碩士學位論文

CoAP기반의 IoT 노드 데이터 정제 방법

濟州大學校 大學院

컴퓨터 工學科

汪 建

2015年6月

CoAP기반의 IoT 노드 데이터 정제 방법

指導敎授 郭 浩 勇

汪 建

이 論文을 컴퓨터 工學 碩士學位 論文으로 提出함

2015 年 6月

汪建의 工學 碩士學位 論文을 認准함

審査委員長 _____㊞

委　　員 _____㊞

委　　員 _____㊞

濟州大學校 大學院

2015年 6月

A Method of Data Refinement between IoT Nodes based on CoAP

Jian Wang
(Supervised by professor Ho-Young Kwak)

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Science

2015 . 06 .

This thesis has been examined and approved.

Thesis director,.....
Thesis director,.....
Thesis director,.....

July 2015

Department of Computer Engineering

GRADUATE SCHOOL

CHEJU NATIONAL UNIVERSITY

감사의 글

2013년 제주대 대학원 생활을 시작하여 어느덧 석사 학위과정을 마치고 학위를 받게 되었습니다. 2년 동안 열심히 했던 대학원 생활을 뒤돌아보니 저에게 도움을 주셨던 모든 분들이 마음 속을 스쳐 갑니다.

먼전 제가 2년간 대학원 생활을 함에 있어 아낌없이 가르침과 지도를 해 주신 지도교수님 이신 곽호영 교수님께 갑사 합니다. 또는 논문 심사에 애써주신 이상준 교수님과 변영철 교수님에게도 감사 합니다. 항상 대학원 생활에 세심한 지도와 가르침을 아끼지 않아주셨던 사공준 교수님, 김도현 교수님, 이상준 교수님, 변상용 교수님, 송왕철 교수님, 곽호영 교수님, 권훈 박사님께 감사드립니다. 그리고 대학원에 쉽게 적응할 있도록 도움을 주신 학과사무실 문숙희 선생님, 오은희 선생님 감사드립니다.

대학원에 같이 입학하여 동고동학한 선배 진남, 선배 진서, 친구 김문권 항상 도움을 주어 감사드립니다. 제주대학교 연구실의 최강, 김연주, 이세중 함께한 시간이 많지는 않았지만 즐거웠고 모두 감사합니다. 특히 중국에 계신 할머니(黄朝英), 어머님(陈学玲), 아버지(汪帮有), 아내(吳明珠)께서 원만하게 유학생활을 할 수 있도록 지지하여 주신 것에 정말 감사합니다. 그리고 제주대학교 정보통신학과의 가장 친한 친국 우학겸, 대학원 생활 동안 도움을 주어 즐거웠고 고맙습니다. 모두들 모든 일이 소망대로 되시기를 바라며 건강하고 하는 일마다 축복이 있기를 기원합니다.

끝으로 부족한 저에게 언제나 힘이 되어주고 조언을 해준 친구 주학, 륙명성, 두명양, 왕륭, 김진선 에게 감사함을 전합니다.

목차

그림 목차

표 목차

국문 초록

CoAP기반의 IoT 노드 데이터 정제 방법

컴퓨터공학과: 왕 건

지도 교수: 곽호영

최근 휴대용 스마트 단말이 널리 사용되고 있다. 더불어 인터넷 기술의 발달로 M2M 네트워크는 IoT의 핵심 분야가 되었으며, 도시 그리고 국가규모로 확장되고 있다[10]. 이에 따라 각종 센서를 연결하기 위한 통신 프로토콜이 다양하게 필요하다. IETF 의 CoAP 프로토콜은 작은 용량의 메모리와 저전력 등 제한된 환경에서 센서나 구동체 노드 간의 통신을 지원한다[6]. 본 논문에서는 스마트 폰과 IoT 노드 사이에 CoAP 프로토콜을 기반으로 인터넷을 연결하고 스마트 폰 상에 데이터 정제를 위한 미들웨어를 설계하고 구현한다. 스마트 폰 측 미들웨어를 CoAP 클라이언트에 구현하였으며 중복데이터와 노이지 데이터를 제거하는 기능을 포함 한다

Abstract

A Method of Data Refinement between IoT Nodes based on CoAP

WANG JIAN

Department of Computer Engineering

Graduate School

Jeju National University

Recently, it is high use of the portable smart terminal. With the development of IoT, M2M Network System has been extended to the city and national scale. So, protocol for communication to connect with various sensors is required. IETF CoAP protocol supports communication between sensor actuator nodes in a constrained environment, such as small amount of memory, and low power. This paper studies Internet between smart-phone and IoT node based on CoAP, the design and implementation of the middleware with smart phone. Smart phone middleware with CoAPClient has the function to eliminate repeated data and outlier data with service repository

Key Words: M2M, IoT, Middleware, Repetition Data, Outlier Data, CoAP Protocol, Refinement.

약어 표

IoT	Internet of Things
REST	Representational State Transfer
URI	Uniform Resource Identifier
CoAP	Constrained Application Protocol
HTTP	Hyper Text Transfer Protocol
API	Application Programming Interface
M2M	Machine to Machine
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
AIDL	Android Interface Description Language
IP	Internet Protocol
UI	User Interface
IETF	Internet Engineering Task Force
RFC	Request for Comments

Ⅰ. 서론

1. 연구 배경

사물인터넷(IoT: Internet of Things)은 지능화된 사물들이 네트워크를 통해 연결되어 사물과 사물, 사물과 사람 간에 상호소통하고 상황인식 기반의 지식이 결합되어 지능적인 서비스를 제공하는 글로벌 인프라이다[15]. 이러한 IoT 에서 "Thing"은 향상된 컴퓨팅 능력과 상황인식능력, 독자적인 정체성(identity)을 지니게 될 것 이며, 정보 네트워크에 항상 접근이 가능하도록 연결될 것이다.

21 세기 "ubiquitous 세계"에 대해 많은 국가 및 관련 국제기구가 점점 관심을 갖게 되고 유비쿼터스 네트워크 응용 프로그램과 서비스는 많은 분야에서 자동화를 향상 시키고 혁명적인 변화를 가져오고 있다. 예를 들어 물류, 환경 보호, 가정 네트워크, 의료보건, 지능빌딩 등의 분야에서 자동화 및 정보화가 진행되고 있다. 한국은 다양한 사회적인 USN 응용을 기반으로 각 산업 및 전체 도시의 정보화 수준을 향상 시키는 USN 프로젝트를 가동하고 있고, 많은 도시에서 U-City를 발전시키고 있다. U-City 서비스는 U-Home(U가정), U-ITS(U지능형 교통), U-FMS(U종합시설물 관리시스템), U-Monitoring(U감시)을 포함한다[8].

최근 스마트 폰/패드 등 휴대용 스마트 단말의 높은 보급과 더불어 스마트 폰에 내장된 근거리 통신기술(Bluetooth, WiFi, WiFi-Direct, NFC 등)을 활용한 다양한 IoT 제품 및 서비스가 시장에 선보이고 있으나[4], 각 장치들이 제공하는 서비스를 사용자가 직접 검색하고 해당 응용 프로그램을 개별적으로 사용하는 것은 쉽지 않기 때문에, IoT 장치에 대한 개별적인 조작 없이 통일된 플랫폼을 통해 장치 및 서비스를 발견하고 제공하는 기술이 필요하다[2].

사물인터넷 시장은 2014년도를 기준으로 2020년도에는 국내는 4.7배, 국외는 3.3배 성장할 것으로 예측되고 있으며, 인터넷에 연결된 개체 수 또한 500억 개에 이를 것으로 예상된다[16]. 포스트 스마트 폰 시대를 이끌어 가기위한 IT기업들의 스마트 단말의 개발이 활발 해짐에 따라 스마트 단말기에 대한 새로운 활용성 증대의 필요성 이 대두되고 있다. IETF 의 CoAP 프로토콜은 작은 용량의 메모리와 저전력 등 제한된 환경에서 센서나 구동체 노드 간의 통신을 지원한다. 그리고 외부환경 때문에 실제 데이터에는 문제가 항상 있다[9].

Name	Programming Language	Client/Server	Implemented CoAP features	License
libcoap	C	Client + Server	Observe, Blockwise Transfers	BSD/GPL
iCoAP	Objective-C	Client	Observe, Blockwise Transfers	MIT
nCoap	Java	Client + Server	Observe (draft 06), Blockwise Transfers (draft 04, partially)	BSD
jcoap	Java	Client + Server		Apache
coapy	Python	Client + Server		BSD
txThings	Python (Twisted)	Client + Server	Blockwise Transfers, Observe (partial)	MIT
TinyOS CoapBlip	nesC/C	Client + Server	Observe, Blockwise Transfers	BSD
RESTful Contiki	C	Server		3-clause BSD
Erbium for Contiki	C	Client + Server	Observe, Blockwise Transfers	3-clause BSD
Californium	Java	Client + Server	Observe, Blockwise Transfers, DTLS	3-clause BSD
Copper	JavaScript (Browser Plugin)	Client	Observe, Blockwise Transfers	3-clause BSD
JSCoAP	JavaScript (Library)	Client + Server	Observe	MIT License
CoAP implementation for TinyOS	nesC/C	??		??
CoAP implementation for Go	Go	Client + Server	Core + Draft Subscribe	MIT
Sensinode C Device Library	C	Client + Server	Core, Observe, Block, RD	Commercial
Sensinode Java Device Library	Java SE	Client + Server	Core, Observe, Block, RD	Commercial
Sensinode NanoService Platform	Java SE	Cloud Server	Core, Observe, Block, RD	Commercial
CoAPSharp	C#, .NET	Client + Server	Core, Observe, Block, RD	LGPL
cantcoap	C++/C	Client + Server		BSD
microcoap	C	Client + Server		MIT
node-coap	Javascript	Client + Server	Core, Observe	MIT
smcp	C	Client + Server	Core, Observe, Block	MIT

표 1. CoAP 개발현황

표 1은 CoAP 개발현황을 나타내고 있다. 현재 많은 CoAP 오픈소스 구현이 있다. Java프로그래밍 언어로 개발된 오픈소스는 Library-Californium, jCoAP 등 있다. C 프로그래밍 언어로 개발된 오픈소스는 Library-Erbium, libcoap 등이 있다. CoAP 은 이미 많은 상업용 상품/시스템에 도입되었다. Sensinode

NanoService 가 있다.

2. 연구목적 및 방법

지금 세계적으로 인터넷을 통하여 주요하게 진행되는 데이터교환은 TCP 및 HTTP 애플리케이션계층 프로토콜을 통해 구현된다. 작은 장치에 인터넷을 통하여 데이터 교환을 최적화하기 위하여 CoAP 프로토콜이 설계되었다. CoAP 은 UDP기반의 애플리케이션계층 프로토콜이다. CoAP 프로토콜 매우 작다. 최소패킷은 4 바이트이다. 참고문헌[19]에서도 밝힌 바와 같이, 현재 CoAP 기반의 스마트 단말기 와 IoT 노드 간 커뮤니케이션에 관한 연구가 적다.

본 논문에서는 CoAP 을 기반으로 스마트 폰과 IoT 노드 사이의 연결 구성을 설계하고, 스마트 폰 미들웨어를 설계 및 구현한다. 스마트 폰과 IoT 노드의 상호 연동을 실현하기 위해서 안드로이드와 리눅스 기반 의 CoAP 클라이언트 및 서버를 설계하고 구현하여 상호 호환성을 검증한다. 센서가 데이터를 수집할 때 외부환경 때문에 이상한 데이터를 수신할 경우가 있다. 이 문제를 해결하기 위해 Pauta Criterion을 이용한다. 중복된 센서 값이 생길 경우는 메타데이터 기술요소를 이용하여 제거한다. 마지막으로 데이터 분석 및 성능평가를 수행한다.

데이터 중복제거는 데이터의 세그먼트를 분할, 중복된 영역을 제거하여 유일한 고유 블록을 단 한번만 저장하는 기술 이다. 데이터 중복제거의 주요특징은 다음과 같다:

1) 처리 방식: 인라인, 오프라인.
2) 기록 방식: 역참조, 순참조.
3) 구성 방식: 파일 기반, 블록 기반.

데이터 중복제거를 위해 사용하는 기술요소는 다음과 같다:

1) 지문: 유일성, 진본성 여부를 확인한다.

2) 해쉬 알고리즘: SHA-1, MD5를 통해 비트레벨 비교를 수행한다.
3) 파이버 채널: Giga Bit 데이터 전송, SAN을 구성한다.
4) 데이터 압축: 엔트로피 무손실 압축을 적용한다.
5) 메타 데이터: 중복 확인 레포지터리 구성, Chuck 별 색인.

본 논문에서는 메타 데이터기술요소를 이용하여 중복된 센서데이터를 처리한다. 그리고 표준편차 값 과 센서 값을 비교하여 센서데이터가 이상이 없는지 확인한다.

3. 논문 구성

본 논문의 구성은 다음과 같다. 먼저 2장에서는 IoT프로토콜과 미들웨어를 서술한다. 3장에서는 CoAP 을 바탕으로 스마트 폰과 IoT 노드 사이의 연결 구성을 설계하고 4장에서는 스마트 폰 미들웨어 구현 및 시험결과를 보여주고. 마지막으로 5장에서는 결론과 향후 연구 계획에 대하여 설명한다.

Ⅱ. 관련연구

1. CoAP 프로토콜

IETF 워킹그룹에서 2010년 초부터 본격적으로 개발되기 시작한 CoAP 프로토콜은 여러 차례의 변화를 거쳐 최근 RFC 7252를 발표했다. 여기서는 RFC 7252 CoAP 프로토콜의 핵심적인 내용을 기술한다[10].

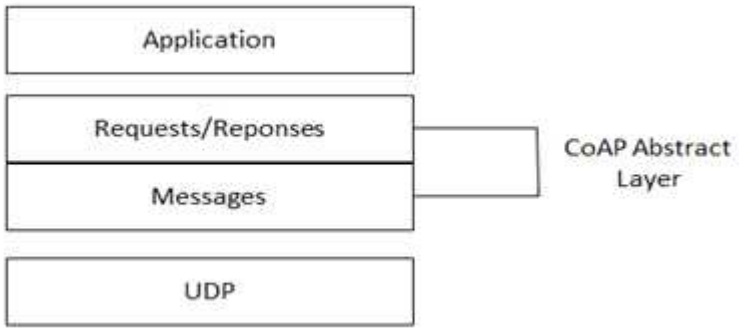


그림 1. CoAP 프로토콜 계층구조

그림 1은 CoAP 의 계층 구조를 나타내고 있다. CoAP 은 UDP를 기반으로 하며, RESTful API 구조의 데이터 모델까지 제공할 수 있는 특징을 가진다. RESTful API 란 HTTP를 통하여 웹 서버 및 데이터베이스로 구성된 서버에게 데이터 작업을 요청하고, 그 결과를 받는 것이다[14].

프로토콜 메시지 유형은 네 가지가 있다.

1) “CON”: 확인 요청을 필요로 한다, CON 요청이 전송되면, 상대방은 반드시 응답해야 한다.

2) “NON”: 확인 요청이 필요없다. NON 요청이 전송되면, 상대방은 응답할 필요

가 없다.

3) "ACK": 응답하는 메시지이고, CON에 대해 수신되는 메시지이다.

4) "RST": 복위 메시지이고, 수신자가 받은 메시지에 오류 결함이 있을 때, 수신자가 무시하면 그때 복위메시지를 보낸다.

RESTFul 프로토콜은 단순성 및 적용성 때문에, 웹 응용 프로그램으로 널리 사용되었다, CoAP은 RESTFul API 구조의 데이터 모델까지 제공하기 때문에, CoAP의 인기도 높아질 것으로 예상된다. CoAP 자원에 대해 URI 묘사를 할 수 있다. 예를 들어, 온도센서에서 온도를 측량하면, 이 온도 센서의 URI는 다음과 같이 묘사된다.

"CoAP://machine.address:5683/sensors/temperature"

CoAP의 기본 UDP 포트번호는 5683 이다.

2. 스마트 IoT 미들웨어 플랫폼(MoRI)

그림 2. MoRI의 개념도

MoRI는 USB, Bluetooth, NFC 등의 통신프로토콜을 사용하는 다양한 IoT 센서

및 구동기를 스마트 단말에 연결시켜 사용자의 다양한 질의처리 및 결과 보고 데이터
를 수집, 처리, 가공하여 사용자 응용과 COMUS[1]와 같은 시맨틱 USN 플랫폼에 제공하기 위해 설계된 미들웨어 시스템이다.

그림 2는 MoRI의 개념도 이다. MoRI는 스마트 단말의 앱 형태로 제공, 설치되어 USB, Bluetooth, NFC 프로토콜을 지원하는 IoT 센서 및 구동기가 스마트단말기에 접속될 때 해당 장치의 메타데이터(장치 식별자 및 처리가능정보)를 플랫폼에 등록하고, 각 장치의 생명주기를 관리한다. 이때 연결된 장치는 다양한 통신 인터페이스를 통해 가상화되어 사용자의 제어를 받게 된다. IoT 미들웨어 플랫폼의 주요 기능은 다음과 같다:

1) IoT 장치등록 및 질의/보고 인터페이스기능(ISO/IEC 30128표준)[3]
2) AIDL(Android Interface Description Language) 기반 응용 앱 인터페이스 기능
3) 센서 데이터 통합처리 및 데이터가공/변환 기능
4) 센서 데이터 및 구동기 질의 처리 엔진 기능
5) COMUS 플랫폼 연동 기능
6) IoT 자원관리 및 서비스 통계 기능
7) 사용자 응용 및 리소스 센서관리자 기능

본 논문에서 센서 데이터 통합처리 및 데이터가공/변환 기능을 참조하여 데이터를 정제한다.

3. 데이터 전처리 기술

전처리 함수 T를 이용하여 원시데이터벡터 Xik 가 새로운 데이터 벡터 Yij로 전환된다.

$$Y_{ij} = T(X_{ik})$$

i. Yij는 Xik 에 있는 "가치있는 정보"를 표현한다.

ii. Yij 는 Xik 의 문제를 해결한다.

iii. Yij 가 Xik 보다 더 유용하다.

i=1, ..., n n =객체의 수.

j=1,..., m m = 전처리 후 기능의 수.

k=1,...,l l = 속성수/전처리 전기능의 수.

일반적으로, m != 1.

데이터 전처리가 필요한 이유는 다음과 같다[7]:

1) 데이터 유형의 분석 수행을 방해 할 수 있는 문제 데이터를 해결한다.

2) 데이터의 특성을 이해하고 더 의미 있는 데이터 분석을 수행한다.

3) 주어진 데이터 중에서 의미있는 지식을 추출한다.

대부분의 애플리케이션에서, 하나 이상의 유형에 대한 데이터 전처리 작업이 필요하다.

3.1 손상 및 노이즈 데이터

실제 데이터에는 문제가 항상 있다. 그림 3에는 문제가 있는 데이터를 처리하는 과정을 나타낸다. 인간이 통제 할 수 없는 경우도 있다.

손상된 데이터는 예를 들어, 센서 고장, 데이터 전송 실패, 부적절한 데이터 입력 등이 있다. 노이즈데이터가 발생하는 몇 가지 원인은 다음과 같다:

1) 데이터 측정과 전송 오류

2) 내재원인. 예를 들어, 데이터가 수집되는 프로세스 나 시스템의 특성.

데이터의 손상 및 노이즈를 제대로 식별하고 적절한 해결방안을 찾아야 된다.

3.2 특징 추출

복잡한 온라인 데이터 분석 애플리케이션에서, 예를 들어, 화학 프로세스, 펄프 및 제지 응용 프로그램 등에서 수백 번의 측량이 있을 수 있지만, 상대적으로 적은 이벤트가 발생 될 수 있다. 적절한 데이터 전처리 과정이 필요하다. 적절한 해석을 통해 데이터를 전처리하여 조건에 맞는 입력 데이터로 쉽게 후속 특징 추출 및 정확도를 가능하게 하는 것이 양식의 특징 추출이다 [20].

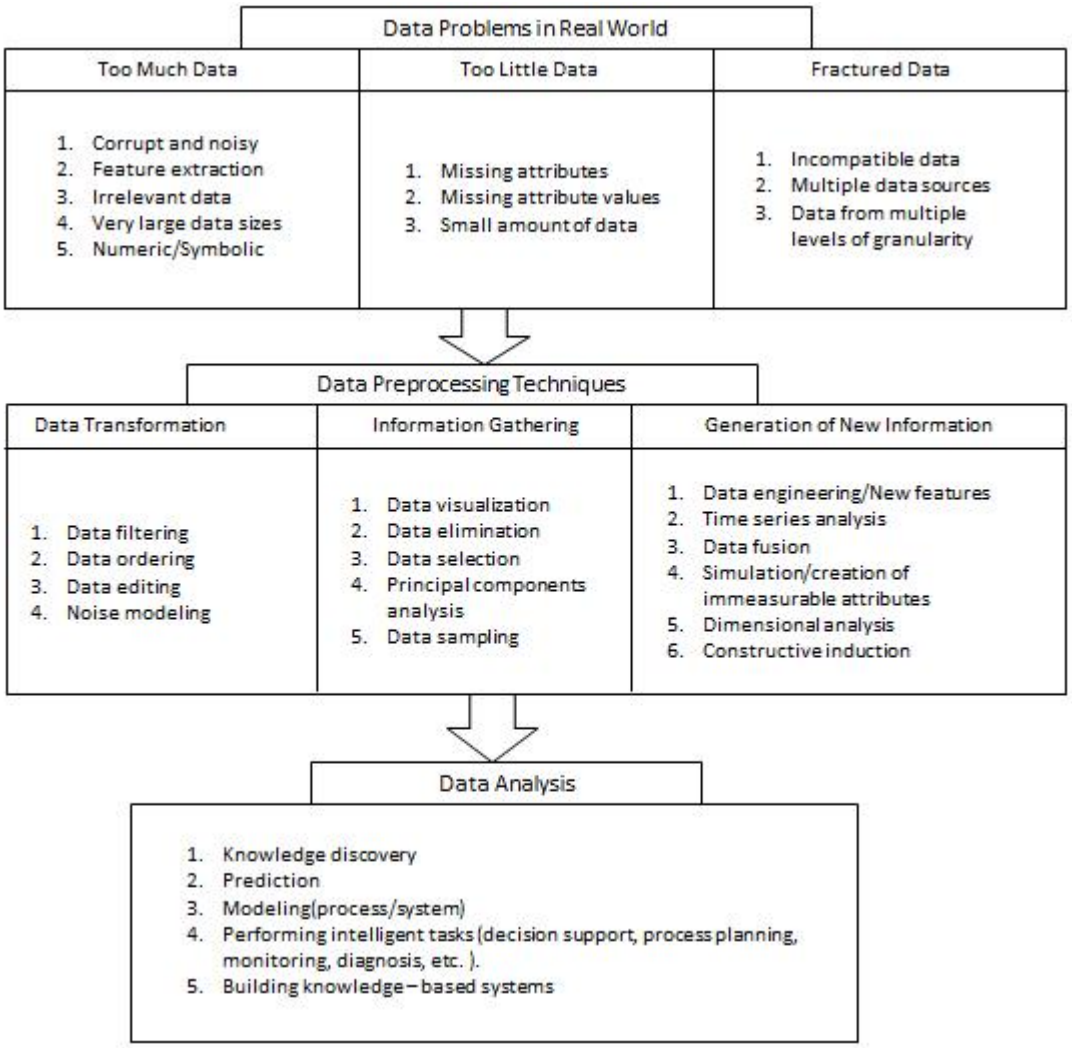


그림 3. 이상한 데이터 및 데이터 전처리 기술

3.3 무관한 데이터

많은 데이터 분석 응용 프로그램은 대규모 데이터 세트에서 의미 있는 데이터의 추출을 필요로 한다. 무관한 데이터가 제거된 경우 복잡성을 크게 줄일 수 있으며 데이터 분석 도구의 성능을 향상시킬 수 있다[20].

III. CoAP 프로토콜을 기반으로 인터넷을 연결하고 스마트 폰 상에 데이터 정제를 위한 미들웨어를 설계

1. CoAP기반의 스마트 폰과 IoT 노드 간 통신 설계

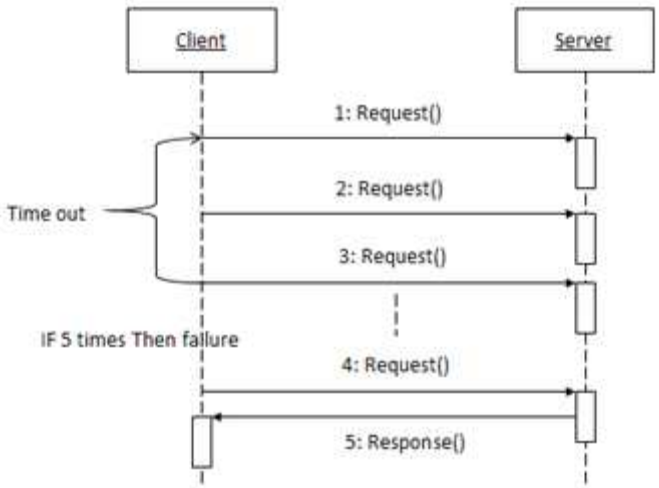


그림 4. CoAP 프로토콜 메시지 전송 시퀀스

그림 4는 CoAP 프로토콜을 통하여 메시지를 전송하는 시퀀스를 나타낸다. 클라이언트에서 서버에 메시지를 전송하면 서버에서는 클라이언트에 응답메시지를 전송한다. CoAP 프로토콜은 UDP상위 프로토콜로서 클라이언트에서 전송한 메시지가 서버에 도착하지 않을 경우가 있다. 이때 클라이언트는 서버로부터 응답메시지를 받을 수 없다. 클라이언트에서 서버에 요청메시지를 전송하고 정해진 시

간 내에 클라이언트가 서버로부터 응답메시지를 받지 못하면 다시 요청 메시지를 전송한다. 이와 같이 반복하여 전송하는 메시지 D는 같은 것으로 전송하며 5번까지 전송하는데 계속 응답메시지를 받지 못하면 이 통신은 실패하는 것으로 한다[9][11].

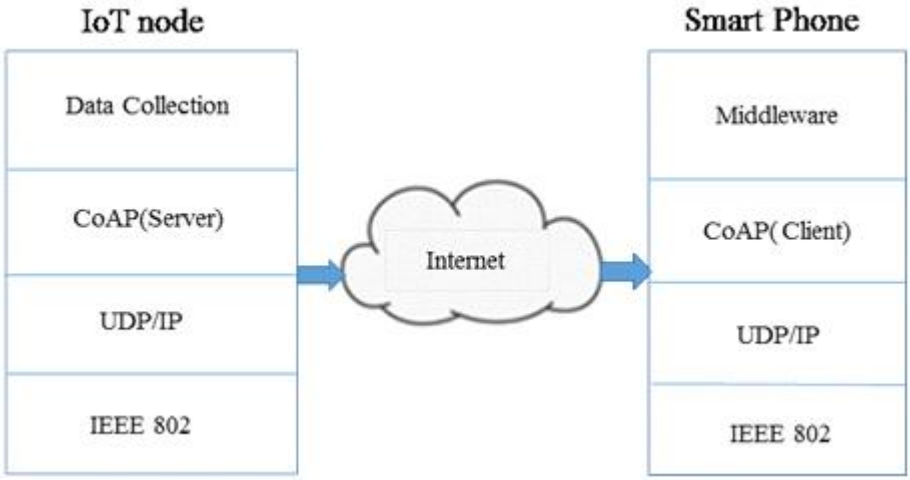


그림 5. IoT 노드와 스마트 폰 연결 기본 구성도

그림 5는 작은 용량의 메모리와 저전력 등 제한된 환경에서 센서나 구동체 노드 간의 통신을 지원하기 위한 CoAP 기반의 IoT노드와 스마폰 연결 기본 구성을 보여주고 있다. 여기서 IEEE 802계층은 LAN, PAN, MAN 표준을 이용할 수 있으며 UDP/IP 를 이용하여 데이터 전송을 수행한다. CoAP 서버는 CoAP 라이브러리를 사용하여 스마트 폰과 통신하기 위한 서버 이며 CoAP 클라이언트는 CoAP을 이용하여 IoT 노드로부터 센싱 데이터를 받기 위한 응용프로그램이다. 미들웨어는 수집된 데이터의 처리를 위해 CoAP 클라이언트를 포함하고 있으며, 중복 데이터와 에러 데이터 처리 기능을 제공한다[10].

2. 스마트 폰 과 IoT 노드의 통신을 위한 시스템 전체 구조

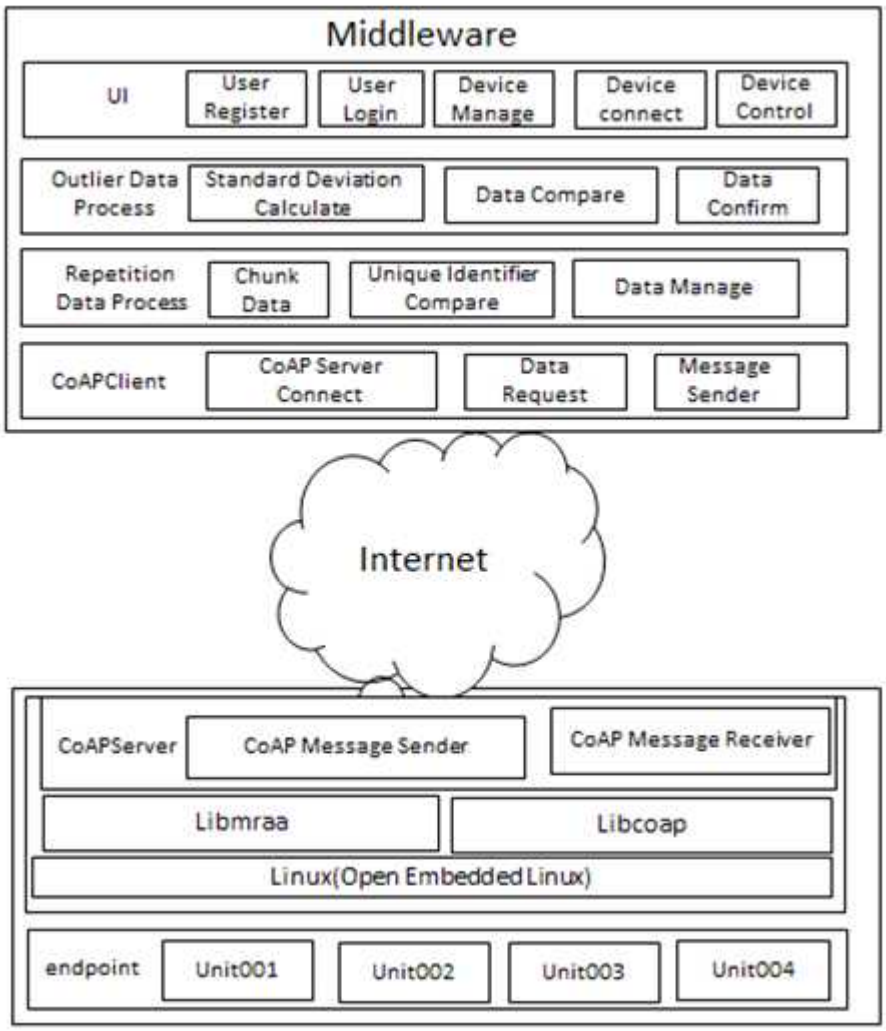


그림 6. 스마트 폰과 IoT 노드간의 통신을 위한 시스템 전체 구조도

그림 6은 스마트 폰과 IoT 노드간의 통신을 위한 시스템 전체 구조도이다. IoT 노드는 endpoint, Linux, Libmraa, Libcoap, CoAPServer등으로 구성된다. 하나의 CoAP endpoint는 CoAP 메시지가 도착되어야 하는 최종 목표를 의미한다. 여기서 Unit001, Unit002, Unit003, Unit004가 포함되어 있다. IoT노드의 운영체제는

오픈 임베디드 리눅스이다. Libcoap은 IoT노드 상의 디바이스와 CoAP Server를 연결하기 위한 라이브러리이다. Libmraa는 자바 스크립트 및 파이썬을 바인딩하고 갈릴레오, 에디슨 및 기타 플랫폼의 IO를 연결하는 라이브러리이다. CoAPServer 는 CoAP Message Sender, CoAP Message Receiver로 구성한다. CoAP Message Receiver는 CoAPClient 에서 보내온 메시지 수신을 의미한다. CoAP Message Sender는 CoAPClient 로 데이터 전송하는 것을 의미한다.

스마트 폰에 포함한 미들웨어는 CoAPClient, Repetition Data Process, Outlier DataProcess, UI로 구성한다. CoAPClient 는 CoAPServer을 연결하는 역할, CoAPServer 에서 데이터를 수신하는 역할, CoAPServer 에게 메시지를 전송하는 역할을 제공한다. Repetition Data Process는 원시 데이터를 청크 데이터로 나누는 역할, 유니크 블록을 비교하는 역할, 유니크 블록이 존재하는지 확인하는 역할을 제공한다. Outlier DataProcess 는 표준편차를 계산하는 역할, 표준편차와 수신한 데이터를 비교하는 역할을 제공한다. UI 는 사용자 생성, 사용자 등록, IoT Node 상의 디바이스를 스마트폰 미들웨어에 추가, 디바이스를 선택하여 디바이스 측에 데이터 수집 및 제어하는 역할을 제공한다.

3. IoT 노드 설계

3.1. IP 및 Endpoint Unit ID 매핑 아키텍처

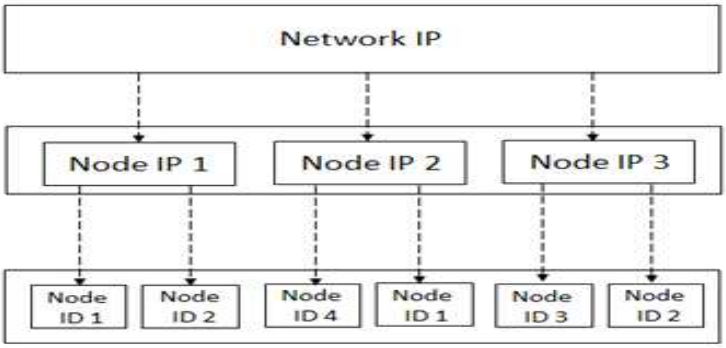


그림 7. IP 및 Endpoint Unit ID 매핑 아키텍처

그림 7은 IP 및 Endpoint Unit ID 매핑 구조도이다. 제안된 Endpoint장치 ID 시나리오에서 일반화된 IP 구조 및 ID 매핑을 제공한다. 네트워크 IP 및 로컬 IP 주소는각각의 노드 및 물리적 노드의 네트워크에 접속하는데 사용된다. 우리는 하나의 노드가 다수의 자원을 가질 수 있음을 제안하며 이들 자원들의 각각은 다수의 서브 - 식별자 (IDS)에 의해 표현될 수 있다. 복합적인 자원을 위한 식별자는 서브Unit ID 라고 하며 노드는 하나 혹은 하나 이상의 Unit ID를 가질 수 있다.

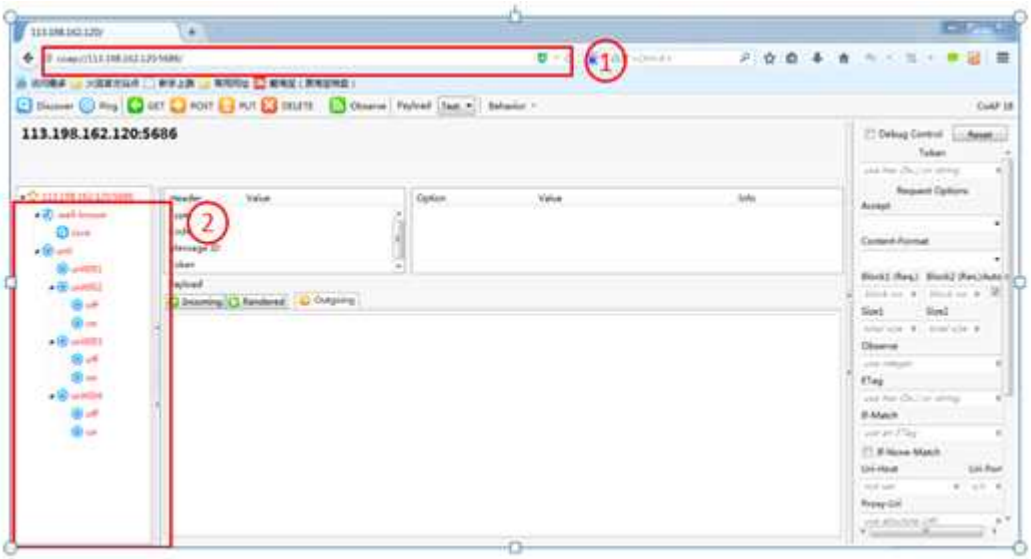


그림 8. IP 및 Endpoint Unit ID 매핑 예

파이어 폭스의 Copper(Cu) CoAP 사용자 에이전트이다. 설치하면 센서 장치는 인터넷을 통하여 상호 작용할 수 있다[17]. 그림 8은 IP 및 Endpoint Unit ID 매핑 예이다. 여기서 노드 IP주소는 113.198.162.120 이다. 이 IP 주소를 통하여 물리노드를 방문할 수 있다. 여기에서 물리 노드는 unit001, unit002, unit004, unit004로 구성 되어 있다. unit001은 온도 센서이다. unit002는 파란색 LED, unit003는 노란색LED, unit004는 빨간색 LED이다.

3.2. Mraa API

Libmraa는 런타임에 특정의 보드 또는 하드웨어의 탐지를 할 뿐만 아니라, 개발자들과 센서 제작자들이 하드웨어 위에서 센서와 액추에이터를 나타내는 것과 고수준 레벨 언어와 구조로 저수준 커뮤니케이션 프로토콜을 제어하는 것을 쉽게 한다.

3.2.1. Mraa API 구조

C API Moidules	C++ API Classes
Gpio	Gpio class
I2c	I2c class
Aio	Aio class
pwm	Pwm class
spi	Spi class
uart	Uart class
common	common

표 2. Mraa API 구조

표 2는 Mraa API 구조도이다. Mraa API는 Gpio 모듈, i2c모듈, Aio 모듈, pwm 모듈, spi 모듈, uart 모듈, common모듈을 포함한다. Mraa API 는 Galileo Gen 1 - Rev D, Galileo Gen 2 - Rev H, Intel Edison, Intel(R) NUC DE3815 등의 시스템을 지원 한다

3.2.2. Gpio 클래스

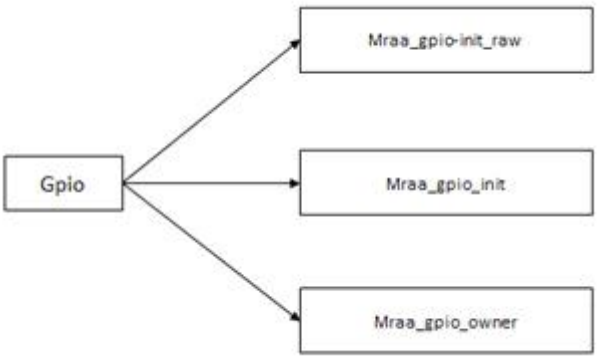


그림 9. 객체를 생성하기 위한 함수의 호출 그래프

본 논문에서 gpio 클래스를 이용하여 CoAP 서버와 센서를 연결한다. 그림 9는 gpio 객체를 생성하기 위한 함수의 호출 그래프 이다. pin, owner, raw 3개 파라메터가 있다. Pin 은 핀 번호이며, owner는 소유자 핀을 설정한다.

3.3. 노드 등록 설계

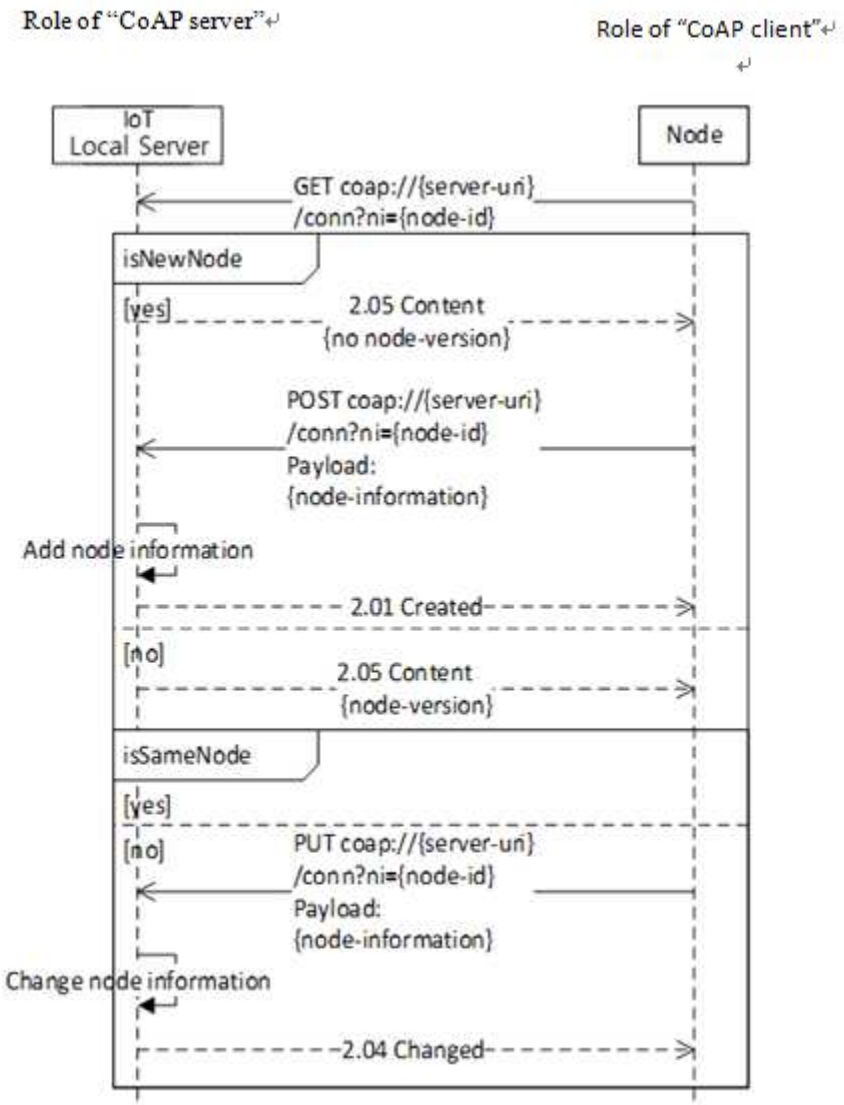


그림10. 노드 등록 시퀀스 다이어그램

그림 10은 노드 등록 시퀀스 다이어그램 이다. 먼저 IoT 노드에서 "ni"를 포함한 GET 메시지를 IoT Local Server에 있는 "conn" 리소스에 전송한다. 노드의 ID는 IoT 미들웨의 데이터베이스에서 IoT노드의 정보를 조회하는데 사용된다. 노드 ID에 의하여 노드정보가 조회되면 IoT미들웨어는 해당 노드의 버전 값을 반환한

다. 조회되지 않으면 IoT미들웨어는 어떤 값을 반환하여 해당 노드가 존재하지 않는다는 것을 알려준다. IoT 노드가 응답메시지를 받으면 "ni" 파라메터와 payload를 포함한 "POST" 요청을 IoT 미들웨어에 전송한다. 해당 노드의 식별자로 정보가 조회되지만 다른 버전일 경우 PUT 요청을 IoT 미들웨어에 전송하여 기존의 정보를 수정한다. 노드의 정보와 버전이 같은 경우에는 본 등록을 종료한다.

3.4. 노드 제어 설계

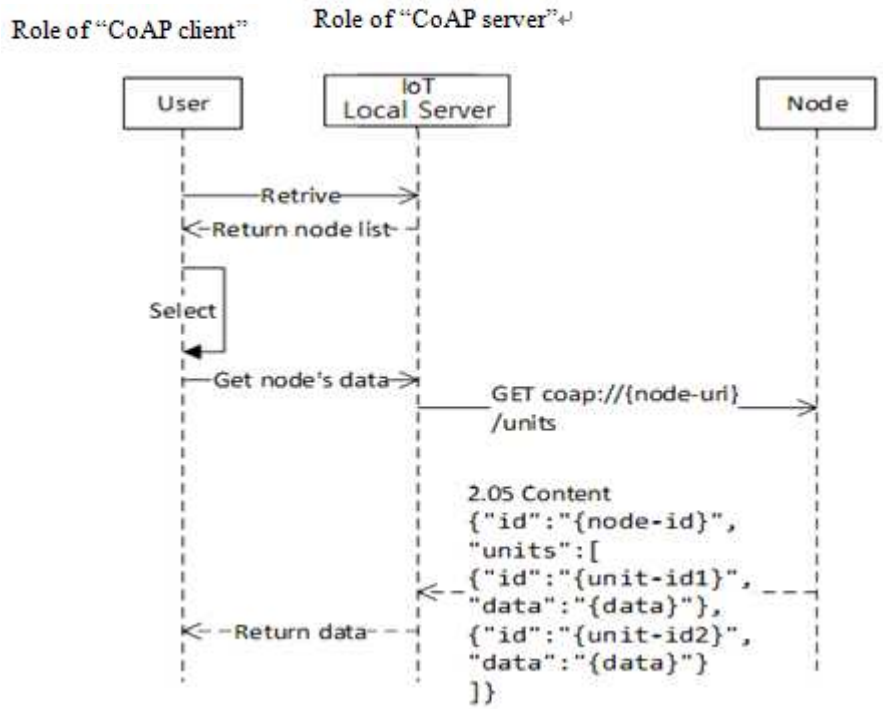


그림 11. IoT노드의 실시간 환경데이터 수집 시퀀스

그림 11은 IoT노드의 실시간 환경데이터 수집 시퀀스이다. 사용자는 IoT local server를 통하여 노드의 목록을 조회한다. 목록에서 특정된 노드를 선택하여 IoT Local Server에 노드의 상황 데이터를 수집하는 요청을 전송한다. IoT Local Server는 노드에 CoAP 요청을 하여 노드가 수집한 실시간 상황데이터를 반환받

아 다시 사용자에게 전송한다.

4. CoAP기반의 스마트 폰과 IoT노드 간 통신하기 위한 스마트 폰의 미들웨어 설계

CoAP기반의 스마트 폰과 IoT 노드 간 통신을 하기 위하여 스마트 폰에 미들웨어로 CoAPClient 설계, 데이터 처리설계, UI설계를 구성한다. 본 장 1절은 CoAPServer를 설계하고 2절은 데이터 처리를 설계한다. 3절은 UI를 설계한다.

4.1. CoAPServer 설계

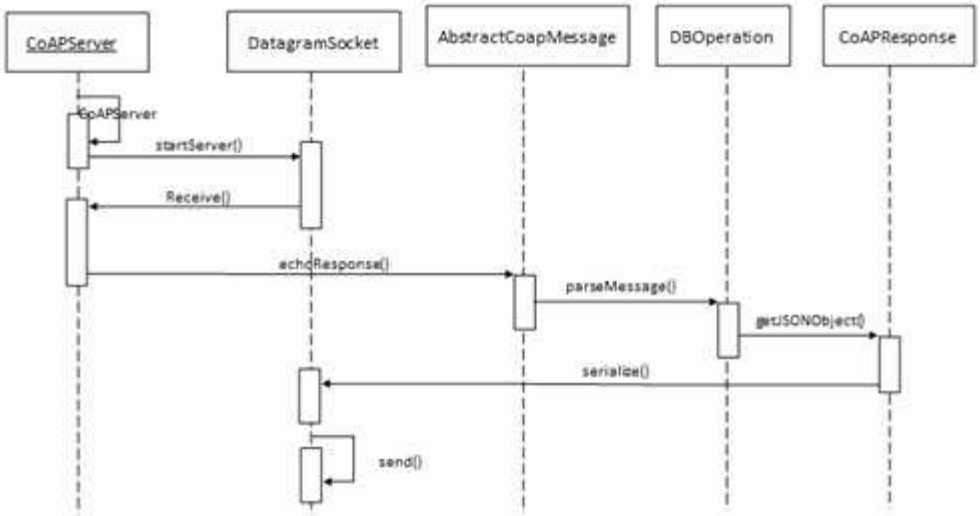


그림 12. CoAPServer 동작 시퀀스 다이어그램

이 그림은 요청 예로 CoAP 서버와 클라이언트를 설명하고 있다. 요청에 URI HOST, URI PORT, URI PATH 와 URI QUERY가 포함된다. URI QUERY를 통하여

식별자를 포함하여 CoAP 요청을 하여 데이터베이스에서 해당 CoAP노드의 정보를 조회한다. 그리고 다시 CoAP 클라이언트로 정보를 Json 포맷으로 전송한다. Json은 여러가지 데이터타입을 표현할 수 있다. CoAP 메시지에 노드의 ID와 Token값을 포함함과 동시 Json데이터도 포함하여 UDP를 통하여 Android 클라이언트로 전송한다[18].

4.2. 데이터 처리설계

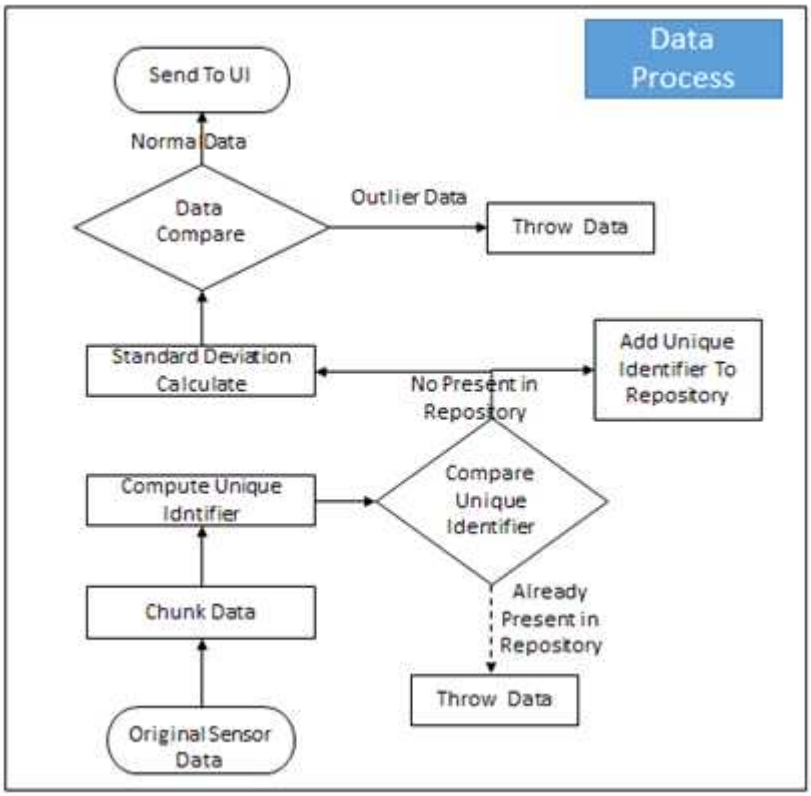


그림13. 데이터 처리 흐름도

그림 13은 데이터 처리 흐름도 이다. 처리 단계는 다음과 같다.

1) 원시 데이터를 청크 데이터로 나눈다.

2) 유니크 식별자를 통하여 유니크 블록을 식별한다.

3) 저장소에 유니크 블록이 존재하는지 확인한다. 없으면 추후 중복을 검사하기

위해 저장소에 등록한다.

4) 표준편차를 계산한다.

5) 표준편차와 수신한 데이터를 비교 한다

6) 조건을 만족하면 UI 에 표출한다. 아니면 데이터를 제거한다.

4.2.1. 중복 데이터 처리 알고리즘

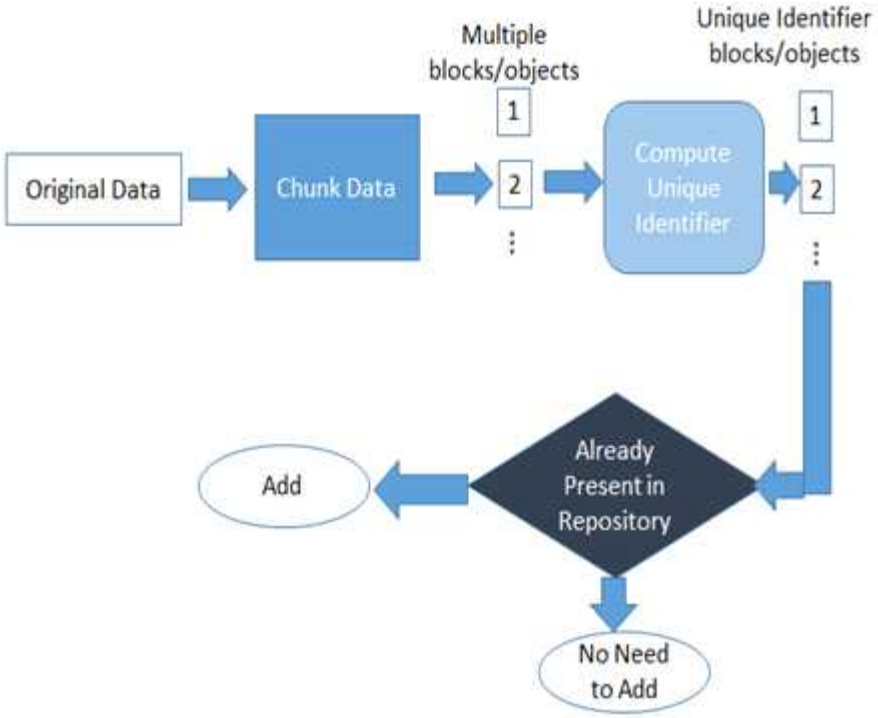


그림14. 중복 데이터 처리 알고리즘

그림 14는 중복 데이터 처리 단계도 이다. 먼저 원시 데이터를 청크 데이터로 나눈다. 다중 블록 또는 오브젝트는 유니크 식별자를 통하여 유니크 블록 또는 오브젝트를 식별한다. 저장소에 유니크 블록 또는 오브젝트의 존재여부를 확인한다. 있으면 중복 데이터를 줄이고 없으면 사용자에게 전달한다.

4.2.2. Outlier 데이터 처리 알고리즘

times	1	2	3	4	5	6	7	8	9	10	11
L(cm)	10.35	10.38	10.3	10.32	10.35	10.33	10.37	10.31	10.34	20.33	10.37

표 3 수집된 원시 데이터

표 3은 시간에 따라 측량된 값의 예 이다. T는 측량할 때의 시간 이고 L 은 측량된 값 이다. 구간 [−3σ, 3σ]에 속할 확률은 정상상태에서 99.8% 를 넘는 반면에 사고 등으로 진전되어 갈수록 점차 줄어드는 것을 볼 수 있다[5].

$$\sigma = \sqrt{\frac{\sum_{i=1}^{11}(Li-\overline{L})^2}{11-1}} = 3.01\text{cm}$$

$$3\sigma = 3.01*3 = 9.03\text{cm}$$

수식 1

수식 1 은 표준편차의 값을 계산하는 수식이다. Li는 측량된 값 이다,$\bar{L}$ 는 센서 값에 대한 평균값 이다. σ 는 표준편차 이며 여기서는 9.03 이다.

$$\Delta L_1 = L_1 - \bar{L}$$

$$= 10.35 - 11.25$$

$$= 0.9 < 3\sigma$$

수식 2

수식 2에 L_1의 값은 10.35 이고, $\bar{L}$ 의 값은 11.25 이다. 수식2에서 계산된 값은 표준편차의 3배수 값보다 작으므로 L_1은 적합한 데이터 이다.

$$\Delta L_{10} = L_{10} - \bar{L}$$
$$= 20.33 - 11.25$$
$$= 9.08 > 3\sigma$$

수식 3

수식 3에 L_{10} 의 값을 20.33 이고, $\bar{L}$ 의 값을 11.25 이다. 수식3에서 계산된 값은 표준편차의 3배수 값보다 크므로 L_{10}은 부적합한 데이터 이다.

4.3. UI 설계

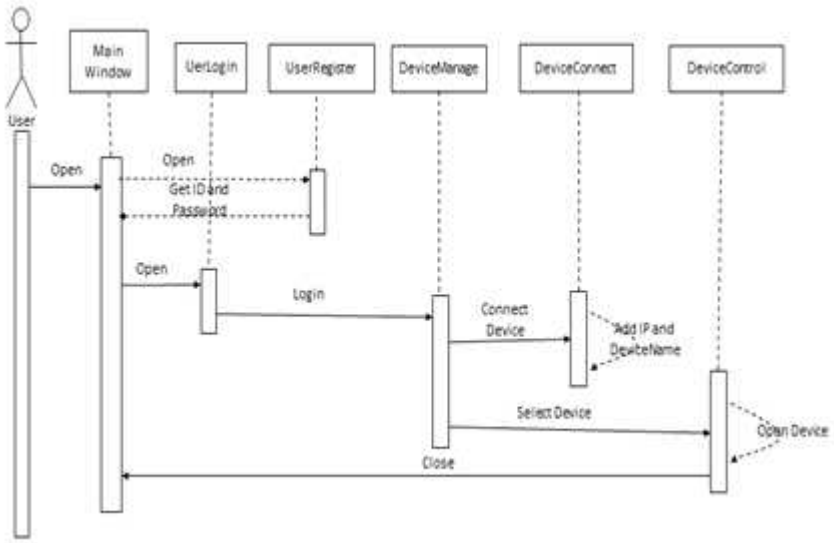


그림15. UI 동작 시퀀스 다이어그램

그림15는 UI 동작 시퀀스 다이어그램이다. 위의 과정은 MainWindow, UserLogin, UserRegister, DeviceManage, DeviceConnect, DeviceControl로 구성되여 있다. 먼저 MainWindow에 접속하여 UserRegister에서 사용자의 ID, Password, 위치, 이름을 입력하여 사용자를 생성한다. UserLogin에서 사용자 등록을 한다. DeviceConnect에서 IoT Node 상의 디바이스의 IP 및 Endpoint를 입력하고 스마트 폰 미들웨어에 디바이스를 추가한다. DeviceManager, DeviceControl은 디바이스를 선택하여 디바이스 측에서 데이터를 수집 및 제어한다.

IV. CoAP 프로토콜을 기반으로 인터넷을 연결하고 스마트 폰 상에 데이터 정제를 위한 미들웨어를 구현

1. IoT 노드 및 스마트 폰 환경

IoT 노드는 에디슨보드를 사용하며 운영체제는 오픈 임베디드 리눅스이다. CoAP 서버를 구현하기 위하여 Libcoap 라이브러리를 이용한다[12]. 스마트 폰에서 사용한 하드웨어는 VEGA IM-A850L 이고 운영체제는 Android 4.1 이다. CoAP 클라이언트를 구현하기위하여 Californium 프레임워크를 이용하여 스마트 폰에서 받은 데이터를 여러 가지로 처리하기 위하여 미들웨어를 적용한다.

개발 환경		
환경	CoAP 서버	미들웨어
시스템	Edison	Android 4.1
컴파일	gcc	Java
툴	eclipse	eclipse
프로그램 언어	c	Java, Android

표 4. CoAP 서버 및 미들웨어를 구현하기 위한 개발환경

표 4 에는 CoAP 서버 및 미들웨어를 구현하기 위한 개발환경을 나다내고 있다.

- 개발환경: Linux OS, JDK1.7, Android SDK, Eclipse juno, MySQL database, Android 4.1 platform, WiFi, Libmraa, C.

- Intel Edison board: IoT 노드
- TINKERKIT: sensor node
- Yocto Linux 1.6

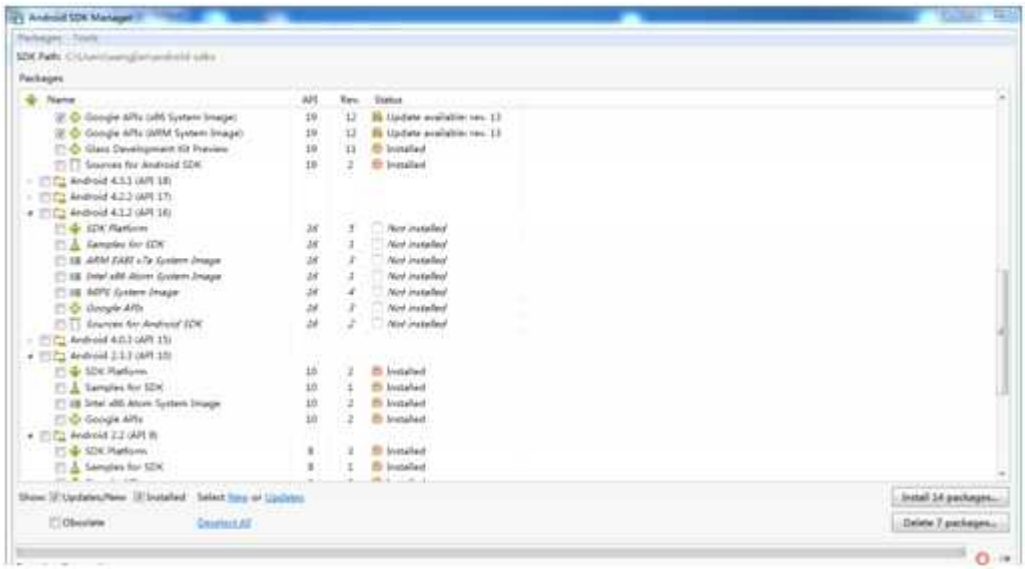


그림16. Android SDK Manager

1.1 Intel Edison

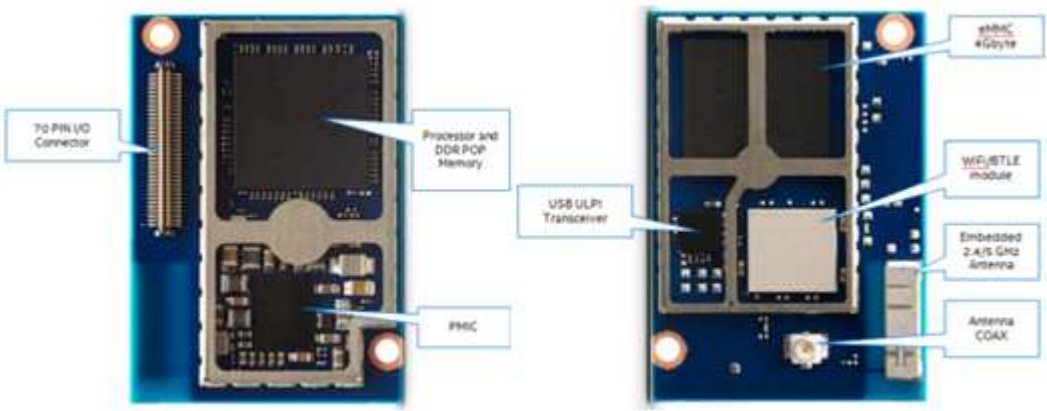


그림 17. Intel Edison

IoT 노드를 구현하기 위하여 Intel Edison보드를 이용한다. 개발환경은 Yocto Linux 1.6, Arduino IDE, C, C++, Python 이다. 22nm 공정 500MHz 듀얼코어 듀얼쓰레드 인텔아톰 CPU & 100MHz 32-bit 인텔 쿼크마이크로 컨트롤러를 제공하며 1GB LPDDR3 POP 메모리가 탑재되어 있다. 802.11 a/b/g/n, 블루투스 4.0을 지원한다.

1.2. TINKER KIT

이 KIT는 에디슨, 갈릴레오 등 개발보드를 위한 센서, 구동체 등을 포함한 패키지이다. 미들웨어와 IoT노드의 통신 및 실행을 확인하기 위하여 이 패키지를 사용하였다. 아래 사용한 센서, 구동체과 실드를 보여주고 있다.

그림 18. TINKER KIT

- Sensor Shield(1)
- 온도 센서(1), 라이트 센서(1)
- LED Light(4)

1.3. Yocto Linux 1.6

Intel Edison 보드에는 기본적으로 Yocto Linux 가 탑재되어 있으며 다음 단계

로 IP를 할당한다.

- putty로 Serial 번호를 이용하여 등록한다.
- "configure_edison -- wifi "명령을 이용하여 wifi를 검색한다.
- WiFi 번호를 선택한다.
- WiFi 의 ID 및 비밀번호를 입력해서 WiFi 에 연결한다.

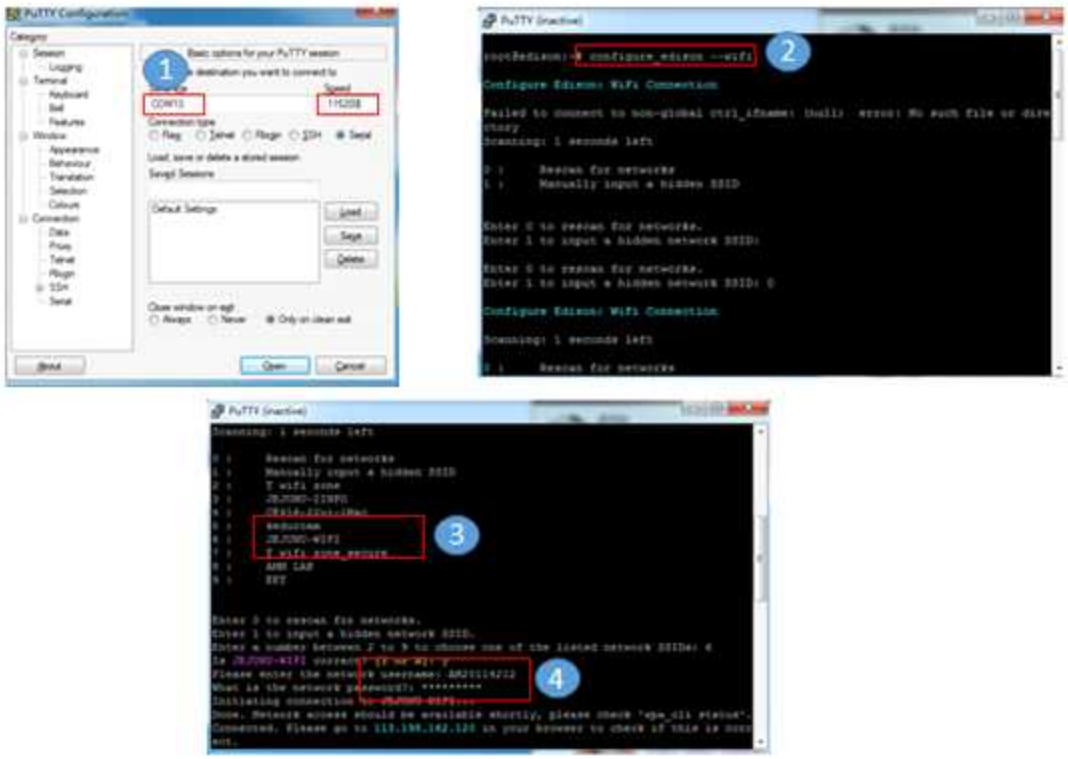


그림 19. IP 설정과정

2. IoT 노드 구현

2.1. 노드 등록

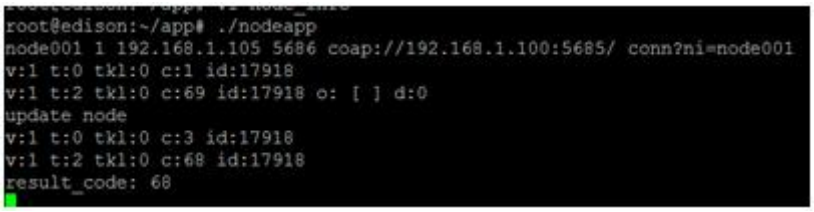


그림 20. putty를 통한 노드의 운영체제 접근화면

그림 20은 putty를 통하여 노드의 운영체제에 접근하는 화면이다. "./nodeapp" 명령을 이용하여 이미 설치한 "nodeapp" 프로그램을 실행한다.

그림 21. 노드 등록 과정 및 버전 확인 화면

그림 21은 노드 등록과정 및 버전을 확인하는 화면이다. 노드를 등록할 때 먼저 본 노드의 존재 여부를 확인하여야 한다. 때문에 질의에 노드의 유일한 식별자를 포함하여 메시지를 전송한다. 응답메시지에 존재하지 않으면 위와 같이 "0" 값을 반환한다.

그림 22. 노드 등록 과정 및 정보 업데이트화면

그림22는 노드 등록 과정 및 정보를 업데이트하는 화면이다. 노드에서 IoT Local Server의 "conn" 리소스에 노드의 정보를 payload에 포함하여 전송한다. 이 정보는 노드의 프로그램 안에 포함되며 노드의 기본정보를 포함한 노드의 식별자, 노드 정보의 버전 등을 포함하고 있다.

2.2. 노드 검색

그림 23. 등록된 노드 목록화면

그림 23은 등록된 노드 목록의 정보를 보여주는 화면이다. 이 목록은 Json 포맷으로 데이터를 보여준다. 사용자가 클라이언트 프로그램을 통하여 HTTP 요청으로 노드의 목록을 해당 질의에 의하여 조회한다. 여기서 사용자는 웹브라우저를 통하여 값을 조회하며 질의에는 아무런 값도 포함하지 않았다. 즉 모든 노드를 조회하게 된다.

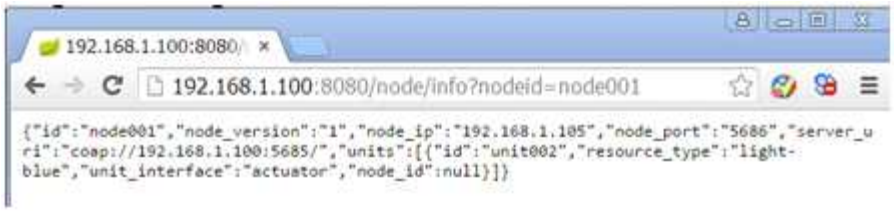


그림 24. 특정노드를 노드id로 검색

그림 24는 특정노드를 노드id로 검색하는 화면이다. 사용자는 노드의 ID를 통하여 특정된 노드의 모든 정보를 Json 포맷으로 가져올 수 있다.

2.3. 노드 제어

사용자는 노드의 식별자 와 해당 unit의 식별자, 그리고 해당 unit에 관한 명령을 통하여 특정된 unit를 제어할 수 있다

그림 25. 센서정보 읽기화면

그림 25는 센서정보를 가져오는 화면이다. 여기서 사용자는 노드의 ID가 "node001"인 unit에 명령을 전송한다. 이 unit은 하나의 온도 센서이다. 이 명령을 통하여 사용자는 실시간 온도 값을 가져온다.

그림 26. LED Light 켜기 화면

그림 26은 LED Light 를 켜는 화면이다. 이 그림에서 unit002는 하나의 전등 구동체이며 "ON" 과 "OFF" 두 가지 명령을 실행할 수 있다. 사용자 웹브라우저를 통하여 "ON" 명령을 이 unit에 전송하여 전등을 켜는 화면이다.

그림 27. LED Light 끄기 화면

그림 27은 LED Light를 끄는 화면이다. 이 그림은 사용자가 웹브라우저를 통하여 "OFF" 명령을 unit에 전송하여 전등을 끄는 화면이다.

2.4. CoAPServer 구현

CoAP 프로토콜은 응용계층프로토콜로 서버 프로그램 구현이 가능하다. CoAP 프로토콜 server는 C로 구현한다. CoAP 프로토콜 server는 CoAP 프로토콜 노드에 설치되므로 메모리가 작고 하드웨어와 직접 연동하여야 하므로 C와 같은 마이크로컴퓨터용 프로그램밍언어로 개발한다. 요청메시지는 IETF에서 제안하는 CoAP 프로토콜 포맷과 같아야 하며 서버에서 정확하게 파싱 하여야 한다. 응답메시지도 요청메시지처럼 포맷에 맞아야 한다.

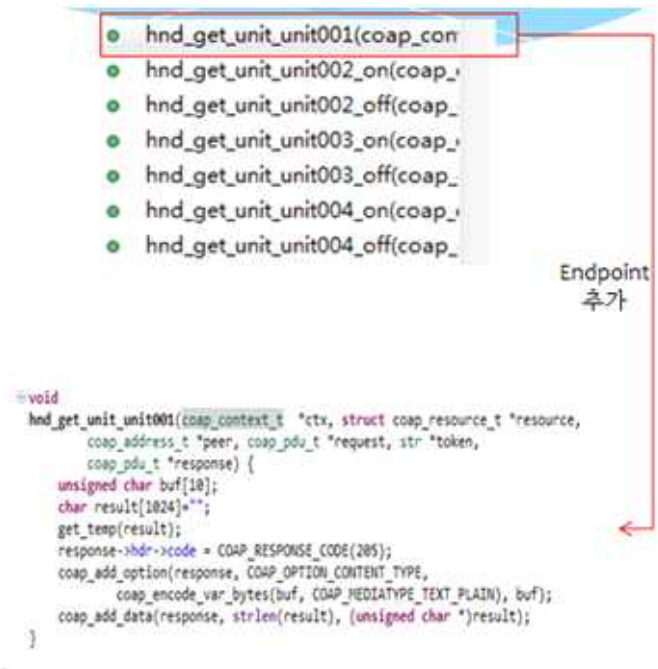


그림 28. CoAPServer에 Endpoint 추가

여기서 Endpoint로 unit001, unit002, unit003, unit004를 추가한다.

IoT 노드에 센서에서 자원을 받으려면 libmraa 라이브러리를 이용하여 센서와 CoAPServer 를 연결해야 된다. 이러한 연결 기본 정의를 그림 29 에서 보여주고 있다.

그림 29. CoAPServer에 디바이스와 서버 연결

3. 스마트 폰 미들웨어 구현

3.1. CoAP Client 구현

CoAP 클라이언트를 구현하기위하여 Java Runtime Environment (JRE) 상에 Java를 이용한 CoAP 프레임워크인 Californium 라이브러리를 이용한다[13]. 표 2 는Californium 라이브러리에 존재하는 주요한 클래스들이다. Server API는 CoapServer, CoapResource, CoapExchange 등의 클래스를 포함한다. 이 클래스들은 CoapResource측에서 자원를 받고, 서버측에 자원을 추가하고 서버를 시동

하는 기능을 가지고 있다. Client API는 CoapClient, CoapHandler, CoapResponse, CoapObserveRelation 등의 클래스를 포함한다. 이 클래스들을 이용하여 대상 URI의 CoapClient를 인스턴스화하는 등의 기능을 사용할 수 있다.

Module Name	Class name	Function
Server API	CoapServer	Implement custom resources by extending CoapResource
	CoapResource	Add resources to server
	CoapExchange	Start server
Client API	CoapClient	Instantiate CoapClient with target URI
	CoapHandler	Use offered methods get(),put(),post(),delete(),observe(),validate(),discover(),orping()
	CoapResponse	
	CoapObserveRelation	Optionally define CoapHandler for asynchronous requests and observe

표 5. Californium 라이브러리 구조

그림 30은 CoAP Client 프로세스 클래스 다이어그램이다. CoAPClient 프로세스는 Respond클래스, Handler클래스, Request클래스, CoAPClient 클래스 등으로 구성한다. CoAPRequest의 부모클래스로서 CoAPClient에서는 CoAPRequest를 통하여 객체를 생성한다. Handler 클래스는 요청이나 응답을 처리하는 클래스이다. CoAPClient 클래스는 요청 및 응답을 모두 제공한다. Respond 클래스와 CoAPClient 클래스는 관련 관계가 있다. Handler 클래스 과 CoAPClient 클래스는 의존관계가 있다. Request 클래스와 CoAPClient 클래스는 관련관계가 있다.

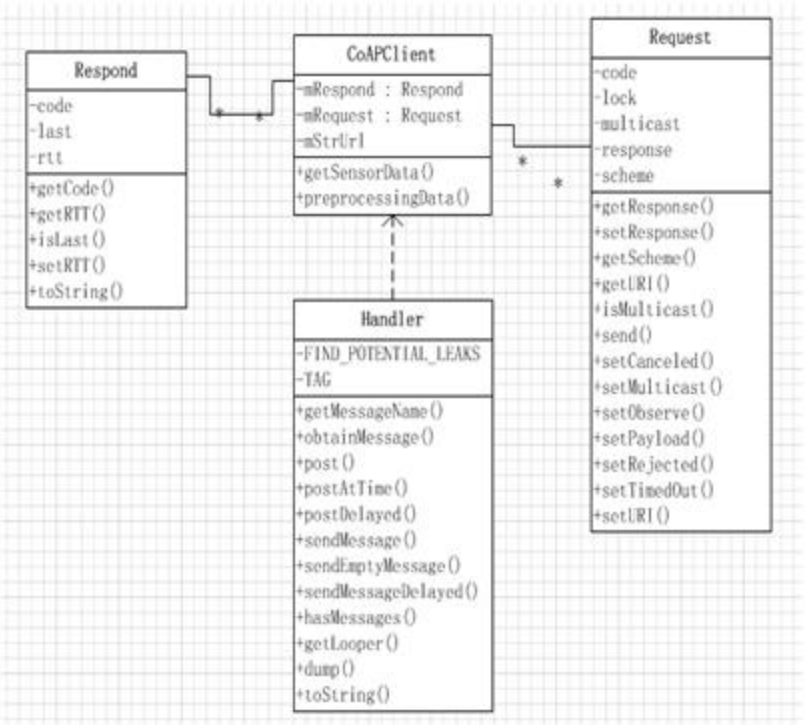


그림 30. CoAP Client 프로세스 클래스 다이어그램

3.2. Repetition Data 프로세스 구현

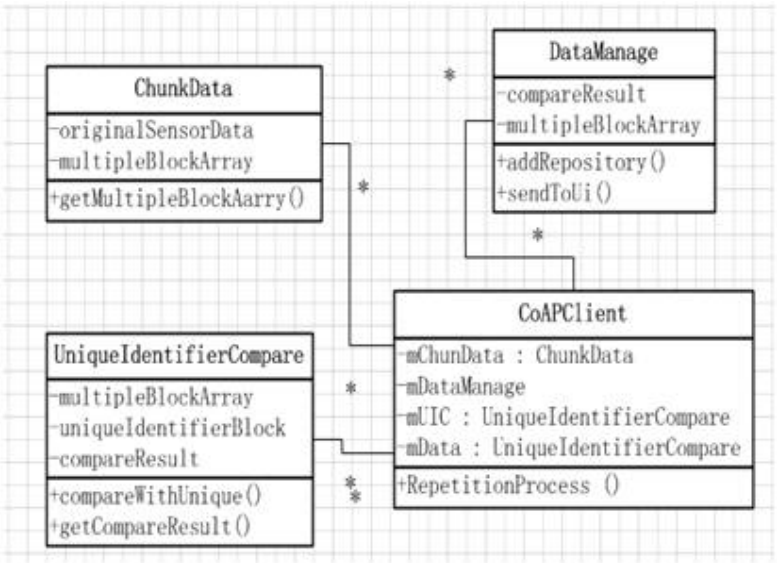


그림 31. Repetition Data 프로세스 클래스 다이어그램

그림31은 Repetition Data 프로세스 클래스 다이어그램이다. Repetition Data 프로세스는 UniqueIdentifierCompare 클래스, ChunkData 클래스, DataManage 클래스로 구성한다. ChunkData 클래스는 원시 데이터를 청크 데이터로 나누는 기능을 제공하며 UniqueIdentifierCompare 클래스는 유니크 블록이 존재하는지 확인하는 기능을 제공하고 DataManage 클래스는 데이터 관리 기능을 제공한다. UniqueIdentifierCompare 클래스, ChunkData 클래스, DataManage 클래스와 CoAPClient 클래스의 관계는 의존관계이다. CoAPClient 클래스를 통하여 각 클래스의 객체생성, 함수를 호출한다.

3.3. Outlier Data 프로세스 구현

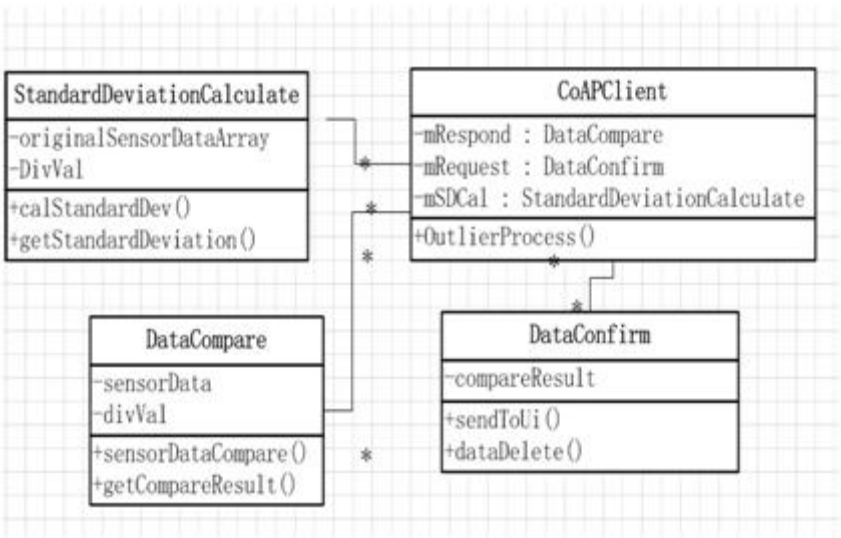


그림 32. Outlier Data 프로세스 클래스 다이어그램

그림 32는 Outlier Data 프로세스 클래스 다이어그램이다. Outlier Data 프로세스는 StandardDeviationCalculate 클래스, DataCompare 클래스, DataConfirm 클래스로 구성한다. StandardDeviationCalculate 클래스는 표준편차 계산 기능을 제공한다. DataCompare 클래스는 표준편차와 수신한 데이터를 비교하는 기능을 제공한다. DataConfirm 클래스는 데이터 제거 및 전송 기능을 제공한다.

CoAPClient 클래스를 통하여 각 클래스의 객체생성, 함수를 호출한다.

3.4. UI 구현

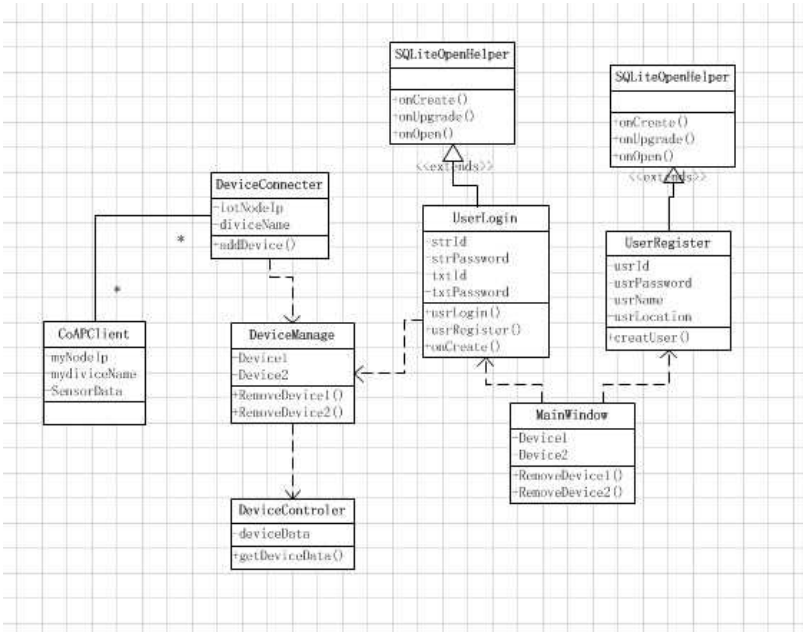


그림 33. UI 구현 클래스 다이어그램

그림 33은 UI 구현 클래스 다이어그램이다. UI 구현는 MainWindow 클래스, UserRegister 클래스, UserLogin 클래스, DeviceConnecter 클래스, DeviceManager 클래스, DeviceControler 클래스로 구성한다. MainWindow 클래스는 MainWindow 출력 역할을 한다. UserRegister 클래스는 사용자의 ID, Password, 위치, 이름을 입력해서 사용자를 생성하는 기능을 제공한다. UserLogin 클래스는 사용자를 등록하는 역할을 한다. DeviceConnecter 클래스는 IoT Node 상의 디바이스의 IP 및 Endpoint를 입력하고 스마트폰 미들웨어에 디바이스를 추가하는 기능을 제공한다. DeviceManager 클래스, DeviceControl 클래스는 디바이스를 선택하여 디바이스 측 데이터를 수집 및 제어한다. MainWindow 클래스와 UserLogin 클래스, UserRegister클래스는 의존관계에 있으며 UserLogin 클래스와 DeviceManager 클래스는 의존관계가 있고

DeviceManager 클래스와 DeviceController 클래스, DeviceConnecter 클래스는 의존관계가 있다. CoAPClient와 DeviceControler 클래스도 의존관계가 있다.

V. 시험 및 성능평가

4 장의 구현방안을 통하여 시스템이 구현되었다. 다음에 유스 케이스를 통하여 시스템이 지원되는 기능을 나타낸다.

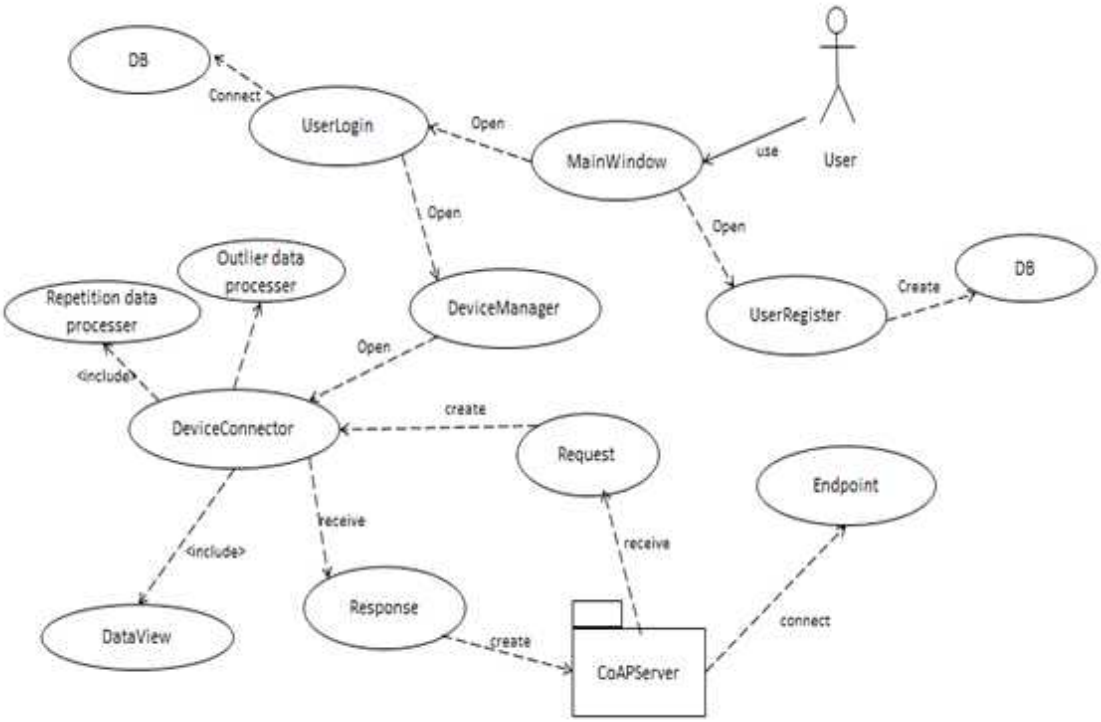


그림 34. 시스템 유스 케이스

그림 34는 시스템 유스 케이스 관계를 나타내고 있다. 사용자는 MainWindow를 통하여 사용자생성, 사용자등록을 수행하고 사용자가 등록되면 Device를 연결한다. DeviceConnector 에서 IP 및 Endpoint 이름을 입력하고 연결을 시키며 CoAP 메시지생성 및 CoAP 메시지 발송기능이 있다. CoAPServer에서 메시지를 받아서 해석한다. CoAPServer는 Endpoint를 연결하고 DeviceConnector에게 응답 데이터를 보낸다. DeviceConnector는 데이터 중복처리, 이상한 데이터처리, 데이

터 뷰 기능 등이 있다.

1. 시험 환경 설정

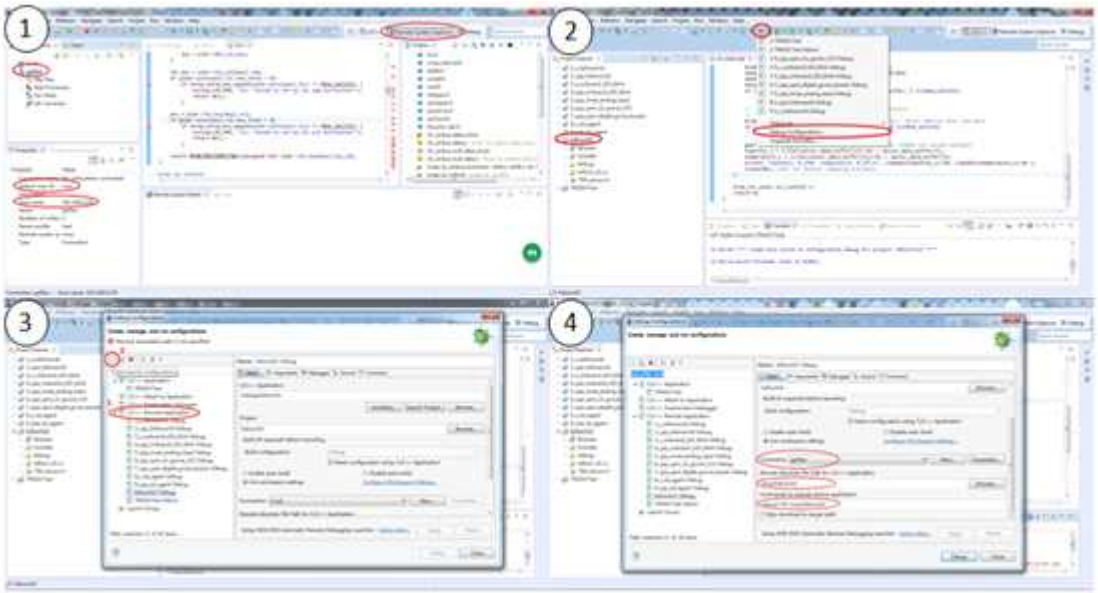


그림 35. CoAP 서버 배치 단계도

CoAP 서버 배치 단계는 다음과 같다

1) 화면에 “galileo”호스트클릭, “Default User”와 “ip”를 입력한다.

2) EdisonI2C 메뉴 선택, “Debug Configurations...”를 클릭한다.

3) “C/C++ Remote Application”를 클릭, EdisonI2C 를 원격 응용 프로그램으로 설정한다.

4) 원격 시스템에 있는 실행 파일 경로를 설정한다.

```
root@edison:~# echo $PWD'>'
/home/root>
root@edison:~#
root@edison:~# chmod 755 /tmp/my_node;/tmp/my_node;exit
```

그림36. CoAP 서버 배치 성공화면

그림 36은 CoAP 서버 배치 성공화면이다. 이제 CoAP클라이언트는 IoT 노드에게 요청을 보내면 자원을 받을 수 있다.

2. 시험 결과

그림 37. 사용자 리스트 및 등록화면

그림 37은 사용자 리스트 및 등록화면 이다. 사용자 ID 및 비밀번호가 없으면 Register를 클릭한다. 리스트화면이 나오고 사용자의 ID, 비밀번호, 이름, 도시 등 정보를 입력하고 사용자를 리스트 한다. 사용자가 리스트 되면 등록화면으로 다시 들어간다. 등록화면에서 사용자의 ID, 비밀번호를 입력하면 등록이 된다.

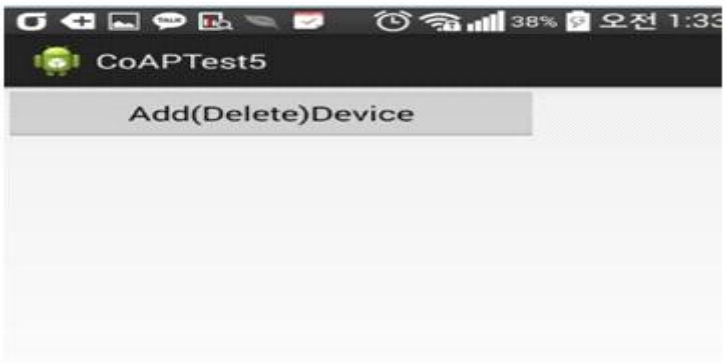


그림 38. 디바이스 추가화면

그림 38은 IoT 노드 추가 화면 이다. Add(Delete)Device로 IoT 노드 리스트 정보를 추가할 수 있다.

그림 39. IoT 노드 리스트 정보 및 IoT 노드와 스마트 폰 연결화면

그림 39는 IoT 노드 리스트 정보 및 IoT 노드와 스마트 폰 연결화면 이다. Add를 클릭한다. IoT 노드 리스트 정보를 입력하고 IoT 노드에 연결한다. IP 는 IoT 노드의 주소이다. 이 주소는 서버의 IP 주소이다. DeviceName 은 IoT 노드상

의 물리 노드 이다. 여기에서 물리노드 이름은 unit001 이다. Unit001는 온도센서 Endpoint 이름 이다.

3. 결과분석

CoAP 클라이언트는 관리자 측에서 동작하는 노드로 서버에게 요청메시지를 전달하고, 서버 측으로 부터 응답메시지를 수신하는 기능을 수행하는 노드이다. 이 노드는 헤더를 생성하여 요청메시지를 전달한다. "T"필드에 Confirmable(0), Non-Confirmable(1), Acknowledgement(2), Reset(3) 중 하나를 선택하여 메시지를 만들고, Code 필드는 GET(1), POST(2), PUT(3), DELETE(4)등을 선택한다. Content-Type, URI_SCHEME, URI-Authority 등과 같은 옵션을 필요한 경우 추가할 수 있다.

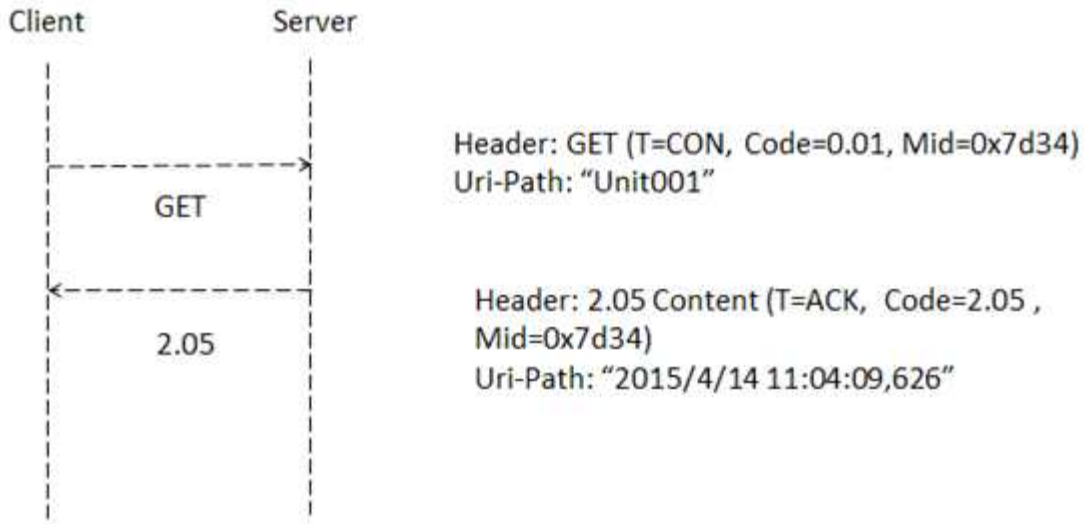


그림40. CoAP 프로토콜 요청 및 응답 흐름도

그림 40은 CoAP 프로토콜 요청 및 응답흐름도 이다. CoAP 프로토콜메시지는 4 byte 헤더로 시작되며 토큰(Token), 옵션(Option)과 페이로드(Payload) 등이 있

다. CoAPClient는 "CON" 메시지를 통해 요청한다. Server가 "CON" 메시지를 수신하면 응답메시지를 응답한다. CoAP 요청은 0.01 GET 방법을 이용한다. 성공하면 CoAP Server가 "2.05 Content"메시지를 보낸다. 요청과 응답에서 "MID"가 일치해야 된다. 여기서는 "0x7d34"이다. 요청응답 토큰 필드는 공백으로 되어 있다. 옵션유형은 "Uri-Path"이며 Option Value유형은 문자열이다. CoAP 응답 값은 온도와 시간이 포함되어 있다. 온도는 626 이다. 시간은"2015/4/14 11:04:09"이다.

Light Original Data	
2015/6/10 10:07:23	306
2015/6/10 10:07:24	310
2015/6/10 10:07:25	312
2015/6/10 10:07:26	313
2015/6/10 10:07:27	317
2015/6/10 10:07:28	321
2015/6/10 10:07:29	323
2015/6/10 10:07:31	327
2015/6/10 10:07:33	322
2015/6/10 10:07:34	324
2015/6/10 10:07:35	326
2015/6/10 10:07:36	66
2015/6/10 10:07:37	329
2015/6/10 10:07:38	332
2015/6/10 10:07:39	335
2015/6/10 10:07:40	78
2015/6/10 10:07:41	335
2015/6/10 10:07:42	337
2015/6/10 10:07:43	337
2015/6/10 10:07:44	336
2015/6/10 10:07:45	335
2015/6/10 10:07:46	336
2015/6/10 10:07:47	335
2015/6/10 10:07:48	65

Light Processing Data	
2015/6/10 10:07:23	306
2015/6/10 10:07:24	310
2015/6/10 10:07:25	312
2015/6/10 10:07:26	313
2015/6/10 10:07:27	317
2015/6/10 10:07:28	321
2015/6/10 10:07:29	323
2015/6/10 10:07:31	327
2015/6/10 10:07:33	322
2015/6/10 10:07:34	324
2015/6/10 10:07:35	326
2015/6/10 10:07:37	329
2015/6/10 10:07:38	332
2015/6/10 10:07:39	335
2015/6/10 10:07:41	335
2015/6/10 10:07:42	337
2015/6/10 10:07:43	337
2015/6/10 10:07:44	336
2015/6/10 10:07:45	335
2015/6/10 10:07:46	336
2015/6/10 10:07:47	335

그림 41. 수집된 라이트센서 데이터

그림41는 온도센서를 이용하여 수집된 데이터이다. 왼 쪽은 처리하기 전의 데이터이다. 오른쪽은 처리 후의 데이터이다. 표기 1,2에 데이터 집합 표준편차의 값은:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{11}(66-296)*(66-296)}{11-1}} = 15.5$$

3 배 표준편차 = 46.5 < 296 -66

그래서 66 이상한 센서 값이다.

표기 1,2 에 데이터 집합 표준편차의 값은 15.5로 계산되었다. 3배 표준 편차는46.5이다. 표기1,2에서 데이터 집합 평균값은 296 이다. 3에 데이터의 값과 평균값의 차가 3배 표준편차보다 더 크므로 이상한 센서 값으로 정한다. 오른쪽은 처리 후의 데이터에 이상한 데이터가 제거된 모습이다.

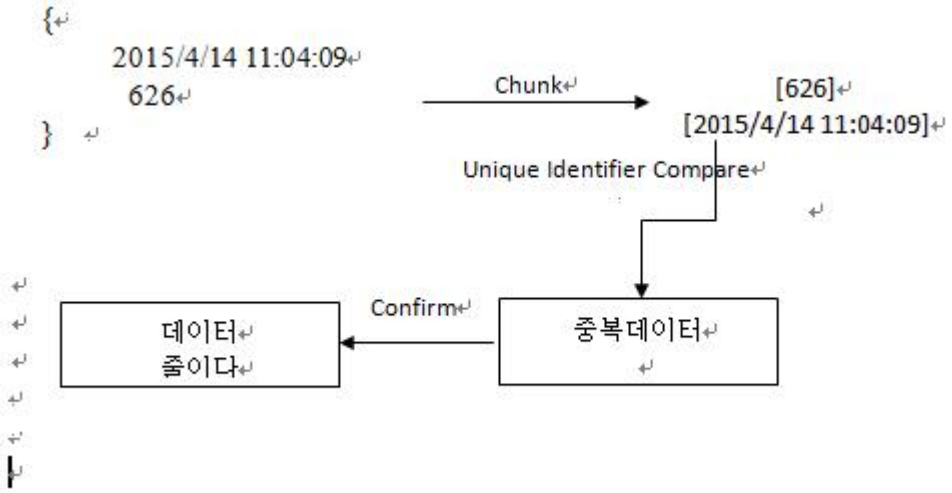
Original Data	
Time	Sensor Data
2015/4/14 11:04:03	606
2015/4/14 11:04:04	619
2015/4/14 11:04:05	620
2015/4/14 11:04:06	619
2015/4/14 11:04:07	618
2015/4/14 11:04:08	619
2015/4/14 11:04:09	620
2015/4/14 11:04:09	626 (1)
2015/4/14 11:04:13	621
2015/4/14 11:04:14	620
2015/4/14 11:04:15	620
2015/4/14 11:04:16	619
2015/4/14 11:04:17	619
2015/4/14 11:04:18	618
2015/4/14 11:04:19	621
2015/4/14 11:04:19	621
2015/4/14 11:04:21	620
2015/4/14 11:04:22	725 (3)
2015/4/14 11:04:23	619
2015/4/14 11:04:24	621
2015/4/14 11:04:25	619

After Processing Data	
Time	Sensor Data
2015/4/14 11:04:03	606
2015/4/14 11:04:04	619
2015/4/14 11:04:05	620
2015/4/14 11:04:06	619
2015/4/14 11:04:07	618
2015/4/14 11:04:08	619 (2)
2015/4/14 11:04:09	620
2015/4/14 11:04:13	621
2015/4/14 11:04:14	620
2015/4/14 11:04:15	620
2015/4/14 11:04:16	619
2015/4/14 11:04:17	619
2015/4/14 11:04:18	618
2015/4/14 11:04:19	621
2015/4/14 11:04:21	620
2015/4/14 11:04:23	619
2015/4/14 11:04:24	621
2015/4/14 11:04:25	619

그림 42. 수집된 온도센서 데이터

그림42는 온도센서를 이용하여 수집된 데이터이다. 왼 쪽은 처리하기 전의 데이터이다. 오른쪽은 처리 후의 데이터이다.

표기1 의 원시 데이터:



표기1의 원시 데이터를 Chunk 처리하여 [626], [2015/4/14 11:04:0]로 나누고 이러한 값이 존재하는지 확인한다. 저장소에 [2015/4/14 11:04:09] 블록이 있으므로 중복 데이터로 확인한다. 오른쪽에 처리된 데이터를 보면 데이터가 줄어들었다.

표기 2,3의 데이터 집합 표준편차의 값은:

$$\text{표준편차} = \sqrt{\frac{\sum_{i=1}^{11}(725-619.7)*(725-619.7)}{11-1}}$$

$$= 10.27$$

$$3\text{ 배 표준편차 } = 30.81 < 725 - 619.7$$

$$\therefore \quad 725 \text{ 이상한 센서 값이다.}$$

표기 2,3의 데이터 집합 표준편차의 값은 10.27로 계산되었다. 3배 표준 편차

는30.81이다. 표기2,3의 데이터 집합 평균값은 619.7 이다. 3에 데이터의 값과 평균값의 차가 3배 표준편차보다 더 크므로 이상한 센서 값으로 정한다.

4 성능 평가

4.1 데이터 프로세스 성능분석

4.1.1 라이트 센서 데이터 분석

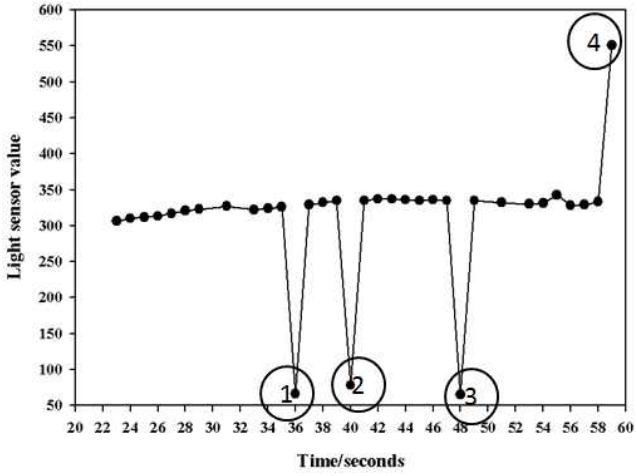


그림 43. Original Light Data

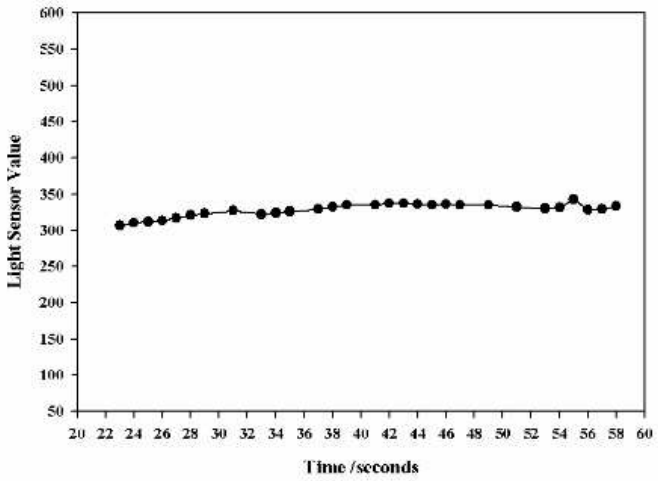


그림 44. After Processing Light Data

그래프의 가로축은 시간은 나타내고 세로축은 측량된 라이트 센서 값이다. 그림 43에서 표기 1,2,3,4는 이상한 데이터이다. Outlier Data 프로세스를 이용하여 처리되며 그림 44에 제거된 데이터가 표시되었다.

4.1.2 온도 센서 데이터 분석

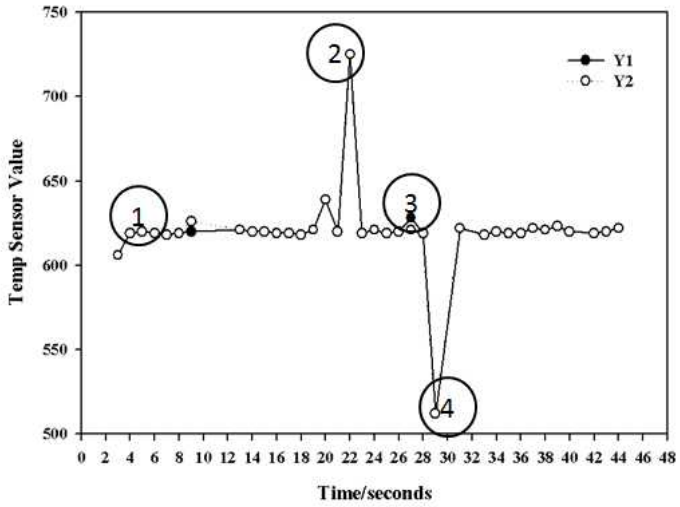


그림 45. Original Temp Data

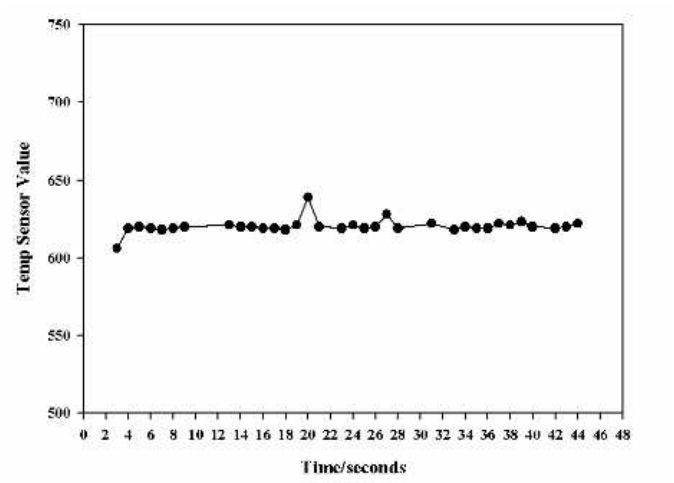


그림 46. After Processing Temp Data

그래프의 가로축은 시간은 나타내고 세로축은 측량된 온도센서 값이다. 그림

45의 표기 1,3 에 중복된 데이터가 있음을 나타낸다. Repetition Data 프로세스를 이용하여 데이터가 처리되면 그림46 처럼 데이터가 줄어든다. 그림 42에서 표기 2,4는 이상한 데이터이다. Outlier Data 프로세스를 이용하여 처리되며 그림 46에 제거된 데이터가 표시되었다.

4.2 CoAP Server vs. HTTP Server

수요분석에 따르면 CoAP 서버가 HTTP 서버보다 더 좋은 성능을 가지고 있다. 다음과 같은 원인으로 설명할 수 있다:

- CoAP 는 HTTP 의 서브세트로 볼 수 있다.
- HTTP 는 8가지 기본 메소드를 정의한다. 그러나 CoAP은 4가지 기본 메소드 정의뿐이다.
- HTTP 응답코드는 HTTP의 서브세트이다. CoAP은 UDP 기반의 프로토콜이며 HTTP는 TCP 기반의 프로토콜이다. TCP 연결 설정은 세 번 핸드셰이크를 사용하므로 UDP보다 더 복잡하다. 그러면서도 CoAP 은 강한 트란스미션 메커니즘이 있다.
- CoAP 프로토콜 매우 작으니까 전송 효율이 더 높다.

Ⅵ. 결론

IETF (Internet Engineering Task Force)의 CoAP (Constrained Application Protocol) 프로토콜은 작은 용량의 메모리와 저전력 등 제한된 환경에서 센서나 구동체 노드 간에 통신을 지원한다. CoAP 프로토콜은 HTTP와 쉽게 상호 변환할 수 있으며, 사물인터넷(Internet of Things : IoT)와 M2M (Machine-to-Machine) 환경에서 저전력 센서와 구동체 네트워크를 통한 기반 시설을 감시하거나 관리할 수 있다. IETF CoRE(Constrained RESTful environments) 워킹그룹(Working Group)에서 2010년에 CoAP 프로토콜에 대한 표준화를 시작하여 최근에 RFC(Request for Comments) 7252로 발표하였다 [9][11].

본 논문에서는 CoAP 프로토콜을 기반으로 안드로이드 스마트 폰과 인텔 에디슨 IoT 노드 사이의 통신을 위하여 미들웨어를 설계 및 구현하였다. 스마트 폰 측 미들웨어를 CoAP 클라이언트로 구현하였으며, 중복데이터와 노이지 데이터를 제거하는 기능을 포함한다. 데이터중복 제거는 데이터의 세그먼트를 분할, 중복된 영역을 제거하여 유일한 고유블록을 단 한번만 저장하는 백업기술을 이용하고 노이지 데이터는 Pauta Criterion을 이용하여 제거한다.

향후 연구는 CoAP 프로토콜을 기반으로 주변의 다양한 센싱 정보를 수집, 스마트폰 미들웨어에 전송하고 분석하여 정확한 정보를 제공하는 것이다. CoAP 프로토콜을 기반으로 가전제품(TV, 에어컨, 냉장고 등)을 비롯해 에너지 소비 장치(수도, 전기, 냉난방 등), 보안기기(도어록, 감시 카메라 등) 이외의 다양한 분야에서의 모든 사물들은 인터넷으로 연결되어 모니터링 및 제어에 유용하게 활용될 것이다.

참고문헌

[1] 박동환, 방효찬. "개방형 시맨틱 USN/IoT 서비스 플랫폼 기술", 전자파학회 제24권 제4호, 2013.7,12-19

[2] S.Bandyopadhyay, M. Sengupta,S.Dutta. "A Survey of Middleware for Internet of Things" Recent Trends in Wireless and Mobile Networks Communications in Computer and Information Science Volume 162,2011,pp. 288-296

[3] ISO/IEC CD 30128 "Generic Sensor Network Application Interface"

[4] 최진철, 이준욱, 박동환, 방효찬. "스마트 단말을 위한 IoT 미들웨어 시스템에 관한 연구"

[5] Seung-Wook Jee, Kyung-Gea Ok, Shi-Kuk Kim, Chun-Ha Lee "표준편차와 확률분포를 이용한 모의전기설비에서 사고 징후 검출"

[6] Z. Shelby, B. Frank, D. Sturek, "Constrained Application Protocol(CoAP)", RFC 7252, June, 2014.

[7] A. Famili, Wei-Min Shen, Richard Weber, Evangelos Simoudis. "Data Preprocessing and Intelligent Data Analysis".

[8] 진남, "실내환경 제어를 위한 구동체 웹과 미들웨어 설계 및 구현" 석사논문 2013년 6월

[9] Wenquan Jin, "A Study of CoAP Extension Based on ID for IoT Node Registration and Message Queuing", 석사논문 2015년2월

[10] 왕건, 사공준, 곽호영, 김도현, "CoAP 기반의 IoT 노드와 스마트폰 간 Connectivity 연구",

[11] 김문권, 김도현 "Implementation and Experiment of CoAP Protocol Based on IoT for Verification of Interoperability"

[12] "libCoAP: C-Implementation of CoAP ",http://libCoAP.sourceforge.net, 2014

[13] “Californium (Californium library) CoAP framework - Java CoAP Implementation", https://github.com/mkovatsc/Californium, 2014
[14] 민경주, 김용운, 유상근, 김형준, 정회경, "CoAP 프로토콜 구현과 USN 환경 적용", 한국 해양 정보통신학회 논문지 제15권 제5호, p1189-1197, May, 2011
[15] 표철식, 강호용, 김내수, 방효찬," IoT(M2M) 기술 동향 및 발전 전망", 한국통신학회지 (정보와 통신) 제30권
[16] 강정호, 김형주, 전문석, "사물인터넷 시장 및 기술 동향", 한국콘텐츠학회지 13(1) 14-17 1598-9437
[17] Copper(Cu),<https://addons.mozilla.org/en-US/firefox/addon/coper-270430>
[18] cJSON,<http://sourceforge.net/projects/cjson/>
[19] CoAP<http://en.wikipedia.org/wiki/Constrained_Application_Protocol>
[20] E. Bontrager et al., GAIT-ER-AID: "An Expert System for Analysis of Gait with Automatic Intelligent Preprocessing of Data", 4th Annual Symposium on Computer Applications in MED CARE (1990), 625-629.