creative commons
COMMONS DEED

저작자표시 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이차적 저작물을 작성할 수 있습니다.
- 이 저작물을 영리 목적으로 이용할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학석사 학위논문

무선 센서네트워크상의 Smart Plug and Play 시스템 설계

컴 퓨 터 정 보 공 학 과

노 순 영

지도교수 김 창 훈

2014년 6월

대구대학교 대학원

Collection

무선 센서네트워크상의
Smart Plug and Play 시스템 설계

이 논문을 공학석사 학위논문으로 제출함.

컴 퓨 터 정 보 공 학 과

노 순 영

지도교수 김 창 훈

노순영의 공학석사 학위논문을 인준함.

2014년 6월

심사위원장 ——————(인)

심 사 위 원 ——————(인)

심 사 위 원 ——————(인)

대구대학교 대학원

Collection

목 차

Collection

Collection

표 목 차

Collection

그림 목차

Collection

Collection

무선 센서 네트워크상의 Smart Plug and Play 시스템 설계

(요 약)

인간 중심의 통신 패러다임에서 사물이 주체로 참여하는 사물인터넷(Internet of Things : IoT)이 미래융합서비스로 각광 받으면서 미국, 유럽, 일본, 중국을 중심으로 사물인터넷 시장을 위한 다양한 정책이 추진되고 이에 따라 점차 시장이 확대될 것으로 예상한다(Uckelmann et al, 2011). 사물인터넷의 주요 기술로는 센싱 기술, 유무선 네트워크 기술, 서비스 인터페이스 기술이 필요하고 특히 센싱 기술은 다양한 상황 및 고차원적인 데이터 센싱을 요구한다. 이를 위해서는 센싱 데이터를 수집하는 게이트웨이에서 센서노드들의 효율적인 관리 및 정확한 제어가 필수이다.

본 논문에서는 CoAP 프로토콜 상에서 센서노드들을 효율적으로 관리하고 제어를 위한 Smart Plug and Play(SPnP)시스템을 제안한다. 제안한 시스템은 데이터 센싱 및 전송하는 센서노드, 수신하는 Zigbee모듈, 수신된 데이터를 처리하는 게이트웨이로 구성되며, 구성 요소간 통신은 Zigbee프로토콜상 CoAP을 구현하여 Non-IP프로토콜의 한계점을 극복하고 확장된 코드를 통하여 게이트웨이에서 센서노드의 등록, 해제, 상태, 요청메시지를 처리하고, 라이브러리화를 통해 다양한 응용서비스에 적용 가능하도록 한다. 제안한 시스템은 기존의 시스템에 비해 확장된 코드의 효율성이 증가 되었고, 코드별70%에서 100%의 메시지 전송성공률을 나타내었다. 따라서 제안한 시스템은 저성능의 소형가전에 적용이 가능할 뿐 아니라 개발에 편의성을 가져다주며 이에 따른 IoT 응용서비스 개발이 활성화 될 것으로 예상한다.

Collection

Ⅰ. 서 론

1. 연구배경 및 필요성

인간 중심의 통신 패러다임에서 사물이 주체로 참여하는 사물인터넷(Internet of Things : IoT)이 미래융합서비스로 각광 받으면서 미국, 유럽, 일본, 중국을 중심으로 사물인터넷 시장을 위한 다양한 정책이 추진되고 이에 따라 점차 시장이 확대될 것으로 예상한다. 국내에서는 2010년 5월 방송통신위원회에서 10대 방송통신 미래서비스(방송통신위원회, 2010)중 사물지능통신을 선정하고 2013년 6월 미래창조과학부에서 인터넷 신산업 육성방안(미래창조과학부, 2013)으로 사물인터넷을 중점적으로 다루는 등 사물인터넷 관련 정책 추진을 본격화 하고 있다.

ITU(International Telecommunication Union-Telecommunication Standardization Sector)는 사물인터넷을 기기 및 사물에 통신 모듈이 탑재되어, 유무선 네트워크로 연결됨으로써 사람과 사물 간, 사물과 사물 간에 정보 교환 및 상호 소통할 수 있는 지능적 환경으로 해석하고 있다(김민식. 정원준,2014, 22-27). 이러한 사물인터넷의 주요구성 요소는 1)센싱 기술 2)유무선 통신 및 네트워크 인프라 기술 3)사물인터넷 서비스 인터페이스 기술이 요구되며 그중 유무선 통신 및 인프라 기술은 사물인터넷 시장이 확대되고, 사물에 부착되는 통신 모듈의 수가 증가하게 됨에 따른 다양한 통신 문제를 해결하기 위하여 최적화된 통신기술이 필요하게 된다. 이를 위해서는 통신 모듈 및 센서 디바이스의 네트워크 효율적인 Plug and Play기술(Hyungjin Song, 2005)이 필수이다. 디바이스의 효율적인 Plug and Play기술 중 하나는 마이크로소프트에서 제공하는 UPnP (Universal Plug and Play)이다. UPnP는 각종 가전제품을 홈 네트워크에 접속하는 인터페이스 규격으로 윈도우에서 가지고 있는 Plug and Play기능을 가전으로 확장시킨 개념으로 네트워크에 연결 및 관리의 편의성을 위해 많이 사용되고 있으나 TV, 오디오등, 물리적 부피가 크거나 PC와 같은 고사양의 가전제품에만 적용되고 있으며 저전력 저성능의 소형가전제품은 적은 메모리와, 용량이 큰 통신 패킷으로

Collection

인해여 적용하기 어려운 단점이 있다. 이러한 점을 보완하기 위하여 소형 가전 제품의 홈 네트워크 연결 및 관리를 위한 기술이 활발히 연구되고 있다.

2. 연구 목표

본 논문은 사물인터넷의 원활한 서비스 제공을 위하여 네트워크에 접속하는 디바이스의 효율적인 관리를 위해서 Smart Plug and Play(SPnP)시스템을 제안한다. 제안한 시스템은 데이터를 센싱하고 전송하는 센서디바이스와 데이터를 수집하고 디바이스를 관리하는 게이트웨이로 구성하며, 센서디바이스의 자동접속기술, 게이트웨이의 능동적인 디바이스 관리 기술을 구현한다. 구현한 기술은 저 전력 통신표준 IEEE 802.15.4의 Zigbee기반 CoAP프로토콜로 동작하기 때문에 소형가전제품에 적용이 가능하다. 그러므로 모든 사물을 인터넷에 연결하는 사물인터넷 구성에 반드시 필요한 기술이라 할 수 있다.

3. 논문의 구성

본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 전개를 위한 CoAP프로토콜 및 기존의 Plug and Play시스템의 기본적인 구성과 동작에 관하여 설명하고, 3장에서는 본 논문에서 제안한 시스템의 설계에 관한 내용을 설명한다. 4장에서는 시스템의 구현 방식과 테스트한 결과에 대해 평가를 내리고, 끝으로 5장에서 논문을 결론 짓는다.

Collection

Ⅱ. 관련연구

본 장에서는 현재 추진 중인 사물인터넷의 주요동향을 살펴보고, 무선통신을 위한 CoAP프로토콜과 디바이스 관리 기술인 UPnP에 대해 알아본다.

1. 사물인터넷 산업동향 및 주요 서비스 사례

사물인터넷의 세계 시장은 2011년 26.82조원에서 2015년 47.07조원으로, 국내시장은 2011년 4,147억원에서 2015년 13,474억원으로 성장할 전망이다. 다음 표 1은 세계(IDATE, 2011) 및 국내(KEIT,2012) 사물인터넷 시장의 성장 전망이다.

표 1. 세계 및 국내 사물인터넷 시장

구분	2011	2012	2013	2014	2015	CAGR
세계시장 (조 원)	26.82	29.18	35.61	42.49	47.07	11.9%
국내시장 (억 원)	4,147	5,674	7,201	10,338	13,474	26.6%

국내 사물인터넷 시장은 이동통신사 중심의 단순 결제 서비스에서, 헬스케어, 스마트 팜등 다양한 서비스로 단계적으로 상용하고 있으며 다음과 같은 주요 서비스 사례가 있다.

1) SKT 스마트 팜 서비스

SK텔레콤은 원격제어 지능형 비날 하우스 관리 시스템인 스마트 팜 서비스를 제공하고 있다. 이 시스템은 비닐하우스내부의 온도 및 습도, 배/급수, 사료공급 등 모든 과정에서 스마트 폰을 이용하여 원격제어가 가능하다(전자신문, 2013, 5, 29).

Collection

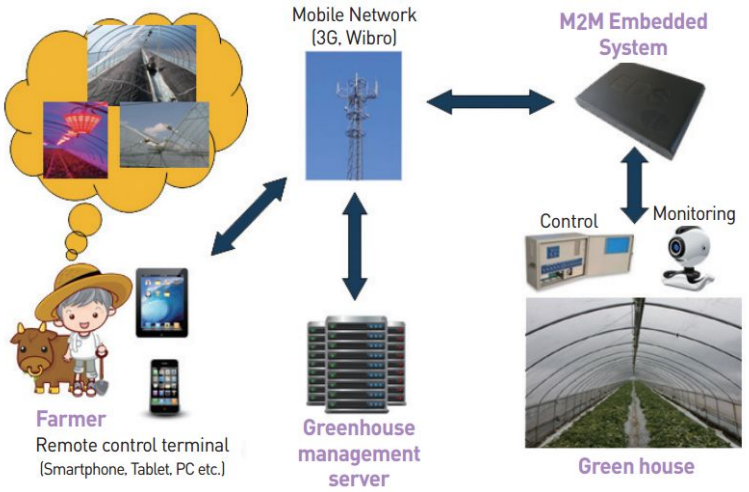

그림 1. SKT 스마트 팜 서비스

2) KT 스마트 홈 서비스

KT는 홈 네트워크 기반의 다양한 서비스를 제공하고 있다. 댁내 방범, 전력 제어, 검침 등 스마트 폰을 이용하여 원격지에서 댁내 환경을 실시간 모니터링을 할 수 있으며, 출입문, 전등 등 원격제어도 가능하다. 또한, 댁내 화재 등 긴급 상황에 대해 알림을 수신할 수 있는 관제서비스도 가능하다(장원규, 이성협, 2013, pp.24-38).

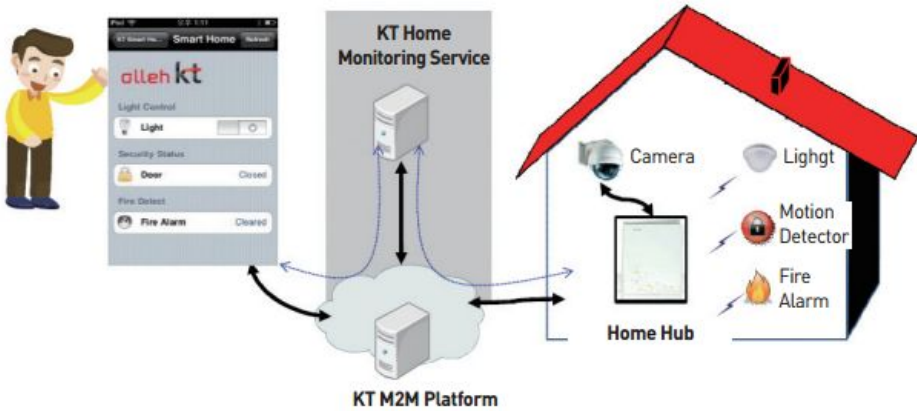

그림 2. KT 스마트 홈 서비스

3) LG U+ 지능형 차량 관제 서비스

LG U+는 실시간 차량관제 서비스를 제공하고 있다. 해당 서비스는 화물차량, 버스, 택시 등을 대상으로 하며 LTE 기반의 사물인터넷 차량관제 시스템 솔루션을 운영하여 승무원, 승객관리, 운행상태와 속도, 이동거리 등 차량 정보를 실시간으로 중앙관제 센터에 전송한다(전자신문, 2012. 5. 6).

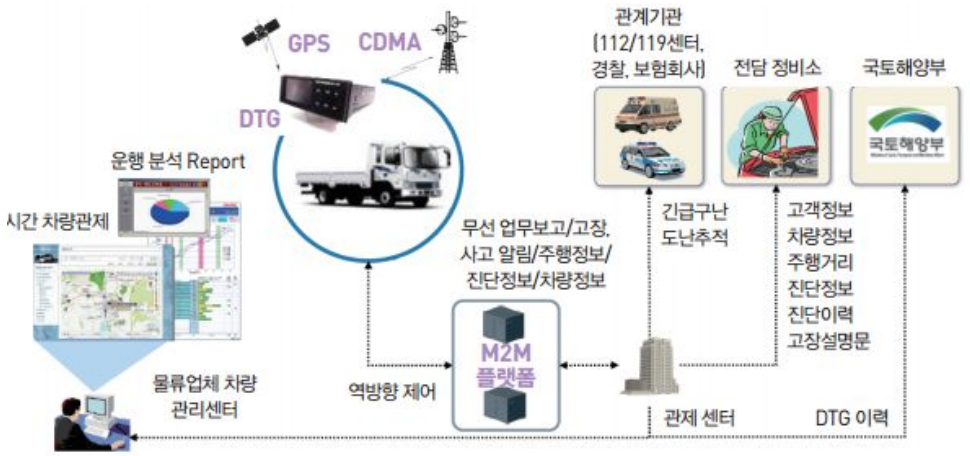

그림 3. LG U+ 지능형 차량 관제 서비스

Collection

2. 관련 연구

1) Constrained Application Protocol (CoAP)

IETF(Internet Engineering Task Force) WG(Working Group)에서 2010년 초에 CoAP (Constrained Application Protocol)을 개발하고, 몇 차례의 수정을 거쳐, 2010년 말 최종 드래프트가 발표되었고, 현재 WG 드래프트18에서 RFC 표준으로 채택되었다(IETF 2013). CoAP프로토콜은 저 전력, 고 손실 네트워크 및 소 용량, 소형 노드에 사용될 수 있는 특수한 웹 전송 프로토콜이다. 소 용량, 소형 노드와 같이 제약이 많은 환경에서 HTTP와 같은 무거운 통신 프로토콜을 사용할 수 없으므로 HTTP처럼 REST형식(IETF, 2012)을 따르면서 웹서비스를 할 수 있는 가벼운 프로토콜개발이 목적이다(고석갑 외, 2013).

(1) CoAP 구성

CoAP은 UDP와 응용계층 사이의 추가 레이어로 웹을 지원하기 위한기술이다. UDP는 가볍고 신뢰도가 낮은 프로토콜이므로 메시지 전송 신뢰도 향상을 위해 Request/Response구조를 구현한다.

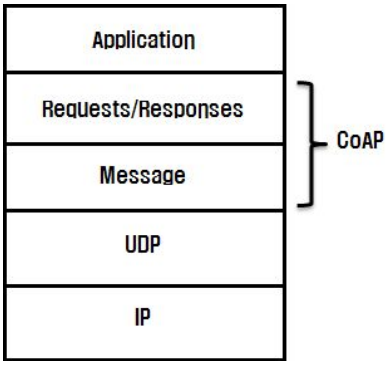

그림 4. CoAP의 추상 계층

그림 4는 CoAP프로토콜의 추상적 계층을 나타낸다. 기본적으로 UDP와 같

Collection

은 비동기적으로 전송되는 프로토콜 위에 추상 계층으로 구성하며 비 신뢰성의 UDP를 사용하므로 신뢰성 있는 전달을 위한 재전송 및 타이머 관리를 옵션으로 포함하고 있다.

(2) CoAP 메시지 포맷

CoAP프로토콜은 데이터 전송 사이즈를 줄이려는 시도를 많이 한다. 이를 위해 그림 5와 같이 단순한 메시지 포맷을 가진다.

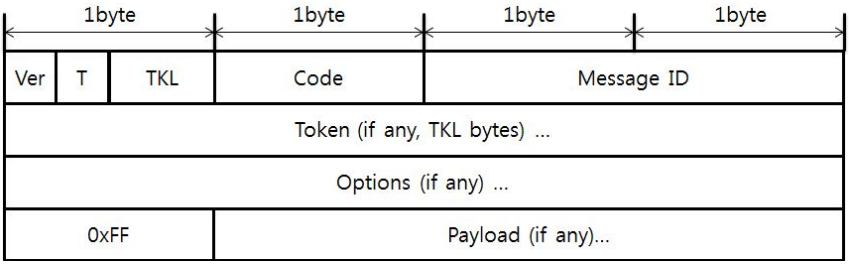

그림 5. CoAP 메시지 포멧

메시지포맷은 간단한 바이너리 포맷으로 인코딩 되며 4바이트 길이의 고정 헤더를 포함한다. 이어서 최대8바이트의 가변길이 토큰이 위치하고 이후 옵션이 온다. 첫2비트 Ver은 버전을 의미하고 두 번째 2비트T는 메시지타입을 의미한다. TKL은 토큰필드의 길이를 나타내고 Code는 메시지의 종류, Message ID는 중복확인 및 확인성/비 확인성 메시지에 대한 짝으로 사용된다. 4바이트로 고정길이 헤더를 구성한 후 최대 8바이트 길이의 토큰이 위치하고 그다음 옵션이 온다.

Collection

(3) CoAP 메시지 전송

■ 신뢰성 있는 메시지

신뢰성 있는 메시지를 전달하기 위해서 CoAP은 CON메시지를 사용하며, CON메시지를 받은 서버는 ACK를 이용해 메시지가 잘 전달되었는지 클라이언트에게 알려준다. 그림 6은 CoAP의 신뢰성 있는 메시지 전송을 나타낸다.

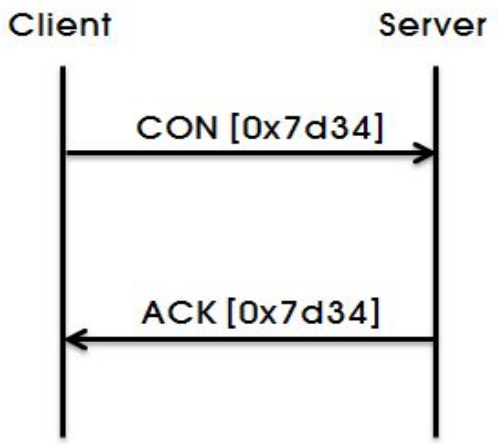

그림 6. CoAP의 신뢰성 있는 메시지 전송

■ 비 신뢰 메시지

비 신뢰 메시지의 전달의 경우 NON메시지를 통해 전송한다. NON메시지를 받은 서버는 또다시 NON메시지를 통해 응답을 전달한다. 비 신뢰 메시지의 경우 Message레이어가 아닌 Request/Response레이어에서 응답을 보내며 그림 7은 CoAP의 비 신뢰 메시지 전송을 나타낸다.

Collection

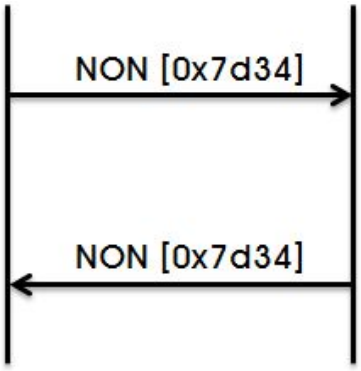

그림 7. CoAP의 비신뢰성 메시지 전송

■ GET요청

서버에서 원하는 데이터를 요청하기 위한 GET요청은 동기식 요청과 비동기식 요청이 있다. 동기식 GET요청은 CON메시지에 요청 메시지를 포함하여 정보를 요청한다. 서버는 그에 대한 응답으로 ACK메시지에 요청한 데이터를 포함해서 전송한다. 이처럼 ACK메시지에 데이터를 포함해서 보내는 것을 피기백 응답이라 하며, 토큰정보는 해당 요청에 대한 응답을 매치하기 위해 사용하는 값이다. 그림 8은 동기식 GET요청을 나타낸다.

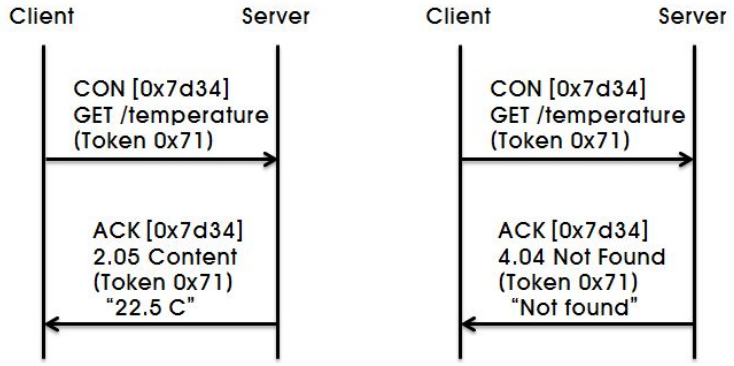

그림 8. CoAP의 동기식 GET요청

그러나 CoAP프로토콜은 센서노드의 슬립, 오류, 정지등 제약사항이 많으므로 위와 같이 동지적으로 사용하기에는 제약사항이 많이 따른다. CON메시지

Collection

를 받은 서버는 해당 센서노드를 바로 사용할 수 없음을 판단하고 메시지 내용이 없는 빈 ACK메시지를 보내고 클라이언트는 응답을 대기한다. 센서노드가 깨어나면 해당 센싱 데이터 값을 읽어 CON메시지에 데이터 값을 채우고 전송한다. 이러한 방식을 비동기식 GET요청이라 한다. 다음 그림 9는 비동기식 GET요청을 나타낸다.

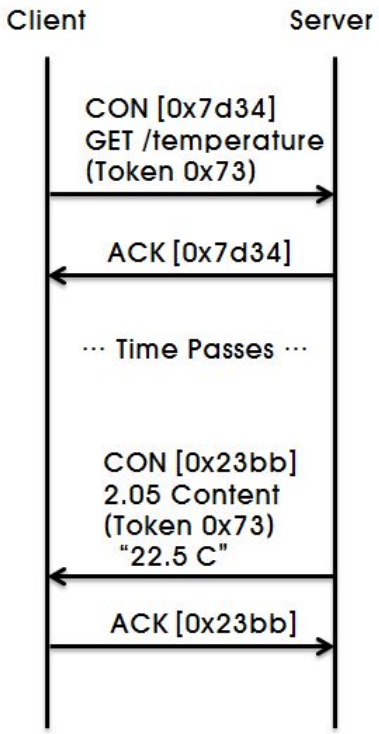

그림 9. CoAP의 비동기식 GET요청

■ NON요청

CON요청과 더불어 NON요청도 자원을 요청할 수 있다. NON메시지에 GET요청을 담아 전송하면 되는데, 서버는 같은 NON메시지에 응답을 실어 전송한다. 그림 10은 CoAP의 NON메시지에 GET요청을 담아 전송하는 것을 나타낸 것이다.

Collection

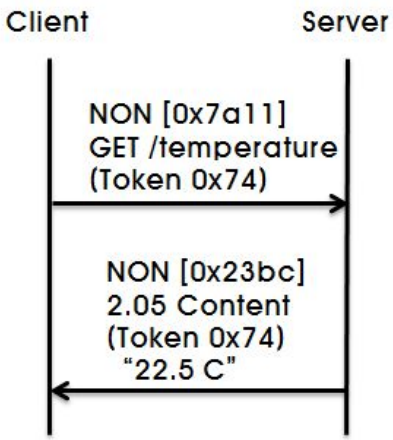

그림 10. CoAP의 NON요청

(4) CoAP 메시지 코드

CoAP은 헤더의 처음 1바이트를 제외한 그다음 1바이트를 메시지 코드로 사용하며 1바이트 중 3비트는 클래스(class)를, 5비트는 자세한 내용(detail)을 의미한다. 클래스와 자세한 내용을 나누어 c.dd로 표현하며 클래스는 0은 요청, 2는 성공적인응답, 4는 클라이언트 에러응답, 5는 서버응답으로 표현한다. 요청의 경우, 0.01은 GET, 0.02는 POST, 0.03은 PUT, 0.04는 DELETE를 나타낸다.

Collection

2) Universal Plug and Play(UPnP)

UPnP는 디바이스를 네트워크에 접속시켰을 때, 인터넷과 웹 프로토콜을 사용하여 서로를 자동으로 인식할 수 있도록 해주는 표준(UPnP Forum, 2011). UPnP는 디바이스가 네트워크에 연결되게 되면 디바이스는 스스로 구성을 완료하며 자신의 주소를 받고 다른 디바이스들에게 존재를 알리기 위해 HTTP 기반의 프로토콜을 사용한다(UPnP Forum, 2008).

(1) UPnP의 구성

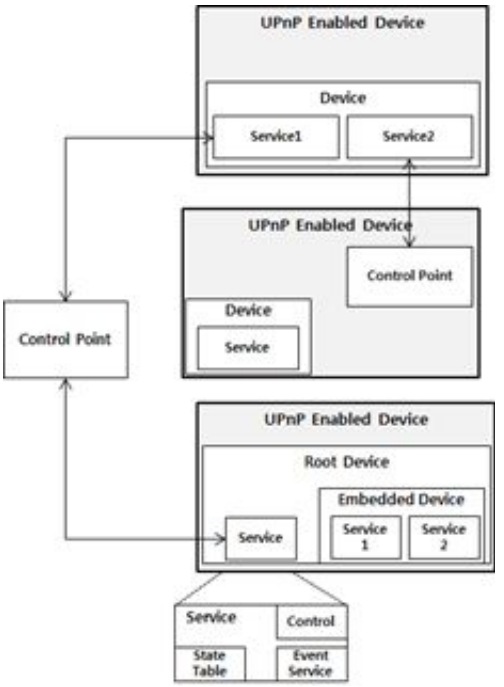

그림 11. UPnP의 구성요소

그림 11은 UPnP의 구성요소를 나타내며 크게 디바이스, 서비스, 제어 포인트(Control Point)로 구성된다.

Collection

■ 디바이스(Device)

UPnP에서 디바이스는 서비스를 제공하는 TV, 오디오등 엔드(End) 디바이스를 의미한다.

■ 서비스

서비스란 디바이스에서 제공 가능한 기능을 의미한다. 예를 들어 TV전원을 on/off 할 수 있으며, 셋탑박스는 원하는 영상을 재생하는 것 등을 의미한다. 디바이스에서 제공하는 서비스의 종류는 XML로 저장되어있는 디바이스 정보에 포함되어 있으며 디바이스 정보는 URL을 통하여 확인할 수 있다. 디바이스가 서비스를 제공하기 위해서는 상태테이블(state table), 제어서버(control server), 이벤트 서버(event server)로 구성되어야하며 상태테이블은 상태 변수를 활용하여 서비스의 상태를 모델화하며 제공되는 서비스의 형태가 변경되면 업데이트를 한다. 제어서버는 디바이스의 동작요청을 수신하여 실행하고 상태테이블에 실행된 결과를 반환하고 이벤트 서버는 서비스 상태가 변경될 때 변경되는 디바이스와 관련된 다른 디바이스들에게 알려주는 역할을 한다.

■ 제어 포인트

네트워크 내의 디바이스들을 검색하고 제어하는 컨트롤러이며 일반적으로 디바이스들을 제어할 수 있는 PC나 노트북을 예로 들 수 있다. 제어 포인트는 디바이스가 검색되면 1)디바이스 정보를 검색하여 관련된 서비스 목록을 확보하고 2)해당 디바이스와 관련되는 서비스의 정보를 검색하며 3)해당 디바이스의 서비스 제어를 실행하고 마지막으로 4)이벤트 서버에 등록한다.

(2) UPnP Protocol

■ UPnP용 네트워킹 매체

UPnP 네트워크상의 디바이스들은 RF, 유무선, 전화선등 모든 통신매체를 사용하여 연결하는 것이 가능하다. 그리고 TCP/IP, HTTP, XML과 같은 개방형 표준 프로토콜을 사용한다. UPnP는 IP프로토콜 기반의 네트워크를 구성하

Collection

므로 Non-IP프로토콜 기반의 다른 네트워크 기술들이 UPnP네트워크상에 존재하기 위해서는 UPnP Bridge와 Proxy를 통하여 네트워크에 활용이 가능하다. 그림 12는 브릿지 장비를 포함한 UPnP네트워크를 나내며 이와 관련된 많은 연구가 이루어지고 있다(Ferreira et al, 2013)(Cong et al, 2012).

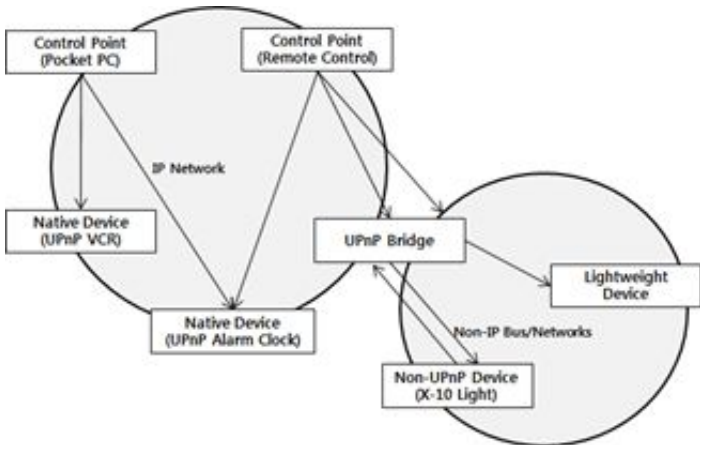

그림 12. Bridge를 포함한 UPnP네트워크

■ UPnP Protocol

UPnP는 표준 프로토콜을 사용함으로써 다양한 업체들이 제공하는 제품들 간의 상호 운영성을 보장한다. 그림 13은 UPnP프로토콜 스택이다.

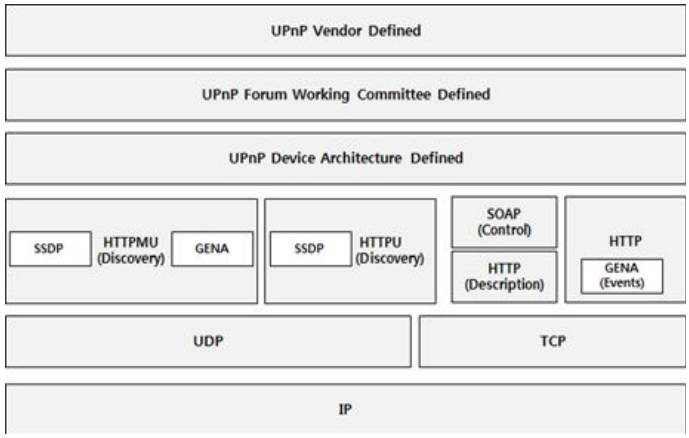

그림 13. UPnP 프로토콜 스택

○TCP/IP

TCP/IP 스택은 UPnP프로토콜을 구축하는 기반 역할을 한다. UPnP는 널리 사용되는 공용표준을 사용하여 디바이스들 간의 상호운용성을 보장한다.

○HTTP, HTTPU, HTTPMU

HTTP는 UPnP의 핵심적인 부분으로 HTTP에서 파생된 프로토콜들이 UPnP의 기반이 된다. HTTPU/HTTPMU는 HTTP의 파생 프로토콜로서 TCP레이어가 아닌 UDP레이어 위에서 메시지를 전송하기 위해 정의되었다.

○SSDP(Simple Service Discorvery Protocol)

SSDP는 HTTP, HTTPU, HTTPMU를 활용하여 네트워크상에서 디바이스와 서비스를 검색하는 방법 및 제어포인트가 네트워크상의 디바이스들에게 가용상태에 있음을 알리는 방법을 정의 한다(IETF, 2000) 디바이스와 제어포인트 모두가 SSDP를 사용하며 제어 포인트는 부팅이 되자마자 HTTPMU를 사용한SSDP를 통해 검색요청 메시지를 전송하여 네트워크에서 활용 가능한 디바이스 및 서비스를 검색한다. 디바이스는 검색요청을 수신하면 일치 여부를 확인하기 위해 검색조건을 점검한다. 일치하는 조건이 발견되면 디바이스의 정보를 HTTPU를 사용한 유니캐스트 SSDP 응답이 제어 포인트로 전송되한다. 또한 SSDP는 디바이스 및 서비스가 네트워크 연결을 원활하게 끊는 방법을 포함한다.

○GENA(Generic Event Notification Architecture)

GENA는 TCP/IP를 통한 HTTP 및 멀티캐스트 UDP를 사용하여 통보(notification)을 송수신하는 기능을 제공한다(IETF, 1999). 디바이스는 가용상태가 되면 GENA포맷을 사용하여 디바이스 정보를 생성하고 SSDP프로토콜을 통하여 전송한다. 또한 이벤트를 수신하여 서비스상태가 변경되었을 때 제어포인트로 알려주는 기능을 한다. 제어포인트는 연결되어있는 디바이스들의 상태를 주기적으로 갱신하고 GENA를 사용하여 연결 및 취소기능을 정의한다.

eCollection

○SOAP(Simple Object Access Protocol)

SOAP는 HTTP기반의 프로토콜들을 사용하여 XML기반의 메시지를 네트워크상에서 교환하는 프로토콜을 정의한 것이다(W3C, 2000). 웹서비스에서 기본적인 메시지를 전달하는 기반이 되며 다양한 형태의 메시지 패턴이 존재하지만 보통의 경우 원격 프로시져호출(Remote Procedure Call:RPC)패턴으로, 클라이언트에서 서버로 메시지를 요청하고 서버는 응답하는 Request/Response패턴을 사용한다. UPnP에서는 SOAP를 사용하여 실행할 동작 및 일련의 매개변수를 포함한 제어메시지를 디바이스들로 전송하며 그 결과 및 오류내용을 제어 포인트로 반환한다(W3C, 2000).

(3) UPnP 네트워킹 단계

■ Addressing

디바이스는 네트워크에 접속하면 주소를 할당받는다. 이때 DHCP서버를 검색하여 존재하면 주소가 자동 할당되며 존재하지 않을 경우 Auto IP를 사용한다. Auto IP는 169.254.16영역에서 주소를 할당하고 IP충돌이 일어나면 새로운 IP를 재할당하는 동작을 반복한다. 디바이스는 Auto IP로 IP가 할당되어도 네트워크상에 DHCP서버가 사용가능한지 주기적으로 계속 검사한다.

■ Discovery-Advertisement

디바이스에 주소가 할당되면 네트워크상의 제어포인트 및 다른 디바이스들에게 접속 상태를 표준형 멀티캐스트용 주소(239.255.255.250:1900 HPPTU)로 멀티캐스트한다. 상태알림 내용 중 타임스탬프를 포함시켜 디바이스의 유효기간을 알려주며 기간이 만료하기 전에 알림 내용을 재전송한다. 재전송하지 않으면 제어포인트는 디바이스가 유효하지 않다고 인식하며 디바이스가 오프라인 모드로 전환 시 오프라인 메시지도 반드시 전송한다. 그림 14는 Discovery-Advertisement에 사용되는 UPnP프로토콜 스택을 나타낸다.

Collection

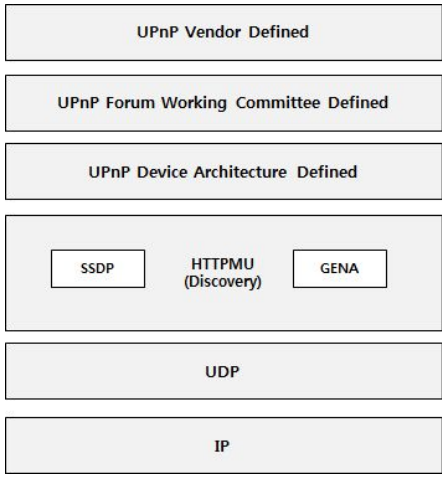

그림 14. Discovery-Advertisement에 사용되는 UPnP 프로토콜 스택

■ Discrvery-Search

제어포인트가 네트워크에 추가 되었거나, 디바이스 및 서비스를 검색하기위해서는 SSDP검색 메시지를 멀티캐스트하여 원하는 디바이스 및 서비스를 검색한다. 모든 디바이스는 메시지를 수신하고 검색조건이 일치하면 응답을 하며 응답은 SSDP헤더를 가진 유니캐스트 UDP를 사용하여 전송한다. 그림 15는 Discrvery-Search에 사용하는 UPnP프로토콜 스택이다

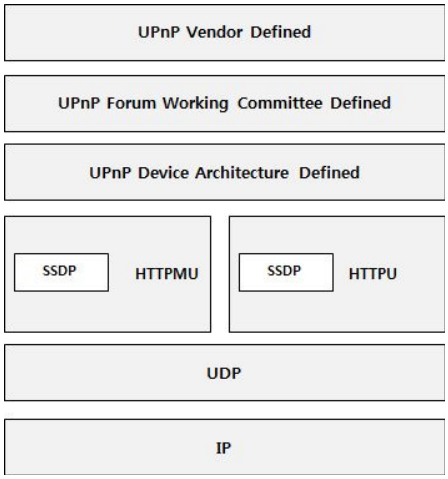

그림 15. Discrvery-Search에 사용하는 UPnP프로토콜 스택

eCollection

■ Description

Discrvery-Search요청을 하면 응답하는 디바이스는 자신의 정보를 저장해 놓은 XML파일의 위치를 URL에 포함하여 응답한다. 제어포인트는 URL을 통하여 HTTP GET요청을 수행하여 파일을 요청하고 디바이스는 해당 파일을 전송해준다. 디바이스 정보에는 서비스 설명도 동일하게 존재하며 HTTP GET요청을 통하여 서비스 검색도 가능하며 공급업체와 관련된 정보, 포함되어있는 디바이스에 대한 정의, 디바이스 공급 URL, 제공하는 서비스 내용과 제어 및 이벤트 URL등에 관한 정보를 포함한다. 그림 16은 Description에서 사용하는 UPnP프로토콜 스택이다.

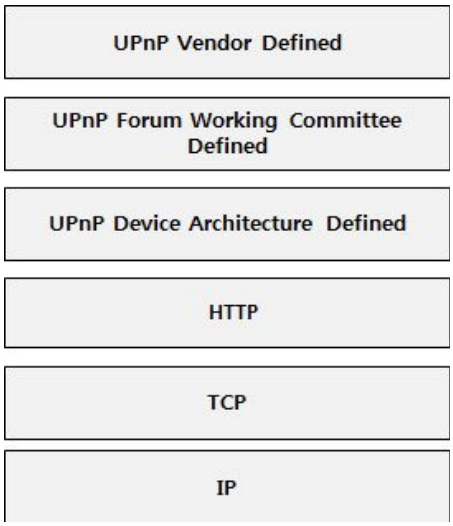

그림 16. Description에서 사용하는 UPnP프로토콜 스택

■ 제어(Contorol)

제어포인트는 제어 URL을 통하여 디바이스를 제어할 수 있다. 디바이스를 동작시키는 것도 일종의 원격프로시저 호출이다. 제어 메시지는 UPnP용 포맷으로 캡슐화 되어 SOAP를 사용하여 포맷된 후 HTTP를 사용하여 전송된다. 제어 요청을 받은 디바이스는 반드시 30초 이내에 응답해야 되며 제어포인트는 특정 서비스의 상태를 요청할 수 있다. 그림 17은 제어 단계에서 사용되는 UPnP의 프로토콜 스택이다.

Collection

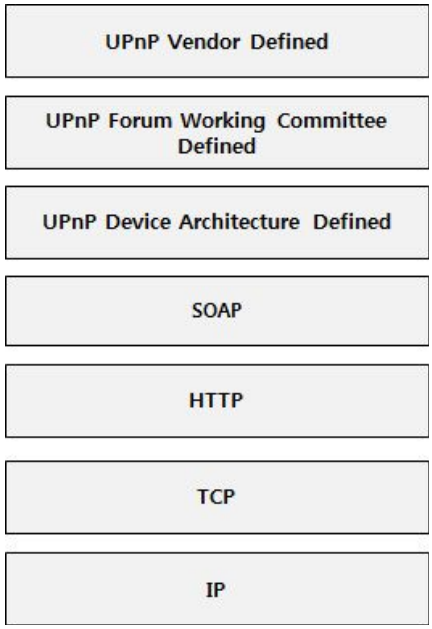

그림 16. 제어(Control)에서 사용하는 UPnP프로토콜 스택

■ Event

디바이스는 네트워크상의 제어포인트에게 자신의 변화 상태를 송신할 수 있다. 제어포인트는 디바이스의 이벤트 수신 취소가 가능하며 취소되면 디바이스는 이벤트 목록에서 해당 제어포인트를 삭제한다. 디바이스가 보내는 이벤트 메시지에는 URL, 상태 기간, 특정 변수 값 및 변수 이름이 포함되어 있으며 GENA를 사용하고 포맷 후 TCP/IP를 사용하여 전송한다. 그림 17을 Event 발생 시 사용되는 UPnP프로토콜 스택을 나타낸다.

Collection

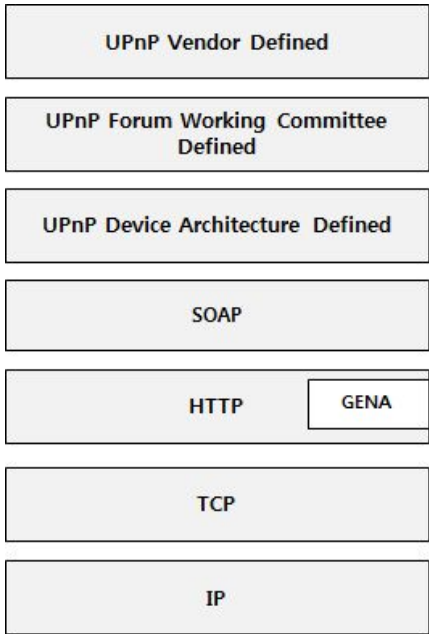

그림 16. 제어(Control)에서 사용하는 UPnP프로토콜 스택

■ Presentation

제어 포인트가 디바이스에서 제공하는 URL을 통해 브라우저와 같은 User Interface형태로 디바이스를 제어하거나 상태를 볼 수 있는 단계이다. 그림 17은 Presentation에서 사용되는 UPnP프로토콜 스택이다.

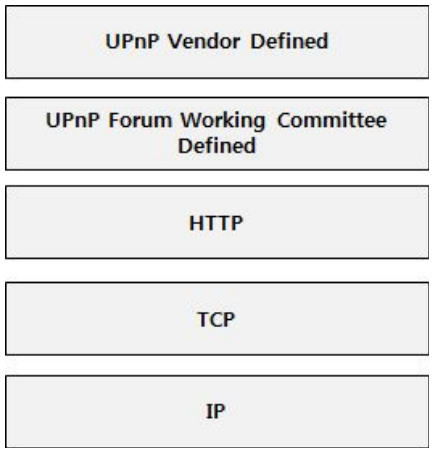

그림 17. Presentation에서 사용하는 UPnP프로토콜 스택

Collection

Ⅲ. 시스템 설계

본 논문에서는 센서네트워크를 위한 SPnP 시스템에 대해서 설계하고 구현하였다. SPnP를 구성하기 위해서는 데이터를 센싱하고 전송하는 센서디바이스, 센서디바이스와 게이트웨이사이 중계역할을 하는 Zigbee노드, 센서디바이스를 관리하는 게이트웨이로 구성되며 센서노드는 저 전력 특성을 위해 C언어 기반으로 구현하였으며 게이트웨이는 다양한 응용프로그램에 적용이 가능하기위해서 안드로이드 SDK(천인국, 2012)기반으로 자바 언어를 이용해 구현하였다. 그림 18을 제안된 SPnP 시스템 구성 및 구조를 나타낸 것이다.

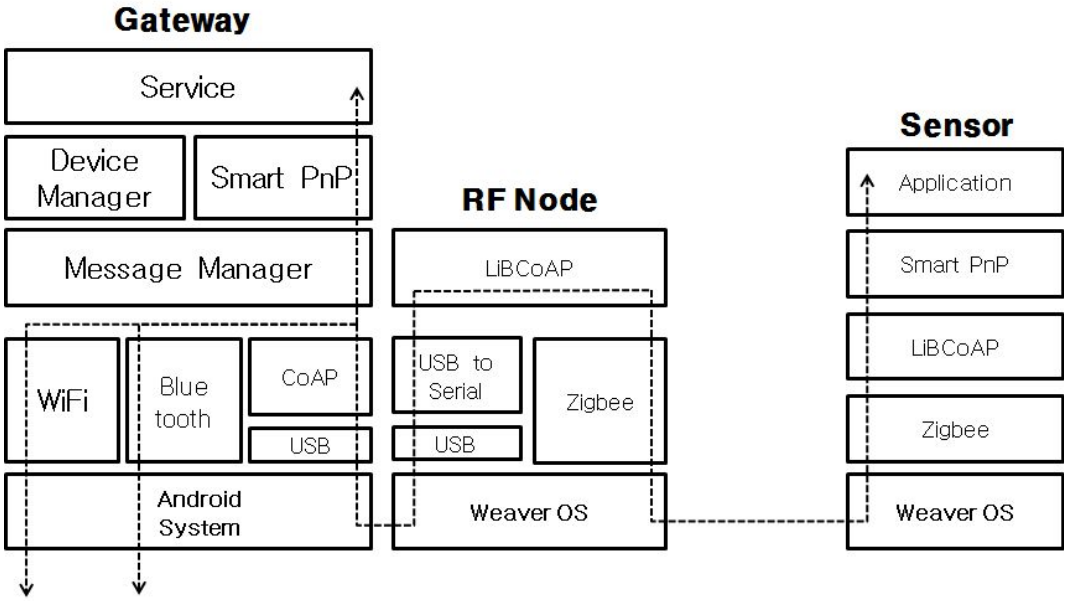

그림 18. 제안된 시스템 및 구조 및 구성

제안된 시스템의 게이트웨이에서는 센서노드 뿐만 아니라 Wifi, 블루투스 통신을 통하여 접속된 디바이스를 통합관리하며 라이브러리화를 통하여 사물인터넷 응용서비스 구현에 편의성을 추가하였다. 또한 센서노드의 SPnP상의 CoAP구현을 통해서 센서디바이스의 자동접속기술을 구현하였다.

Collection

1. SPnP를 위한 CoAP 구현

본 절에서는 Zigbee프로토콜 위에 CoAP 구현을 통하여 Non-IP프로토콜의 한계를 극복하고 확장된 SPnP CoAP 코드를 통해 게이트웨이에서 센서노드의 다양한 정보를 효율적으로 관리하는 법을 제시한다.

1) Zigbee over CoAP

본 시스템은 기존의 Non-IP프로토콜인 Zigbee프로토콜의 능동적인 인터넷 접속 및 개별적인 노드관리의 단점을 극복하기 위해 LibCoAP(익명)을 이용하여 Non-IP기반 프로토콜인 Zigbee상에서 구현하였다.

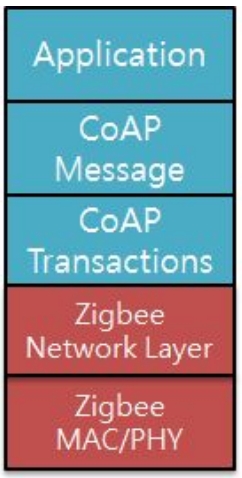

그림 19. Zigbee over CoAP스택

그림 19는 Zigbee상의 CoAP스택을 나타내며 기존의 IP계층과 UDP계층을 대신 Zigbee의 물리계층과 네트워크계층을 사용하여 네트워크 연결과 물리적인 통신은 Zigbee프로토콜이 담당한다. 본 논문의 CoAP은 Zigbee네트워크 계층 위에 CoAP헤더로 포맷변환을 수행한다.

게이트웨이에서는 Zigbee프로토콜로 수신하는 데이터를 인터넷에 연결하기

Collection

위해서 가상의 IP맵핑 테이블을 가지며 센서노드의 Zigbee주소와 게이트웨이에서 생성한 Auto-IP를 동시에 관리한다. Auto-IP는 169.254.16영역대의 IP를 할당한다.

2) CoAP 확장 코드

(1) SPnP 코드 정의

기존 CoAP코드에 확장된 코드를 추가하여 게이트웨이에서 능동적으로 센서노드를 관리할 수 있도록 구현하였다. 표 2는 제안된 시스템의 확장된 CoAP코드를 보여준다.

표 2. 확장된 CoAP 코드

분류	코드 번호	코드 타입
기존 코드	0.xx	Request
기존 코드	2.xx	Success
기존 코드	4.xx	Bad Request
기존 코드	5.xx	Server Error
제안된 확장 코드	7.xx	SPnP Code

기존코드의 사용하지 않는 7.xx대의 영역을 SPnP Code로 정의를 하며 이는 224에서 255까지의 상수를 가질 수 있다. 이중 7.01부터 7.15까지 SPnP 요청코드로 정의하며 7.16부터 7.31까지 SPnP응답코드로 정의한다. 표 3과 표 4는 제안한 시스템의 확장된 요청코드와 응답코드를 나타낸다.

Collection

표 3. 제안된 시스템의 요청코드

분류	요청 코드 번호	코드 상수	코드 타입
기존 코드	0.01	001	Get
기존 코드	0.02	002	Post
기존 코드	0.04	003	Put
기존 코드	0.05	004	Delete
제안된 확장 코드	7.01	224	Join
제안된 확장 코드	7.02	225	Event
제안된 확장 코드	7.03	227	Power
제안된 확장 코드	7.04	228	State
제안된 확장 코드	7.05	229	Broadcast

표 4. 제안된 시스템의 응답코드

요청 코드 번호	코드 상수	코드 타입
7.16	240	Join_Sucess
7.17	241	Join_Another
7.18	242	Event_Sucess
7.20	244	Power_Remain
7.21	245	State_OK
7.22	246	State_Busy

확장된 코드를 사용하여 게이트웨이는 능동적으로 센서노드의 다양한 정보를 요청하고 업데이트하여 응용프로그램에 최적화된 노드정보 및 메시지 관리를 할 수 있다. 또한 메시지 파싱과정에 확장된 코드를 감지할 경우 SPnP 메시지 처리모듈로 전달하여 처리하므로 사물인터넷 응용서비스를 구현하는 개발자에게 센서노드 상태 및 정보를 은닉하는 효과를 가져 온다.

(2) SPnP 코드 메시지 흐름

SPnP 코드는 게이트웨이에서 센서노드의 효율적인 관리를 목적으로 구현하였으므로 게이트웨이나 센서노드가 동작되면 바로 메시지를 전송한다. 그러므로 센서노드는 동작을 시작하면 주변에 게이트웨이와 연결되어있는 Zigbee노

Collection

드를 감지하며[]. 감지가 되면 게이트웨이에 접속요청 메시지를 전송한다. 게이트웨이에서 SPnP를 사용 중이면 요청메시지에 대한 응답이 전송이 되고 사용하지 않으면 일정시간이 지난 후 다른 게이트웨이를 검색한다. 주위에 다른 게이트웨이가 없을 경우 처음 감지된 Zigbee노드의 정보를 가지고 있으며 해당 Zigbee노드와 연결된 게이트웨이에서 SPnP를 사용할 때 까지 대기한다. 그림 20은 제안한 시스템의 기본적인 메시지 흐름을 보여준다.

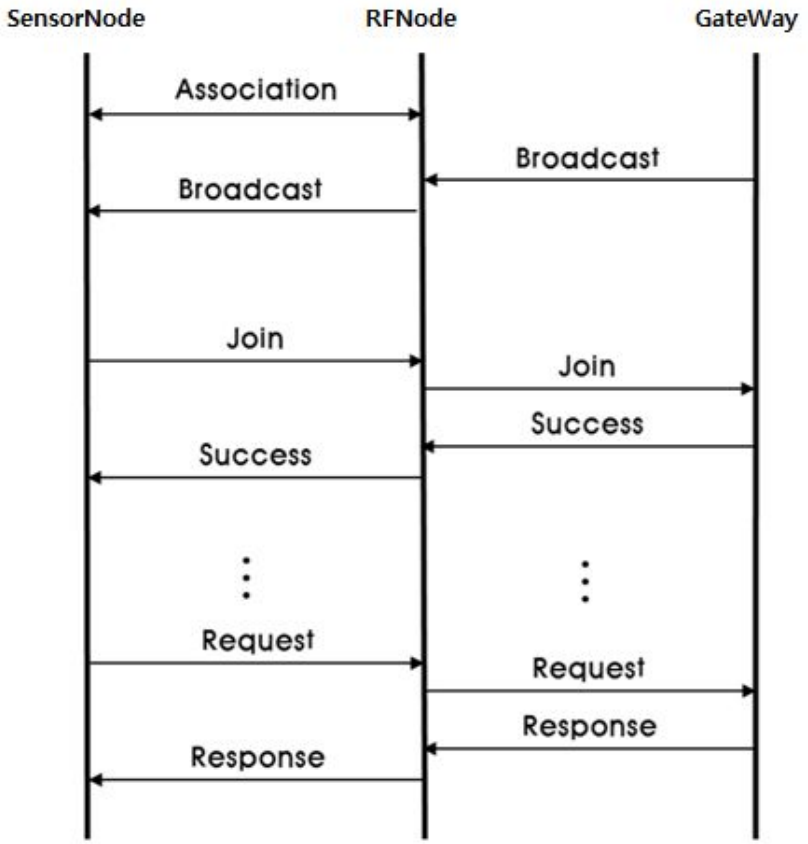

그림 20. 제안한 시스템의 메시지 흐름

3.2.1. 에서 정의한 확장된 코드는 게이트웨이와 센서노드의 상태에 따라 특정 과정을 수행하며 해당 코드의 메시지 흐름은 다음과 같다.

Collection

■ Join

처음 센서노드가 동작할 때 게이트웨이에 최초로 전송하는 접속요청 메시지다. 접속요청과 동시에 센서노드의 정보를 포함하여 전송하고 게이트웨이는 해당 코드를 수신하면 센서노드의 정보를 확인하여 센싱되는 데이터가 수집해야할 데이터인지 다른 센서노드 직접 통신을 해야 하는 역할을 하는지 판단한다. 수집되는 데이터가 필요한 경우 게이트웨이는 해당 센서노드를 등록하며 Join_Sucess 응답메시지를 전송하며 메시지를 수신 한 센서노드는 자신의 상태를 Join으로 변경한다. 직접통신을 할 경우 Join_Another 응답메시지를 해당 센서노드의 주소를 포함하여 전송한다. 센서노드 간 직접통신은 제안된 시스템의 CoAP코드를 사용하지 않고 기존의 Zigbee통신으로 통신을 수행한다.

■ Event

Event코드는 SPnP코드중 유일하게 센서노드에서 전송하는 요청메시지이며 센서노드는 자신의 센싱 데이터 정보가 변경됨을 감지하면 게이트웨이에 센싱 정보가 변경되었음을 요청한다. 게이트웨이는 센서노드의 정보를 변경한 후 Event_Sucess 응답메시지를 전송한다.

■ Power

주기마다 센서노드의 배터리 잔량을 확인하는 Power요청 코드는 센서노드는 해당 요청코드를 수신하면 배터리 잔량 90%이상 / 60%이상 / 30%이상 / 30%미만으로 확인 후 Power_Remain 응답메시지에 포함하여 게이트웨이에 전송한다. 게이트웨이는 배터리잔량이 30%미만의 센서노드를 별도로 관리하며 시간당 해당요청메시지의 전송횟수를 증가시킨다.

■ State

주기마다 센서노드의 상태를 확인하는 코드로 센서노드는 메시지를 수신하면 현재 데이터 센싱여부를 판단한다. 데이터 센싱 중이면 State_Busy 응답메시지를 전송하며 대기 중일 경우 State_OK 응답메시지를 전송한다.

Collection

■ Braodcast

게이트웨이가 SPnP를 실행 할 때 발생하는 요청코드로 주위의 모든 센서노드들에게 데이터수신이 가능함을 멀티캐스트를 통해 알린다. 주위의 모든 센서노드는 멀티캐스트를 감지하고 자신의 현재 접속 상태를 확인한다. Join상태가 아니면 센서노드는 게이트웨이에 Join요청코드를 전송하며 자신의 상태가 Join이면 별도의 응답메시지를 전송하지 않고 게이트웨이는 해당 센서노드의 존재를 알 수 없게 된다.

Collection

2. 제안한 시스템 구성

1) 센서노드

센서 노드는 다양한 데이터를 센싱하여 게이트웨이 및 다른 센서노드와 통신을 수행하며 그림 21은 제안한 시스템의 센서노드 시스템 구성을 나타낸다.

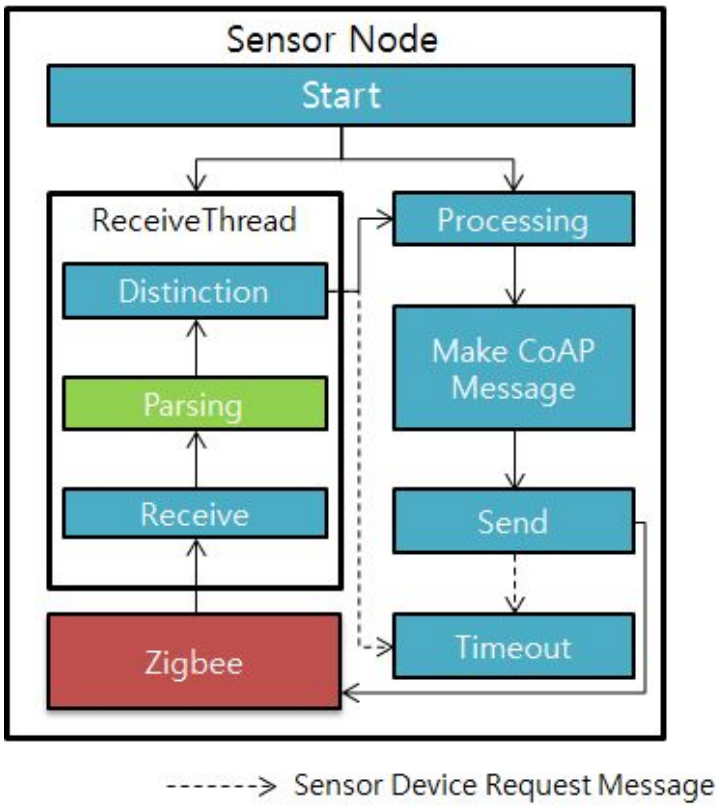

그림 21. 센서노드의 시스템 구성

제안한 시스템의 센서노드는 CoAP메시지로 데이터를 송수신하는 역할을 하며 응용프로그램이 제안한 시스템을 실행하면 게이트웨이 혹은 다른 센서노드에서 데이터를 수신하기위한 스레드가 동작된다. 메시지 송수신 과정은 다음과 같다.

Collection

(1) 메시지 송신

센서노드는 메시지를 송신하기 위해서 전송할 데이터를 CoAP포맷으로 변환한다. 다음 자신과 연결되어 있는 게이트웨이로 CoAP메시지를 전송한다. 센서노드의 요청메시지일 경우 동시에 타이머를 실행하여 일정시간 응답이 없을 경우 게이트웨이가 동작하지 않음을 인지하고 주위의 다른 게이트웨이를 검색하며 응답메시지일 경우 타이머를 동작하지 않고 메시지만 전송한다.

(2) 메시지 수신

센서노드는 메시지를 수신하면 CoAP포맷을 파싱한 다음 게이트웨이의 요청메시지인지 응답메시지인지 확인한다. 요청메시지일 경우 원하는 정보를 다시 전송하며 응답메시지일 경우 타이머를 중지시키고 응답메시지에 대한 처리를 한 다음 대기상태로 전환한다.

2) Zigbee 노드

안드로이드 제공하는 무선 통신은 Wifi와 블루투스를 제공하고 제안된 시스템에서 사용하는 802.15.4기반의 Zigbee통신을 제공하지 않는다. 따라서 제안된 시스템은 센서노드와의 원활한 통신을 위해서 게이트웨이에 USB인터페이스를 통하여 센서노드(Zigbee노드)를 연결하고 시리얼통신을 통하여 Zigbee노드에서 수신한 메시지를 게이트웨이에서 확인할 수 있도록 설계 하였다. Zigbee노드는 단순 메시지 전송의 중간역할을 할뿐만 아니라 게이트웨이에서 센서노드로 전송하는 메시지들을 CoAP프로토콜 포맷으로 변환하는 작업도 수행한다.

Collection

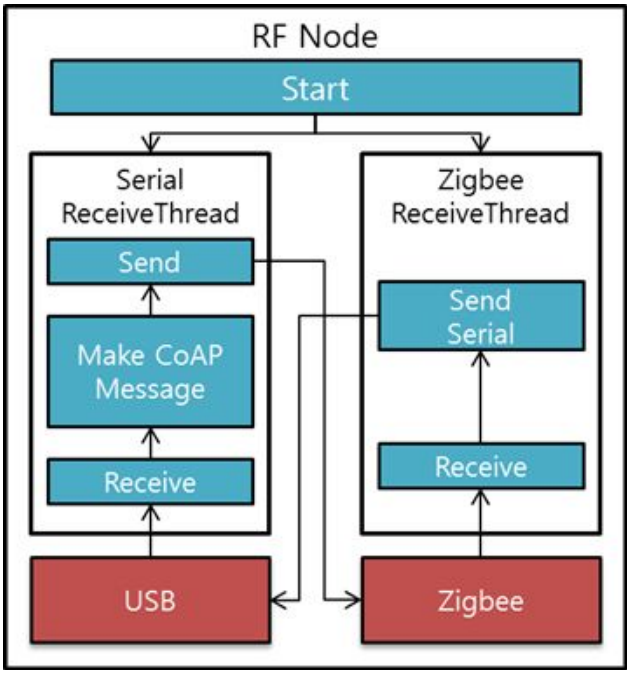

그림 22. Zigbee노드의 시스템 구성

그림 22은 Zigbee노드의 시스템 구성을 나타낸다. 두 개의 송수신 스레드를 수행하며 바이트단위로 마샬링하여 게이트웨이와 USB를 통한 시리얼 통신을 수행한다.

3) 게이트웨이

제안한 시스템에서 게이트웨이는 다수의 센서노드 및 다른 디바이스를 관리하게 되며 데이터 송수신모듈, 메시지 처리모듈, 디바이스 관리 모듈, SPnP처리 모듈로 구현하였으며 전체 구성은 그림 23과 같다. 게이트웨이는 C언어 기반의 센서노드와 Zigbee모듈과 달리 자바언어로 구현한다.

Collection

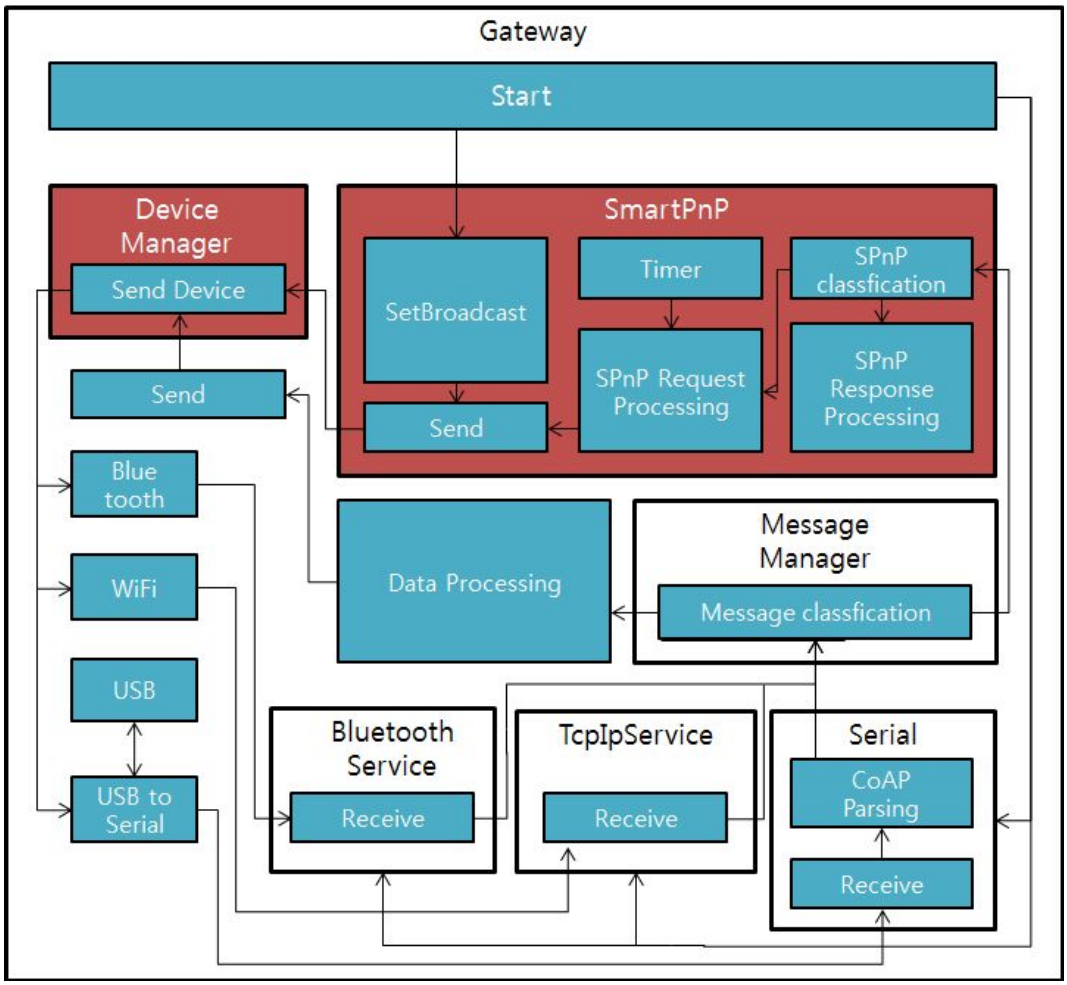

그림 23. 게이트웨이 시스템 구성

(1) 데이터 송수신모듈

게이트웨이는 Zigbee노드와 USB통신을 할 뿐만 아니라 Wifi, 블루투스 통신도 같이 수행한다. 각 통신마다 별도의 백그라운드 서비스로 동작하며 Wifi와 블루투스의 경우 서버소켓을 생성하여 다른 기기들과 통신하며 Zigbee노드와 통신을 위한 USB는 Zigbee노드와 USB를 통한 시리얼 통신을 하기위하여 변환작업이 필요하다[]. 변환 후 수신된 CoAP메시지 형태의 데이터를 파싱하여 메시지 처리 모듈로 전송한다.

Collection

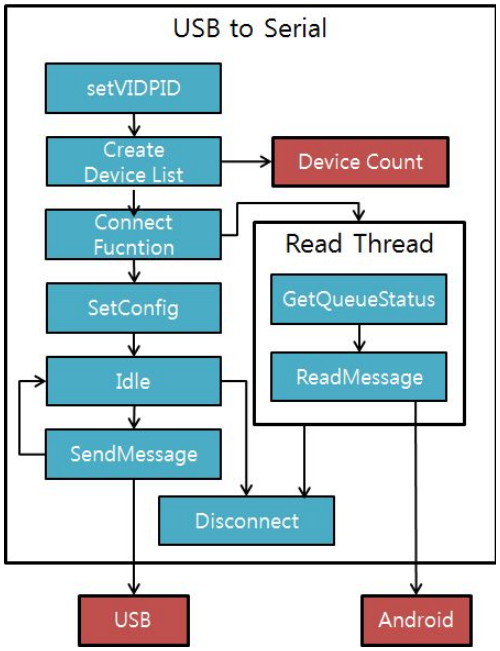

그림 24. USB to Serial흐름도

그림 24는 게이트웨이에서 USB인터페이스로 시리얼 통신을 하기위한 흐름도로 최초 디바이스에 대한 정보를 저장하기위해 변수들을 설정하고 이를 통해 연결된 Zigbee노드의 디바이스 드라이버를 조정하고 Baurate와 패리티비트 등 시리얼 통신 값들을 결정한다. 이후 연결된 디바이스 목록을 가져와 현재 연결된 디바이스를 확인한다. USB에 연결된 디바이스 수는 Device Count에 저장되어 관리한다. 목록에서 원하는 디바이스와 연결하고 연결된 디바이스의 정보는 별도로 관리한다. 제대로 연결이 되면 읽기 스레드를 실행하여 USB에서 전송하는 데이터에 대한 읽기 연산을 수행한다. 이후 SetConfig에서 시리얼 포트에 대한 연결 옵션을 선택한 후 대기상태로 메시지 송신요청이 오기를 기다린다. 메시지 읽기 스레드는 먼저 USB메시지 큐에 메시지가 들어와 있는지 확인한 후 큐에 저장되어있는 메시지 길이만큼 메시지를 수신한다. 이후 게이트웨이나 Zigbee모듈이 동작 불능 상태가 되면 Disconnect를 통해 리소스를 해제한다.

Collection

(2) 메시지 처리 모듈

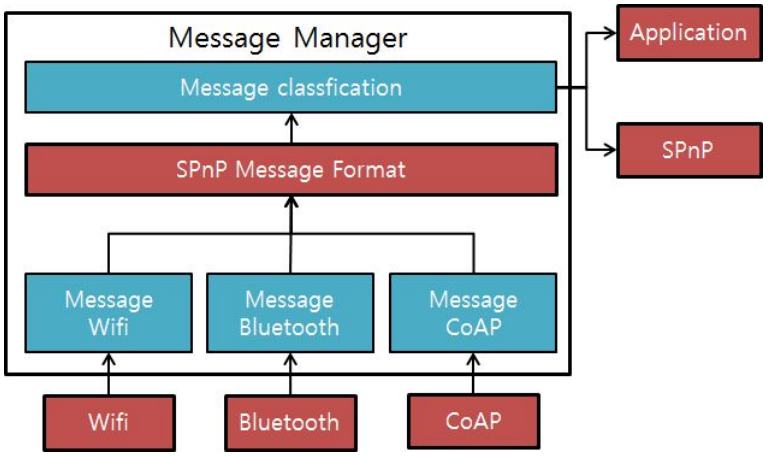

그림 25. 메시지 처리모듈 흐름도

메시지 처리 모듈은 데이터 송수신 모듈에서 받은 데이터를 분류 및 통합하는 작업을 수행하며 그림 25와 같다. Wifi, 블루투스, CoAP으로 수신된 메시지는 각각의 통신에 맞는 객체로 생성하여 데이터를 1차 저장하고 응용프로그램과 SPnP처리모듈에서 사용의 편의성을 위해 SPnP메시지 포맷으로 변환한다. 메시지 포맷에는 수신된 데이터와 통신방식이 저장되어 있다. 변환된 메시지는 SPnP code에 의해 SPnP모듈에서 필요한 것인지 응용프로그램에서 필요한 것인지 파악 후 필요한 곳에 메시지를 전달한다.

(3) SPnP 처리모듈

메시지 처리모듈에서 SPnP 처리모듈로 전달된 메시지는 요청코드와 응답코드로 분류한다. 요청코드면 타이머를 실행하고 요청코드를 정의한 후 디바이스 관리모듈로 전달한다. 센서노드에서 보낸 응답코드면 요청코드 전송 시 실행했던 타이머를 중지하고 해당 응답코드에 대한 처리를 하고 디바이스 정보가 변경될 경우 디바이스 관리모듈로 디바이스 상태 변경정보를 전달한다. 응용프로그램에서 SPnP를 처음 실행하면 Broadcast를 위해 메시지를 정리한다.

Collection

그림 26은 SPnP처리 모듈의 흐름도를 나타낸다.

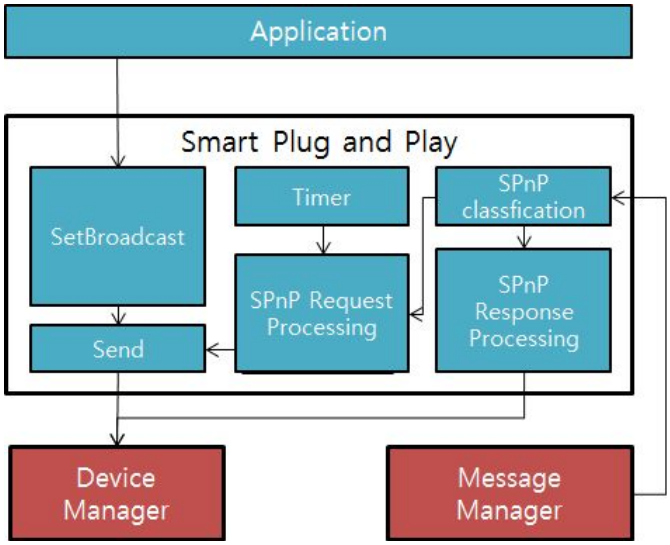

그림 26. SPnP 처리모듈 흐름도

(4) 디바이스 관리 모듈

접속된 디바이스들을 관리하는 작업을 수행하며 게이트웨이와 접속한 모든 디바이스들은 통신방식의 여부와 관계없이 Device객체로 추상화하여 관리한다. Device객체 담긴 정보는 표 5와 같다.

표 5. 디바이스관리 모듈의 저장변수

변수명	저장되는 정보
int type	통신방식
String name	이름
int state	상태
int power	전원 잔량
int sensing _data	센싱하는 데이터
int addr	주소

Collection

sensig_data변수는 Wifi, 블루투스 통신을 하는 다른 디바이스의 경우 사용하지 않게된다. 디바이스 주소의 경우 센서노드는 위한 IP가 아닌 Zigbee주소를 저장한다.

응용프로그램이네 SPnP 처리모듈에서 메시지 전송을 요청하면 디바이스 관리모듈은 송신할 메시지의 목적지를 확인한다. 목적지가 확인이 가능할 경우 저장된 주소로 메시지를 전송하고 확인이 안 될 경우 가장 최근 수신한 메시지를 보낸 디바이스의 주소로 메시지를 전송한다.

Collection

3. Smart PnP Library(SPL)

제안한 시스템의 목적을 위해 시스템의 자바 라이브러리화(Russel Bateman, 2012)를 통해 사물인터넷의 응용서비스 구현에 필요한 다양한 기능들을 제공한다. 응용프로그램에서는 SPnP를 세팅하고 실행하게 되면 원하는 데이터만 수집할 수 있고 필요하지 않는 정보 및 내용은 제안된 시스템에서 능동적으로 처리가 가능하다. SPL을 통해 응용프로그램에 제공되는 주요 기능은 다음과 같다

■ setSPnP

SPnP 사용을 위한 세팅을 하며 게이트웨이에 접속하는 디바이스의 목록의은닉 여부를 결정한다. 또한 상태확인 요청메시지나 전원잔량 요청메시지의 송신주기를 결정하며 정의하지 않을 경우 정의된 기본 값을 사용한다. 이를 통해 응용프로그램은 자신이 원하는 데이터만 전달 받을 수 있다.

■ getDeviceList

현재 게이트웨이에 접속되어있는 디바이스 목록을 확인할 수 있으며 디바이스 목록을 은닉했을 경우 확인이 불가능하다.

■ getState / getPower

응용프로그램은 Device객체의 이름을 통해 디바이스를 확인할 수 있고 상태를 확인할 수 있다. 또한 응용프로그램은 특정 디바이스에 대해 현재 상태나 전원잔량을 요구할 수 있으며 이는 SPnP에서 동작하는 주기적인 요청메시지와는 별도로 동작한다.

■ getMessage / sendMessage / sendAllMessage

메시지 송수신을 하는 기능으로 디바이스 목록 은닉상태면 최근 수신한 디바이스로 송신가능하며 확인상태면 원하는 디바이스에 전송이 가능하다. senAllMessage는 브로드캐스트를 통해 현재 접속되어있는 모든 디바이스들에게 메시지를 전송하는 역할을 한다.

Collection

Ⅳ. 구현 및 성능 평가

본 논문에서 설계한 Smart Plug and Play시스템에 대해 구현한 후 테스트를 위해 하드웨어를 구성하고 메시지 송수신을 통한 전송 성공률을 비교 및 분석한다.

1. 하드웨어 환경설정

제안한 시스템의 하드웨어 환경설정은 그림 27과 같으며 자세한 사양은 센서노드는 표 6, 게이트웨이는 표 7과 같다. 표6의 센서모듈은 센서노드에만 부착하며 데이터 센싱이 필요 없는 Zigbee노드는 부착하지 않는다. 센서노드와 Zigbee모듈은 본 논문의 제안된 통신을 수행한다. 게이트웨이는 안드로이드 4.0(android, 2011) 기반의 운영체제를 사용하고 Zigbee노드와 USB인터페이스로 연결한다.

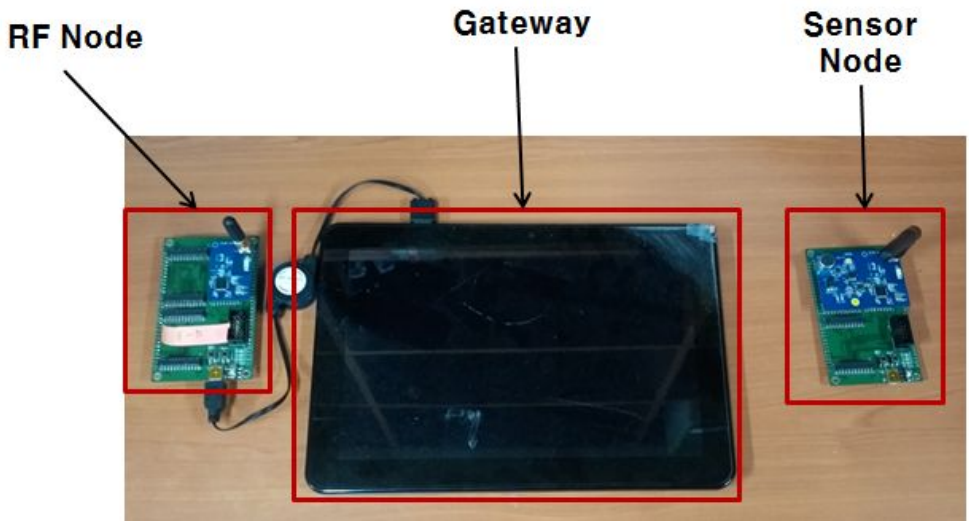

그림 27. 제안한 시스템의 하드웨어 환경

Collection

표 6. 센서 / Zigbee노드의 하드웨어 사양

분류	사양
Processor	Atmel Atmega 2561
Memory	SRAM 256Kbyte
Zigbee module	TI CC2420 Z
LED	Power, RED, GREEN, YELLO 4EA
Switch	Reset, Interrupt 2EA
Sensor module	TTCLM-CSM(temp, tilt, mic, photo, lux, LED)

표 7. 게이트웨이의 하드웨어 사양

분류	사양
Processor	Samsung Exynos4412 Prime Cortex-A9 Quad Core 1.6Ghz with 1MB L2 cache
Memory	2GB LP DDR2 880Mega data rate
Storage	16GB eMMC
Broadband	Cellular Data: HSPA+/WCDMA/GSM/GPRS/EDGE
WiFi	802.11 a/b/g/n dual band 2.4Ghz / 5Ghz
Bluetooth	2.1 + EDR

Collection

2. 시스템 구현 결과

1) SPnP 구현결과 분석

(1) 확장된 CoAP 구현결과 분석

기존의 시스템에서 확장된 CoAP코드를 적용한 사례(Jin Mitsugi et al, 2011)와 제안한 시스템의 확장된 CoAP코드를 비교분석하였으며 표 8은 기존의 시스템에서 확장된 CoAP코드를 적용한 사례와 제안한 시스템의 확장된 CoAP코드를 비교한 것이다.

표 8. 기존시스템과 제안한 시스템의 확장된 CoAP코드 비교

기존 시스템		제안한 시스템	
Join	등록	Join	등록
Leave	해제	Power	배터리확인 / 해제
Alive	동작확인	Event	센싱정보 변경
Ping	상태확인	State	상태확인
CollectAll	전체확인	Broadcast	전체확인 / 알람

Join은 기존의 시스템과 같이 게이트웨이에 등록하는 동일한 동작을 하며 기존 시스템의 Leave와 Alive코드는 게이트웨이의 상태를 물어보는 비슷한 역할을 하므로 State코드로 통합하고 센서노드의 센싱 상태를 추가하여 코드의 수를 줄이고 하나의 코드에서 더 많은 정보를 확인할 수 있도록 구성하였다. 센서노드를 해제하는 Leave코드는 센서노드의 특성상 배터리 방전 시 전송이 힘들다는 것을 확인하여 Power명령어로 대체하였고 배터리 잔량이 30% 이하일 경우 응답이 없을시 자동으로 해제하도록 하여 기존시스템에 비해 배터리 방전이 되기 전에 센서노드를 능동적으로 해제할 수 있는 기회가 늘어났음을 확인하였다. 또한 기존의 시스템은 센서노드의 on/off상태만 확인하는 코드로 정의 되어 있으나 제안한 시스템은 Event와 Power와 같이 센서노드의

Collection

배터리 잔량과 센싱 정보도 관리하므로 하나의 센서노드에 대해 더 많은 데이터를 관리함을 확인할 수 있다.

(2) SPnP 동작 분석

동일한 하드웨어 환경에서 2.2.2.을 적용해본 결과 128k 바이트의 메모리에 모두 적재가 안 되어 메모리 과부화 문제가 발생하여 시스템이 동작하지 않음을 확인했다. 따라서 3.1.2. 에서 정의한 제안한 시스템의 코드를 전송하고 메시지 전송 성공률을 확인하였다. 그림 28은 센서노드와 게이트웨이 사이의 Join메시지를 요청하고 응답하는 과정을 확인 한 것이다.

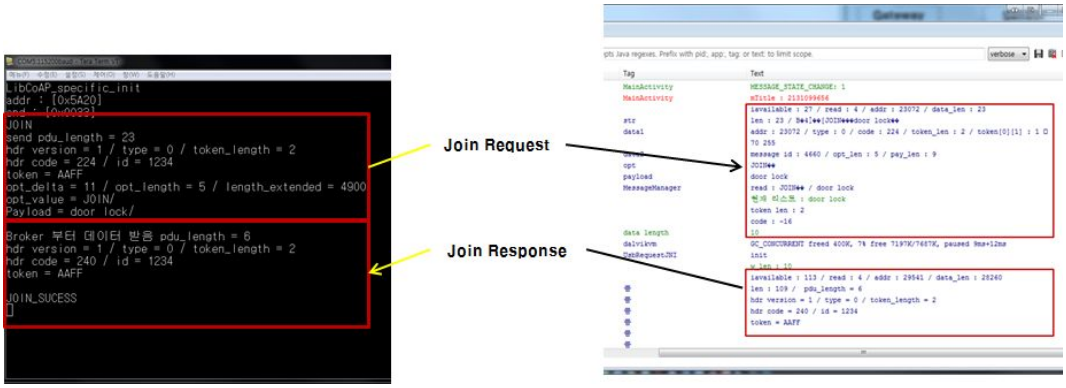

그림 28. Join 요청메시지에 대한 메시지 전송 및 응답확인

표 9. 제안한 시스템 메시지 전송 성공률 비교

요청 메시지	횟수	성공	성공률
Join	100	100	100%
Event(1)	100	100	100%
Event(2)	10	7	70%
Power	1000	998	99.8%
State	1000	1000	100%
Broadcast	10	10	100%

Collection

제안한 시스템의 메시지 전송확인과 성공률은 표 8과 같으며 게이트웨이의 요청메시지에 따른 센서노드의 응답 성공률은 전원잔량을 제외한 모든 메시지에서 100%전송 성공률을 보였으며 센서노드의 요청메시지인 Event메시지의 경우 센서 자체의 센싱정보를 변경할 경우 100% 요청 메시지가 전송되지만 센서모듈을 교체할 경우 70%의 비교적 낮은 성공률을 보였다.

.

2) SPL 구현결과 분석

제안한 시스템을 다양한 사물인터넷 응용 서비스에 적용 가능하게 하기위해서 자바 라이브러리화를 실시하였으며 이를 통해 응용프로그램은 SPnP를 실행하면 별도의 연결 설정 없이 센서노드 및 Wifi, 블루투스로 연결된 디바이스를 활용한다.

본 논문에서는 SPL성능을 테스트하기위해 그림 28 과 같이 간단한 디바이스 제어 응용프로그램을 제작하였다. 제작된 응용프로그램은 SPnP를 실행하면 디바이스가 자동으로 연결되고 목록을 업데이트하여 GUI를 통하여 접속된 디바이스를 확인할 수 있으며 해당 디바이스의 ON/ OFF기능을 활성화한다. ON/OFF제어 메시지를 100회 수행 한 결과 100% 수행성공률을 보였다.

그림 29. SPL 구현확인을 위한 응용프로그램

Collection

Ⅴ. 결론

본 논문에서는 Smart Plug and Play 시스템을 제안하고 이를 위해 Zigbee 프로토콜상의 CoAP 및 확장코드를 구현하고 센서노드, Zigbee노드, 게이트웨이를 구성하고 각 역할에 맞는 메시지처리 기능을 구현하였다. Zigbee프로토콜상의 CoAP은 Zigbee 네트워크계층 위에 CoAP메시지 처리 블럭을 구현하여 Non-
IP프로토콜의 한계점인 인터넷 및 웹 연결을 극복하였으며 다섯 가지의 확장된 코드를 정의하여 메시지를 송수신하면 게이트웨이에서 센서디바이스의 자동접속 및 능동적인 관리가 가능하다. 각 구성별 동작을 확인하면 센서노드는 제안한 시스템의 CoAP통신을 송수신하는 역할을 수행하며 이를 위해 CoAP 메시지 파서를 구현하였다. Zigbee노드는 게이트웨이와 센서노드사이의 메시지 중계 역할을 수행하며 센서노드와는 Zigbee통신을, 게이트웨이와는 USB인터페이스를 통한 시리얼 통신을 구현하였다. 마지막으로 게이트웨이는 크게 메시지 송수신 모듈, 메시지처리모듈, SPnP처리모듈, 디바이스관리모듈을 개발하였으며 SmartPnP Library로 라이브러리화를 통해 응용프로그램에서 활용할 수 있도록 구현하였다.

제안된 시스템의 성능평가를 위해 기존의 UPnP를 소형노드에 적용했으나 메모리 과부화로 동작하지 않음을 확인하였고, 제안한 시스템은 코드별로 70%에서 100%메시지 전송성공률을 나타내고, 따라서 제안된 시스템은 기존의 디바이스 관리 방법보다 경량화된 소형디바이스에 최적화된 모습을 보였다. 또한 게이트웨이에서 응용프로그램을 개발하여 SPL의 활용성을 확인하였고 디바이스 및 통신방식 은닉을 통해 응용프로그램에서 원하는 정보만 확인할 수 있음을 확인하였다.

향후 본 시스템을 활용하면 저 전력 특성의 소형가전제품부터 고성능의 대형가전제품까지 동시에 관리가 가능하고 응용프로그램 구현의 편의성을 제공하여 사물인터넷의 다양한 응용서비스가 개발되어 국내 사물인터넷 시장을 발전시키는 주요기술이 될 것으로 판단한다.

Collection

참고 문헌

김민식, 정원준 (2014). 사물인터넷(IoT)관련 가치사슬 및 시장 구성요소 현황. 정보통신방송정책 제 26권 8호 통권 576호. 22-27p.

고석갑, 박일균, 손승철, 이병탁 (2013). IETF CoAP 기반 센서 접속 프로토콜 기술 동향. 전자통신 동향분석 제28권 제6호. 133-140p.

미래창조과학부 (2013). 인터넷 신산업 육성방안

방송통신위원회 (2010). 10대 방송통신 미래서비스

장원규, 이성협 (2013). 국내외 사물인터넷 정책 및 시장동향과 주요 서비스 사례. 동향과 전망 : 방송·통신·전파. 통권 제24호. 24-38p.

전자신문 (2012). LG U+, 2012년 여수 세계박람회 LTE 차량관제 시스템 구축

전자신문 (2013). SKT 스마트 팜, 창조농업의 꿈을 스마트 팜으로 실현한다.

천인국 (2012). 그림으로 쉽게 설명하는 안드로이드 프로그래밍. 서울 : 생능출판사

KEIT (2012). IoT플랫폼 기술 동향 및 발전방향.

Hyungjoo Song, Daeyoung Kim, Kangwoo Lee, Jongwoo Sung (2005). UPnP-Based Sensor Network Management Architecture. ICMU. ICMU 2005, April 13 - 15, 2005 Osaka University Convention Center, Osaka, JAPAN

Android(2011). Android 4.0 Icecream sandwich. http://developer.android.com/

Dieter Uckelmann, Mark Harrision, Florian Michahelles (2011). Architecting the Internet of Things. Springer, Berlin

Hiro Gabriel Cerqueira Ferreira, Edna Dias Canedo, Rafael Timóteo de Sousa Junior (2013). IoT Architecture to Enable Intercommunication Through REST API and UPnP Using IP, ZigBee and Arduino. Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference

IDATE (2013). Internet of Things : Market report. Digiworld by IDATE

Internet Engineering Task Froce (1999). General Event Notification Architecture Base, draft-cohen-gena-p-base-01

Internet Engineering Task Froce (1999). Muticast and Unicast UDP HTTP Message.

Internet Engineering Task Froce (2000). Simple Service Discovery Protocol 1.0, draft-cai-ssdp-v1-03.

Internet Engineering Task Froce (2012). Constrained RESTful Environment (CoRE)

Collection

Link Format

Internet Engineering Task Froce, CoRE Working Group (2013). Constrained Application Protocol(CoAP), draft-ietf-core-coap-18

Jin Mitsugi, Shigeru Yonemura, Hisakazu Hada, Tatsuya Inaba (2010). Brdging UPnP and Zigbee with CoAP : protocol and its performance evaluation. IoTSP '11 Proceedings of the workshop on Internet of Things and Service Platforms

libcoap (2013). C-implementation of CoAP. http://sourceforge.net/projects/libcoap

Russel Bateman (2012). Jar libraries tutorial. http://www.javahotchocolate.com/tutorials/jar-libraries.html#jar

UPnP Forum (2008). UPnP device architecture 1.1. UPnP Forum

W3C (2000). Simple Object Access Protocol (SOAP) 1.1

Xiangchao Cong,Shi Jieqin, Ke He (2012). A Communication Network Based on IEEE1394 and UPnP Technology. Instrumentation and Control Technology (ISICT), 2012 8th IEEE International Symposium

Collection

Design of Smart Plug and Play System on wireless sensor network

No, Soon Young

School of Computer and Information Engineering
Graduate School, Daegu University
Gyeongbuk Korea

Supervised by prof. Kim, Chang Hoon

(Abstract)

In the network paradigm of anthropocentricity, as Internet of Things(IoT) which things participate as the subject is issued as convergence service of future, various policies for markets of IoT are propeled in US, Europe, Japan, and China , and according to this, it is expected that the market will be expanded. Main technology of IoT requires sensing, wired/wireless network, and service interface technology, especially sensing demands variety environment and high quality data sensing. For this, effective management and certain control of sensor nodes are necessary in the gateway which collects sensing date

In this paper propose the Smart Plug and Play(SPnP) to effective control and management to sensor nodes on the CoAP protocol. This system is consist of data sensing, sensor node for transmission, RF node for receive and gateway for processing received date. The communication among each components is based on the Zigbee protocol CoAP so that it overcome the weakness of Non-IP protocol. By using extension code, it processing the join, delete, state and request message for sensor node at gateway. Furthermore it also able to apply for other various applications by making library. The system showed the 70% ~ 100% transmit success in each code. Therefore this system is not only able to apply to small home appliances in low spec but also it provide convenience to development. That will bring vitalization in IoT application services development.

Collection