Journal of KIIT. Vol. 18, No. 6, pp. 117-127, Jun. 30, 2020. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2020.18.6.117

시맨틱 기반 IoT 매쉬업 플랫폼 개발

남성현*, 이용주**

Developing a Semantic-based IoT Mashup Platform

Seonghyeon Nam*, Yongju Lee**

이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임. (No. 2016R1D1A1B02008553). 이 논문은 2019년 대한민국 교육부와 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2019S1A5A2A03049067). 본 논문은 교육부 및 한국연구재단의 BK21 플러스 사업(경북대학교 컴퓨터학부 Smart Life실현을 위한 SW인력양성사업단)으로 지원된 연구임(21A20131600005)

요 약

국내에서 시맨틱 IoT 매쉬업 처리 기술은 아직 초보 수준에 머무르고 있으며 M2M, USN 환경을 고려한 체계적인 연구개발은 아직까지 되어 지고 있지 않다. 이러한 상황에서 현재 IoT 기술 및 응용에 대한 핵심 이슈는 시맨틱 기반 IoT 매쉬업 플랫폼의 개발이다. 본 논문에서는 시맨틱 기반 IoT 매쉬업 플랫폼을 위한 처리 기술들을 개발하였다. 구현된 플랫폼은 클라우드 컴퓨터로부터 IoT 데이터를 수집하고, 이들을 RDF로 변환한 후 시맨틱 주석 처리한다. 변환된 데이터는 IoT 온톨로지 모델을 통해 온톨로지로 구축되고 지식베이스에 저장된다. 매쉬업 시스템과 온톨로지 저장소는 Amazon 클라우드 서비스인 AWS 환경에서 Apache Jena 시맨틱 웹 프레임워크를 사용하여 구현되었다.

Abstract

The semantic IoT mashup processing technology is still at the beginning level in Korea, and the systematic research and development considering M2M and USN environments has not yet been conducted. In this situation, the key issue for current IoT technologies and applications is the development of a semantic-based IoT mashup platform. In this paper, we developed processing technologies for the semantic-based IoT mashup platform. The implemented platform collects IoT data from the cloud computer, converts them into RDF, and annotates them with semantics. The transformed data are built into the ontology and stored into the knowledge-base through the IoT ontology model. The mashup system and ontology storage were implemented using the Apache Jena semantic web framework in the AWS environment, an Amazon cloud service.

Keywords

IoT mashup platform, RDF, SPARQL, IoTMakers, Amazon AWS, apache jana

* 경북대학교 IT대학 컴퓨터학부 석사과정
- ORCID: https://orcid.org/0000-0003-4516-1089
** 경북대학교 IT대학 컴퓨터학부 교수(교신저자)
- ORCID: https://orcid.org/0000-0002-1705-4967

· Received: Apr. 14, 2020, Revised: May 15, 2020, Accepted: May 18, 2020
· Corresponding Author: Yongju Lee
School of Computer Science and Engineering, Kyungpook National University, 80, Daehak-ro, Buk-gu, Daegu 41566, Korea,
Tel.: +82-53-950-7285, Email: yongju@knu.ac.kr

Ⅰ. 서 론

최근 스마트폰이 새로운 생태계를 창출한 것처럼 IoT(Internet of Things)도 지능화됨으로써 기존에 얻지 못했던 새로운 차원의 정보를 제공하게 되었다. 이는 새로운 서비스 생태계 형성에 크게 기여할 것으로 예상된다[1][2]. 근래 하루가 멀다 하고 새로운 IoT 디바이스가 출시되고 있다. 이런 추세라면 시장조사 기관들이 전망한 것처럼 2023년에는 그 수가 300억 개에 다다를 것으로 예상한다. 이렇게 IoT 디바이스의 수가 증가하고 그에 따르는 데이터의 양이 폭증됨에 따라 이들을 수집하고 새롭게 재가공할 수 있는 기술의 필요성이 증가하고 있다[3]. 이에 따라 더욱 가치 있는 데이터 정보로 가공되고 활용되기 위한 시맨틱 기반의 IoT 서비스 매쉬업(Mashup)에 대한 연구가 필요하다.

IoT 서비스 매쉬업이란 이동통신사 중심으로 단말의 연결성을 중요시하는 M2M 플랫폼, 센싱 정보나 수집 정보를 활용해 서비스를 강조한 기존의 IoT 플랫폼, 그리고 단말간 협업과 웹을 통한 서비스가 가능한 WoT(Web of Things) 플랫폼에 시맨틱 기술이 추가되어 지능적인 서비스를 제공되는 것을 말한다[4][5]. 이는 LOD(Linked Open Data) 등과 결합하여 새롭고 값어치 있는 서비스를 재생산할 수 있다.

IoT 플랫폼에서 요구되는 기술은 3가지로 압축된다. IoT 디바이스에서 얻을 수 있는 센서기술, 확보된 센서 데이터를 다양한 네트워크를 통해 서버로 보내는 통신 기술, 그리고 이렇게 모여진 데이터를 새로운 가치로 만들어 낼 수 있는 매쉬업 서비스 기술이다[2]. 또한 이러한 기술들이 접목된 IoT 서비스를 제공하기 위해서는 센서 데이터의 개방이 필수적이며 다양한 IoT 디바이스 데이터를 저장하고 새로운 데이터로 가공할 수 있는 오픈 데이터를 제공하여야 한다. 즉, 서비스 개발자들에게 고부가가치 매쉬업 서비스를 개발할 수 있는 환경을 제공하는 개방형 데이터 플랫폼이어야 한다. 하지만 개방형 데이터 플랫폼을 위해 수년전부터 네이버, 다음, 구글 등 국내외 주요 포털 사이트를 중심으로 다양한 OpenAPI들을 제공하고 이를 활용한 매쉬업 개발이 활발하였으나, 센서 데이터의 경우 외부 개발자들이 용이하게 정보를 제공 받아 활용할 수 있는 플랫폼은 거의 찾아보기 힘들었다.

따라서 본 논문에서는 IoT OpenAPI로부터 모여진 센서 데이터를 보다 가치 있는 데이터 정보로 가공 및 활용하기 위하여 시맨틱 기반 IoT 매쉬업 플랫폼을 위한 처리 기술들을 개발하고, 이를 위한 IoT 센서 온톨로지 설계하고, IoT 매쉬업 시스템을 구현한다. 이를 통해 가공된 시맨틱 IoT 데이터를 공유할 수 있는 다양한 환경을 제공한다. 우리는 이미 시맨틱 기반 IoT 플랫폼을 제안한 바 있다[2][3]. 하지만 기존 플랫폼의 가장 큰 문제점은 온프레미스(On-premise) 환경에서 서비스 되었고, Jena의 Fuseki 서버를 사용하여 효율적인 대용량 RDF 쿼리를 수행하기에는 문제점이 많았었다.

본 논문은 이러한 문제점을 해결한 개방형 IoT 매쉬업 플랫폼을 개발하기 위해 다음과 같이 구성하였다. 2장에서 관련 연구를 살펴보고, 3장에서 시맨틱 기반 IoT 매쉬업 플랫폼을 제안한다. 4장에서 IoT 매쉬업 시스템을 설계 및 구현하고, 5장에서 시스템 성능 분석을 수행한다. 그리고 6장에서 결론을 내린다.

Ⅱ. 관련 연구

2.1 국내외 관련 연구 동향

과거의 M2M 영역은 디바이스간 통신을 바탕으로 위치추적, 텔레메틱스, 원격검침 정도였으나 최근에는 M2M 서비스 분야가 각 산업별로 그 영역을 확장하고 있고 B2B, B2C가 중요시 되어가는 추세이다. 수년전부터 네이버, 다음, 구글 등 국내외 주요 포털 사이트를 중심으로 다양한 OpenAPI 제공 및 이를 활용한 매쉬업 개발이 활발해지고 있으나, 센서 데이터의 경우에는 외부 개발자가 용이하게 정보를 제공 받아 활용할 수 있는 플랫폼은 찾아보기 힘든 상황이다.

국내의 경우 센서 데이터 관리 및 질의처리를 위하여 COSMOS[6], uSeed, Spiroad-USN 등 여러 USN (Ubiquitous Sensor Network) 미들웨어 솔루션들이

개발되었으나 현재는 개발이 중지된 상태다. ETRI의 시맨틱 USN 서비스 플랫폼인 COSMOS는 다양한 디바이스들이 생성한 센싱 정보를 온톨로지 기반 시맨틱 정보로 변환하여 시맨틱 기반 연관 정보 검색 및 추론 기능을 제공하는 플랫폼이다. COSMOS는 USN 자원의 메타데이터를 RDF로 변환하고 이벤트 정보 추론을 통해 상황인지 등을 수행하지만 개발자들이 쉽게 창의적인 매쉬업 서비스를 지원하는 기능은 제공하고 있지 않다. 우리가 개발중인 시맨틱 기반 IoT 매쉬업 플랫폼은 IoT OpenAPI 데이터를 RDF로 변환하고 이벤트 정보 추론을 위해 시맨틱 데이터로 가공하여 상황 인지를 수행하여, 외부 개발자들이 다양한 환경에서 용이하게 정보를 제공 받을 수 있도록 한다. 하지만 지금까지 발표된 논문[2][3]에서는 클라우드 컴퓨팅 환경은 지원하지 못하였고 대용량 시맨틱 매쉬업 처리를 위한 트리플 저장소 기능이 부족하였다.

W3C에서는 SWE(Sensor Web Enablement)와 센서 네트워크 온톨로지와의 연계 표준 기술을 개발하였고 이를 활용하여 ebbits[7], OpenIoT[8] 등 EU의 FP7 프로젝트들이 다수 진행되고 있다. 특히 EU를 중심으로 IoT 장치간 상호운용성 확보를 위해 시맨틱 기술의 중요성이 부각됨에 따라 향후 시맨틱 IoT의 기술이 매우 활발히 연구될 것으로 전망된다. WoO(Web of Object) 연구[9]는 유럽의 EUREKA 프로젝트에서 진행 중인 IoT 기술이며, 사물을 오브젝트화하여 다양한 콘텍스트 정보와 서비스들을 결합하여 새로운 서비스를 창출하는 IoT 시맨틱 연구를 수행하고 있다. 또한, IoT 서비스 지능화를 통해 사물간 상호 작용하고 디바이스간 매쉬업을 제공하여 기존의 서비스보다 향상된 서비스를 지원하는 연구들로 DIYSE[10], DPWS[11], 그리고 SOCRADES[12] 프로젝트들이 있다.

2.2 시맨틱 IoT 매쉬업 플랫폼 구현 기술

공공 인프라적 성격의 센서 정보를 함께 활용하기 위해서는 IoT 데이터의 개발과 공유가 필연적이며, 개방된 데이터가 외부에 널리 활용되기 위해서는 개발자들이 쉽게 데이터를 찾고 활용할 수 있는 개방형 서비스 플랫폼을 지원해야 한다. 공개되는 IoT 데이터는 시맨틱 데이터 표현 방식인 RDF (Resource Description Framework) 표준에 따라 각 USN 자원의 메타데이터로 변환되고 센싱 값에 의한 이벤트 정보 추론을 통해 추론된 정보를 시맨틱 데이터로 가공되어 고부가가치 정보를 재생산하게 된다[2].

IoT 서비스는 상황 정보를 끊임없이 수집하여 상황에 적합한 행동을 수행해야 한다. 온톨로지는 IoT 환경의 사람 및 다양한 사물들로부터 수집되는 상황 정보의 의미를 논리적으로 표현하기 적합하며, 사람뿐만 아니라 IoT를 구성하는 다양한 기계들 사이에서 상황 정보에 대한 의미적인 상호운용성을 지원할 수 있다[13].

매쉬업 플랫폼에서 공급자는 IoT 디바이스 정보를 REST(REpresentational State Transfer) 형식으로 매쉬업 서버에 등록한다. 등록된 REST 메타데이터는 등록 모듈을 거쳐 온톨로지로 구축되고 지식베이스에 저장된다. 플랫폼의 API는 OpenAPI 형태로 제공되고 시맨틱 기반 검색 엔진에 의해 IoT 서비스가 지능적으로 검색된다. 또한 매쉬업 엔진에 의해 다른 IoT 서비스와 기존의 OpenAPI, LOD 등과 결합되어 새롭고 값어치 있는 서비스를 재생산 할 수 있도록 지원된다.

III. 시맨틱 기반 IoT 매쉬업 플랫폼

3.1 Open API를 통한 데이터 수집 및 변환

본 연구에서는 IoT OpenAPI를 제공하는 여러 포털 사이트들 중 KT의 IoTMakers[14]에서 RESTful방식의 OpenAPI를 통해 디바이스로부터 수집된 데이터를 받아온다. IoTMakers는 손쉽게 IoT 디바이스를 연결하여 테스트 할 수 있고, 수집된 데이터를 관리할 수 있으며, 제공되는 OpenAPI를 통하여 IoT 서비스를 만들 수 있는 KT의 개방형 IoT 플랫폼이다. 또한 실제 디바이스가 없어도 가상 디바이스와 공개 디바이스를 통해 원하는 정보를 제공받을 수 있고 제어할 수 있다.

IoTMakers로부터 데이터를 수집하기 위해서는 인

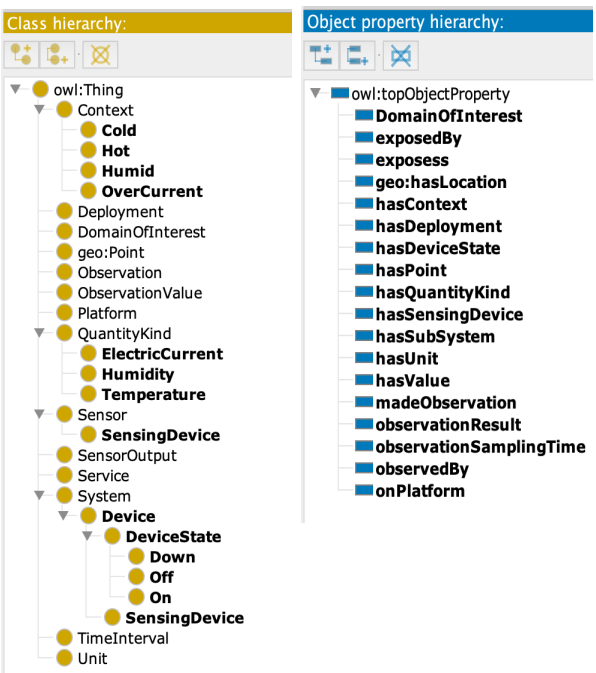
증 토큰을 받아 API를 사용해야 하는데 이를 위한 인증 방식은 Basic을 사용해야 한다. Basic 인증 방식은 IETF(Internet Engineering Task Force)의 RFC 7617[15]에 정의되어 있으며, 이는 base64를 이용하여 인코딩된 사용자 아이디와 비밀번호 쌍의 인증 정보를 전달하는 방식이다. 두 번째 인증 과정으로 Basic 인증을 통해 얻은 인증 토큰으로 Bearer 인증을 한다. Bearer 인증이란 API에 접속하여 사용하기 위해서는 인증 토큰을 API 서버에 제출해서 인증해야 하는데, 이 때 사용하는 방법이 Bearer Authentication이고 이 방법은 OAuth를 위해 고안된 방법이다. 인증 메커니즘 중에서 상대적으로 가장 간단한 방법이고 Bearer 인증은 Access token via HTTP Header 방식을 따른다. Bearer 인증을 통해 IoTMakers의 태그스트림으로부터 JSON 포맷의 데이터를 받아온다.

받아온 JSON 데이터에서 필요한 정보를 추출하기 위하여 파싱하여 시맨틱 데이터 표현 방식인 RDF로 변환하고 추론된 정보를 시맨틱 데이터로 가공하여 정보를 재생산하게 된다. RDF 변환과 추론은 온톨로지 API와 추론 API를 제공하는 Apache Jena 시맨틱 웹 프레임워크[16]를 사용한다. 또한, W3C의 RDF 규격에 따라 데이터 모델이 올바르게 생성되었는지 확인하기 위하여 W3C의 RDF Validator[17]를 사용해 변환된 RDF를 검증한다. W3C에서 지정한 RDF 표준 규격은 N-Triples, N-Quads, RDF/XML, Turtle 등이 있지만 본 연구에서 사용한 규격은 RDF/XML이다.

3.2 IoT 센서 온톨로지 모델링

복잡한 온톨로지를 구축하기 위해서는 툴 사용이 필수적이며 현재 가장 널리 쓰이는 OWL 저작툴은 Protégé[18]다. 온톨로지를 구축할 때는 기존의 온톨로지를 매핑(Mapping)하여 사용하는 것이 좋다. 매핑이란 필요한 데이터가 기존의 온톨로지에 구축되어 있다면 이를 재활용하여 사용하는 것이다. 본 연구에서는 H2020의 Fiesta IoT 프로젝트[19]의 Fiesta IoT Ontology를 매핑하고 수정하였다. Fiest IoT Ontology는 LOV4IoT 프로젝트[20]를 기반으로 기초적인 테스트베드 자원의 설명과 센서 등에서 수집된 데이터의 측정에 초점을 맞추고 있다. 이와 비교하여 본 연구에서 모델링하고 구축한 온톨로지는 IoT 디바이스의 센서 등에서 수집된 데이터의 측정에 초점을 맞추었다. 그리고 수집된 데이터를 통해 발생하는 이벤트를 파악할 수 있는 이벤트 정보를 모델링하였다. 이를 통해 온톨로지를 기반으로 하여 실제로 시스템이 서비스 되었을 때 해당 시스템과 IoT 디바이스에서 발생하는 이벤트 정보를 파악하기 쉽고 해당 상황을 해결하는데 용이하다.

클래스 속성 정의는 그림 1(a)와 같이 Context, Deployment, DomainOfInterest, Point, Observation, ObservationValue, Platform, QuantityKind, Sensor, SensorOuput, Service, System, TimeInterval, Unit로 정의하였고, 속성 정의는 그림 1(b)와 같이 ssn 온톨로지에서 hasDeployment, hasSubSystem, hasValue, hasQuantityKind, onPlatform, madeObservation, observationResult, observationSamplingTime, observerdBy, iot-lite 온톨로지에서 hasUnit, geo 온톨로지에서 hasLocation을 매핑한다. 그리고 hasContext를 추가한다.



(a) Class property (b) Object property

그림 1. 클래스/객체 속성 계층

Fig. 1. Class/object property hierarchy

이렇게 만들어진 온톨로지를 기반으로 Apache Jena 프레임워크에서 제공하는 온톨로지 API인 OntModel[21]을 사용하여 온톨로지 모델을 만들었다. Jena는 OWL을 잘 지원할 뿐 아니라 OWL Reasoner를 제공하고 있어 OWL 온톨로지 구축 및 추론에 많이 사용되고 있다. 온톨로지 추론을 위해서 제공하는 Jena 룰은 온톨로지를 사용하여 서비스를 기술하는데 적합하다. 그림 2는 Protégé의 OntoGraf 플러그인을 사용하여 데이터 모델링을 기반으로 구축된 온톨로지를 시각적인 그래프로 표현한 것이다.

IV. IoT 매쉬업 시스템 설계 및 구현

4.1 IoT 매쉬업 시스템 설계

매쉬업 시스템과 트리플 저장소는 Amazon의 클라우드 서비스인 AWS 환경[22]에서 서비스된다. AWS로 시스템 환경을 구성한 이유는 매쉬업 시스템과 트리플 저장소를 위한 물리적 서버를 구성할 필요가 없고 추후에 시스템을 확장할 때 보다 쉽게 확장할 수 있는 이점이 있기 때문이다. 디바이스로부터 데이터를 수집하거나 SPARQL 엔드포인트로 데이터를 공유받기 위한 접근이 이루어졌을 때 발생할 수 있는 보안 측면의 이슈를 방지하기 위해 네트워크 ACL(Access Control List)과 보안그룹의 사용 또한 물리적 서버에서 보안 체계를 구성하여 관리하는 것보다 뛰어나기 때문에 시스템이 데이터를 수집, 관리하여 공유하기에 편리성과 안정성을 부여해 준다. 또한 Amazon VPC(Virtual Private Cloud)를 사용하여 Amazon EC2(Elastic Compute Cloud) 인스턴스들을 독립된 네트워크 환경에서 작동하게 만들어 준다. VPC를 사용하지 않으면 하나의 인스턴스만 추가되도 모든 인스턴스의 네트워크 환경을 재구성 해주어야 하고 각각의 인스턴스들이 복잡하게 연결되어 시스템의 복잡도를 증가시킨다. 이러한 이유에서 VPC를 구성해 주었고, 인터넷을 연결해주는 역할을 하기 위해 NAT 게이트웨이(NAT gateway)와 인터넷 게이트웨이를 구성하였다. 이러한 시스템의 네트워크 구성은 그림 3과 같다.

매쉬업 서버는 EC2에서 리눅스를 기반으로 Apache Jena 프레임워크를 사용하고, 시스템에서 사용되는 트리플 저장소는 Amazon Neptune을 사용하였다.

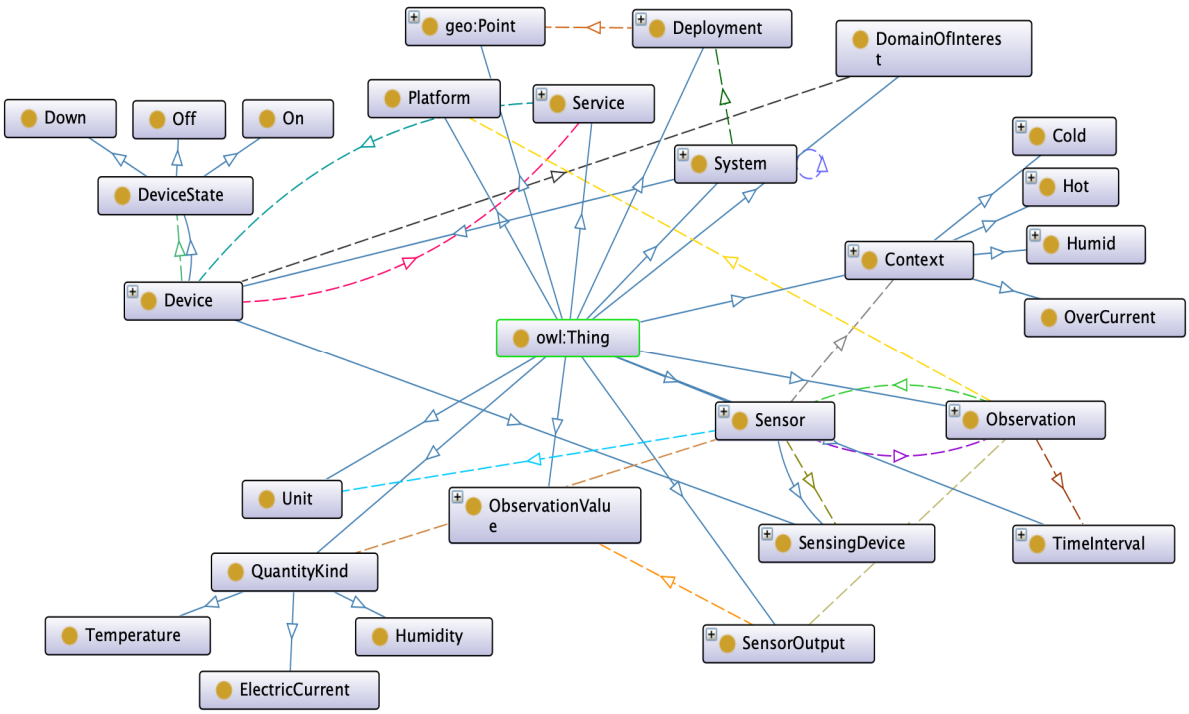


그림 2. IoT 온톨로지 모델
Fig. 2. IoT ontology model

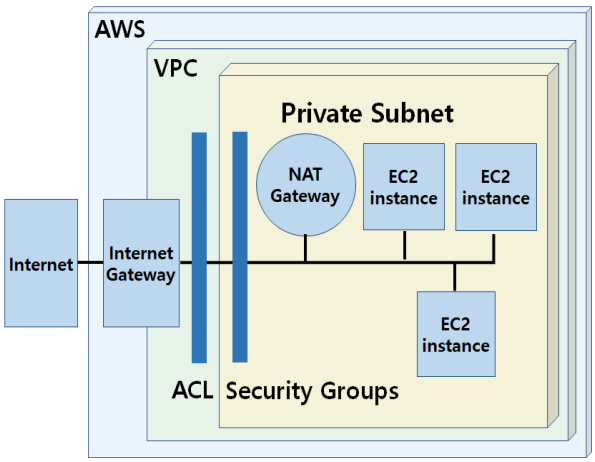


그림 3. 네트워크 구성도
Fig. 3. Network diagram

IoT 매쉬업 시스템 구성도는 그림 4와 같다. IoT 디바이스로부터 수집된 데이터는 AWS에서 구성한 인터넷 게이트웨이를 거쳐 매쉬업 서버로 전달된다. IoTMakers에서 수집된 디바이스 데이터를 얻기 위해 매쉬업 서버의 Gathering Module에서는 REST 방식의 OpenAPI를 통해 IoTMakres로 접근하기 위해 일련의 인증 과정을 거쳐 인증 토큰을 얻고 데이터를 얻기 위한 권한을 획득한다. Gathering Module을 통해 수집한 JSON 포맷의 IoT 센서 데이터를 기반으로 Parsing Module에서는 시맨틱 데이터로 가공하기 위한 데이터를 추출하기 위해 파싱을 수행한다. 파싱된 데이터는 Converting Module에서 시맨틱 데이터 표현방식인 RDF 표현 방식으로 변환된다. 이렇게 변환된 RDF 트리플은 Annotating Module에서 시맨틱 어노테이션을 추가하여 OWL 온톨로지로 구축되고 RDF 트리플은 트리플 저장소에 저장된다. 매쉬업 서버의 모듈을 거쳐 RDF 트리플로 만들어진 데이터는 첫 번째로 Amazon S3(Simple Storage Service)에 저장된다. S3는 Amazon에서 제공하는 스토리지 서비스로 물리적인 저장 공간을 확보하지 않더라도 IoT 데이터를 저장할 수 있도록 해준다. 또한 트리플 저장소인 Neptune에 RDF 트리플을 저장하기 전에 S3에 RDF 트리플을 저장함으로써 데이터의 손실을 미연에 방지해준다.

그림 5는 매쉬업 서버에서 데이터가 처리되는 과정을 좀 더 세부적으로 나타낸 순서도이다. OpenAPI를 사용하여 매쉬업 서버의 데이터를 수집하기 위해서는 두번의 인증 과정을 거친다. 첫 번째 인증 과정은 IoTMakers의 OpenAPI를 사용하기 위한 권한을 획득하는 Basic 인증 과정이다. 사용자의 아이디와 패스워드 그리고 인증 토큰이 정상적인지를 확인하는 Check Credentials를 거쳐 인증 토큰이 정상적이지 않으면 다시 Basic 인증을 수행한다.

정상적으로 인증 토큰을 얻어 권한을 획득하면 두 번째 인증 과정으로 IoTMakers의 데이터를 받기 위한 Bearer 인증 과정을 거친다. Basic 인증 과정에서 획득한 인증 토큰과 OpenAPI를 사용해 필요한 데이터를 획득하고, Check Credentials를 통해 획득한 인증 토큰과 URL이 정상적인지를 판단한다. 이러한 인증 과정을 거친 후 JSON 포맷을 가진 센서, 디바이스, 이벤트 데이터를 수집한다.

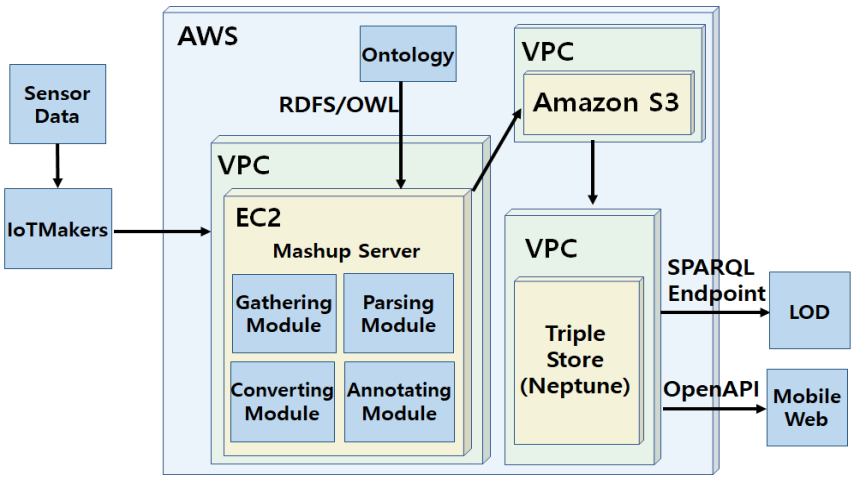


그림 4. 시스템 구성도
Fig. 4. System diagram

수집된 데이터는 Parsing Module의 룰에 의해 파싱되고 {키, 값} 형태로 데이터를 다루는 HashMap으로 저장된다. 이는 RDF 변환 과정을 보다 쉽게 수행하기 위함이다. 또한 데이터 파싱을 위해서 자바 JSON 라이브러리를 사용하였다. JSON 라이브러리에는 JSON.simple, GSON, Jackson, JSONP 등의 다양한 종류의 라이브러리가 있지만 본 연구에서 사용한 jackson은 JAX-RS이다. JAX-RS는 RESTful 웹 서비스를 위한 자바 API로 자바 SE5에 소개된 어노테이션을 사용하여 정의된 사양이다. 대표적인 어노테이션으로는 Path, GET, POST, PUT 등이 있다. 또한 jackson은 JSON 뿐만 아니라 XML, YAML, CSV 등 다양한 형태의 데이터를 지원하는 데이터 처리 툴이다.

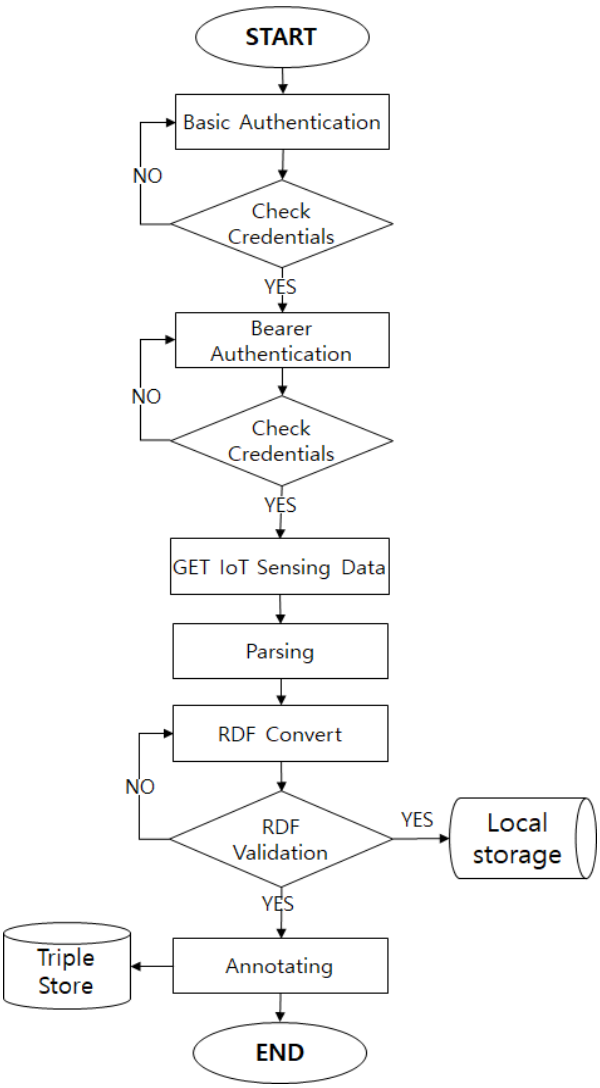


그림 5. 시스템 처리 순서도
Fig. 5. System processing flowchart

4.2 IoT 매쉬업 시스템 구현

그림 6은 매쉬업 서버에 등록되어 있는 IoT 디바이스의 위치를 Daum Map API를 통해 한 눈에 파악할 수 있는 실시간 디바이스 위치 화면이다. 디바이스 위치 검색을 위한 알고리즘과 SPARQL 쿼리는 그림 7과 같다.

그림 6. 실시간 디바이스 위치 화면
Fig. 6. Real-time device location screen

```
IF SEARCHING DEVICE COMMAND
   THEN
      CREATE A DEFAULT MAP
      FINDING POINT(LAT, LONG) AND GENERATE MARKER
   ELSE
      DEVICE SEARCHING FAILURE MESSAGE
```

(a) Retrieval algorithm

```
SELECT DISTINCT ?id ?lat ?long
WHERE {
    ?n iot-lite:id ?id .
    ?n ssn:hasSubSystem ?sensingDevice .
    ?sensingDevice ssn:hasDeployment
    ?deployment
    ?deployment iot-lite:hasPoint ?point
    ?point geo:lat ?lat
    ?point geo:long ?long
}
```

(b) SPARQL query

그림 7. 위치 검색 알고리즘
Fig. 7. Location-based retrieval algorithm

매쉬업 서버에서 수집한 데이터를 추론한 상황 정보는 그림 8과 같이 웹에서 확인할 수 있다. 이는 차트 표현을 위한 Open API인 HighCharts[23]를 이용하여 구현하였다. 반원과 막대 차트에서 상황 정보(context)의 수를 확인할 수 있고, 상황 정보는 Hot, Cold, Humid, Over, Good으로 구성되어 있다. 그리고 scatter, bubble 차트에서 수집된 데이터의 Temperature, Humidity, Current 분포를 쉽게 파악할 수 있다.

4.3 트리플 저장소

본 연구에서 사용한 트리플 저장소는 AWS 환경에서 동작하는 상업용 트리플 저장소인 Neptune이다. Amazon Neptune은 RDF 데이터의 업로드와 수정을 지원하며 클라우드 환경에서 빠르고 안정적인 그래프 데이터베이스를 제공한다. 스키마에 자유로운 환경을 제공하며 RDF 1.1, SPARQL 1.1, TinkerPop Gremlin 3.3을 지원한다. 또한 자바뿐만 아니라 C#, Python, JavaScript, PHP 등의 다양한 언어를 지원한다. Neptune에 접근하기 위해선 “https:// neptune-endpoint:port/sparql”과 같은 형태를 가지는 엔드포인트 URL이 필요하다. 인스턴스의 엔드포인트를 사용하여 DB 클러스터의 특정 DB 인스턴스에 연결하는 것보다는 클러스터 엔드포인트나 리더 엔드포인트를 DB 클러스터에 사용하는 것이 좋다. curl명령을 통해 SPARQL 질의를 수행할 수도 있지만 Neptune에서는 URI로 UPDATE LOAD가 지원되지 않아 AWS S3를 사용하여 데이터 파일을 업로드한다.

그림 8. 실시간 상황 정보
Fig. 8. Real-time situation information

S3에 업로드하기 위해 IAM(Identity and Access Management)에 S3에 대한 권한을 추가하고 AWS CLI(Command Line Interface) 명령 “aws s3 cp data-file-name s3://bucket-name/object-key-name”을 통해 파일을 업로드 한다. 소량의 RDF 트리플은 curl에 직접 SPARQL 질의를 해서 Neptune에 저장해도 되지만 대용량 데이터를 저장하기에는 비효율적이다. 따라서 대용량의 RDF 트리플을 저장하기 위해서 S3에 업로드된 RDF 트리플을 Neptune Loader를 이용하여 Neptune에 저장하였다.

V. 시스템 성능 분석

시맨틱 기반 IoT 매쉬업 플랫폼은 현재 연구가 활발히 진행되고 있는 분야이므로 구체적인 개발이 완료된 시스템은 아직 찾아보기 힘든 상황이다. 따라서 규모나 개발 범위가 다른 시스템간 성능 분석은 큰 의미가 없으므로 표 1과 같이 본 논문에서 구현된 시스템을 기능이 유사한 시스템들과의 기능 스펙을 비교하였다. 시맨틱 아키텍처, 위치 기반 검색, 개방형 플랫폼 사용 유무, 서비스 환경(PC 또는 모바일), 그리고 클라우드 환경에서 시스템이 서비스 되는지를 비교하였다.

표 1에서 Barnaghi[24]는 LOD 기반 시맨틱 센서 플랫폼(즉, Sense2Web)으로서 Jena API를 사용하여 LOD에 쿼리하여 센서 데이터와 위치 정보를 제공받는다. 웹에서 사용자가 키워드를 입력하고 관련 데이터를 제공 받을 수 있도록 하고 AJAX를 사용하여 결과를 웹 화면에 표시한다. Google Maps API를 사용하여 위치 기반 검색 기능을 제공하지만 개방형 플랫폼은 사용하지 않는다. Heo[25]는 Oliot[26] 플랫폼을 사용하여 IoT 매쉬업 애플리케이션 플랫폼을 설계하고 구현하였다. 모듈형 소프트웨어와 라이브러리 개발 및 배포를 위한 자바 프레임워크 OSGI(Open Service Gateway Initiative)를 사용했다. Android 환경에서만 서비스되며 위치 기반 검색 기능은 제공하지 않는다. Patni[27]는 2만여개의 기상 센서로부터 얻은 데이터로, 센서 데이터셋을 대용량의 RDF 트리플로 변환하고 Virtuoso[28]에 저장하며 온톨로지를 구축한다. 시스템은 지도에서 위치를 선택하거나 GeoNames[29]에서 위치에 대한 공간 정보, 시간 정보, 주제별 정보(온도, 바람 세기 등)을 검색하여 데이터에 대한 위치 기반 검색과 접근을 제공한다. 하지만 개방형 플랫폼은 사용하지 않는다.

Zhang[30]은 시맨틱 센서 데이터를 위한 아키텍처를 제안하고 Java GUI로 개발하였다. 주로 SSN을 기반으로 하는 매핑을 나타내는 방법에 초점을 맞추었으며 SASML을 기반으로 XML 파일에 수동으로 어노테이션을 달고 SDRM 매핑 알고리즘을 통해 자동으로 SSN 온톨로지 인스턴스로 변환한다. 하지만 위치 기반 검색과 개방형 플랫폼을 사용하지 않았고, 웹 브라우저 환경을 제공하지 않는다. Ryu[31]는 웹에서 실행되고 HTML과 jQuery를 사용한다. 비동기 자바스크립트와 XML(Ajax)을 사용하여 입력된 값을 ISSS에 연결하고 전송한다. 온톨로지 모델이 서비스 도메인 내의 지식을 공식적/명시적으로 표현하기 위해 서비스 도메인에 대한 온톨로지를 만들 수 있는 웹 기반 저작 도구를 개발했다. 개방형 플랫폼을 사용하고 위치 정보를 수집하지만 위치 기반 검색 기능은 제공하지 않는다.

표 1. 시스템 간 기능 비교
Table 1. Function comparison with other systems

	Proposed system	Barnaghi [24]	Heo [25]	Patni [27]	Zhang [30]	Ryu[31]
Semantic architecture	Yes	Yes	No	Yes	Yes	Yes
Location based search	Yes	Yes	Yes	Yes	No	No
Open platform	KT IoTMakers	No	Oliot	No	No	Mobius
Service environment	Web (PC, Mobile)	Web (PC)	Android	Web (PC)	PC	Web (PC, Mobile)
Cloud environment	Yes	No	No	No	No	No

VI. 결 론

본 논문에서는 시맨틱 기반 IoT 매쉬업 플랫폼을 위한 처리 기술들을 개발하고, IoT 센서 온톨로지를 설계하고, IoT 매쉬업 시스템을 구현하였다. 본 시스템에서는 IoTMakers에서 OpenAPI 인증 과정을 거쳐 수집된 데이터를 매쉬업 서버에서 시맨틱 데이터로 만들어 트리플 저장소의 SPARQL 엔드포인트로 질의할 수 있으며 링크드 데이터로 활용할 수 있다. 또한 IoT 매쉬업 시스템은 가공된 데이터를 다양한 환경에서 시맨틱 기반 IoT 정보 활용 및 시맨틱 데이터를 공유할 수 있는 환경을 제공한다. PC와 모바일 환경에서 Daum Map API를 이용하여 위치 기반 검색을 통해 실시간으로 센서의 데이터를 확인할 수 있으며 HighCharts API를 사용해 차트를 통한 상황 정보의 종류와 분포를 쉽게 파악할 수 있다. 그리고 디바이스 아이디를 통한 검색 기능을 이용하여 리스트로 센서에서 수집한 데이터 및 상황 정보를 한 눈에 파악할 수 있다.

시스템 성능 분석에서 비교한 시스템들 중 유일하게 클라우드 컴퓨팅을 사용하여 AWS 환경에서 서비스되었고 트리플 저장소인 Neptune 또한 AWS에서 서비스되었다. 클라우드 컴퓨팅을 통해 리소스에 대한 가변 비용을 줄일 수 있었으며 시스템의 확장성과 시스템을 서비스하기 좋은 인프라로 구축할 수 있었다. 향후 연구에서는 본 논문에서 제안한 시스템의 데이터 처리 기능 및 시맨틱 어노테이션의 자동화를 통해 좀 더 고도화된 시스템을 구현할 필요가 있다.

References

[1] S. H. Lee, "An Analysis on the Changes of Logistics Industry using Internet of Things", JAITC, Vol. 9, No. 1, pp. 57-66, Jul. 2019.

[2] Y. J. Lee, H. S. Seok, S. H. Nam, and Y. E. Lee, "Semantic-based IoT Platform Architecture", Korean Computer Congress 2019, pp. 328-330, Jun. 2019.

[3] H. Seok and Y. Lee, "Ontology-based IoT Context Information Modeling and Semantic-based IoT Mashup Services Implementation", Journal of the KIECS, Vol. 14, No. 4, pp. 671-678, Aug. 2019.

[4] C. Pyo, "M2M Technology and Its Standardization Trends", oneM2M 2013 Seoul International Conference, Tutorial Session, Jun. 2013.

[5] L. Mainetti, V. Mighali, and L. Patrono, "A Software Architecture Enabling the Web of Things", IEEE Internet of Things Journal, Vol. 2, No. 6, pp. 445-454, Dec. 2015.

[6] M. R. Kim, J. W. Lee, Y. J. Lee, and J. C. Ryou, "COSMOS: A Middleware for Integrated Data Processing over Heterogeneous Sensor Networks", ETRI Journal, Vol. 30, No. 5, pp. 696-706. Oct. 2008.

[7] EBBITS Project, http://cordis.europa.eu/project/id/257852 [accessed: Aug. 01. 2019]

[8] OpenIoT Project, "Open Source Blueprint for Large Scale Self-organizing Cloud Environments for IoT applications", http://cordis.europa.eu/project/id/287305 [accessed: Apr. 21. 2017]

[9] Web Of Object, http://itea3.org/project/web-of-objects.html [accessed: Feb. 06. 2015]

[10] DIYSE Project, http://itea3.org/project/diy-smart-experiences.html [accessed: May 01. 2012]

[11] DPWS, http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01 [accessed: Jul. 01. 2009]

[12] SOCRADES Project, http://www.socrades.net [accessed: Dec. 17. 2014]

[13] D. H. Woo, M. K. Yoo, and Y. H. Kim, "A Study on Ontology for Semantic-Based Service Exploiting the Context Information in IoT Environment", The Journal of Society for e-Business Studies (Society for e-Business Studies), Vol. 21, No. 3, pp. 1-13, Jan. 2016.

[14] http://iotmakers.kt.com [accessed: May. 15. 2019]

[15] http://www.ietf.org/rfc/rfc7617.txt [accessed: Sep. 01. 2015]

[16] http://jena.apache.org [accessed: May. 15. 2019]

[17] http://www.w3.org/RDF/Validator [accessed: Feb. 28. 2006]

[18] http://protege.stanford.edu [accessed: May 15. 2019]
[19] Fiest-IoT Ontology, http://fiesta-iot.eu [accessed: Jul. 143. 2018]
[20] LOV4IoT, http://lov4iot.appspot.com [accessed: May. 01. 2019]
[21] http://jena.apache.org/documentation/javadoc/jena/org/apache/jena/ontology/OntModel.html [accessed: Apr. 21. 2019]
[22] http://aws.amazon.com/ [accessed: Apr. 21. 2019]
[23] http://www.highcharts.com [accessed: Apr. 21. 2019]
[24] P. Barnaghia, F. Ganza, and H. Abangara, "Sense2Web: A Linked Data Platform for Semantic Sensor Networks", Semantic Web - Interoperability, Usability, Applicability an IOS Press Journal, Vol. 2, No. 1, pp. 1-11, 2011.
[25] S. Heo, S. Woo, J. Im, and D. Kim, "IoT-MAP: IoT Mashup Application Platform for the Flexible IoT Ecosystem", 5th International Conference on the Internet of Things, pp. 163-170, Oct. 2015.
[26] http://www.oliot.org [accessed: Apr. 21. 2019]
[27] H. K. Patni, S. S. Sahoo, C. A. Henson, and A. P. Sheth, "Provenance Aware Linked Sensor Data", http://corescholar.libraries.wright.edu/knoesis/558 [accessed: Apr. 21. 2019]
[28] http://virtuoso.openlinksw.com. [accessed: Apr. 21. 2019]
[29] http://www.geonames.org. [accessed: Apr. 21. 2019]
[30] X. Zhang, Y. Zhao, and Y. Liu, "A Method for Mapping Sensor Data to SNN Ontology", International Journal of u- and e-Service, Science and Technology, Vol. 8, No. 6, pp. 303-316, Aug. 2015.
[31] M. Ryu, J. Kim, and J. Yun, "Integrated Semantics Service Platform for the Internet of Things: a Case Study of a Smart Office", Sensors (Basel, Switzerland), Vol. 15, No. 1, pp. 2137-2160, Jan. 2015.

저자소개

남 성 현 (Seonghyeon Nam)

2017년 : 안동대학교 컴퓨터공학전공(학사)
2020년 : 경북대학교 IT대학 컴퓨터학부(공학석사)
관심분야 : 시맨틱 웹, 빅데이터, 웹 데이터베이스

이 용 주 (Yongju Lee)

1985년 : 한국과학기술원 정보검색전공(공학석사)
1997년 : 한국과학기술원 컴퓨터공학전공(공학박사)
1998년 8월 ~ 현재 : 경북대학교 IT대학 컴퓨터학부 교수
관심분야 : 링크드 데이터, 시맨틱 웹, 빅데이터, 웹 사이언스