, 피난시설 등을 고려하여 설정한 피난경로가 포함되어야 한다.**

③ 제1항의 **소방안전관리대상물의 관계인은 피난시설의 위치, 피난경로 또는 대피요령이 포함된 피난유도 안내정보를 근무자 또는 거주자에게 정기적으로 제공하여야 한다.**

④ 제1항에 따른 피난계획의 수립·시행, 제3항에 따른 피난유도 안내정보 제공에 필요한 사항은 총리령으로 정한다.

[본조신설 2014.12.30.]

Ⅲ 스마트N키퍼 SW 개요

3.1 개요

□ 진행사항

- 스마트N키퍼의 서버 및 클라이언트에 대한 전체 시스템 분석 및 설계과정을 통해서 충분한 요구사항 반영 및 적절한 기능 구현을 위한 시스템 설계
- SW개발에 필요한 세부 모듈 별 개발을 진행 중이며, 단위 모듈 별로 성과 도출

□ 개발성과

- 스마트N키퍼 안내 영상
- 스마트N키퍼 분석 및 설계도서
- 상황인지 알고리즘
- 피난유도 알고리즘
- 관리기능 화면UI
- 가상데이터를 이용한 스마트N키퍼 테스트

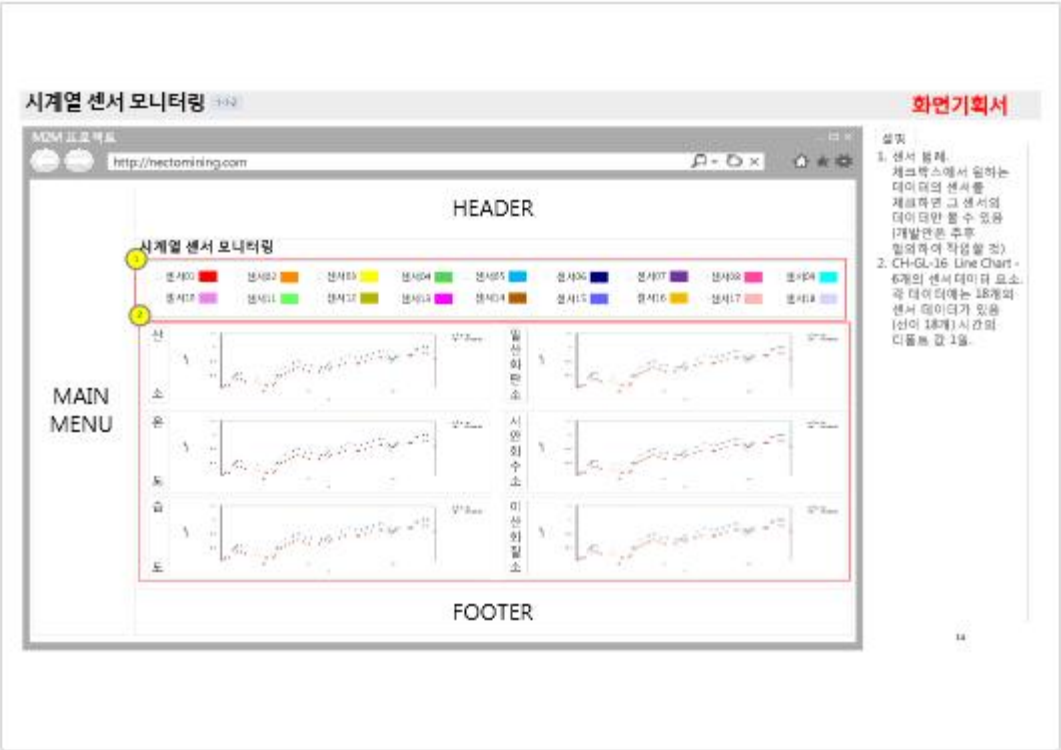
3.2 재난전조 시스템 UI 설계

□ 개요

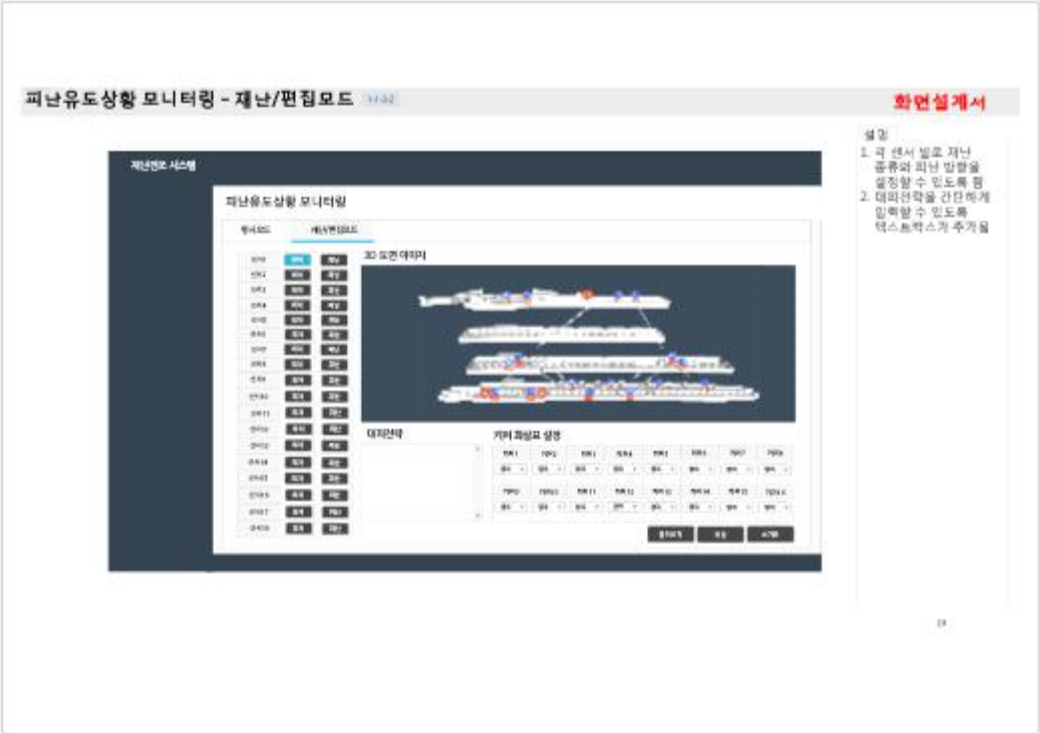
- 1차년도에서 작성한 지하철 재난 안전 교육용 영상 내용에 추가적으로 스마트N키퍼의 동작 설명 및 상황발생 시 대응 방안 내용 추가를 위한 시스템 설명 E-BOOK 제작
- 기본 내용 제작 후에 내용 검증을 거치고 최종적으로는 일러스트적인 요소를 추가한 비주얼한 E-BOOK 완성 후 스마트N키퍼 클라이언트에 탑재



<그림. 실시간 센서 모니터링 capture>



<그림. 시계열 센서 모니터링 capture>



<그림. 피난유도설정 모니터링 capture>



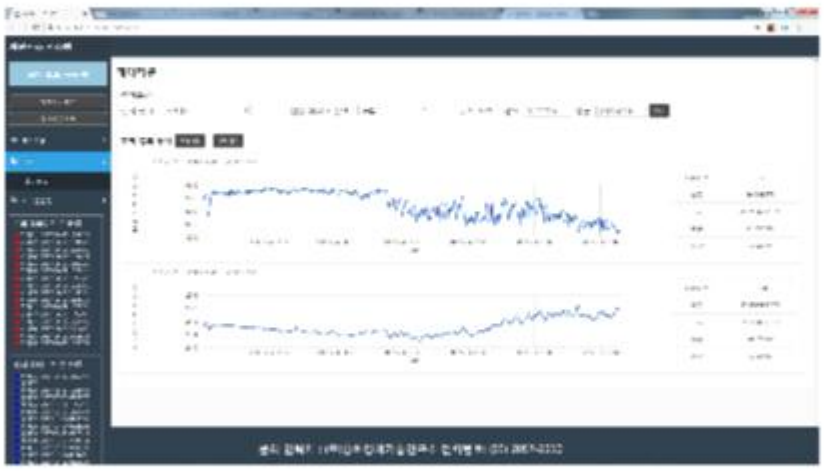
<그림. 피난유도상황 모니터링 평시모드>

재난 상황 판단 화면캡쳐



<그림. 재난상황 판단 모듈>

데이터뷰 화면캡쳐



<그림. 데이터 뷰>

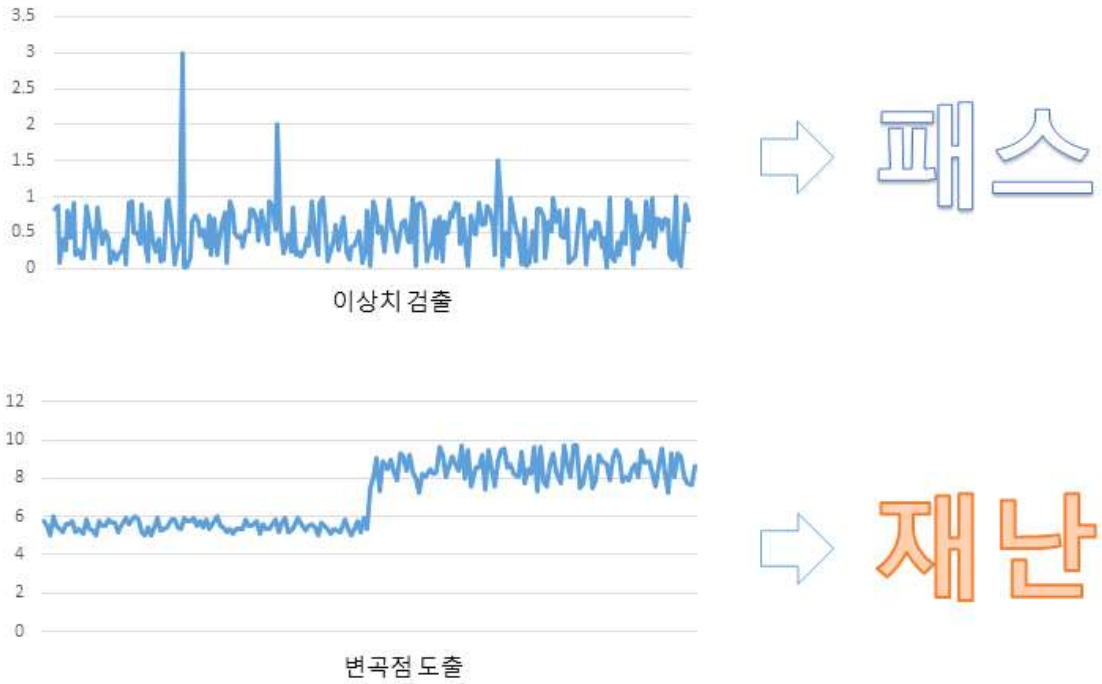
3.3 상황인지 알고리즘 및 모듈 개발

□ 개요

- 상황인지에 대한 정의 및 SOP에 대한 시나리오 검토 후 가상 데이터를 이용한 상황인지 모델 개발 및 SW 구현

□ 재난전조 감지 기술

- 실시간 센서 데이터를 감지 중 재난 발생이라고 생각되는 데이터의 변화 발생시 이를 분석하여 재난 발생이라고 판단하는 기술



<그림. 상황인지 개념도>

- 재난 종류로 화재 및 가스 발생을 선정했으며, 상황으로는 이상과 피난의 두 가지를 정하였음
- 이상은 원인 파악 및 점검이 필요한 상황이며, 피난은 즉각 대피를 해야 하는 상황으로 상정

□ 상황인지 알고리즘 개발을 위한 기계학습 알고리즘 분석 및 검토

- 센서 데이터를 바탕으로 반복적으로 정상, 전조, 위험이라는 3단계의 영역에 대한 상황인지에 대한 학습을 수행함
- 학습을 통해서 최종적으로는 알고리즘에서 사용가능하도록 상황룰셋이 도출되어야 함
- 상황룰셋[1)]
  - ◦ 변수종류, 변수의 범위, 조건 등의 조합으로 분기되어야 함
  - ◦ 상황판단을 체계화하기 위해서 구조화되어야 함

- 학습모델
  - ◦ 의사결정 나무를 이용한 회귀트리(검토중)
- 예측모델을 적용하기 위해서는 학습 데이터를 생성해야 함
- 목적변수[2)]
  - ◦ 상태(정상, 전조, 재난)
  - ◦ 재난종류

- 입력변수
  - ◦ 센서 데이터 : 센서 종류, 센서 값[3)], 센서
  - ◦ 센서 위치(검토중)[4)]
  - ◦ 센서 데이터는 시계열데이터이므로 시계열 데이터를 입력변수로 변환하기 위한 변환과정이 필요

---

1) 상황룰셋은 상황제어에서 사용할 수 있는 코드화된 결과물로서 도출필요. 블랙박스형이 아닌, 논리적인 설명이 가능한 방식이 필요

2) 학습에서 재난종류 및 상태까지 동시에 예측을 해야 한다면, 변수는 총 재난종류 * 상태(3)가 되어야 함. 이를테면 가스전조, 화재전조, 화재확산 등이 됨. 재난 종류를 범위 확정 필요

3) 단순한 센서의 값에 대한 정보를 이용한 상황결정은 학습을 통해서 도출될 수도 있고, 사전 파라미터에 의해서 결정될 수도 있음. 사용자의 의사결정을 반영하기 위한 파라미터와 학습 결과가 상충될 경우에 이에 대한 의사결정을 어딘가 프로세스에 반영을 해야할 필요가 있음

4) 센서의 위치 정보를 학습데이터로 사용하기 위해서는 공간이 상황인지에 어떤 영향을 미치는지에 대한 리서치 및 팩터 추출이 필요

- 일반적인 시계열 분석 방법에서 시간에 대한 연속적인 데이터임
- 특정시점에서 연속적인 데이터를 이용해서 예측모델에 의한 분류를 위해서는 연속데이터를 이산화하고 이를 예측모델의 입력 변수로 사용하기 위한 방법론이 필요

- 검토중인 방법론
  - ◦ 시계열분석 방법(ARIMA[5])
    - 시점에서의 관측값이 모델을 따르는지에 대한 여부 판별
    - 다중데이터에 대한 ARIMA 모델 적용 방법론[6] 필요
    - 시간 t에서의 측정데이터가 정상인지 비정상인지에 대한 판별값 도출
  - ◦ 확률분포에 의한 센서간 경계치 도출
    - 각 센서들의 과거 일정기간의 데이터를 이용한 확률 분포 생성
    - 확률분포에 의해서 센서별로 정상인 값의 범위를 도출 가능

- 의사결정나무
  - ◦ 관측값과 목표값을 연결시켜주는 예측모델
  - ◦ 결정트리를 사용
    - 분류트리 - 목적변수가 이산적일 때
    - 회귀트리 - 목적변수가 연속적일 때

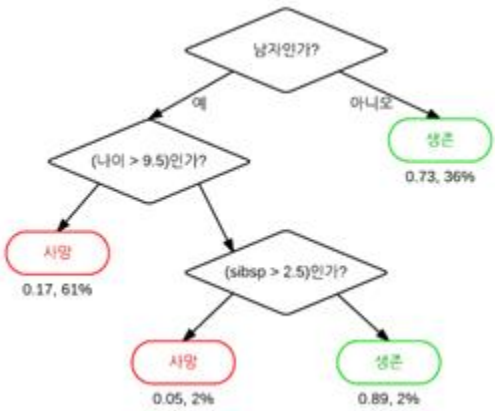
  - ◦ 분류 방법

---

5) Auto-Regressive Integrated Moving-Avearge model. 시계열 데이터 y(t)의 과거치들이 설명변수인 AR과 과거의 오차항들이 설명변수인 MA 모델의 합성
*AR(1) : 과거의 패턴이 지속되면 y(t)는 과거 관측치y(t-1)에 의해 예측될 수 있다. 어느정도까지 멀리 있는 과거 관측치 사용 가중치 적용 필요
**MA : y(t)의 값이 과거오차 e(t-1),,,들에 의해서 설명이 될 때 MA(q)-차수가 q인 moving-average(이동평균) 모형을 따른다고 함

6) 복합센서 데이터에 대해서 전체 데이터의 경향을 나타내기 위한 상태정보를 추출하기 위한 방법론 필요. 주성분분석을 이용하여 데이터 종류를 줄여서 모델 예측에 필요한 연산시간을 줄일 수 있음

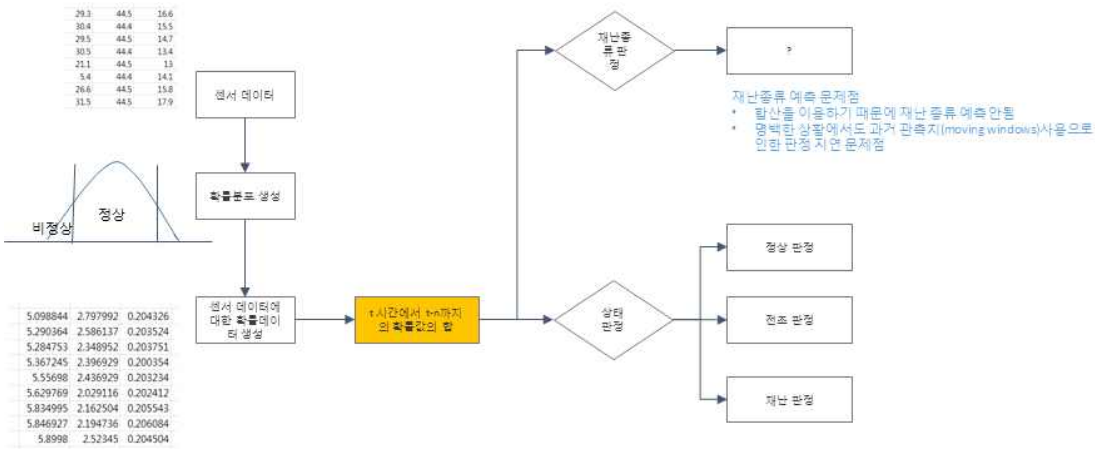
- 범주형 - 지니지수[7](M2M에서는 범주형(정상, 전조, 재난)임)
- 연속형 - 분산의 감소량



<그림. 분류 트리 예>

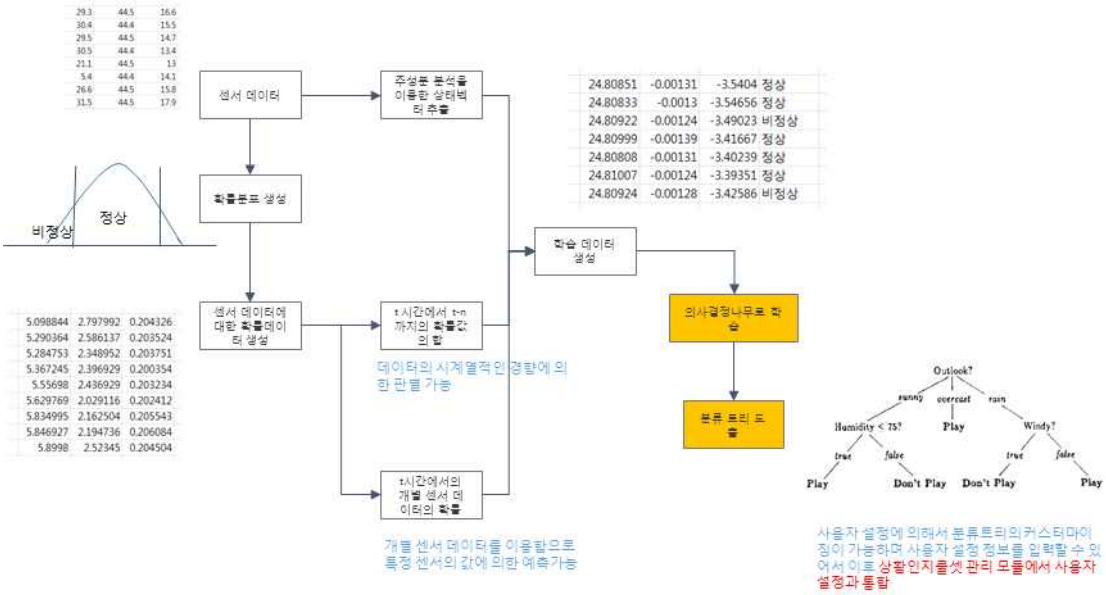
<표. 의사결정나무 분기트리 코드화 예>

내부 트리	분할
Age >= 30	Age >= 30 and Gender = Male
	Age >= 30 and Gender = Female
Age < 30	Age < 30 and Gender = Male
	Age < 30 and Gender = Female



<그림. 센서 데이터에 대한 확률 분포를 이용한 시계열 예측모델>

7) 분순도(impurity)를 측정지수로 사용



<그림. 센서 데이터에 대한 MA 혹은 선형회귀 오차를 이용한 의사결정나무 예측모델>

3.4 피난유도 기능 개발

□ 피난유도 기능개발

- 알고리즘 개요
  - 그래프 구조의 각 노드에서 시작점(sink)로 이동하기 위한 최소거리 및 경로를 노드의 웨이트를 이용해서 산출할 수 있도록 한 node-weighted spanning tree
  - spanning tree(신장 트리)의 일종임
    - 신장트리는 노드V 에 대해서 에지(E)의 개수가 |V| - 1인 트리
  - 너비우선탐색(BFS), 깊이우선탐색(DFS)와 유사하게 각 노드를 탐색하면서 트리를 구현
    - 기본적으로 BFS의 알고리즘을 차용하며, 중복적으로 각 노드의 웨이트를 비교해서 최소 웨이트를 산출
- 계산량
  - 현재 $\Theta \quad v!$ 계산량
  - 초기 버전에서는 계산량을 고려하지 않고 알고리즘 개발, 차후 속도를 보고 개선
- 입력 데이터 형식
  - 인접 리스트(adjacency list) 형태
- 알고리즘 비교
  - MST와의 차이
    - 도로의 관점에서 본다면 기존 도시 도로망에서 MST(minimum spanning tree)는 각 수요처를 연결하는 최소 도로 길이를 구하는 관점에서 볼 수 있고, MNWST는 각 지점에서 특정 지역으로 최단거리 이동하기 위한 주요

도로를 지정 하는것으로 이해를 하면 됨

- BFS 와의 차이점
  - DFS는 특정 노드를 큐에 넣은 후에 다시 확인을 하지 않으나 본 알고리즘은 최소 가중치를 구해야하기 때문에 반복적으로 계산을 수행

□ 알고리즘

- MNWST(G,s)

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

-Minimum Node Weight Spanning Tree

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

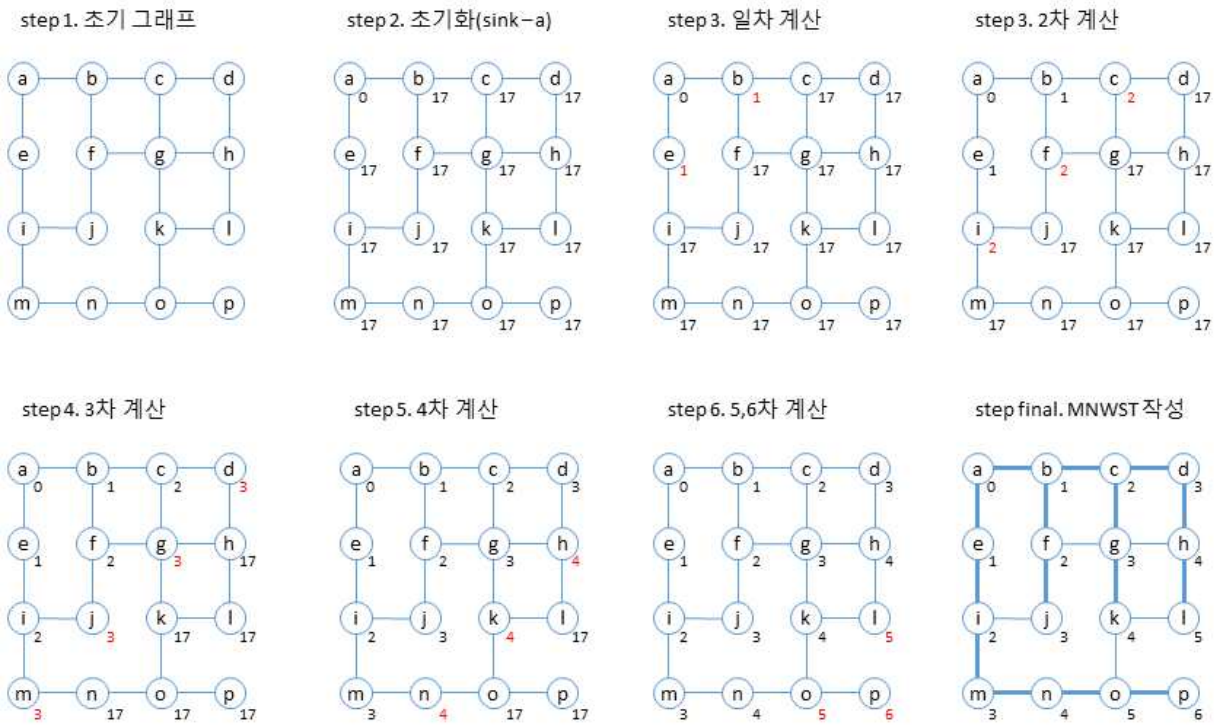
-s : 시작점, sink, 대피소

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

보다 작은 값을 부여받게 됨

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

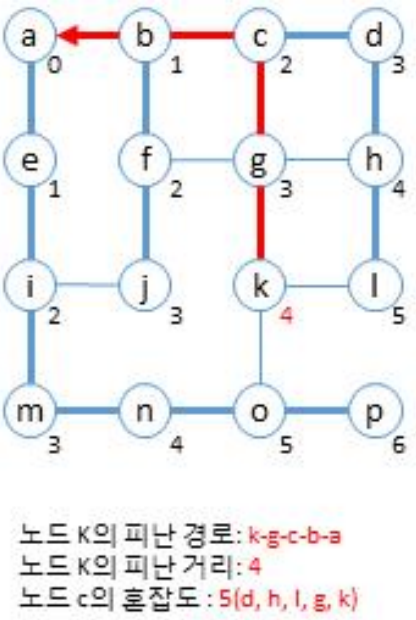
□ 알고리즘 도식 해설



<그림. 알고리즘 도식 해설>

□ MNWST의 특징

- 모든 노드에서 최단 경로 검색 가능
  - MNWST 및 각 노드의 가중치를 활용하면 특정 노드에서 피난처(sink)를 찾아가는 경로는 현재 노드에 연결된 간선 중에서 웨이트가 낮은 노드로 이동하는 경로임
- 최단 피난 거리 산출
  - 각 노드의 웨이트는 sink까지의 최단 피난 거리임
- 노드 혼잡도 계산
  - 각 노드가 이동경로로 이용되는 빈도로서 노드의 혼잡도는 각 노드의 자식 노드 수로서 구할 수 있음



<그림. 노드 계산>

☐ MNWST 공간 적용

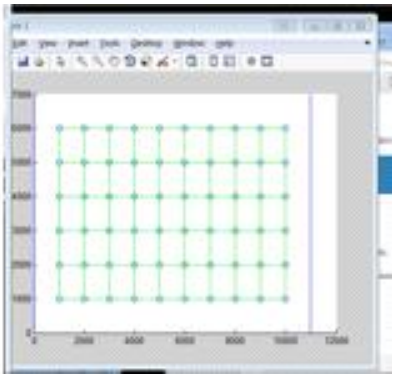
- 그래프 데이터 작성
  - 대상 공간을 그리드로 나누고 문이나 기둥에 의해서 단절되는 부분 이외를 그래프로 작성
  - 단절이 되지 않은 노드끼리만 연결을 하고 대상 사이트 내부에 대해서 그래프 생성
- 대피처 설정
  - 대피처를 하나 이상 설정 한 후에 알고리즘에 의해서 각 노드의 웨이트를 계산함
- MNWST 계산
  - 알고리즘에 의해서 각 노드의 가중치 및 스패닝 트리가 산출됨

☐ 알고리즘 검증 - 기본 그리드 테스트

- 검증 설정 및 결과
  - 정방형의 그리드를 대상으로 매트랩에서 알고리즘테스트

- 원 공간 데이터





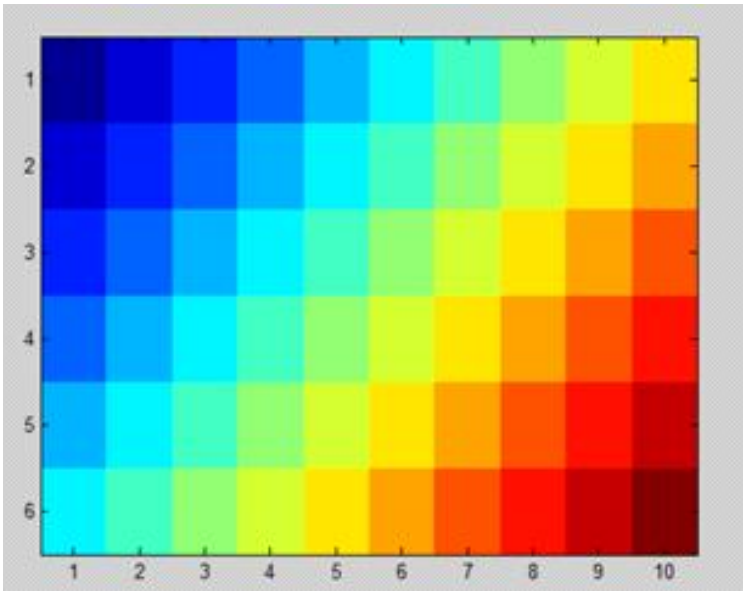
<그림. 10*6 정방형 격자>

- 씽크를 좌상의 포인트로 정한 후 계산 결과

	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9
2	1	2	3	4	5	6	7	8	9	10
3	2	3	4	5	6	7	8	9	10	11
4	3	4	5	6	7	8	9	10	11	12
5	4	5	6	7	8	9	10	11	12	13
6	5	6	7	8	9	10	11	12	13	14

<그림. 좌상 포인트 계산 결과>

- 결과 검토
  - 초기에 설정한 대로 가중치 계산이 완료됨



<그림. 계산 완료>

□ 캐드 데이터를 이용한 공간 그래프 작성 방법

1. 격자 간격 설정

2. 바운더리 전체 구간에 대해서 격자 생성

이때 노드 번호 및 에지 번호, list를 매트릭스 구조로 변환한 노드 매트릭스 도출

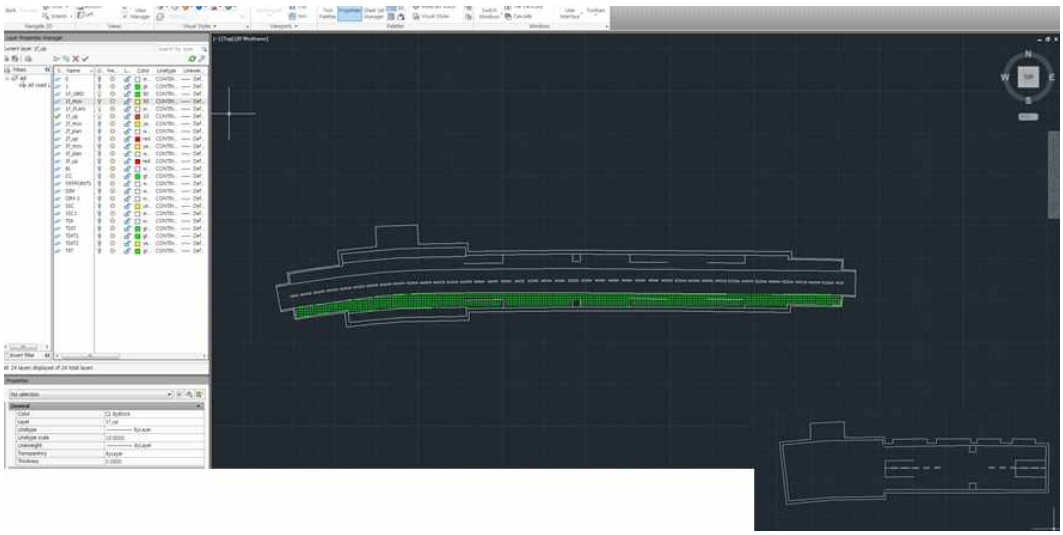
3. 격자와 plan 간의 교차 분석을 통해서 교차되는 간선 삭제

4. Move point 에서 가장 가까운 격자 포인트를 move node 로 설정

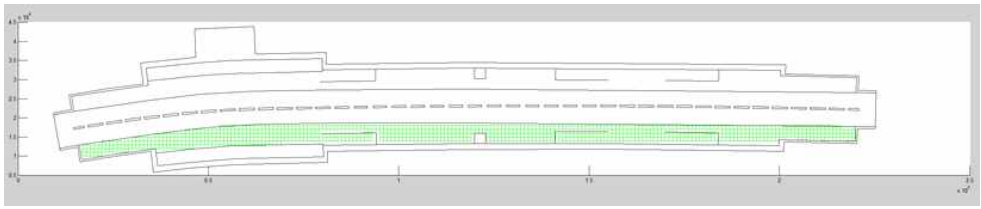
5. 만들어진 공간 그래프에 대해서 move node에서 연결된 노트 탐색. 이때 연결된 노드를 발견하면 그 노드까지의 경로 간선이 MNWST에 포함됨

□ 개포동역 적용

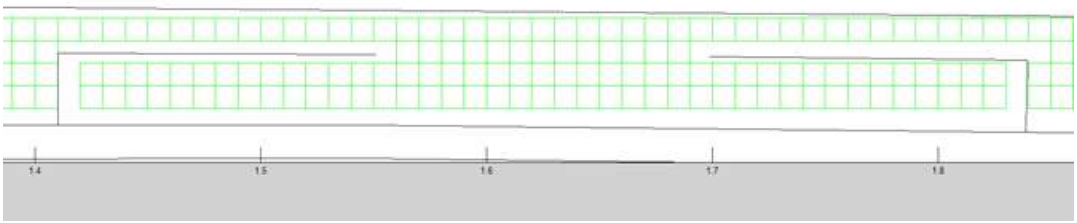
- 필요 레이어
  - 1F - plan : 건물 벽체 레이어
  - 1F - grid : 공간 그래프 레이어
  - 1F - up : 상층으로 이어지는 혹은 탈출구 지점



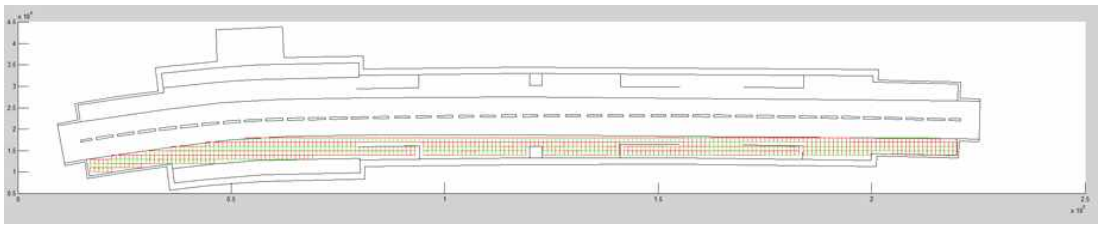
<그림. 개포동역 캐드 도면>



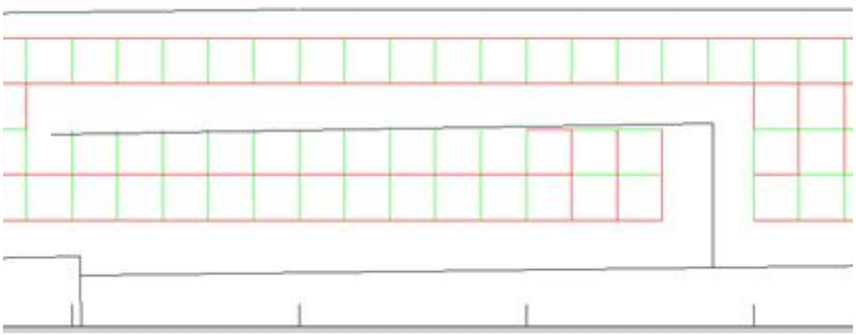
<그림. 공간 그래프(녹색)>



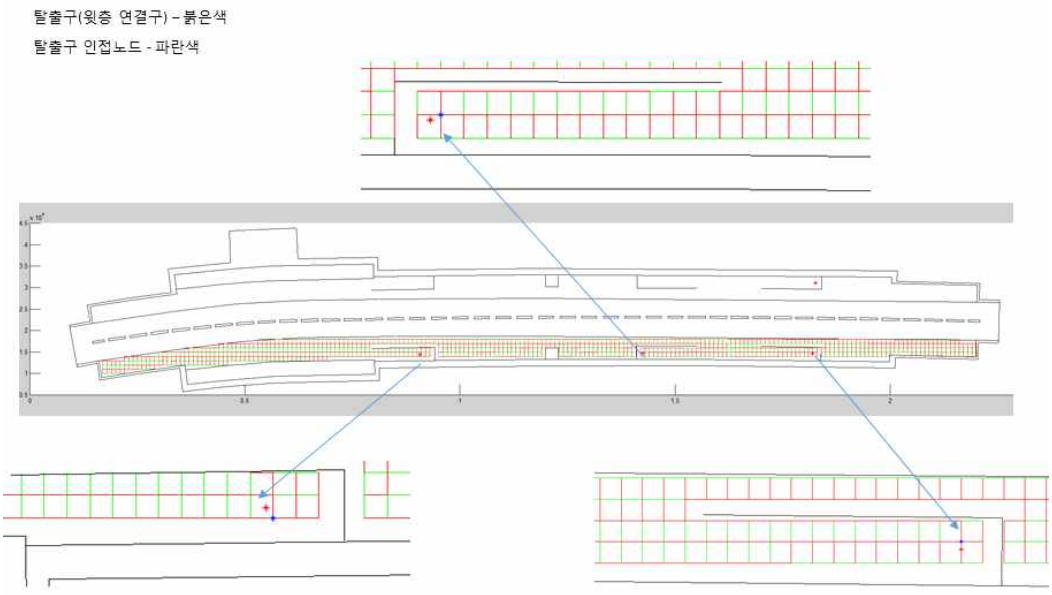
<그림. 계단부 확대(녹색)>



<그림. 공간 그래프(녹색)+MNWSP(붉은색)>



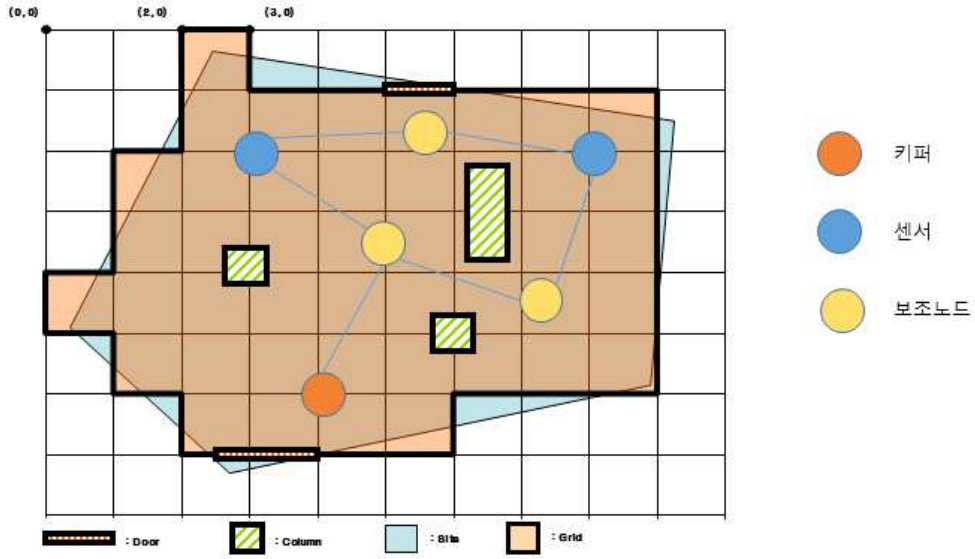
<그림. 계단부 확대(녹색+붉은색)>



<그림. 탈출구>

- 결론 및 ToDo
  - 베타버젼은 완성했으나 아직 value 검증 진행 중
  - 좀 더 검증을 통해서 디버깅 작업을 진행할 예정
  - 공간 그래프 작성의 문제를 해결하기 위한 자동화 필요 여부 검토

- 추가 제안
  - 피난 유도를 위해서는 앞에서 실험한 것과 같이 복잡한 그리드를 작성할 필요가 없음
  - 재난 발생 위치를 시스템에서는 센서의 위치로 파악하므로 대피처, 키퍼, 센서 위치로만 연결된 간단한 그래프로서 피난 유도 가능
  - 보조 노드를 사용해서 격자형태 및 각 노드간 거리를 동일하게 유지 가능하며, 오퍼레이터가 적절하게 수작업으로 작성 가능하며, 기본 그래프는 MNWSP를 기준으로 함
  - 스마트N키퍼의 실시간 피난 유도 시스템 구현 예정



<그림. 키퍼의 실시간 피난 유도 시스템>

□ 알고리즘의 활용

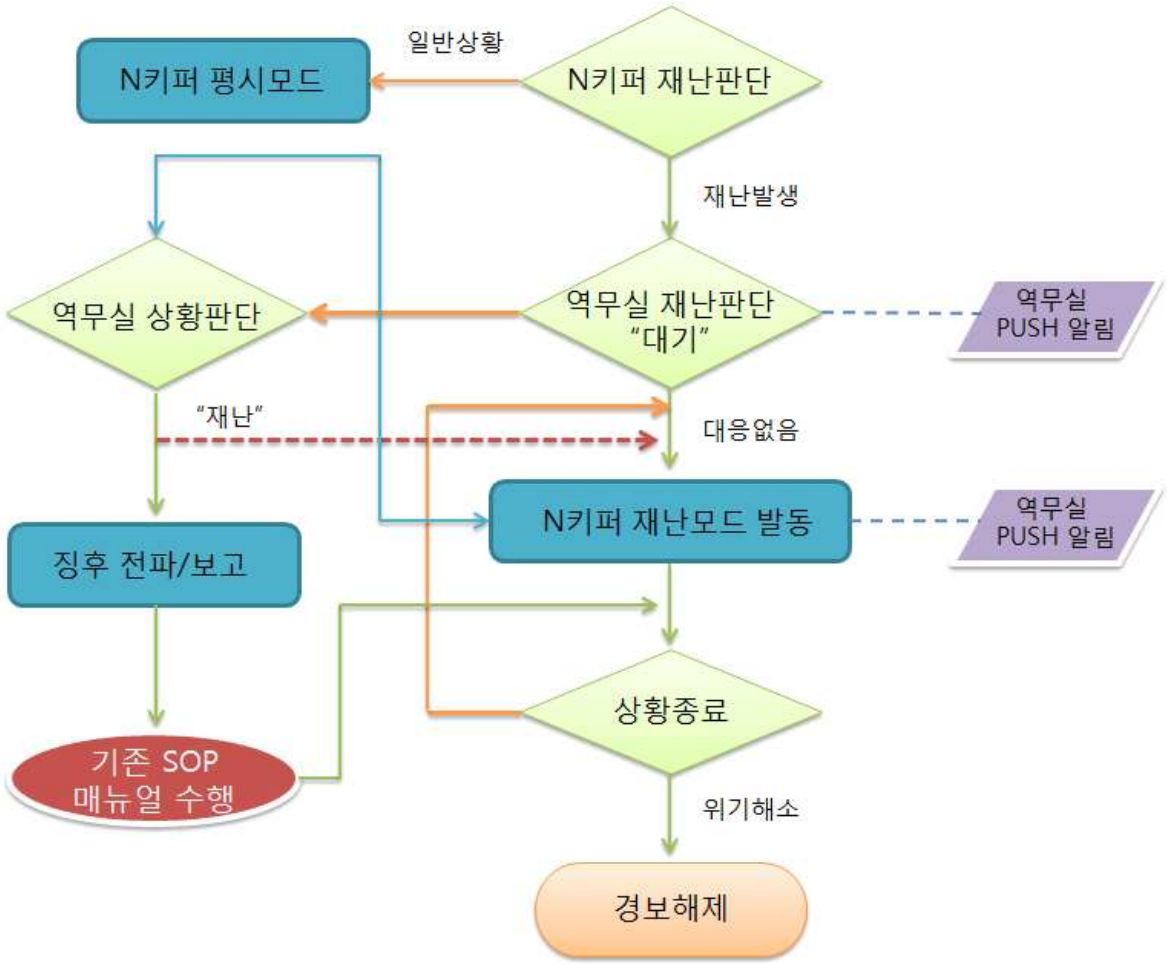
- 방재
  - 피난 경로
    - 공간을 그리드화 한 그래프 공간 구조에서 각 노드에서 대피처까지의 최단 경로를 그래프를 이용해서 도출
  - 피난 동선 밀도 계산
    - 모든 노드에 대한 피난 경로 계산을 통해서 각 노드의 사용 빈도를 구할 수 있음
    - 사용빈도는 에지의 사용회수를 구하거나 타임라인 별로 각 노드에서의 피난에이전트 수를 구함으로서 간이 피난 시뮬레이션을 수행하는 것과 동일한 효과 발생
- 재난전조 예측시스템
  - 주어진 공간 내에서 대피처(sink)로 향하는 경로를 찾는 알고리즘
  - 일반적으로는 최단경로 탐색을 이용하면 되나, 고정적인 평면(그래프 구조)에서 대피처(sink)가 정해진 상태이면 모든

출발지(노드)에서 대피처까지의 경로를 case by case로 계산하는 것이 아니라, 모든 노드에서 최단경로를 위해서 이동 가능한 다음 목적지를 계산한 그래프, 즉 대피처까지의 거리를 속성으로 가지는 그래프를 작성

- 일정 시간대에 정체가 높아지면 다른 대안 선정 기능도 고려할 수 있음

3.5 스마트N키퍼를 이용한 SOP(안) 개발

□ 재난(화재, 테러)발생 시 스마트N키퍼는 스스로 재난에 대하여 판단하여 역무실의 역무원에게 “알람”을 주게되며, 이 알람을 받은 역무원이 재난에 대한 상황판단 이후 적절한 조치를 취하여야 한다. 하지만 N키퍼의 알람에도 일정 시간(약 10초)동안 아무런 조치가 없을 경우 N키퍼는 스스로 재난모드를 발동하여 승객에게 알람 및 대피유도를 하도록 되어있으며, 자세한 흐름도는 다음과 같다.



<그림. M2M 기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크 시스템을 이용한 위기경보 발령체계도>

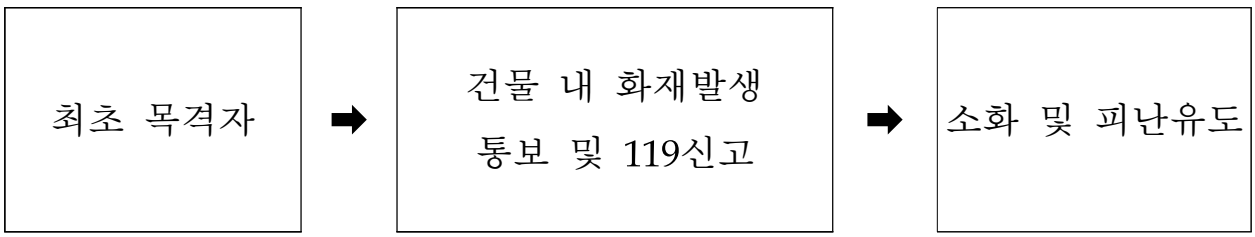
□ 기존의 SOP에서 철도, 방재전문가와의 회의를 통해 스마트N키퍼의 역할과 기능을 고려한 SOP 베타버전을 개발하였으며 내용은 다음과 같다.

- 재난(화재,테러)상황시 행동요령 및 피난계획

□ 대공원역

- 재난(화재,테러)상황시 행동요령

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



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

소방안전관리자는 사업장별 실정에 적합한 세부행동요령을 별도로 정할 수 있다.

- 피난계획

▪ 맞이방/대합실(지하 1층)

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

▪ 승강장(지하 2층)

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

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

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

- 재난(화재,테러)상황시 행동요령 및 피난계획

□ 개포동역

- 재난(화재,테러)상황시 행동요령

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

최초 목격자	➡	건물 내 화재발생 통보 및 119신고	➡	소화 및 피난유도

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

소방안전관리자는 사업장별 실정에 적합한 세부행동요령을 별도로 정할 수 있다.

- 피난계획

▪ 맞이방/대합실(지하 1층)

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

▪ 중간층(지하 3층)

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

▪ 승강장(지하 4층)

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

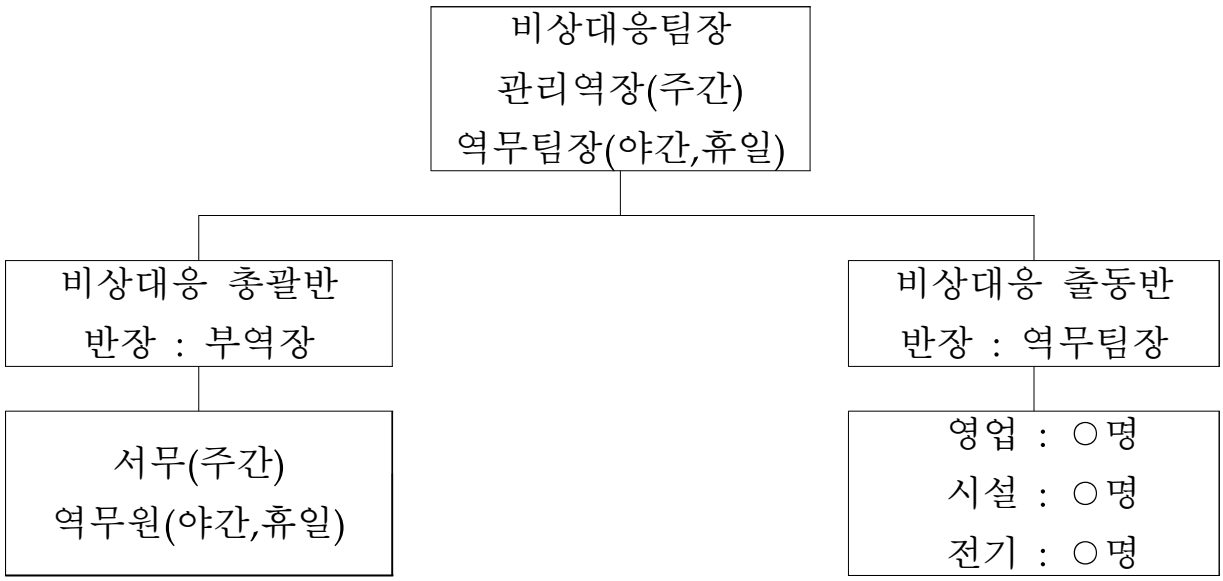
경우 게이트 및 비상게이트를 개방하여 신속히 대피할 수 있도록 한다.

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

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

소방대 조직표

□ **대공원역/개포동역**



- 위기정보 단계별 조치사항

구 분	조치주체	조치내용
주의 (Yellow)	역(역무원)	• 접수, 보고/전파 • 역사내 임대매장 정전대비 상황 통보 • 정전 대비 장비 및 임무 확인
	전기사업소(전기처)	• 현장출동 • 상황파악 및 보고 • 예비전원 설비 집중 점검

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

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

- 반별 임무

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

- 재난상황별 개인임무카드

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

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

- 시간별 개인임무카드

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

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

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

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

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

Ⅳ 테스트베드 개요 및 현장평가 계획

4.1 테스트베드 개요

□ 분당선 개포동역(한국철도공사 관리 역사)

- 시험위치 : 지하 1층 ~ 4층 고객이동 통로 공간(승강장, 대합실)

□ 선정사유

- 다양한 지하 공간 특성을 고려한 통합관리지원체계 개발을 목적으로 대심도 추가 공간을 확보 및 실험

□ 현장 개요

- 2차년도 테스트베드 구축에 사용된 스마트N키퍼 및 센서 설치 개수는 다음과 같다.

위치	구조/용도	N키퍼 설치 개수	센서 설치 개수
지하 1층	대합실, 개찰구	4 개	2 개
지하 2층	기계실/계단	-	-
지하 3층	공조실	4 개	2 개
지하 4층	승강장	8 개	14 개
계		16 개	18 개

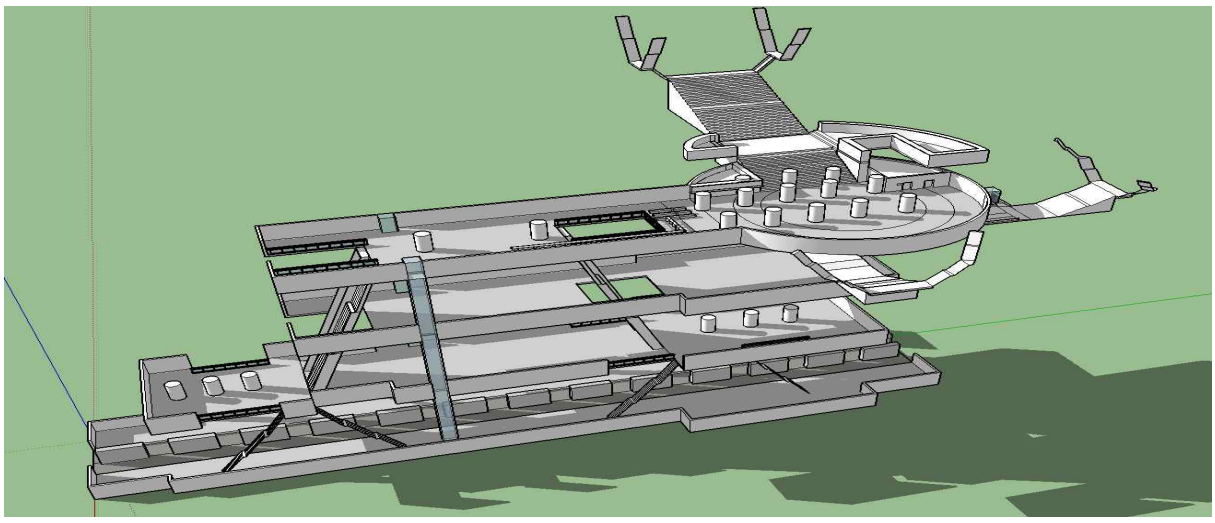
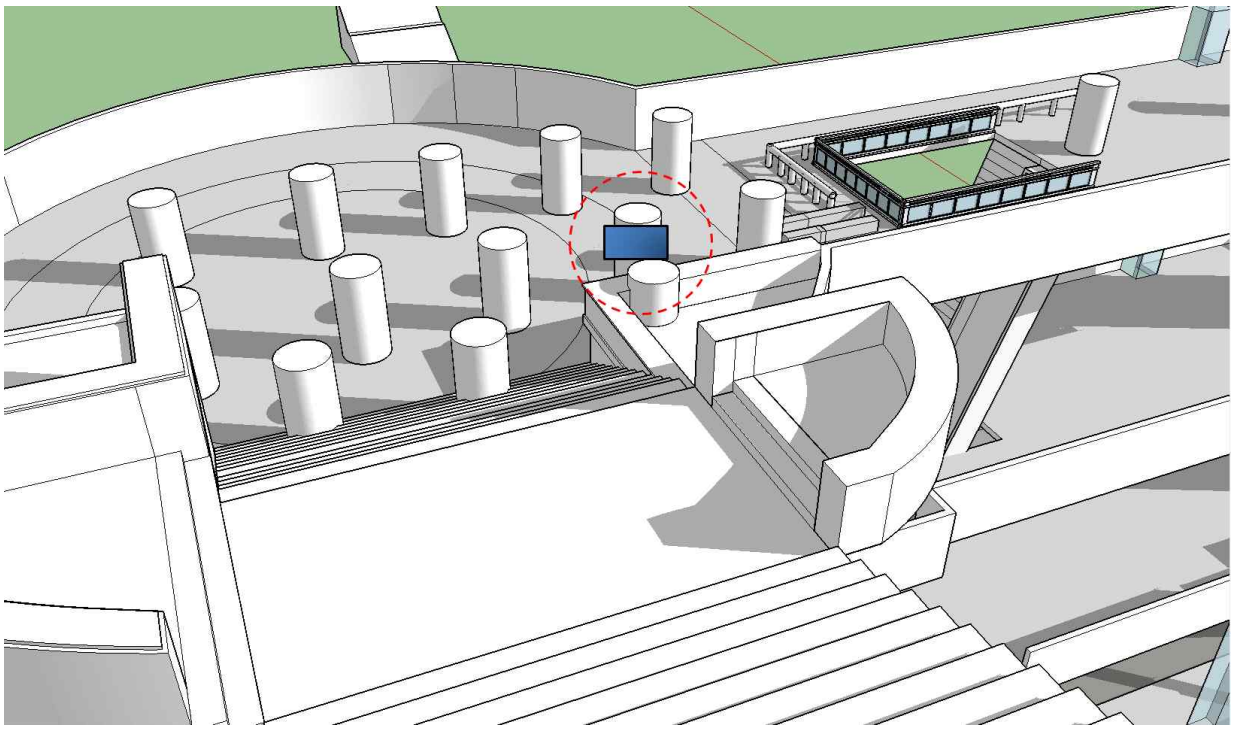


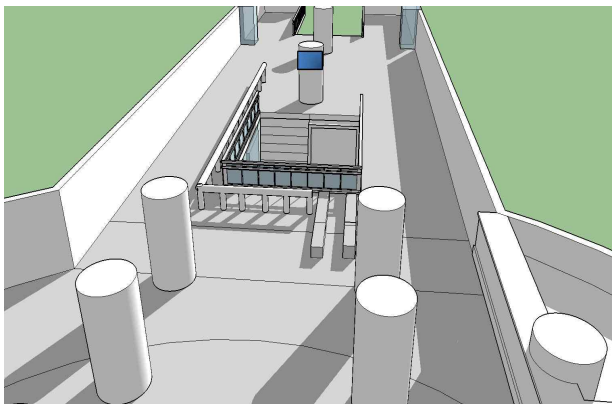
그림. 개포동역 3D 렌더링

4.2 N키퍼 설치위치

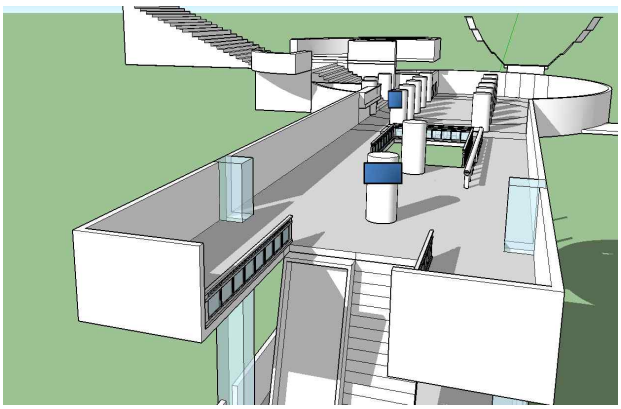
□ 개포동역 역사에서의 원활한 무선 데이터통신을 위하여 최대한 LOS(Line Of Sight)를 고려하여 N키퍼의 위치를 선정하였으며, 특히 승강장의 경우 동선이 나뉘는 위치마다 N키퍼를 설치하여 재난이 발생하였을 때 승객들에게 최적 대피경로를 유도할 수 있도록 하였다.



a) 2번 출구방향

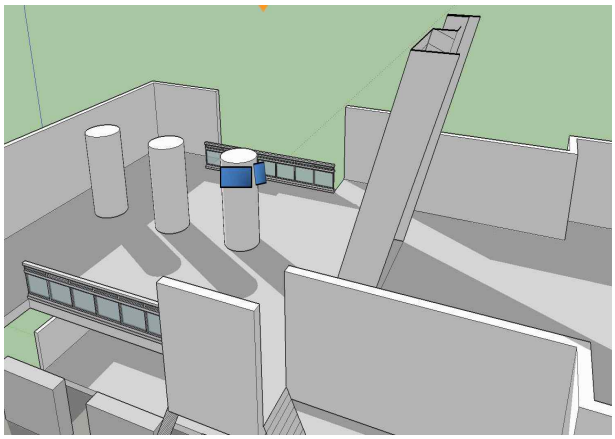


b) 에스컬레이터/계단 방향

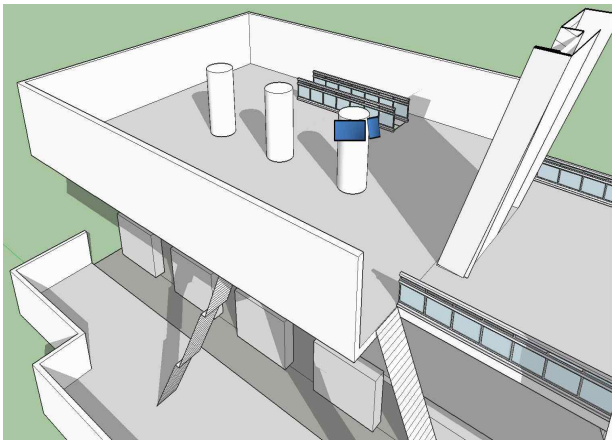


c) 에스컬레이터/계단 방향 및 개찰구 방향

그림. 지하1층(대합실) 스마트N키퍼 설치 위치

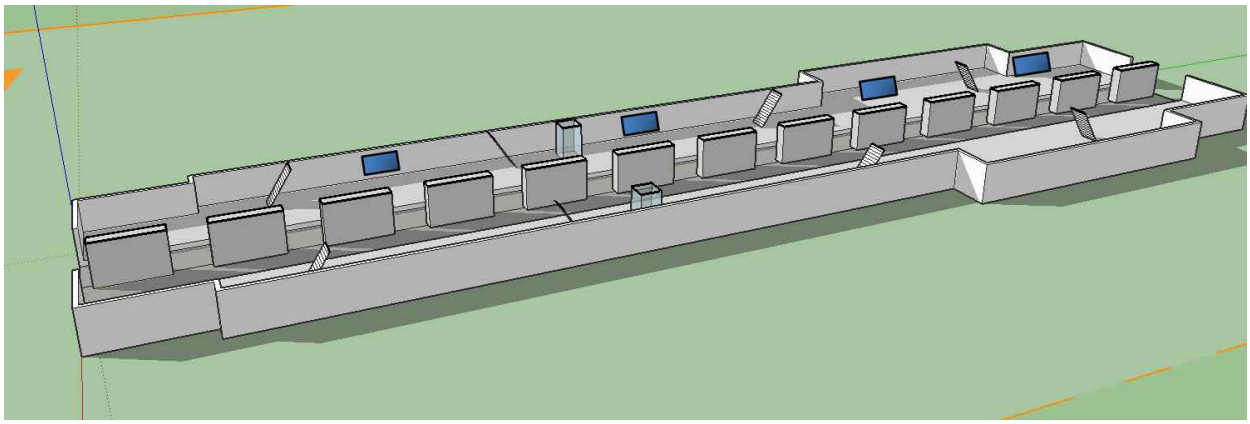


a)지하3층/이동방향

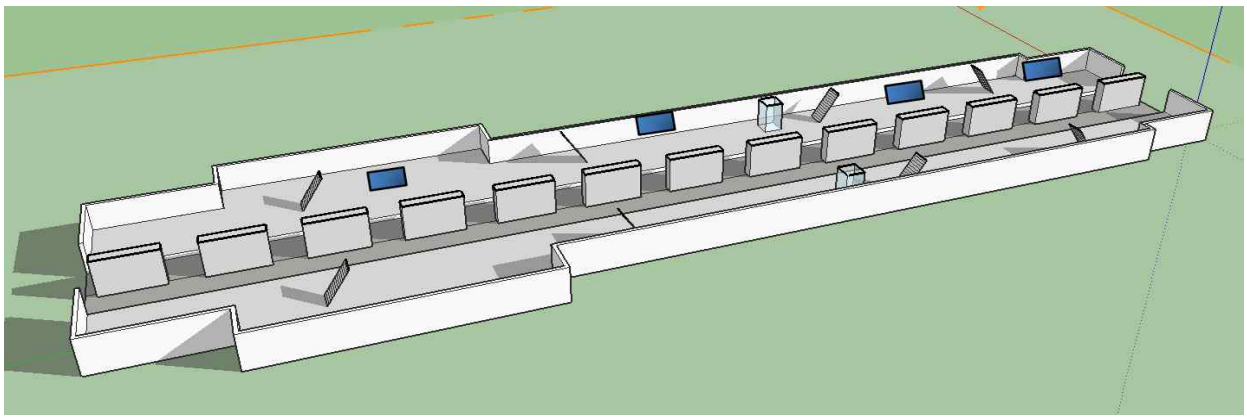


b)지하3층/이동방향

그림. 지하3층(공조실) 스마트N키퍼 설치 위치



a)대모산입구방향



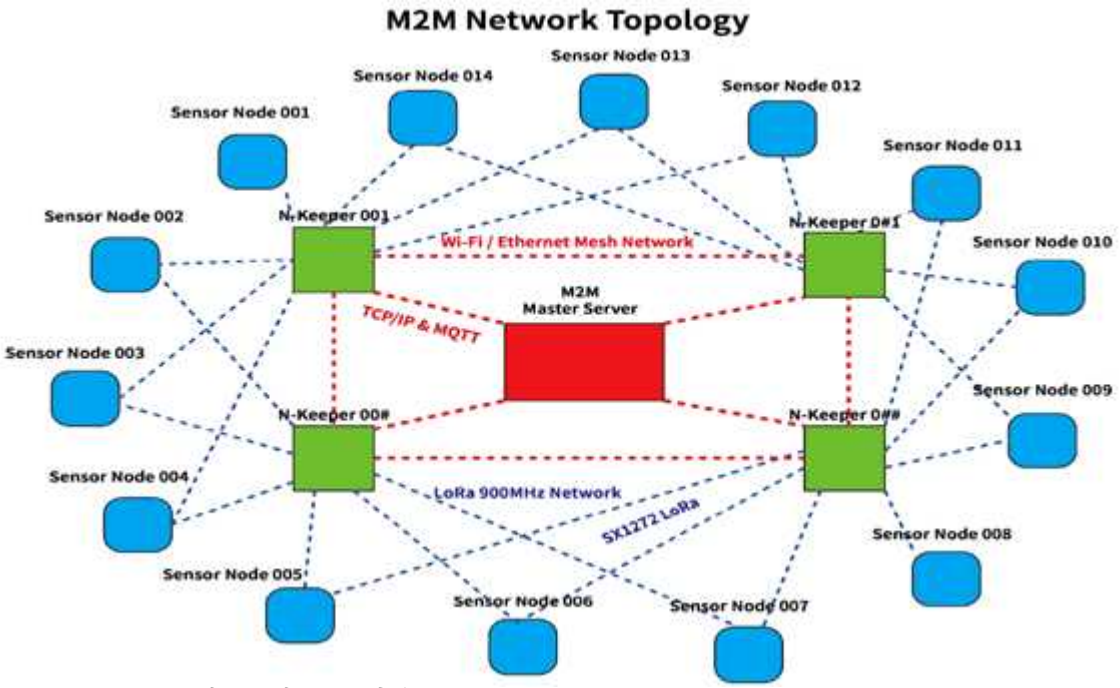
b)구룡방향

그림. 지하4층(승강장) 스마트N키퍼 설치 위치

4.2 재난전조 대응 M2M 네트워크의 구축

□ M2M 네트워크 구성

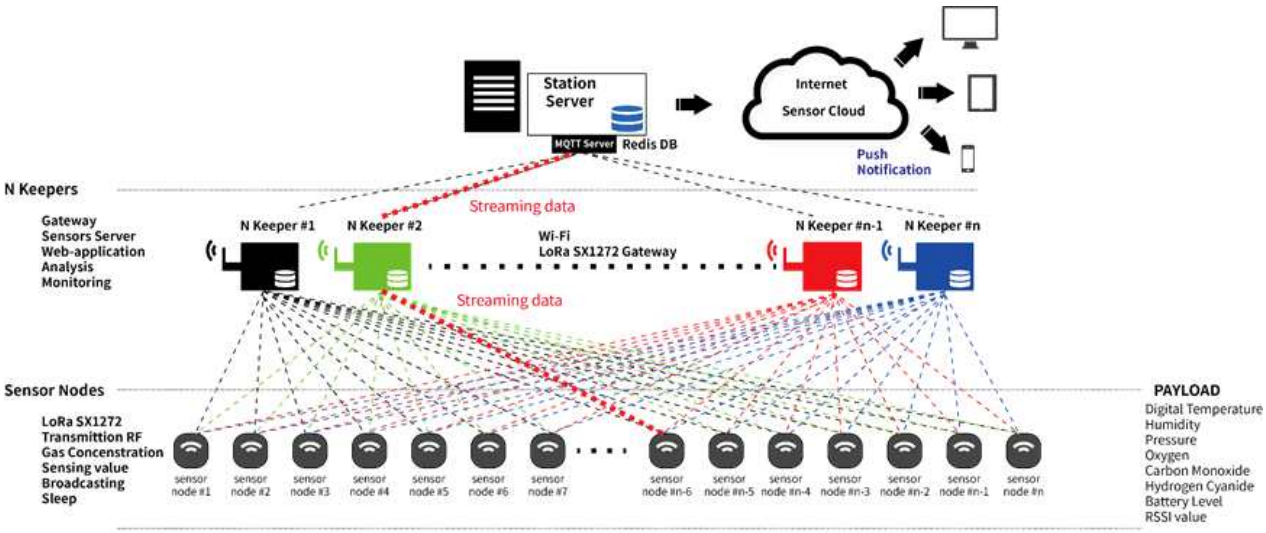
재난전조 대응 M2M 네트워크는 아래그림과 같다. 스마트N키퍼는 센서노드와 N키퍼 그리고 마스터 서버로 구성되어진다. 센서노드는 N키퍼와 900MHz 대역 LoRa SX1272 네트워크로 LoRa프로토콜로 데이터를 주고받는다. 이때 센서노드는 여러대의 N키퍼를 도착 주소로 갖게 되며 주변 N키퍼가 많기 때문에 센서노드의 네트워크 hop은 N키퍼가 MQTT 프로토콜을 사용하여 대신한다. 따라서 아래 그림과 같이 N키퍼의 통신에서 센서노드의 중복 데이터 수집하게 되고, 센서노드에서 데이터가 도달하지 않는 네트워크의 사각지대가 사라지게 된다.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



□ M2M 네트워크 계층도

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

통신하며 각 계층마다 역할은 다음과 같다.



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

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

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

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

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

□ 재난 전조 대응 N키퍼 모듈 네트워크 구성

1) 개요

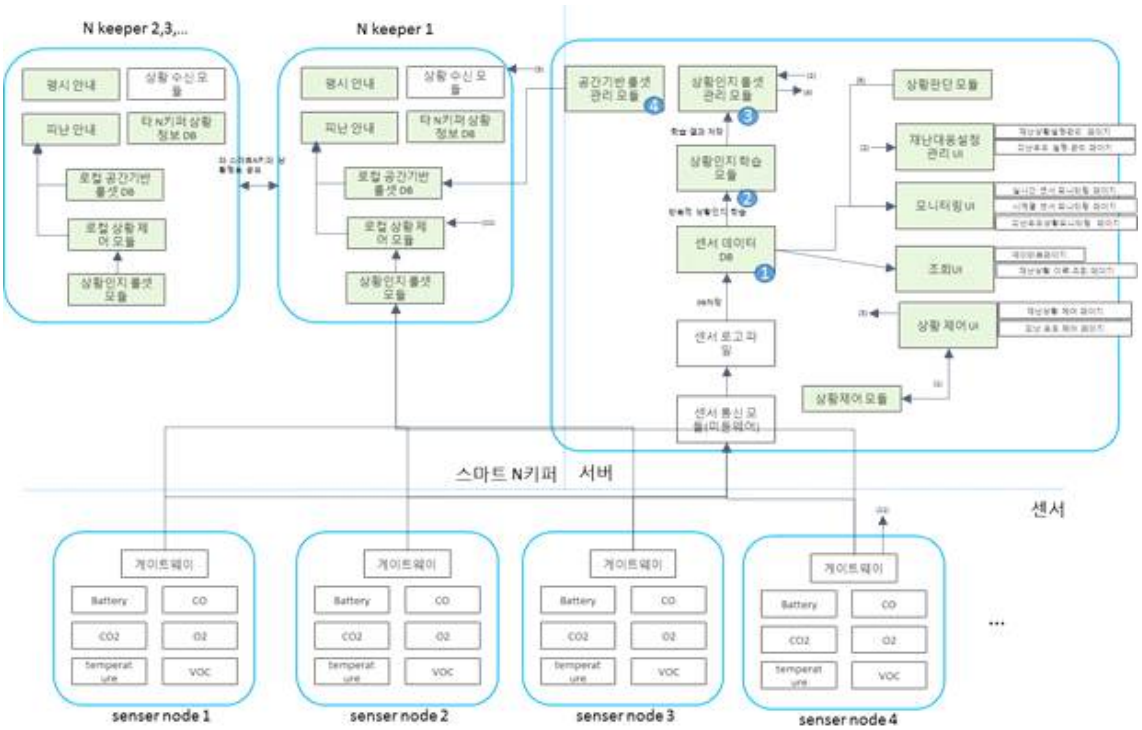
○ 시스템은 구성도에서 우상단은 마스터의 기능을 나타내며, 좌상단은 키퍼의 기능을 나타낸다.

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

○ 저장된 데이터는 JSON 형식으로 각 머신에서 제공이 된다.

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

○ 키퍼 및 마스터는 Ajax로 통신을 하며 필요한 정보를 교환한다.



2) 주요 서비스 내용

○ 전체 시스템은 역무실에 있는 관리 시스템인 마스터와 역사에 있는

키퍼로 이루어지며, 각 시스템의 서비스 내용은 다음과 같다.

가) 마스터

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

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

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

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

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

나) 키퍼

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

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

가) 개발환경

○ 모든 개발은 윈도우 7 및 우분투 16.04의 환경에서 파이썬 3.5.3 및 장고 1.6.4의 환경에서 개발되었다.

○ 서비스는 웹으로 표출되며, 프론트엔드(웹페이지)는 우분투 16.04 환경의 파이어폭스을 대상으로 개발하였다.

나) 운용환경

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

4.3 현장 검증 개최

□ 분당선 개포동역(한국철도공사 관리 역사)

- 일 시 : 일 시 : 2017년 7월 14일
- 현장 테스트 장소 : 지하 1층 ~ 4층 고객이동 통로 공간(승강장, 대합실)

□ 검증방법

- 방재전문가를 초청하여 스마트N키퍼의 구동 확인 및 SOP모듈 작동 확인/ 재난모드 발동에 따른 대피경로 안내에 대한 전문가 평가의견서 작성

□ 전문가 자문평가

- 참석 : 임종권 부원장 (한국건설VE연구원), 정종문 교수 (연세대학교), 김정곤 박사 (한국전자부품연구원)

□ 시연 내용

- 재난 대응 M2M 네트워크 시스템 소개
- 현장 센서데이터 뷰
- 재난 대응 SOP 모듈 소개
- 재난 대응 SOP 모듈 시연
- 대피경로 안내 e-book 시연
- 대피경로 안내 화살표 시인성 확인
- 통합 EL 디스플레이 모듈 소개 및 시연
- 화재테스트 영상 시연

□ 시연 전경



□ 평가의견서 결과

1. 재난전조 예측/재난 대응 모듈 고도화

① 임종권 부원장 (90점)

재난을 센서와 기계만으로 자동 예측하기는 실질적으로 힘든 부분이 많으며, 화재 및 테러 등에 의한 화재 발생은 사전에 재난의 전조를 예측하기는 대단히 어려우므로 사전 예측 보다는 사후 신속한 대응을 통한 피해 최소화에 모든 기능적인 면을 고려해야 함. 현재 과제를 통해 개발된 시스템은 이러한 신속한 재난에 대응할 수 있는 필요한 기능을 구현할 수 있도록 구성되었다고 보나, 피해를 최소화할 수 있는 피난유도 기능에 대해서는 좀 더 다양한 시나리오에 의한 검토가 필요함.

또한 재난 대응에 대한 기능을 뚜렷하고 구별하고 그 기능에 대한 목표 성능을 명확히 하여 연구목표를 뚜렷하게 하여 최종 상용화될 때 시스템 기능 구현에 대한 관계자와의 커뮤니케이션 및 기술 개발 변화 과정에 대한 관리가 필요함.

② 김정곤 박사 (96점)

재난을 예측하기는 실질적으로 힘든 부분이 많으며, 잘못 예측된 정보를 바탕으로 대응을 하는 경우 더 큰 문제가 발생함. 특히, 본 과제의 경우는 전조나 예측이 어려운 사회재난에 속하고 센서를 통한 상황인식 기능이 중심이므로 신속한 위험 발견과 대응에 주안점을 두는 것이 현실적임.

지하철 역사의 경우 지하공간이기 때문에 승객의 안전을 확보하기 위해서는 무엇보다 신속한 지상으로의 탈출이 우선시 되어야 하며 현재 과제를 통해 개발된 시스템은 유효한 기능을 할 수 있는 대응시스템으로 판단됨.

다만 위험 상황의 판단에 있어서는 위험요소의 위치와 함께 확산방향에 대한 고려도 필요할 것으로 사료됨.

③ 정종문 교수 (99점)
• 네트워크 단절시 각 N키퍼의 역할 정의와 재연결시 마스터서버 및 N키퍼의 대응 알고리즘 개발 탑재가 성공적으로 개발이 되었음.
• 2차년도 전문가 자문 평가 개선 제안 사항들도 잘 반영된 것으로 판단됨.
• 통합 디스플레이 모듈 및 EL Display 제어 모듈이 잘 개발 된 것으로 판단됨.
• 개발된 N키퍼 시스템과 통합 및 EL Display 모듈들이 재난전조 예측/재난 대응 모듈 고도화에 잘 기여 할 것으로 판단됨. (99점)

2. 센서 노드 통신 데이터 수집 및 분석 모듈

① 임종권 부원장 (95점) : 의견 없음

② 김정곤 박사 (96점)
화재 및 테러 등을 상정하여 연기 이외에도 독성가스를 검출 할 수 있도록 센서노드를 구성하였으며 데이터를 상시적으로 수집 및 축척하여 변화를 감지할 수 있는 기능을 구축하여 시간에 따른 변화 추이를 파악할 수 있는 분석기능도 개발되었음.
향후 축척된 데이터를 활용하면 지하 공간의 공기질 문제 등에도 축척된 데이터가 활용될 수 있을 것으로 사료됨.

② 정종문 교수 (99점)

• LoRa 및 디스플레이 통신 모듈의 양방향 통신 개선 사항들을 확인하였음.
• 데이터 품질 개선 및 센서링 타임에 의한 보정 및 최적화 제어 잘 이뤄졌음.
• LoRa 네트워크의 시각동기화 구현 고도화 작업 목적을 달성 한 것으로 판단됨.

3. SOP 및 시나리오 통합 대피경로 안내 모듈

① 임종권 부원장 (95점)

현재 구성되어 있는 SOP 및 시나리오 통합 대피경로 안내 모듈은 계획한대로 개발된 것으로 판단되나 모든 대피경로를 자동으로 구현하게 되면 현장에서 생기는 돌발적인 상황에 전혀 대처가 안되는 문제가 있을 수 있으므로 본 시스템과 시스템을 관리하는 관리자와의 유기적인 정보교환을 통한 최적의 대피경로를 탐색해 낼 수 있도록 하는 기능으로 시스템과 관리자와의 상호 인터페이스 시스템으로 발전되는 것이 필요함.

② 김정곤 박사 (90점)

시나리오를 구성하여 발생 가능한 유형의 사고들을 검토 및 고려한 접근방법은 유효적절하다 판단됨. 다만 대응에 있어서는 피난경로 유도가 개찰구를 통한 출구방향 유도로 되어 있어 입구에서 화재나 테러가 발생한 경우에 대응이 어려울 수 있음으로 필요시 철도를 통한 피난이 가능하도록 검토가 필요함. 또한 위험상황에서 개찰구가 피난 흐름에 방해가 될 수 있음으로 이에 대한 개선방안도 제안하기 바람.

③ 정종문 교수 (98점)

- 최초 재난 발생 구역을 기점으로 대피 시나리오 적절한 것으로 판단 됨.

• 센서 별 화재 및 화학테러 시나리오 룰셋 고도화 작업이 목적을 달성한 것으로 판단됨.

4. M2M 통합보드 대피경로 안내용 보조패널의 방향기 인지 / 시인성

① 임종권 부원장 (95점)

대피경로를 효율적으로 안내하기 위한 보조장치로서 유효한 기능을 할 것으로 판단됨. 화재 상황을 가정하여 눈에 분명히 인지가 되는지 실험이 필요하며 실제 상용화되기에 앞서 실제 상황에 제대로 작동되는지 좀 더 현실적인 실험 계획이 필요함. (95점)

② 김정곤 박사 (96점)

대피경로를 효율적으로 안내하기 위한 보조장치로서 유효한 기능을 할 것으로 판단됨. 다만, 현재 개발된 패널의 경우 화살표 크기가 화재가 발생한 경우 시계가 불량하게 되는 점을 고려하면 인식성이 떨어질 우려가 있음으로 화살표의 크기를 크게 할 필요가 있으며, 글씨 등은 가급적 제외하고 사인위주로만 제작하는 것이 효과적일 것으로 사료됨.
분기점에서 방향 안내도 중요하지만 경로를 연속적으로 안내하는 것이 더 중요함. (96점)

③ 정종문 교수 (99점)

• M2M 통합 N키퍼 보드 시제품이 네트워크 단절시 각 N키퍼의 역할 정의에 맞게 잘 개발됨. 즉, 네트워크 단절 후 재 연결 시 마스터서버 및 N키퍼의 대응 알고리즘이 제대로 탑재 되어서, 2차년도 전문가 자문 평가 개선 제안 사항들이 반영이 제대로 된 것으로 판단됨.

• 통합 디스플레이 모듈 및 EL Display 제어 모듈 개발이 잘 된 것으로 판단됨.

• 대피유도 안내 시스템과의 연동 및 N키퍼와 EL Display의 제어 통합 방향 제어 기능이 잘 개발 되어, 2차년도 전문가 자문 평가 개선 제안 사항이 잘 반영 된 것으로 판단됨.

5. 테스트베드 운용을 통한 시제품 테스트 : 유사재난 상황별 대응능력 검증

① 임종권 부원장 (95점) : 의견없음
② 김정곤 박사 (96점)
현장적용을 통한 시제품의 테스트는 유효하며, 취득된 데이터를 지속적으로 분석하여 주기적으로 현장최적화를 실시하기 바람
테러가운데 사린가스 등을 활용한 형태의 경우 본 연구과제에서 개발한 시스템을 통해서 위험가스의 검출 등 일부 대응 기능을 할 수 있을 것으로 사료됨. (96점)

② 정종문 교수 (95점)
• 실무적용이 가능한 시제품 시스템이 잘 개발 된 것으로 판단됨.
• 현실적인 현장검증을 잘 실시 된 것으로 판단됨.
• 테스트 베드 운용을 통한 시제품 테스트, 실측데이터 기반 에뮬레이션 데이터 생성하여 재난 상황 시뮬레이션 테스트, 및 캘리브레이션 재료를 통한 센서노드 재난상황 시뮬레이션 모두 잘 된 것으로 판단 됨.
• 지하철역 마다 구조가 다르므로 추가 실험을 통한 재난 상황 기반의 평가를 권유함.

6. M2M 웹기반 관리시스템

① 임종권 부원장 (90점)

기본적인 관리는 웹기반 시스템으로 관리하는 것이 필요하나 시스템 개선 및 시스템 유지보수 등에 대한 대안이 마련되어야 할 것임.

② 김정곤 박사 (96점)

M2M시스템의 상시적 관리를 위해서는 웹기반의 시스템을 활용하여 데이터 모니터링 셋팅 및 관리를 하는 것이 편리할 수 있음.

메뉴는 사용자의 편리성을 고려하여 복잡한 기능을 최대한 단순화시켜야함. (90점)

③ 정종문 교수 (98점)

- M2M 웹기반 관리시스템 안정화 및 버그 수정이 잘 되었음.
- 테스트베드 역무실 마스터 서버에서 외부 관리 서버로 상황 및 데이터 전송 및 관리 체계 잘 구성 된 것으로 판단됨.
- 통신모듈 설정 및 개별 모듈 구동 안정화 잘 된 것으로 판단됨.
- M2M 시스템 보안 설정 적합한 것으로 판단됨.
- 매뉴얼 작성 제대로 된 것으로 판단됨.

7. 종합평가

① 임종권 부원장 (종합 점수 : 94점)

본 연구는 큰 틀에서 상용화가 가능한 수준의 시스템 개발과 현장검증이 이루어진 것으로 판단된다. 모든 문제를 센서를 통해서만 결정하기보다는 인적자원과 정보도 함께 활용하는 방법에 대해 고민하여 사고발생시 기능이 무용지물이 되지 않도록 시나리오에 대한 모든 가능성을 검토하여 실무적용에 효과적인 시스템으로 발전시켜주기를 바랍니다.

② 김정곤 박사 (종합 점수 : 94점)

실무적용이 가능한 수준의 시스템 개발 및 현장검증을 실시하였다 판단됨. 다만 현장별로 여건이 다르고 다양한 돌발적 상황이 발생할 수 있음으로 확산적용을 위해서는 지속적인 기능 검증 및 보완 등 후속조치가 필요함..

③ 정종문 교수 (98점)

- 재난전조 예측/재난 대응 모듈 고도화 성공한 것으로 판단됨.
- 센서 노드 통신 데이터 수집 및 분석 모듈 잘 개발 된 것으로 판단됨.
- SOP 및 시나리오 통합 대피경로 안내 모듈 잘 개발 된 것으로 판단됨.
- M2M 통합보드 대피경로 안내용 보조패널의 방향기가 잘 개발 것으로 판단됨.
- 테스트 베드 운용을 통한 시제품 테스트를 통한 유사재난 상황별 대응능력이 검증 되었으나 만약을 대비하여 구조가 다른 타 지하철역에서도 추가 실험을 권유함.
- M2M 웹기반 관리시스템 잘 개발 것으로 판단됨.
- 종합적으로, 과제가 매우 성공적으로 잘 진행 되는 것으로 판단 됨.

평가의견서

[M2M기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크 시스템개발]

1. 평가항목별 점수

▶ 재난전조 예측/재난 대응 모듈 고도화	90	/100
▶ 센서 노드 통신 데이터 수집 및 분석 모듈	95	/100
▶ SOP 및 시나리오 통합 대피경로 안내 모듈	95	/100
▶ M2M 통합보드 대피경로 안내용 보조패널의 방향기호 인지 / 시인성	95	/100
▶ 테스트 베드 운용을 통한 시제품 테스트 : 유사재난 상황별 대응능력 검증	95	/100
▶ M2M 웹기반 관리시스템	90	/100

2. 평가항목별 의견

■ 재난전조 예측/재난 대응 모듈 고도화

재난을 센서와 기계만으로 자동 예측하기는 실질적으로 힘든 부분이 많으며, 화재 및 테러 등에 의한 화재 발생은 사전에 재난의 전조를 예측하기는 대단히 어려우므로 사전 예측보다는 사후 신속한 대응을 통한 피해 최소화에 모든 기능적인 면을 고려해야 함. 현재 과제를 통해 개발된 시스템은 이러한 신속한 재난에 대응할 수 있는 필요한 기능을 구현할 수 있도록 구성되었다고 보나, 피해를 최소화할 수 있는 피난유도 기능에 대해서는 좀 더 다양한 시나리오에 의한 검토가 필요함.
또한 재난 대응에 대한 기능을 뚜렷하고 구별하고 그 기능에 대한 목표 성능을 명확히 하여 연구목표를 두력하게 하여 최종 상용화될 때 시스템 기능 구현에 대한 관계자와의 커뮤니케이션 및 기술 개발 변화 과정에 대한 관리가 필요함.

■ 센서 노드 통신 데이터 수집 및 분석 모듈

■ SOP 및 시나리오 통합 대피경로 안내 모듈

현재 구성되어 있는 SOP 및 시나리오 통합 대피경로 안내 모듈은 계획한대로 개발된 것으로 판단되나 모든 대피경로를 자동으로 구현하게 되면 현장에서 생기는 돌발적인 상황에 전혀 대처가 안되는 문제가 있을 수 있으므로 본 시스템과 시스템을 관리하는 관리자와의 유기적인 정보 교환을 통한 최적의 대피경로를 탐색해 낼 수 있도록 하는 기능으로 시스템과 관리자와의 상호 인터페이스 시스템으로 발전되는 것이 필요함.

■ M2M 통합보드 대피경로 안내용 보조패널의 방향기 인지 / 시인성

대피경로를 효율적으로 안내하기 위한 보조장치로서 유효한 기능을 할 것으로 판단됨. 화재 상황을 가정하여 눈에 본명히 인지가 되는지 실험이 필요하며 실제 상용화되기에 앞서 실제 상황에 제대로 작동되는지 좀 더 현실적인 실험 계획이 필요함.

■ 테스트 베드 운용을 통한 시제품 테스트 : 유사재난 상황별 대응능력 검증

■ M2M 웹기반 관리시스템

기본적인 관리는 웹기반 시스템으로 관리하는 것이 필요하나 시스템 개선 및 시스템 유지보수 등에 대한 대안이 마련되어야 할 것임.

■ 종합 평가 의견

본 연구는 큰 틀에서 상용화가 가능한 수준의 시스템 개발과 현장검증이 이루어진 것으로 판단된다. 모든 문제를 센서를 통해서만 결정하기 보다는 인적자원과 정보도 함께 활용하는 방법에 대해 고민하며 사고 발생시 기능이 무용지물이 되지 않도록 시나리오에 대한 모든 가능성을 검토하여 실무적용에 효과적인 시스템으로 발전시켜주기를 바랍니다.

종합 평가 점수(100점 만점) : 94
소 속 : 한국건설VE연구원
직 위 : 부원장
일 시 : 2017년 7월 21일
작성자 : 임종권

[임종권 부원장]

평가의견서

[M2M기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크 시스템개발]

1. 평가항목별 점수

▶ 재난전조 예측/재난 대응 모듈 고도화	96	/100
▶ 센서 노드 통신 데이터 수집 및 분석 모듈	96	/100
▶ SOP 및 시나리오 통합 대피경로 안내 모듈	90	/100
▶ M2M 통합보드 대피경로 안내용 보조패널의 방향기호 인지 / 시인성	96	/100
▶ 테스트 베드 운용을 통한 시제품 테스트 : 유사재난 상황별 대응능력 검증	96	/100
▶ M2M 웹기반 관리시스템	90	/100

2. 평가항목별 의견

■ 재난전조 예측/재난 대응 모듈 고도화

재난을 예측하기는 실질적으로 힘든 부분이 많으며, 잘못 예측된 정보를 바탕으로 대응을 하는 경우 더 큰 문제가 발생함. 특히, 본 과제의 경우는 전조나 예측이 어려운 사회재난에 속하고 센서를 통한 상황인식 기능이 중심이므로 신속한 위험 발견과 대응에 주안점을 두는 것이 현실적임.
지하철 역사의 경우 지하공간이기 때문에 승객의 안전을 확보하기 위해서는 무엇보다 신속한 지상으로의 탈출이 우선시 되어야 하며 현재 과제를 통해 개발된 시스템은 유효한 기능을 할 수 있는 대응시스템으로 판단됨.
다만 위험 상황의 판단에 있어서는 위험요소의 위치와 함께 확산방향에 대한 고려도 필요할 것으로 사료됨.

■ 센서 노드 통신 데이터 수집 및 분석 모듈

화재 및 테러 등을 상정하여 연기 이외에도 독성가스를 검출 할 수 있도록 센서노드를 구성하였으며 데이터를 상시적으로 수집 및 축적하여 변화를 감지할 수 있는 기능을 구축하여 시간에 따른 변화 추이를 파악할 수 있는 분석기능도 개발되었음.
향후 축적된 데이터를 활용하면 지하 공간의 공기질 문제 등에도 축적된 데이터가 활용될 수 있을 것으로 사료됨.

■ SOP 및 시나리오 통합 대피경로 안내 모듈

시나리오를 구성하여 발생 가능한 유형의 사고들을 검토 및 고려한 접근방법은 유효적절하다 판단됨. 다만 대응에 있어서는 피난경로 유도가 개찰구를 통한 출구방향 유도로 되어 있어 입구에서 화재나 테러가 발생한 경우에 대응이 어려울 수 있음으로 필요시 철로를 통한 피난이 가능하도록 검토가 필요함. 또한 위험상황에서 개찰구가 피난 흐름에 방해가 될 수 있음으로 이에 대한 개선방안도 제안하기 바람.

■ M2M 통합보드 대피경로 안내용 보조패널의 방향기 인지 / 시인성

대피경로를 효율적으로 안내하기 위한 보조장치로서 유효한 기능을 할 것으로 판단됨. 다만, 현재 개발된 패널의 경우 화살표 크기가 화재가 발생한 경우 시계가 불량하게 되는 점을 고려하면 인식성이 떨어질 우려가 있음으로 화살표의 크기를 크게 할 필요가 있으며, 글씨 등은 가급적 제외하고 사인위주로만 제작하는 것이 효과적일 것으로 사료됨.
본기점에서 방향 안내도 중요하지만 경로를 연속적으로 안내하는 것이 더 중요함.

■ 테스트 베드 운용을 통한 시제품 테스트 : 유사재난 상황별 대응능력 검증

현장적용을 통한 시제품의 테스트는 유효하며, 취득된 데이터를 지속적으로 분석하여 주기적으로 현장최적화를 실시하기 바람
테러가운데 사린가스 등을 활용한 형태의 경우 본 연구과제에서 개발한 시스템을 통해서 위험가스의 검출 등 일부 대응 기능을 할 수 있을 것으로 사료됨.

■ M2M 웹기반 관리시스템

M2M시스템의 상시적 관리를 위해서는 웹기반의 시스템을 활용하여 데이터 모니터링 셋팅 및 관리를 하는 것이 편리할 수 있음.
메뉴는 사용자의 편리성을 고려하여 복잡한 기능을 최대한 단순화시켜야함.

■ 종합 평가 의견

실무적용이 가능한 수준의 시스템 개발 및 현장검증을 실시하였다 판단됨. 다만 현장별로 여건이 다르고 다양한 돌발적 상황이 발생할 수 있음으로 확산적용을 위해서는 지속적인 기능 검증 및 보완 등 후속조치가 필요함.

종합 평가 점수(100점 만점) : 94
소 속 : 전자부품연구원
직 위 : 책임연구원
일 시 : 2017년 7월 21일
작성자 : 김정곤

[김정곤 박사]

평가의견서

[M2M기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크 시스템개발]

1. 평가항목별 점수

▶ 재난전조 예측/재난 대응 모듈 고도화	99	/100
▶ 센서 노드 통신 데이터 수집 및 분석 모듈	99	/100
▶ SOP 및 시나리오 통합 대피경로 안내 모듈	98	/100
▶ M2M 통합보드 대피경로 안내용 보조패널의 방향기호 인지 / 시인성	99	/100
▶ 테스트 베드 운용을 통한 시제품 테스트 : 유사재난 상황별 대응능력 검증	95	/100
▶ M2M 웹기반 관리시스템	98	/100

2. 평가항목별 의견

■ 재난전조 예측/재난 대응 모듈 고도화

- 네트워크 단절시 각 N키퍼의 역할 정의와 재연결시 마스터서버 및 N키퍼의 대응 알고리즘 개발 탑재가 성공적으로 개발이 되었음.
- 2차년도 전문가 자문 평가 개선 제안 사항들도 잘 반영된 것으로 판단됨.
- 통합 디스플레이 모듈 및 EL Display 제어 모듈이 잘 개발 된 것으로 판단됨.
- 개발된 N키퍼 시스템과 통합 및 EL Display 모듈들이 재난전조 예측/재난 대응 모듈 고도화에 잘 기여 할 것으로 판단됨.

■ 센서 노드 통신 데이터 수집 및 분석 모듈

- LoRa 및 디스플레이 통신 모듈의 양방향 통신 개선 사항들을 확인하였음.
- 데이터 품질 개선 및 센서링 타임에 의한 보정 및 최적화 제어 잘 이뤄졌음.
- LoRa 네트워크의 시각동기화 구현 고도화 작업 목적을 달성 한 것으로 판단됨.

■ SOP 및 시나리오 통합 대피경로 안내 모듈

- 최초 재난 발생 구역을 기점으로 대피 시나리오 적절한 것으로 판단 됨.
- 센서 별 화재 및 화학테러 시나리오 룰셋 고도화 작업이 목적을 달성 한 것으로 판단됨.

■ M2M 통합보드 대피경로 안내용 보조패널의 방향기 인지 / 시인성

- M2M 통합 N키퍼 보드 시제품이 네트워크 단절시 각 N키퍼의 역할 정의에 맞게 잘 개발됨. 즉, 네트워크 단절 후 재 연결 시 마스터서버 및 N키퍼의 대응 알고리즘이 제대로 탑재 되어서, 2차년도 전문가 자문 평가 개선 제안 사항들이 반영이 제대로 된 것으로 판단됨.
- 통합 디스플레이 모듈 및 EL Display 제어 모듈 개발이 잘 된 것으로 판단됨.
- 대피유도 안내 시스템과의 연동 및 N키퍼와 EL Display의 제어 통합 방향 제어 기능이 잘 개발 되어, 2차년도 전문가 자문 평가 개선 제안 사항이 잘 반영 된 것으로 판단됨.

■ 테스트 베드 운용을 통한 시제품 테스트 : 유사재난 상황별 대응능력 검증

- 실무적용이 가능한 시제품 시스템이 잘 개발 된 것으로 판단됨.
- 현실적인 현장검증을 잘 실시 된 것으로 판단됨.
- 테스트 베드 운용을 통한 시제품 테스트, 실측데이터 기반 에뮬레이션 데이터 생성하여 재난 상황 시뮬레이션 테스트, 및 갤리브레이션 재료를 통한 센서노드 재난상황 시뮬레이션 모두 잘 된 것으로 판단 됨.
- 지하철역 마다 구조가 다르므로 추가 실험을 통한 재난 상황 기반의 평가를 권유함.

■ M2M 웹기반 관리시스템

- M2M 웹기반 관리시스템 안정화 및 버그 수정이 잘 되었음.
- 테스트베드 역무실 마스터 서버에서 외부 관리 서버로 상황 및 데이터 전송 및 관리 체계 잘 구성 된 것으로 판단됨.
- 통신모듈 설정 및 개별 모듈 구동 안정화 잘 된 것으로 판단됨.
- M2M 시스템 보안 설정 적합한 것으로 판단됨.
- 메뉴얼 작성 제대로 된 것으로 판단됨.

■ 종합 평가 의견

- 재난전조 예측/재난 대응 모듈 고도화 성공한 것으로 판단됨.
- 센서 노드 통신 데이터 수집 및 분석 모듈 잘 개발 된 것으로 판단됨.
- SOP 및 시나리오 통합 대피경로 안내 모듈 잘 개발 된 것으로 판단됨.
- M2M 통합보드 대피경로 안내용 보조패널의 방향기가 잘 개발 것으로 판단됨.
- 테스트 베드 운용을 통한 시제품 테스트를 통한 유사재난 상황별 대응능력이 검증 되었으나 만약을 대비하여 구조가 다른 타 지하철역에서도 추가 실험을 권유함.
- M2M 웹기반 관리시스템 잘 개발 것으로 판단됨.
- 종합적으로, 과제가 매우 성공적으로 잘 진행 되는 것으로 판단 됨.

종합 평가 점수(100점 만점) : 98
소 속 : 연세대학교
직 위 : 교수
일 시 : 2017년 7월 21일
작성자 : 정종문

[정종문 교수]