Journal of the Korea Society of Disaster Information Vol.13 No.1 ▌pp. 26 - 34

Available online at www.sodi.or.kr

통합 서비스를 위한 지하매설관 하부 이벤트 정보전달 인터페이스 개발

The Development of sub-event Information interface for Integrated management of underground pipelines

Sookwon Chae[a,1], Jaesoon Seo[b,2], Joonseok Kim[c,*]

[a] *Department of Environmental Health and Safety, University of Eulji, 553 Sanseong-Daero, Seongnam, 461-713, Republic of Korea*
[b] *Department of R&D, WACON Co., Ltd, 15, Nonhyeon-ro, 26Beon-gil, Namdong-gu, Incheon, 405-825, Republic of Korea*
[c] *Department of civil and envirnmental engineering, Chungwoon University, 113 Sukgol-ro, Nam-gu, Incheon, 402-803, Republic of Korea*

A B S T R A C T

In this study , for the integrated information management of underground pipelines, each information management server software must have it's inter-operability. So Many kinds of smart-city integrated platform softwares were surveyed and the best platform software was selected. The interface software modules developed in this study was installed at the test system. Through this test system, when a sub system transfers an event message to the upper integrated server system, the inter-operability test between the upper system and a sub operating system was performed and its operability was resolved.

KEYWORDS

Real-time Monitoring
Underground Pipeline
Accident Prevention
Interface
Integrated
Management
Inter-Operability

지하매설배관의 실시간 통합 유지관리를 위하여 최상위의 관제 및 유지관리 서버 소프트웨어와 정보전달 체계를 만들기 위해서는 호환성을 유지하기 위한 인터페이스가 필요하다. 이를 위하여 스마트 시티를 끌어가는 국내의 다양한 플랫폼 운영 소프트웨어를 조사하고. 시장 지배력이 가장 큰 플랫폼을 선정하여 이들이 가지고 있는 정보전달 인터페이스에 맞도록 소프트웨어 모듈 개발을 수행하였다. 또한, 기존 하위 지하매설배관 운영프로그램에 개발된 인터페이스 모듈을 이식하고, 상위 도시통합관제 플랫폼 시스템과 연동하였다. 이 과정에서 예방, 파손, 누수와 같은 하위의 정보가 상위 도시통합 플랫폼 시스템에 전달되어 정상적으로 동작이 되는지를 확인하였다.

실시간 감시
지하매설배관
사고예방
인터페이스
통합관리
상호호환성

* Corresponding author. Tel. 82-32-770-3701. Fax. 82-32-770-8138. Email. jskim@chungwoon.ac.kr
1 Tel. 82-31-740-7146. Email. cskwen@eulji.ac.kr
2 Tel. 82-32-421-1672. Email. jsseo21c@naver.com

ARTICLE HISTORY
Recieved Nov. 23, 2016
Revised Dec. 08, 2016
Accepted Dec. 22, 2016

http://dx.doi.org/10.15683/kosdi.2017.03.31.26

1. 서 론

최근 국가적 도시 통합 관제를 위한 스마트 시티 구축에 관한 관심이 증대되고 있으며, 이를 위한 다양한 도시 인프라를 구축하기 위한 도시 운영 시스템들이 등장하고 있다. 이 중에는 전기, 통신, 상수도, 하수도, 에너지, 수자원, 안전, 방재 등 다양한 요소들이 존재한다(Management Consulting Research Society(2015).

이러한 도시 운영 시스템들은 모두 자기 고유의 솔루션을 제공하기 위해서 개발되고 있다. 따라서, 최근에는 이들 운영 시스템들에서 제공되는 서비스를 통합하여, 도시 운영의 효율화를 추구하여야 할 필요성이 제시되고 있다. 이러한 발전 추세에 따라서 어떠한 시스템들이 있는지를 조사하는 것이 우선되어야 한다(Korea Information Society Agency(2015). 향후 도시 통합 운영 시스템 구축에 필수적으로 사용되는 도시 통합 운영 플랫폼 소프트웨어를 선정하여, 이와 상호 연동할 수 있는 지하매설배관의 운영 시스템의 정보 인터페이스 방법을 구현할 필요가 있는 것으로 조사되었다. 본 논문에서는 상기와 같이 상호 연동할 수 있는 지하매설배관 서비스 운영정보인 예방감지, 파손감지, 누수감지 메시지를 상위의 도시통합 운영 시스템과 연결하고, 상호 호환성 및 전송처리 반응속도를 측정하는 인터페이스를 개발하였으며, 시험을 통하여 그 개발결과를 확인할 수 있었다.

2. 개발 시스템 모델 구성

2.1 개발 시스템 구성

본 시스템은 서비스 운영 시스템인 지하매설배관 운영관리 시스템 부분과 도시통합 운영관리 부분 등 크게 2 부분으로 구성된다. 그림1에 두 부분에 대한 운영 흐름을 블록도로 도시하였다. 상세 연구개발 부문의 첫 번째로는 서비스 정보 정규화로서 하위 시스템 속에 혼재한 정보 데이터를 상위 시스템으로 전송하기 위한 메시지 구조로 변환하기 위한 메시지의 정규화이다(O'Reilly(2016). 두 번째로는 서비스 프로토콜 개발로서 정규화된 메시지를 실어서 전송할 프로토콜의 선정 및 구현이다. 세 번째로는 서비스 표출모형 개발로서 구현된 서비스 프로토콜을 이용한 상위 운영 시스템과의 연계 동작상황에 대한 확인이다.

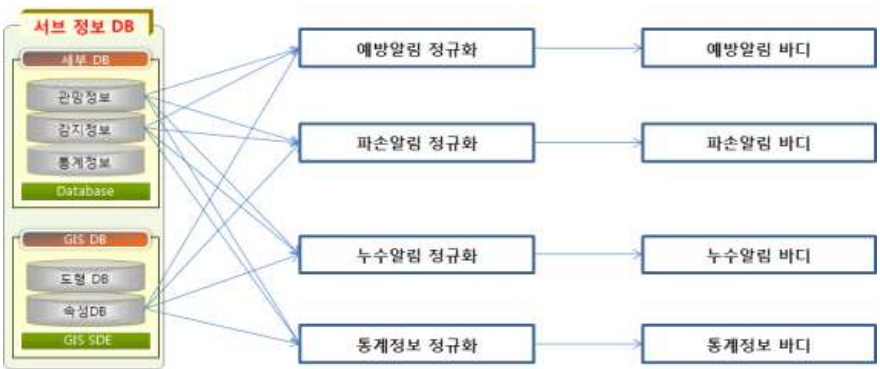


Fig. 1 Normalized event message

2.2 서비스 이벤트 메시지 정규화

정규화는 혼재된 데이터를 가공하여 서비스를 수행하기 위한 정형된 형태의 문장으로 표현하는 것을 의미하는 것으로서, 이를 통하여 전송 프로토콜에 운영정보를 실어서 상위 시스템에서 메지지 내용을 확인할 수 있도록 하는 과정이다. 본 연구에서는 상위 도시통합 운영 플랫폼 소프트웨어에서 하위 시스템으로부터 제공받는 데이터를 표준적으로 인식할 수 있도록 메시지를 구성할 필요가 있다. 따라서, 상위 통합운영 시스템에 하위 서비스 정보(예방, 파손, 누수 위치정보)를 전달하기 위한 정규화 표현식을 다음과 같이 도출하였다.

Table 1. Normalized expression

```
payload : {
  "values": [
        {"timestamp" : "20150413095012125", "value": [1-3]},          // 이벤트 구분 코드
        {"timestamp" : "20150413095012125", "value": [1-3]},          // 상태( 파손/누수/복구 )
        {"timestamp" : "20150413095012125", "value": 36.361195,},   // 위치좌표(위도)
        {"timestamp" : "20150413095012125", "value": 127.378461,}, // 위치좌표(경도)
        {"timestamp" : "20150413095012125", "value": 1000}          // 위치좌표(높이)
    ]
}
```

파라미터	타입	설명
timestamp	String	데이터 값 측정 일시 (default: 현재 시간) 포맷 - yyyyMMddHHmmssSSS (17자리)
value *	String	데이터 값

		// 이벤트 구분코드
		1 : 예방 / 2 : 파손 /3 : 누수
		// 상태 구분코드
		1 : 파손 / 2 : 누수 / 3 : 복구
		// 이벤트 위치 위도
		이벤트 발생 위치좌표 위도 값.
		// 이벤트 위치 경도
		이벤트 발생 위치좌표 경도 값.
		// 이벤트 위치 높이
		이벤트 발생 위치좌표 높이 값.

2.3 서비스 인터페이스 프로토콜 개발

서론에서 언급한 것처럼, 다양한 통합운영 시스템의 인터페이스 프로토콜을 조사한 결과, 대부분이 자기 고유의 전송 프로토콜을 가지고 있는 것으로 확인되었으며, 전용의 TCP/IP socket 프로토콜, FTP 프로토콜, HTTP 프로토콜 등으로 다양하였다. 그 중에서 최근에 사용빈도가 높은 것은 HTTP 전송 프로토콜로 확인되었다. 이것은 서비스가 웹을 기반으로 구현되고 있다는 점 때문에 가장 많이 선택되고 있는 것으로 추정된다.

이에 본 연구에서도 향후 시장성과 호환성을 높이기 위해서는 HTTP 전송 프로토콜을 사용하여 연계 연동시키는 것이 효율적인 것으로 판단되어 HTTP 전송 프로토콜을 개발하였다. HHTP 전송 프로토콜은 다양한 서비스 메소드를 제공하고 있는데, 대표적인 것으로서 'GET', 'PUT', 'POST' 등 다양한 종류가 제공되고 있다. 이는 서비스의 기능에 따라서 구분되어 사용된다. 본 연구에서는 정해진 시간에 정보를 수집하는 것이 아니고, 언제 발생할지 모르는 현장의 사고 발생 이벤트를 처리하기에 가장 좋은 방법으로 판단되는 'POST' 메소드를 사용하여 구현하였다. 따라서, POST 메소드에 들어가 메시지 구조를 아래와 같이 정의함으로써 HTTP POST 메소드를 만족시켰다.

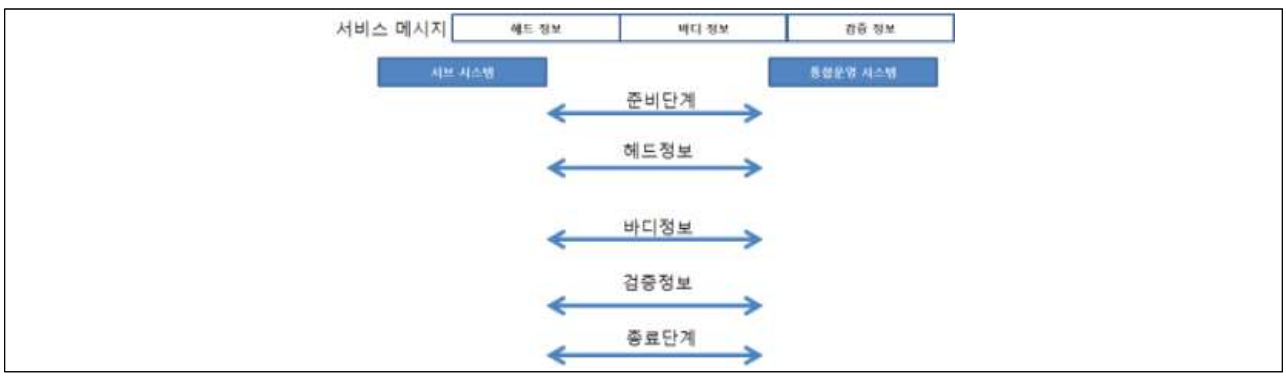


Fig. 2 Service protocol

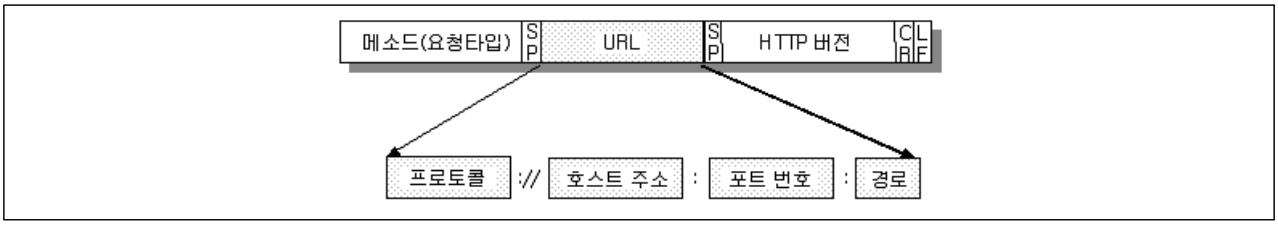


Fig. 3 HTTP service protocol

2.3.1 스트림 값 전달(배열)

데이터 스트림 값(배열)을 서버로 전달하는 스트림 값 전달(배열)(Post Data Stream Values)은 아래와 같이 구성하였으며, 요청 주소와 요청헤더, 요청변수 예제는 아래 표2~표7에 예시하였다.

Table 2. Data stream

```
POST] http://iot.url/devices/{device_id}/streams/{stream_name}/values
var msg = {
  action : “postDataStreamValues”,
  device_id : “f97085afbec60a3de29ffff5f1aa701b”,
  stream_name : “Stream Data”,
  payload : {
    "values": [
          {"timestamp" : "20150413095012125", "value": 1},            // 이벤트 구분 코드
          {"timestamp" : "20150413095012125", "value": 1},            // 상태( 파손 )
          {"timestamp" : "20150413095012125", "value": 36.361195,}, // 위치좌표(위도)
          {"timestamp" : "20150413095012125", "value": 127.378461,}, // 위치좌표(경도)
          {"timestamp" : "20150413095012125", "value": 1000}          // 위치좌표(높이)
        ]
  }
}
return msg;
```

Table 3. Request URL

[POST] http://iot.url/devices/{device_id}/streams/{stream_name}/values

Table 4. Request header

헤더	타입	설명
IoT-KEY	String	API 키

Table 5. Request parameters

파라미터	타입	설명
device_id *	String	디바이스 아이디
stream_name *	String	데이터 스트림명
timestamp	String	데이터 값 측정 일시 (default: 현재 시간) 포맷 - yyyyMMddHHmmssSSS (17자리)
value *	String	데이터 값

Table 6. Request example

```
Content-Type: application/json
{
  "values": [
          {"timestamp" : "20150413095012125", "value": 1},           // 이벤트 구분 코드
          {"timestamp" : "20150413095012125", "value": 1},           // 상태( 파손 )
          {"timestamp" : "20150413095012125", "value": 36.361195,}, // 위치좌표(위도)
          {"timestamp" : "20150413095012125", "value": 127.378461,}, // 위치좌표(경도)
          {"timestamp" : "20150413095012125", "value": 1000}         // 위치좌표(높이)
        ]
}
```

Table 7. IoT Flow Designer function Node example

```
var msg = {
  action : "postDataStreamValues",
  device_id : "f97085afbec60a3de29ffff5f1aa701b",
  stream_name : "Stream Data",
  payload : {
    "values": [
          {"timestamp" : "20150413095012125", "value": 1},           // 이벤트 구분 코드
          {"timestamp" : "20150413095012125", "value": 1},           // 상태( 파손 )
          {"timestamp" : "20150413095012125", "value": 36.361195,}, // 위치좌표(위도)
          {"timestamp" : "20150413095012125", "value": 127.378461,}, // 위치좌표(경도)
          {"timestamp" : "20150413095012125", "value": 1000}         // 위치좌표(높이)
        ]
  }
}
return msg;
```

파라미터	타입	설명
action	String	IoT 인터페이스 노드 API 명
device_id	String	디바이스 아이디
stream_name	String	데이터 스트림 명

payload	Object	// 이벤트 구분코드
		1 : 예방
		2 : 파손
		3 : 누수
		// 상태 구분코드
		1 : 파손
		2 : 누수
		3 : 복구
		// 이벤트 위치 위도
		이벤트 발생 위치좌표 위도 값.
		// 이벤트 위치 경도
		이벤트 발생 위치좌표 경도 값.
		// 이벤트 위치 높이
		이벤트 발생 위치좌표 높이 값.

2.3.2 서비스 인터페이스 표출모형 구현

그림4에서 HTTP 전송 프로토콜에 실어서 보낼 메시지 구조에 따라서, 하위 운영 시스템에서는 데이터베이스에 저장된 운영 데이터에서 메시지 구조에 마추어 메시지를 생성할 필요가 있다. 이렇게 생성되어야 하는 메시지에는 파손예방 이벤트 처리 메시지, 파손 이벤트 처리 메시지 및 누수 이벤트 처리 메시지로 구성된다. 이를 구성하기 위해서는 아래와 같은 절차에 따라서 구현하였다. 소스코드는 표8과 같다.

① 위치정보는 관망정보 데이터베이스에 저장되어 있음.
② 이벤트 발생정보는 이벤트 정보 데이터베이스에 저장되어 있음.
③ 따라서, 현장에서 발생하는 이벤트는 하위 시스템의 화면에 위치를 표출함.
④ 표출된 위치정보는 다시 상위 통합관리 시스템으로 위치정보를 전송하기 위함.
⑤ 본 논문에서 개발된 연동 프로토콜 메시지 형식에 따라서 메시지를 생성함.
⑥ 상위 통합관리 시스템으로 전송함.
⑦ 전송된 메시지는 상위 통합관리 시스템의 이벤트 관리 데이터베이스에 저장됨
⑧ 저장된 메시지는 다시 위치정보 상태에 따라서, 상태기호를 통하여 위치 좌표값에 따라서 화면에 표출하게 됨.
⑨ 표시 상태는 파손, 누수 표시로 나타나며, 복구 이벤트가 발생하면 화면에서 사라지게 됨.

Table 8. Source code

예방 메시지 처리 소스코드	파손 메시지 처리 소스코드	누수 메시지 처리 소스코드

szIoTKey = 'hh936986622101f1a15b61a7b7b8efeb'
szBodys := '{"value":"TDR0001,0,'+dLatitude+','+dLongitude+'"}';
 	
JsonToSend := TStringStream.Create(Utf8Encode(szBodys));
try
JsonToSend.Position := 0;
try
 	
IdHTTP1.Request.ContentType := 'application/json';
 	
IdHTTP1.Request.Connection := 'close';
 	
IdHTTP1.Request.CustomHeaders.Values['IoT-KEY'] := szIoTKey;
 	
sResponse := IdHTTP1.put('http://'+szUrl+':'+intToStr(nPort)+szPath , JsonToSend);
except
on E:Exception do Memo1.lines.Add('ER='+ E.Message);
end;
Memo1.lines.Add(sResponse);
 	
finally
JsonToSend.Free;
end;
	szIoTKey = 'hh936986622101f1a15b61a7b7b8efed'
	szBodys := '{"value":"TDR0001,0,'+dLatitude+','+dLongitude+'"}';
	
	JsonToSend := TStringStream.Create(Utf8Encode(szBodys));
	try
	JsonToSend.Position := 0;
	try
	
	IdHTTP1.Request.ContentType := 'application/json';
	
	IdHTTP1.Request.Connection := 'close';
	
	IdHTTP1.Request.CustomHeaders.Values['IoT-KEY'] := szIoTKey;
	
	sResponse := IdHTTP1.put('http://'+szUrl+':'+intToStr(nPort)+szPath , JsonToSend);
	except
	on E:Exception do Memo1.lines.Add('ER='+ E.Message);
	end;
	Memo1.lines.Add(sResponse);
	
	finally
	JsonToSend.Free;
	end;
		szIoTKey = 'hh936986622101f1a15b61a7b7b8efee'
		szBodys := '{"value":"TDR0001,0,'+dLatitude+','+dLongitude+'"}';
		
		JsonToSend := TStringStream.Create(Utf8Encode(szBodys));
		try
		JsonToSend.Position := 0;
		try
		
		IdHTTP1.Request.ContentType := 'application/json';
		
		IdHTTP1.Request.Connection := 'close';
		
		IdHTTP1.Request.CustomHeaders.Values['IoT-KEY'] := szIoTKey;
		
		sResponse := IdHTTP1.put('http://'+szUrl+':'+intToStr(nPort)+szPath , JsonToSend);
		except
		on E:Exception do Memo1.lines.Add('ER='+ E.Message);
		end;
		Memo1.lines.Add(sResponse);
		
		finally
		JsonToSend.Free;
		end;

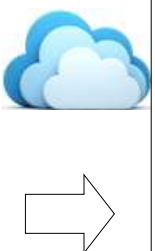
Fig. 4 Create service messages algorithm and transfer process

3. 시스템 동작 시험

3.1 시험 시스템 구성

본 연구에 사용된 재료는 시험을 위한 동일한 하드웨어 환경을 구축하고, 하위 운영 시스템은 기존에 운영하고 있는 지하매설배관 운영 시스템 PipeGuard 제품에 개발된 프로토콜을 이식하여 설치하였으며, 상위 운영 시스템은 가장 많은 시장 점유율을 가지고 있는 ESE 도시통합 관제 운영 시스템을 적용하여 시험을 수행하였다(IoT API Documentation V2.0 ESE Co.). 자세한 시험 환경 구성은 아래 표9에 기술하였다.

Table 9. Operating system

하위 운영 시스템	인터넷 연결 (클라우드)	상위 운영 시스템	하위 운영 시스템 사양		상위 운영 시스템 사양	
			부품	규격	부품	규격
 하위 운영 시스템		 상위 운영 시스템	CPU	intel Core i5-6400	CPU	intel Core i5-6400
			RAM	DDR3L 12800 8GB	RAM	DDR3L 12800 8GB
			Data Storage	SSD 250GB+SATA 500GB	Data Storage	SSD 250GB+SATA 500GB
			비디오	VGA-Intel HD GRAPHIC/ODD-SUPER MULTI	비디오	VGA-Intel HD GRAPHIC/ODD-SUPER MULTI
			O.S	Win 7 Pro 64Bit	O.S	Win 7 Pro 64Bit
			실험 SW	WACON PipeGaurd	실험 SW	ESE 통합 데시보드

3.2 시험 장치 및 방법

본 연구의 성능시험 결과 값을 구하기 위하여 3종의 메시지에 대하여 수행하였으며, 시험절차는 ①각 메시지 발생 명령어를 실행 ②이벤트 발생 메시지를 제거하는 명령어를 수행 ③ 상기의 수행을 10회 반복 수행으로 진행하였다. 시험의 수행은 전문 공인 시험기관에 의뢰하여 스톱워치를 사용하는 방법으로 시험수행을 진행하였다.

4. 시험 결과 분석

4.1 파손예방, 파손, 누수 메시지에 대한 인터페이스 반응속도

3종의 메시지에 대한 생성에서 전송 및 수신에 이르는 과정의 시간을 측정한 결과 값들을 표10에 수록하였다. 원격의 인터넷에 연결된 상위 통합운영 시스템에 도달하여, 정보가 표시될 때까지의 시간을 의미한다.

본 시험의 결과가 상시 운영에 들어가서 발생되는 전체의 결과를 의미하는 것은 아니며, 다만, 실험실에서 수행한 결과 값으로서 인터넷 클라우드 환경은 항상 통신 트래픽의 부하가 바뀔 수 있기 때문에 실제 운영시에는 본 결과보다 오래 걸릴 수 있다. 본 실험결과 목표치 대비하여 비교적 좋은 결과를 도출할 수 있었다.

Table 10. Interfacing event message, response speed measurements

회	시험항목	측정결과(초)	시험항목	측정결과(초)	시험항목	측정결과(초)
1	예방	0.74	파손	0.98	누수	0.80
	복구	0.74	복구	0.70	복구	0.67
2	예방	0.86	파손	0.77	누수	0.74
	복구	0.84	복구	0.75	복구	0.91
3	예방	0.79	파손	0.81	누수	0.78
	복구	0.81	복구	0.82	복구	0.77
4	예방	0.72	파손	0.95	누수	0.76
	복구	0.80	복구	0.80	복구	0.69
5	예방	0.71	파손	0.77	누수	0.85
	복구	0.83	복구	0.79	복구	0.90
6	예방	0.89	파손	0.88	누수	0.79
	복구	0.75	복구	0.71	복구	0.69
7	예방	0.79	파손	0.79	누수	0.82
	복구	0.91	복구	0.89	복구	0.86
8	예방	0.85	파손	0.79	누수	0.78
	복구	0.84	복구	0.77	복구	0.76
9	예방	0.88	파손	0.92	누수	0.82
	복구	0.83	복구	0.81	복구	0.76
10	예방	0.71	파손	0.89	누수	0.95
	복구	0.73	복구	0.78	복구	0.76
평 균	0.80		0.82		0.79	
목표치	50		50		50	

5. 결 론

본 연구에서 개발된 메시지 성능검증 3가지 목표치를 달성할 수 있었다. 다만, 실제 현장에 적용될 경우에는 현장의 상황에 따른 다양한 변수로 인하여 영향을 받을 수 있지만, 향후 현장 적용시에 특별한 문제가 없다면 , 인적재난정보를 실시간으로 제공하는데 기여를 할 수 있을 것으로 판단된다. 본 연구를 통하여 다음과 같은 결론을 얻을 수 있었다.

(1) 지하매설관 하부 이벤트 정보(파손예방, 파손, 누수 이벤트 정보)를 상위 통합 시스템으로 전달하기 위한 메시지의 정규화를 수행하여, 메시지 구조를 설계할 수 있었다.
(2) 설계된 3종의 메시지를 통합 운영 플랫폼에서 사용하는 HTTP 전송 프로토콜의 POST 메소드를 이용하여, 3종의 메시지가 전달될 수 있도록 인터페이스 프로토콜을 구현할 수 있었다.
(3) 또한, 개발된 인터페이스 프로토콜 모듈을 하위의 지하매설관 운영 프로그램 PipeGuard 프로그램에 이식하여, 상위 운영 프로그램 EST 도시통합 플랫폼과 연계 시험을 수행할 수 있었다.
(4) 연구시스템을 공인시험기관에 의뢰하여, 각 메시지의 성능시험을 수행하였으며, 파손예방 0.8초, 파손 0.82초, 누수 0.79초 등으로 개발 목표인 50초 이내의 비교적 우수한 값들을 얻을 수 있었다.

감사의 글

이 논문은 한국산학연협회 중소기업기술개발지원사업-연구마을(과제코드:C0363132)의 지원을 받아 수행된 연구 결과이며 이에 감사드립니다.

References

Management Consulting Research Society(2015), “For the prevention and efficient asset management of Ground Subsidence smart sheet system of cost - benefit case study”, 2,

Korea Information Society Agency(2015), “U-City IT Detailed construction of infrastructure guidelines. V2.0.” Korea Information Society Agency

O’Reilly(2016), “Regular expression to start first ‘Regualar Expressions”, Hanbit media, pp.19

IoT API Documentation V2.0 ESE Co.