공학박사 학위논문

오픈소스를 활용한 사물인터넷 환경의 스마트 홈 시스템에 관한 연구

A Study on the Smart Home System under Internet of Things using Open Source

지도교수 류 길 수

2017년 2월

한국해양대학교 대학원

컴 퓨 터 공 학 과
성 창 규

본 논문을 성창규의 공학박사 학위논문으로 인준함.

위 원 장 공학박사 김 재 훈 (인)

위 원 공학박사 이 장 세 (인)

위 원 공학박사 하 윤 수 (인)

위 원 공학박사 이 기 욱 (인)

지도교수 공학박사 류 길 수 (인)

2016년 12월 27일

한국해양대학교 대학원

목 차

List of Tables

List of Figures

A Study on the Smart home system under Internet of Things using Open source

Chang-Gyu Seong

Department of Computer Engineering,

Graduate School of Korea Maritime and Ocean University

Abstract

Home service of IoT(Internet of Things) has been briskly provided, mainly by domestic mobile network operators and appliance companies using specific smart devices and the communications network of theirs. Drawbacks of this home service of IoT, such as having to use dedicated equipment and having to a flat fee monthly, do not satisfy the consumers' demands. These shortcomings of IoT home service is causing the appearance of users who try to design the environment of IoT that responds their demands and naturally, attempts to have home automation system through open-source hardware like Arduino. Open source offers end users the paradigm of ICT (Information and Communications Technologies) and DIY (Do-It-Yourself) that let them make their own thing that meets their requirements

and design a smart IoT environment through their thing. Anyone can construct an IoT service platform at a low cost applying the paradigm offered by open source. Thus, this thesis designs and implements an IoT-based smart home service system on the data and specification produced from various devices through the structure of IoT home service environment that consists of open-source IoT platform and IoT home gateway. The proposed platform uses Arduino and Raspberry Pi which are distributed as open-source hardware and forms a thing that is capable of interacting with surroundings by linking multiple sensors to the platform. It uses a lightweight, bidirectional messaging featured protocol, MQTT. Also, every piece except the gateway uses WiFi connection in order to resolve the inconvenience of wired environment and the difficulty of installment. Web Socket, the web standards for the next generation, is employed to be hooked up to web browser. An application that provides a stable, cost-efficient connectivity for the thing in IoT environment and build a broker that interwork with cloud service platform of IoT using Node-RED. Any information gathered is visualized through various ways such as web page, mobile page, social network, and e-mail. The effect of the system is seen through the implementation of smart home system in IoT environment based on the designed content and through the performance comparison to messaging processing using HTTP protocol. The suggested system is expected to be available in multiple areas such as homes, industries, and public services with the capability of satisfying various requirements of IoT environment using open-source platform and with the possibility to be extended to service-oriented application. More research on the system that is appropriate for customized service in every area with

performance and safety enhancement through various solutions of IoT this thesis does not cover is needed in the future.

KEY WORDS: Internet-of-Things; 사물인터넷; Open source platform; 오픈소스 플랫폼; Smart home; 스마트 홈; Message Queueing Telemetry Transport; MQTT;

제 1 장 서론

1.1 연구 배경 및 목적

최근 정보통신기술 분야에서는 사물인터넷(IoT : Internet of Things)에 대한 관심이 증대되고 있다. 사물인터넷의 목표는 사람이 직접 제어하지 않는 상태에서 인터넷으로 연결된 사물들이 스스로 정보를 주고받으며 상황에 맞는 유용한 서비스를 제공하는 것이다. 특히, 인간의 일상생활에서 가장 친밀한 공간인 가정에서 모든 사물이 인터넷에 연결되어 정보를 공유하는 환경은 생활 속의 편의와 위험요소에 대한 예측 등 인간 생활에 많은 변화를 일으키고 있다. 이에, 현재 통신 3사, 가전사업자, 플랫폼 사업자 등을 중심으로 사물인터넷을 사용한 홈 자동화 서비스가 제공되고 있다[1].

하지만 국내외 사업자의 사업현황 및 미래 비전을 살펴보면 사업자의 필요 및 역량에 치중되어 사용자의 요구사항을 충분히 파악하지 못해 만족도면에서 문제점을 보이고 있다. 또한 대부분의 서비스는 월정액과 같은 정기적인 요금을 지불해야 하고 특정 사업자 전용의 스마트 기기를 사용해야 하며, 기기의 종류나 기능이 사용자 요구수준에 비해 제한적이라는 등의 문제점을 갖고 있다[2].

이러한 문제점들로 인해 저렴한 제품을 사용하면서도 자신의 요구 사항에 맞는 사물인터넷 환경을 직접 설계하려는 사용자들이 생겨나고 있고, 아두이노와 같은 오픈소스 하드웨어를 통한 홈 자동화 시도 또한 증가하고 있다. 오픈소스 하드웨어는 전문가가 아니더라도 일반 사용자들이 자신의 요구사항에 맞춰 직접 자신만의 사물(thing)을 만들고 이를 통해 스마트 사물인터넷 환경을 설계하는 ICT(Information and Communications Technologies) DIY(do-it-yourself) 패러다임을 제공할 수 있어 누구라도 사물인터넷 서비스 플랫폼을 구축할 수 있게 해 준다[3].

여기에 리눅스 커널 및 관련 GNU 소프트웨어, 아파치 웹서버, 구글 크롬 웹

브라우저, MySQL 데이터베이스 시스템, python, node.js 언어, 이클립스 툴 등과 같은 사물인터넷 관련 오픈소스 소프트웨어도 수많은 개발자들에 의해 개발되고 있다. 오픈소스 소프트웨어는 소스코드가 공개되어 있어, 일반적으로 자유롭게 사용・복제・배포・수정할 수 있다. 특히, 상호 호환성을 가진 기기 통신을 지원하는 데이터 인프라와 효율성을 가진 사물인터넷 데이터에 대한 미들웨어의 경우 오픈소스 플랫폼과 기술이 적용되어야 하는 필수 솔루션이다[4]. 따라서 오픈소스 소프트웨어에 의한 사물인터넷 패러다임의 변화와 관련한 파급력은 엄청날 것이다.

이에, 본 논문에서는 오픈소스 하드웨어와 소프트웨어 플랫폼을 적용하여 스마트 홈 시스템을 설계하고 구현하는 것을 목적으로 한다. 현재까지, 사물인터넷 플랫폼에 대한 정의는 협의된 바 없으며, 다만 플랫폼의 기본적 역할에서부터 사물인터넷 플랫폼이 무엇이고, 어떠한 기능을 제공할 것인지, 어떤 제품으로 형상화될 것인지 정도만 확인할 수 있다. 즉, 사용자가 요구하는 서비스 제공을 위한 플랫폼 선정의 가이드라인이 전무한 상태인 것이다. 따라서 본 연구의 목적인 사물인터넷기반의 홈서비스를 구축하기 위해 첫째, 사물인터넷 오픈소스 플랫폼 분석을 통해 각 플랫폼의 기능과 특성을 정의해야 한다. 둘째, 스마트 홈서비스를 위한 요구 사항을 수집한다. 셋째, 스마트 홈서비스를 제공할 수 있는 오픈소스 플랫폼을 구성하고, 마지막으로 플랫폼을 이용한 시스템을 설계 및 구현한다.

1.2 연구 내용과 범위

일반적으로 스마트 홈 시스템 설계 과정에서 고려되어야 할 기본적 특징은 다음과 같다.

- 실시간성(real-time) : 사물에서 발생되는 데이터는 실시간으로 수집되고 처리되어서 시간과 장소에 상관없이 언제 어디서나 서비스되어야 한다.
- 양방향성(full-duplex) : 사물과 사물 또는 사물과 인간은 단방향의 통신이 아니라 서로 정보를 주고받을 수 있는 양방향 통신이어야 한다.

- 저전력 및 낮은 대역폭(low power & low bandwidth) : 일부 사물인터넷 기기는 전기 시스템에서 전력을 공급받을 것이지만 장소에 따라 배터리를 사용하게 될 것이므로 통신과정에서 저전력과 낮은 대역폭의 무선 통신을 할 수 있는 통신 프로토콜을 사용해야 한다.

- 다양한 사용자 인터페이스 (user interface) : 사용자는 언제 어디서나 접속 기기에 상관없이 다양한 방법으로 정보를 제공받을 수 있어야 한다.

- 확장성(scalability) : 설치가 어려운 장소인 경우 무선 통신으로 사물을 쉽게 배치하고 위치를 변경할 수 있어야 한다.

- 쉬운 개발(easy development) : 서비스를 새롭게 구축하거나 확장하는 경우 쉽고, 빠르게 개발이 가능하여야 한다.

- 저비용(low-cost) : 사물인터넷 서비스를 구축하고 운영하는데 소요되는 비용이 저렴해야 한다. 이를 위해서는 하드웨어, 소프트웨어의 비용과 개발기간 및 노력이 최대한 적게 소요되어야 한다.

본 논문은 언급한 기본 특징을 가지는 오픈소스 플랫폼을 선별하여 우리가 거주하는 집에서 생산되는 다양한 데이터를 효율적으로 중계 및 소비할 수 있는 사물인터넷 기반 스마트 홈 시스템의 제안 및 개발을 목적으로 한다. 사물인터넷 플랫폼은 서비스 제공자와 소비자를 무엇으로 보느냐에 따라 다양한 정의가 가능하다. 사물인터넷에서 핵심 기술은 사물과 사물, 사물과 서버, 서버와 인간의 연결을 담당하는 양방향 통신, 네트워크 기술과 사물 구축 및 서비스를 제공하기 위한 응용과 인터페이스 기술을 통합하는 플랫폼 제공이라 할 수 있다[5]. 그러므로 오픈소스 사물인터넷 기반의 스마트 홈서비스 환경에서는 데이터의 생산, 소비 및 표현의 관점으로 우리가 거주하는 집에서 생산되는 다양한 데이터를 효율적으로 중계 및 소비할 수 있어야 한다.

이에 본 논문에서는 사물인터넷 환경의 스마트 홈 시스템의 환경구조와 가정 내 기기들의 경제적이고 안정적인 연결과 생성되는 정보의 중계 및 소비를 지원하는 오픈소스 중심의 플랫폼을 제안한다. 제안하는 플랫폼은 오픈소스 하드웨어 플랫폼으로는 아두이노(Arduino)와 라즈베리 파이(Raspberry Pi)를 사용하

고 다양한 센서를 플랫폼에 연결함으로써 주변 환경과 상호 작용이 가능한 사물을 구성한다. 사물간의 통신 프로토콜로는 국제표준협력체 oneM2M이 권장하는 Pub/Sub 모델의 가벼우면서 양방향성 메시징 특성을 가진 MQTT(Message Queueing Telemetry Transport) 프로토콜을 이용한다. 또한, 유선 환경의 불편함과 설치상의 어려움을 제거하기 위해 게이트웨이를 제외한 모든 사물들은 와이파이(WiFi, IEEE 802.11b High Rate) 통신만으로 연결하고, 웹브라우저와 연결을 위해서는 차세대 웹 표준인 웹 소켓(Web Socket)을 이용하도록 구성하여 수집된 정보는 웹 페이지, 모바일 페이지, 소셜네트워크, 이메일 등 다양한 방법을 통해 제공된다.

본 논문의 구성은 다음과 같다.

2장에서는 사물인터넷 기술의 종류와 동향, 오픈소스 하드웨어 플랫폼, 오픈소스 소프트웨어의 종류와 특징과 관련한 문헌들을 살펴보고 스마트 홈서비스를 위한 플랫폼 설계 모델에 연구 내용을 분석하여 제안 시스템 설계에 활용한다.

3장에서는 사물인터넷 환경의 스마트 홈 시스템을 위한 오픈소스 플랫폼 구성을 제안하고 서비스 구조 및 요구사항들에 대해 정의한다.

4장에서는 제안 내용을 적용하여 스마트 홈서비스 시스템을 구현하고, 서비스 결과를 HTTP 프로토콜을 이용한 경우와 성능 비교하여 그 성능에 대해 검증한다.

마지막으로 5장에서는 본 연구에 대한 결론 및 향후 연구 방향에 대하여 기술한다.

제 2 장 관련 연구

2.1 사물인터넷 기술

사물인터넷은 최근 갑자기 등장한 기술이 아니라 오래 전부터 정보통신기술의 발전과 함께 진화되어 왔다. 정보통신 기술의 발전에 따라 점차 기술과 개념이 변화되었다[6]. RFID, USN(Ubiquitous Sensor Networks), M2M등이 그 대표적인 개념들이다. 사물인터넷의 개념은 1999년 매사추세츠공과대학(MIT)의 오토아이디 센터(Auto-ID Center)의 케빈 애시턴(Kevin Ashton)이 RFID와 여러 센서 등을 활용하여 사물에 탑재된 인터넷이 발달할 것이라 예측한 것에서 비롯되었다[7]. 사물인터넷은 표 2.1과 같이 다양하게 정의된다. 국제전기통신연합(ITU : International Telecommunication Union)은 기존의 통신환경에서 사람과 사물, 사물과 사물간의 통신이 새로운 통신 유형의 등장을 사물인터넷으로 정의하였고, 유럽통신표준기구(ETSI : European Telecommunications Standards Institute)는 인간의 직접적인 개입이 꼭 필요하지 않은 둘 혹은 그 이상의 객체 간에 일어나는 통신이라는 M2M의 개념으로 사물인터넷을 정의하고 있다.

사물인터넷 기술은 무선센서 네트워크(WSN : Wireless Sensor Network)와 사물통신의 다양한 기술들이 융합되면서 진화하고 있다. 무선센서 네트워크와 사물통신 및 사물인터넷 기술들은 센싱과 피드백을 통하여 인간의 개입을 최소화하거나 개입 없이 사물 간에 정보를 수집하고, 수집된 정보를 융합하여 인간에게 지식과 서비스를 제공하여 편리한 삶을 누릴 수 있게 할 수 있으므로 다양한 분야에서 사용자 서비스 요구가 계속적으로 증가할 것이다[8-9].

사물인터넷은 모든 사물을 통신기술을 통해 상호 연결함으로써 다양한 정보를 공유하고, 그 정보를 바탕으로 새로운 서비스를 구현할 수 있다. 사물인터넷 환경에서 서비스의 형태와 규모는 데이터 수집에 사용되는 센서의 수량과 종류의 다양성에 의해서 달라질 수 있어, 대규모의 센서를 통하여 수집된 정보를 효율적으로 관리하기 위한 다양한 방법들이 연구되고 있다[10]. 대규모의 센서를 이용한 사물인터넷 환경은 시스템의 운용 효율성 등을 위하여 센서와 서버

사이에서 센서로부터 전송된 데이터를 수신하고 서버로 중계하는 사물인터넷 게이트웨이가 필요하다. 사물인터넷의 기본적인 특성으로 인해 실시간으로 발생하는 대량의 데이터를 수집하여 처리할 수 있는 효율적인 방안도 고려되어야 한다[11-12].

표 2.1 주요 표준화 기구의 사물인터넷 정의
Table 2.1 Internet of things concept by major institutions

기 관	정 의
유럽통신표준기구(ETSI) European Telecommunications Standards Institute	M2M : 사람의 개입 없이 사물간 통신이 이루어지는 것 Communication between two or more entities that do not necessarily need any direct human intervention
전기전자기술자협회(IEEE) Institute of Electrical and Electronics Engineers	M2M : 사람의 개입 없이 사물에 연결된 통신국 사이에서 정보가 교환되는 것 Information exchange between a Subscriber station and a Server in the core network(through a base station) or between Subscriber station, which may be carried out without any human interaction
국제전기통신연합(ITU) International Telecommunication Union	IoT : 통신기술을 바탕으로 물리적 가상적 사물들이 상호 통신을 하는 것 A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on, existing and evolving, interoperable information and communication technologies
국제인터넷표준화기구(IETF) Internet Engineering Task Force	IoT : 표준 통신규약에 따라 통신접속 주소를 가진 사물들 사이에서 이루어지는 통신 A world-wide network of interconnected objects uniquely addressable, based on standard communication protocols

출처 : 박재현, 임정선, M2M 시장 현황과 국내외 성장 전망, KT경제경영연구소 DIGIECO, 2013[13]

사물인터넷은 지능화된 사물들이 인터넷에 연결되어 네트워크를 통해 사람과 사물, 사물과 사물간에 상호 소통하고 상황인식 기반의 지식이 결합되어 지능적인 서비스를 제공하는 글로벌 인프라이며, 스마트 디바이스, 클라우드, 빅데이터 기술 등과 융합하여 개방과 공유를 지향하는 초연결사회의 핵심이 될 전망이다. 2020년에는 인터넷에 연결되는 사물의 수가 약 260억 개까지 증가할 것으로 전망되며(PC, 태블릿, 스마트폰 제외), 약 3,000억 달러의 시장(서비스, 제품) 창출과 1.9조 달러의 경제적 파급효과가 기대되며, 그 중에서 제조와 헬스케어 분야의 파급효과가 가장 클 것이다[14].

사물인터넷 기술은 디바이스, 네트워크, 플랫폼, 분석/소셜 비즈니스 및 응용산업 서비스로 구분되며, 사물에 부착 또는 내장되는 사물인터넷 디바이스는 사물인터넷 구현을 위해 서비스 환경과 사물의 형태에 따라 매우 다양한 성능과 함께 향후 10년간 가격 하락, 전원공급 문제의 해결, 더 넓은 연결 등 기술의 성숙이 요구된다[15].

사물인터넷의 핵심 요소 기술은 크게 센서/기기 기술, 네트워킹 기술, 인터페이스 기술로 구분할 수 있다[16-17].

2.1.1 센서/기기 기술

센서 기술은 온도, 습도, 수분, 가스, 조도 및 초음파 센서와 같은 다양한 센서를 이용하여 원격감지, 위치 및 모션 추적 등을 통해 사물과 주위 환경으로부터 정보를 획득하는 기능이다. 사물인터넷의 센서 기술은 사물이 정보를 인식하고 생산하는 단계, 감지 대상과 방식, 구현기술과 적용분야에 따라 다양한 형태로 진화하고 있다. 측정 대상으로부터 물리/화학/생물학적 정보를 측정하여 시스템이 해석할 수 있는 신호로 변환하여 제공하는 기존 센서와, MCU가 내장되고 SoC(System on Chip) 기술이 접목되어 제어, 판단, 저장, 통신 등의 기능을 포함하여 성능이 향상된 스마트 센서, 센서 기기와 센서 네트워크 기술의 발달로 스마트 센서를 내장한 스마트 기기 및 사물인터넷 기기의 발전이 최근

주를 이루고 있다. 홈 환경에서도 다양한 형태의 스마트 센서 및 스마트 센서를 내장한 스마트 기기들이 등장하고 있으며, 스마트폰, 스마트 가전 등 기존의 내장 센서 외에도 외부의 센서 정보와 외부 인터넷 환경을 활용하여 다양한 응용 서비스의 개발도 진행 중이다.

2.1.2 네트워킹 기술

사물인터넷의 네트워킹 기술은 인간과 사물, 서비스 등 분산된 환경 요소들을 서로 연결시킬 수 있는 유무선 네트워킹 기능이며, 무선 네트워크를 중심으로 발전하고 있다. 기기 간의 연결을 위해 와이파이, 3G/4G/LTE 등과 같이 인터넷 연결을 통해 정보를 주고받는 IP 기반의 프로토콜 기술이 사용되고 있으며, IP를 사용하지 않는 Bluetooth, ZigBee, Zwave, RFID 등의 Non-IP 프로토콜을 사용하는 기기/센서 간의 통신에는 싱크노드(Sink Node)를 통해 인터넷 혹은 다른 기기와 연결하여 정보를 공유할 수 있다.

스마트 홈서비스를 위한 홈 네트워크 환경은 주로 유무선 공유기(AP, Access Point)를 중심으로 와이파이를 통해 기기들이 연결된다. 최근 SmartThings[18], Withings[19], Philips 등 다양한 업체에서 홈 환경에서 사용 가능한 스마트 전구, 센서 등의 제품들을 출시하고 있다. 이러한 스마트 센서 및 액추에이터(Acutuator)들은 기기간 정보교환을 위해 주로 ZigBee/ZWave를 사용하고 있으며, Hub/Bridge라고 불리는 싱크노드를 통해 인터넷과 연결되어 제어 및 상태 메시지를 송수신한다.

2.1.3 인터페이스 기술

인터페이스 기술은 사물인터넷의 주요 구성요소를 통해 특정 기능을 수행하는 응용서비스와 연동하는 역할을 담당한다. 따라서 사물인터넷 망을 통해 저장, 처리 및 변환 등 다양한 서비스를 제공할 수 있는 인터페이스 역할을 실행

할 수 있어야 한다. 인터페이스 기술은 정보의 검출, 가공, 정형화, 추출, 처리 및 저장기능을 의미하는 검출정보 기반기술과 위치정보 기반기술, 보안기능, 데이터 마이닝(data mining)기술, 웹 서비스 기술 등으로 구성된다. 이러한 사물인터넷의 인터페이스 기술은 다양한 업체에서 다양한 플랫폼을 이용한 응용/서비스를 통해 제공되고 있으며, 오픈소스를 기반으로 한 단일보드, 컨트롤러, 센서 및 액추에이터 연결을 통한 다양한 기능을 제공하는 하드웨어 플랫폼(hardware platform), 네트워크 게이트웨이 계층에서 장비에 대한 관리, 구동 등 지원과 함께 응용프로그램/서비스 개발을 지원하는 게이트웨이 중심 플랫폼(gateway centric platform), 기기간의 연결을 중심으로 서비스를 개발할 수 있는 연결 중심 플랫폼(connectivity centric platform)등의 형태로 구분된다[17].

2.1.4 사물인터넷 국내외 기술 동향

최근 사물인터넷 기반 산업은 국내·국외 관계없이 다양한 분야에서 활발히 연구되고 있다. 국내 및 국외 사물인터넷의 시장전망으로 2011년부터 꾸준히 증가 추세를 보이고 있다. 또한 사물인터넷의 기술적 구현보다 사물의 네트워크로 생성되는 데이터를 활용한 새로운 기술의 등장할 것으로 내다보고 있으며 향후 차별화된 부가가치 창출이 가능한 서비스 중심의 시장 발전할 것으로 전망하고 있다.

가. 국내 산업 동향

SK텔레콤은 제주도 서귀포와 경북 성주지역에 비닐하우스 내부의 온도와 습도, 급수와 배수, 사료공급 등까지 원격 제어 지능형 비닐하우스 관리 시스템인 스마트 팜 서비스를 제공하고 있다[20].

KT는 스마트 폰을 활용한 댁내 방범, 전력제어, 검침 등의 다양한 사물인터넷 서비스를 제공하고 있다. 원격지에서 거주자가 스마트 폰으로 KT의 사물인

터넷 플랫폼을 통해 실시간으로 댁내 환경을 모니터링 할 수 있으며, 간단한 스마트 폰 작동을 통해 전등, 출입문 등을 제어할 수 있다. 또한, 실시간 침입 및 화재 경보를 수신할 수 있으므로 스마트 원격 관제 서비스가 가능하다[21].

LG U+는 DTG(Digital Tacho Graph)와 사물인터넷 플랫폼과의 연동을 통하여 실시간 차량 관제 서비스를 화물차량, 버스, 택시 등을 대상으로 제공하고 있다. 또한, 2012 여수 세계박람회 기간 동안 LTE 기반의 사물인터넷 솔루션을 적용한 차량관제 시스템을 운영하여 승무원, 승객관리, 운행상태와 속도, 이동거리 등의 차량 정보를 실시간으로 교통관제 센터에 전송하는 서비스를 제공하였다[22].

나. 국외 산업 동향

현재 약 20억 명이 인터넷을 사용하는 것으로 추정되는데, IBM은 2020년 500억 개의 사물이 인터넷에 연결되는 사물인터넷 시대를 전망했다. 이와 같은 전망아래, IBM은 ‘똑똑한 지구(Smarter Planet)’라는 새로운 혁신 프로젝트를 전개하고 있다. 이는 모든 자연과 사람을 연결해 기능화・지능화 에너지・교통・금융・유통・제조・공공안전・도시 관리 등 다양한 분야에 똑똑한 시스템을 만들자는 것이 핵심이다. 세상의 수많은 디바이스와 소프트웨어가 네트워킹이 가능하도록 하는 IT기술의 확장을 통해 관련 사물인터넷 전 분야를 IBM의 시장영역으로 만들기 위한 전략을 가지고 있다[23].

Cisco도 ‘Smart+Connected Communities’라는 혁신 프로젝트를 추진하고 있다. 네트워크로 연결・통합된 커뮤니티와 도시 활동을 통해 지속적 경제 성장과 자원 관리, 운영 효율을 통한 환경보전을 가능하게 하고 삶의 질 향상을 위한 솔루션으로 제시하고 있다. ‘Community+Connect’ 솔루션을 통해 집, 학교, 교통 분야에서 삶의 질을 향상시킬 수 있는 정보와 서비스를 시민에게 제공한다. ‘Community+Exchange’ 솔루션을 통해서는 정부 및 지역 파트너들이 해당 시민들에게 자유롭게 거주하며 일하고 삶을 즐길 수 있는 안전한 커뮤니티를

제공할 수 있도록 한다는 것이 시스코의 사물인터넷 비전이다[24].

영국의 Pachube는 인터렉티브 환경에서 센서로 부터 들어오는 실시간 데이터를 관리할 목적으로 2008년에 처음 설립된 회사다. 사람, 사물, 애플리케이션을 사물인터넷에 연결하기 위한 목적으로 개발됐다. 수집한 데이터를 실시간으로 자사 서버에 전송하고 수집한 데이터를 누구나 이용할 수 있도록 오픈 API를 제공함으로써 웹 기반 서비스를 통해 전 세계의 데이터를 실시간으로 관리할 수 있다. 회원은 전 세계로부터 수집한 정보를 공유하고 협업할 수 있는 환경을 제공한다. 현재 100개 이상의 국가로부터 수백만 개의 데이터 포인트가 등록되어 있고 이곳에서 측정한 방사능량, 에너지 소비 및 비용, 기후 관련 정보가 공공안전, 농업, 서비스, 빌딩 자동화 등에 이용되고 있다. 현재 Pachube는 LogMeIn사에 인수되어, 회사명이 COSM으로 변경되었으며, 사물인터넷 상용 서비스 플랫폼인 Xively 클라우드 서비스에 통합 돼 서비스를 제공하고 있다[25].

EVRYTHNG은 기존 제품을 웹으로 연결해 보다 스마트하게 만들기 위한 'WoT(Web of Things)' 기술을 개발하는 회사이며, 스마트 폰과 각 제품을 나타내는 고유한 'Intelligent Identity'를 이용해 제품 생산자와 소비자, 파트너를 직접 웹 상으로 연결시켜주는 서비스를 제공한다. EVRYTHNG의 엔진기술을 통해 어떤 물리적 제품이든 개인화된 디지털 서비스를 위한 채널 및 일대일 통신을 가능하게 한다. 이를 통해 제품 생산자가 소비자에게 가까이 다가갈 수 있고 제품에 대한 생산, 판매 및 사용에 관련된 실시간 정보를 얻을 수 있다[26].

2.2 사물인터넷에서의 오픈소스

'오픈소스(open source)'는 IT분야의 자유 소프트웨어 운동의 대안으로, 리차드 스톨만(Richard Stallman)이 주축이 되어 소프트웨어의 소스코드를 공개하여 누구든지 프로그램 개발에 참여하게 함으로써 소프트웨어 기술의 독점화를 반대한 이념으로, 이는 유용한 기술을 공유함으로써 전 세계의 누구나가 자유

롭게 소프트웨어 개발·개량에 참여할 수 있게 하는 것이 우수한 소프트웨어를 만드는 데 도움이 된다는 생각에 바탕을 두고 있다.

1984년 스톨만은 자유 소프트웨어 재단을 설립하고, GNU라는 운영체제 개발 프로젝트를 구성하였다. GNU 프로젝트의 목적은 어느 누구도 소프트웨어 대하여 비용을 지불하지 않도록 하는 것이었고, 스톨만은 실행 프로그램을 구성하는 지식(소스코드)이 공개되어야 한다고 주장하였다. GNU 프로젝트와 자유 소프트웨어 재단의 기본적인 신조는 소스 코드가 컴퓨터 과학을 발전시키는데 기본적인 것이며, 변화가 계속되기 위해 정말로 필요한 것은 바로 자유롭게 사용할 수 있는 소스코드라는 것이다[28].

그 뒤 1997년 기존 자유 소프트웨어를 둘러싼 아이디어들을 진흥시킬 수 있는 방법을 찾기 위한 회의를 통해 시장 점유뿐만 아니라 대중들의 인식까지도 점유할 수 있는 마케팅 캠페인의 일환으로 새로운 용어인 오픈소스가 나타났다.

2.2.1 오픈소스 하드웨어

최근 IT 업계에서는 오픈소스 소프트웨어(Open Source Software, OSS)에 이어 오픈소스 하드웨어(Open Source Hardware, OSHW)를 새로운 기술 혁신 트렌드로 주목하고 있다. 오픈소스 하드웨어도 오픈소스 소프트웨어와 크게 다르지 않다. 오픈소스 소프트웨어가 소프트웨어를 구성하는 소스코드를 공개하듯, 오픈소스 하드웨어는 하드웨어를 구성하는 회로도, 파트리스트, 회로도 등을 대중에게 공개한 제품을 말한다. 그리고 오픈소스 하드웨어 협회(Open Source Hardware Association)에서는 오픈소스 하드웨어에 대한 정의를 명문화해서 공개하고 있으며 현재 1.0버전이 공개되어 있고 위키 페이지에 한글 번역본도 공개되어 있다[28].

오픈소스 하드웨어는 하드웨어를 배우고, 수정하고, 배포하고, 제조하고 팔 수 있는 디자인이 공개된 하드웨어이다. 하드웨어를 만들기 위한 디자인 소스

는 그것을 수정하기에 적합한 형태로 구할 수 있어야 한다. 오픈소스 하드웨어는 각 개인들이 하드웨어를 만들고 이 하드웨어의 사용을 극대화하기 위하여, 쉽게 구할 수 있는 부품과 재료, 표준 가공 방법, 개방된 시설, 제약이 없는 콘텐트 그리고 오픈소스 디자인 툴을 사용하는 것이 이상적이다. 오픈소스 하드웨어 플랫폼은 하드웨어 디자인을 자유롭게 교환함으로써 지식을 공유하고 상용화를 장려하여 사용자들이 자유롭게 기술을 제어할 수 있도록 한다[29].

대표적인 사물인터넷 오픈소스 하드웨어 플랫폼들은 크게 싱글보드 마이크로컨트롤러(Single-board micro controllers)와 싱글보드 컴퓨터(Single-board computers)로 분류할 수 있다[30]. 현재 가장 많이 알려져 있는 오픈소스 하드웨어 플랫폼으로는 아두이노와 라즈베리 파이가 있으며 각 플랫폼에 따른 특성의 차이는 표 2.2와 같다.

표 2.2 오픈소스 하드웨어 플랫폼 비교

Table 2.2 An overview of some IoT embedded platforms.

Brand	Arduino	Raspberry Pi	Intel	Beagle Board
Models	20+ and many clones (Spark, Intel and so on)	A, A+, B, B+, 2, 3, Zero	Edison	BeagleBone Black, X15, and so on
CPU	ATmega, 8-64 MHz, Intel Curie, Linino	ARMv6 or v7, 700 MHz -1.2 GHz	Intel Atom 500 MHz	AM335x 1 GHz ARMv7
RAM	16 KB ~ 64 MB	256 ~ 1 GB	1 GB	512 MB ~ 2 GB
+	Largest community	Full Linux, GPU, large community	X86, full Linux	Stability, full Linux, SDK
Price	~30 USD	~5-35 USD	~50 USD	~50 USD
Type	RTOS, Linux, hobbyists	Linux, hobbyists	Linux, hobbyist to industrial	Linux, hobbyist to industrial
Connectivity	Pluggable extension boards (Wi-Fi, GPRS, BLE,ZigBee, and so on)	Ethernet, extension through USB, BLE (Pi3)	Wi-Fi, BLE	Ethernet, extension through USB and shields
Division	Single-board micro controllers	Single-board computers	Single-board computers	Single-board computers

가. 아두이노

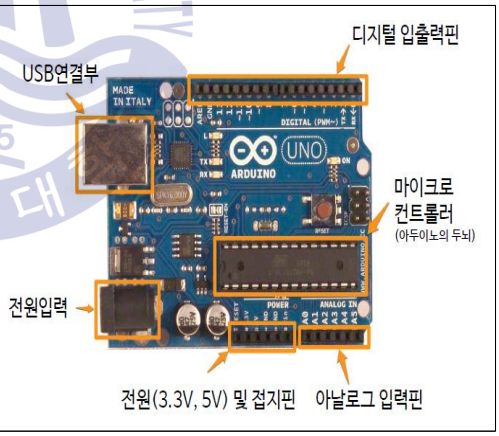
2005년 이탈리아에서 탄생한 아두이노는 현재 가장 유명하고 널리 활용되는 오픈소스 하드웨어 플랫폼이다. AVR을 사용하는 오픈소스 마이크로컨트롤러 보드(MCU)로서 임베디드 개발 경험이 전혀 없는 이용자들도 쉽게 활용할 수 있도록 개발 툴이나 회로도 등을 오픈소스로 제공하고 있다. 특히 아두이노 보드에 펌웨어 프로그램을 만들어 탑재할 수 있는 쉽고 간단한 개발환경은 아두이노 확산의 핵심 요소이다. 아두이노는 8비트 AVR CPU를 탑재한 저사양의 마이크로 컨트롤러 보드이지만, 센서와 액추에이터를 이용할 수 있는 여러 개의 디지털 핀과 아날로그 핀이 있다. 이를 통해 조도, 온도, 습도 등을 측정하

는 다양한 센서는 물론 스피커, LED, 모터 등의 다양한 액추에이터를 연결하는데 적합하다. 여기에 GSM, 와이파이, 이더넷 등의 통신 연결 모듈인 쉴드(Shield)나 LCD 스크린, USB 어댑터 등의 액세서리를 결합시키면 아두이노의 활용도는 더욱 높아진다. 아두이노의 외형과 주요사양은 표 2.3과 같다. 가격은 비교적 저렴하며 윈도우, 맥OS, 리눅스 등의 다양한 OS도 지원가능하다. 다양한 활용 사례가 커뮤니티를 통해 공개되어 있는데, 회로연결에 대한 내용이나 펌웨어 개발 소스 코드, PC 및 모바일 단말 전용소스 코드 등이 포함된다. 따라서 아두이노를 이용하면 맞춤형 시계나 조명 시스템, 애완견과 놀아주는 장난감, 화재경보기, 웨어러블 컴퓨터 등 다채로운 제품을 저렴한 가격으로 간단하게 제작 가능하다.

표 2.3 아두이노 우노 R3 사양

Table 2.3 Arduino UNO R3 Specification

마이크로컨트롤러	ATmega328
동작전압	5V
입력전압	7V ~ 12V
디지털 입출력 핀	14(6 PWM output)
아날로그 입력 핀	6
플래시 메모리	32KB
SRAM	2KB
EEPROM	1KB
클락 주파수	16 MHz



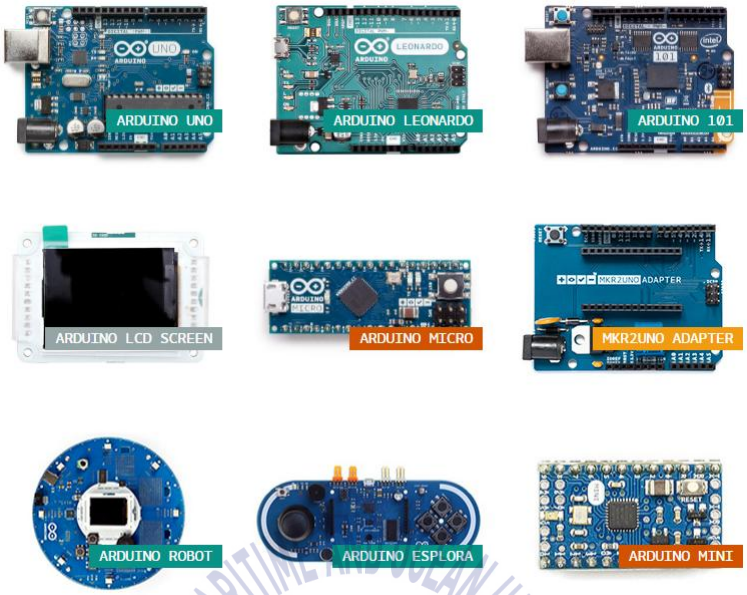


그림 2.1 아두이노 제품 종류
Fig 2.1 Arduino products

현재 오픈소스 하드웨어 플랫폼인 아두이노의 버전은 그림 2.1과 같이 아두이노 우노(UNO)를 비롯하여 디시밀라(Diecimila), 듀이(Due), 레오나르도(Leonardo), 메가(Mega), 나노(Nano) 등 다양한 제품으로 판매되고 있다. 다른 오픈소스 하드웨어에 비해 모델의 종류가 월등히 많으며 그 사이즈도 매우 다양하기 때문에 여러 형태 및 규모의 제품 개발에 적용이 유리하다[31].

나. 라즈베리 파이

라즈베리 파이는 영국의 라즈베리 파이 재단(Raspberry Pi Foundation)이 컴퓨터 및 과학 교육을 목적으로 개발한 초소형 싱글 보드 컴퓨터로서, 기존의 데스크톱 PC와 유사하게 키보드, 모니터 등의 주변기기와 연결이 가능하다. 라즈베리 파이는 700MHz ARM CPU와 그래픽 처리 장치(Graphic Process Unit, GPU), 디지털 신호 처리 장치(Digital Signal Processor, DSP), SD RAM 등이 탑

재된 미국 브로드컴(BroadComm)의 BCM2835 SoC를 기반으로 하고 있다. 램의 종류 및 USB 포트 수에 따라 모델 A(25달러)와 모델 B(35달러) 두 가지 종류로 구분된다. 모델 A에는 256MB의 램이 탑재되어 있으며 1개의 USB 2.0 포트가 내장된 반면, 모델 B에는 516MB의 램이 탑재되고 2개의 USB포트 및 이더넷 네트워크 기능이 내장되어 있다. 2016년 5월 현재, Raspberry Pi 2에 이어 Raspberry Pi 3까지 출시되었다.

출시 초반에는 영국을 중심으로 판매되었지만 점차 북미와 유럽 전역으로 퍼져나가 현재는 아시아, 오세아니아 지역에서도 판매중이다. 아두이노와 마찬가지로 다양한 센서와 액추에이터를 연결해 다양한 기능을 구현할 수 있는 가운데, 전용 카메라 모듈도 발매하는 등 활용성을 향상시켜 나가고 있다.

라즈베리 파이는 라즈비안(Rasbian)이라는 운영체제를 사용한다. 라즈비안은 라즈베리 파이와 데비안을 합친 말이다. 데비안은 리눅스의 한 종류로, 실제로 라즈베리 파이 재단에서는 라즈비안을 비롯하여 아치 리눅스(Arch Linux)의 다운로드를 제공하고 있다[32].

라즈베리 파이의 대표적 모델은 그림 2.2와 같고 주요사양은 표 2.4와 같다.

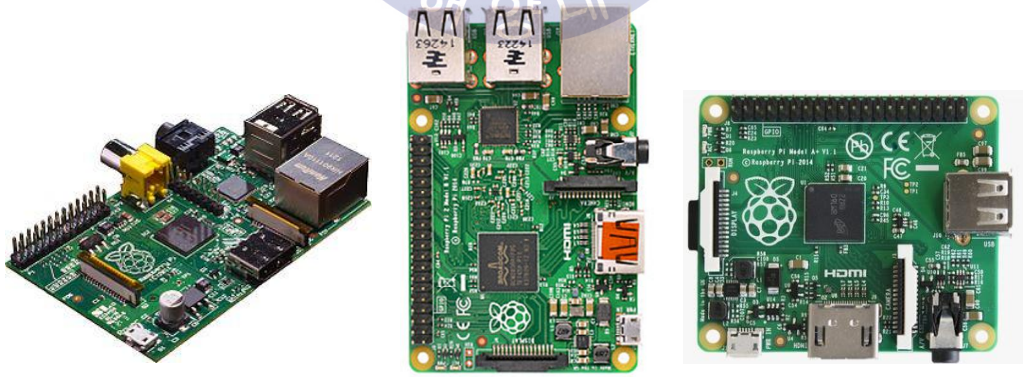


그림 2.2 라즈베리 파이 제품 종류
Fig 2.2 Raspberry Pi Products

표 2.4 라즈베리 파이 모델별 사양

Table 2.4 Raspberry Pi Model Specification

	Raspberry Pi B+	Raspberry Pi 2	Raspberry Pi 3
Price	US$25	US$35	US$35
SOC	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2837
CPU	single-core ARM11@700MHz	quad-core ARM Cortex-A7@900 MHz	quad-core ARM Cortex-A53@1.2 GHz
Memory	512 MB	1 GB	1 GB
USB 2.0 port	4 (via the on-board 5-port USB hub)		
Video outputs	HDMI, Composite video, MIPI display interface		
On-board network	10/100 Mbit/s Ethernet USB adapter on the USB hub	10/100 Mbit/s Ethernet USB adapter on the USB hub	10/100 Mbit/s Ethernet, 802.11n wireless, Bluetooth 4.1
Low-level peripherals	17× GPIO plus the same specific functions, and HAT ID bus		
Power source	5 V via MicroUSB or GPIO header		
Size	85.60 mm × 56.5 mm (3.370 in × 2.224 in)		
Weight	45 g (1.6 oz)		

표 2.5는 오픈소스 하드웨어 플랫폼 중에서 대표적인 아두이노와 라즈베리 파이의 차이점을 비교하여 나타내고 있다.

표 2.5 아두이노와 라즈베리 파이 비교
Table 2.5 Arduino vs. Raspberry Pi

	Arduino	Raspberry Pi
Character	micro controller	micro computer
Memory	0.002 MB	512 ~ 1 MB
Speed	16 MHz	700 MHz
Flash	32 KB	SD Card (2 ~ 16 GB)
OS	None	Linux distributions
Multitasking	No	Yes
IDE	arduino sketch	anything with Linux support
On-board network	Limit	Yes

2.2.2 오픈소스 소프트웨어

사물인터넷에서의 사물의 정의는 다소 광범위하다. 어떤 기기까지를 사물인터넷에서 사물로 볼 것인가는 아직 명확하지 않은 상태이고, 다만 사물인터넷에서 사물은 인터넷에 반드시 연결되어야 하므로 기본적으로 인터넷 접속에 필요한 소프트웨어적 기능들(네트워크 연결, 인터넷 프로토콜, 메시지 송수신 등)은 가져야한다는 정도로 정리할 수 있다.

이와 같은 사물인터넷의 기술적 전제로 볼 때 가장 중요한 요구사항은 기기간 연결이다. 이는 기기간 연결(Peer-to-Peer, Machine-to-Machine 등) 및 때로는 클라우드 등 다양한 인프라와의 연결을 포함한다. 이와 같은 연결은 다양한 네트워크 기술로 구현이 가능하다. 즉, 하나의 통일된 네트워크 기술로 사물들이 연결될 필요는 없으며, 다양한 네트워크 연결 위에서 상호 이해할 수 있는 통일된 통신 프로토콜이 가능하다면 사물인터넷의 구현이 가능하다. 최근 다양

한 무선 네트워크 기술로 와이파이, ZigBee, Bluetooth 등이 사물인터넷에 널리 활용되고 있으며, 특히 초경량 사물에도 적용하기 위해 저전력 기술들이 지속적으로 개발되고 있다. 그림 2.3과 같이 다양한 네트워크로 연결된 사물들간 호환 가능한 통신은 사물인터넷에 필수이며 이 기술들은 소프트웨어적으로 각 사물에 탑재되어 동작되어야 한다.

그림 2.3 사물인터넷에서 사물 연결
Fig 2.3 Connection on Internet of Things

사물인터넷은 이전에 연결되지 않았던 기기들이 점점 더 많이 인터넷에 연결되는 '사물'들의 폭발적인 양적 성장이 이뤄지고 있다. 예를 들어 집, 공장, 빌딩의 조명이 모두 인터넷에 연결되어 사물로 인식, 인터넷을 통해 서로 통신을 할 수 있게 되는 것이다.

연결된 기기는 인터넷을 사용해야 하므로 IETF(Internet Engineering Task Force)에서 제정한 인터넷 프로토콜을 따라야 한다. 그러나 인터넷은 전통적으로 많은 전력과 메모리, HTTP 통신 프로토콜과 같은 자원이 풍부한 기기들을 연결해 왔다. 그와 같은 프로토콜은 사물인터넷 애플리케이션에 적용하기에는 너무 무겁다. 따라서 적은 전력을 사용하면서 제한된 리소스를 갖는 사물에도 탑재할 수 있는 새롭고 더 가벼운 프로토콜을 필요로 한다[33]. 다음은 이러한 특성을 가진 대표적 사물인터넷 오픈소스 프로토콜들이다.

가. MQTT

MQTT(Message Queuing Telemetry Transport)는 원격 제어, 원격 검침을 위한 경량의 통신 프로토콜이다[34]. MQTT 프로토콜은 스마트폰과 같은 제한된 컴퓨팅 성능과 빈약한 네트워크 연결 환경에서 동작하도록 설계된 대용량 메시지 전달 프로토콜로 낮은 대역폭과 긴 지연시간에도 불구하고 메시지 전송의 신뢰성을 제공할 수 있도록 설계되었다. 사물과 같은 임베디드 장치의 제한된 통신 환경을 고려하여 디자인된 MQTT는 현재 사물인터넷 분야에서 최적의 프로토콜로 주목받고 있다[35]. 1999년 IBM에서 초기 버전을 발표하였고, 2004년 MQTT.org에서 스펙 및 라이브러리를 공개하였으며, 2013년 OASIS(Organization for the Advancement of Structured Information Standards)에 의해 사물인터넷을 위한 표준 메시지 프로토콜로 선정되었다.

MQTT의 특징은 다음과 같다.

- 실시간 푸시 전송을 하면서도 전력 사용은 최소화하며, 경량 메시지를 사용하며 80~100kb 정도 코드 메모리만 사용한다.
- 느리고 품질이 낮은 네트워크의 장애와 단절에 대비한다.
- 클라이언트 애플리케이션 동작에 자원 활용이 극히 제한적임을 고려한다.
- 다수의 클라이언트 연결에 적합한 Publish/Subscribe 네트워크를 적용한다.
- 신뢰성 있는 메시징을 위한 QoS(Quality of Service) 옵션을 제공한다.
- 개방형 표준 메시징 프로토콜을 지향한다.

MQTT 구조는 발행/구독(Publish/Subscribe)방식의 모델로 데이터를 송신하고 수신하기 위해 그림 2.4와 같이 발행자(Publisher), 토픽(Topic), 브로커(Broker), 구독자(Subscriber)로 구성된다[36]. 일반적인 통신 구조는 발행자가 토픽을 발행하면 브로커가 이를 중계하고, 구독자는 브로커를 통해 관심 있는 토픽을 구독하는 것이다. MQTT 서버라고 하지 않고 브로커라고 하는 이유는 MQTT가 정보를 생산하는 발행인과 정보를 소비하는 구독자가 메시지를 주고 받을 수

있도록 중계 역할만을 담당하기 때문이다. 발행자와 구독자가 직접 메시지를 주고받지 않기 때문에 비동기 방식이며, 메시지 큐 방식, 발행/구독 또는 출판/가입 방식이라고도 한다.

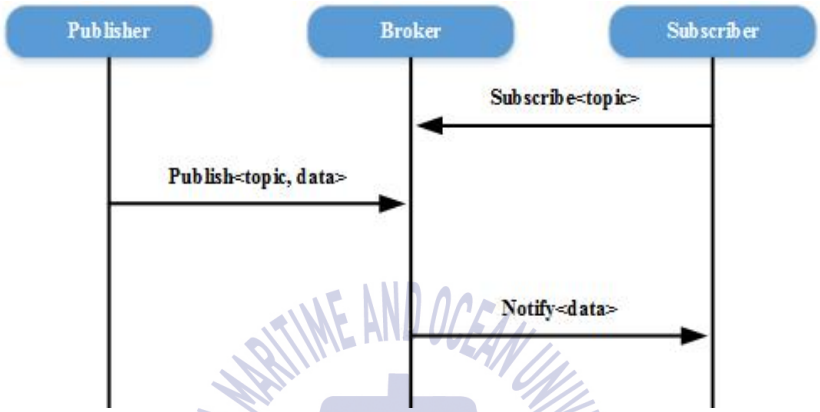


그림 2.4 MQTT 통신의 일반적인 구조
Fig 2.4 General structure of MQTT communication

그림 2.5는 사물인터넷 연결을 위한 MQTT의 발행/구독 구조이다. MQTT 송수신 과정에서 발행자는 다수의 클라이언트와 연결하여 1:N 통신을 할 수 있다. 발행자와 구독자는 서로 직접 연결하지 않고 브로커를 통해 메시지를 주고받는다. 발행자와 구독자 사이에 전송되는 메시지는 다양한 주제의 하위 주제로 분류되어 송수신될 수 있다. 각 메시지의 특정 주제가 토픽(Topic)이다. 구독자가 수신 받고 싶은 메시지에 대한 토픽을 브로커에 등록하면, 발행자가 메시지를 생성해 토픽과 함께 전송하고, 이를 브로커가 수신한 후 토픽에 매칭되는 구독자에게 메시지를 전송한다. 다시 말해 MQTT 브로커는 각종 장치들에 해당하는 MQTT 클라이언트가 보내주는 메시지를 수집하고 이걸 다시 필요한 MQTT 클라이언트들에게 전송해주는 중계 서버의 역할을 담당한다. MQTT 클라이언트는 MQTT 브로커에게 메시지를 전달하고 필요한 메시지를 받기만 하면 되고 MQTT 프로토콜의 구조도 간단하기 때문에 처리 능력이 낮은 임베디

드 장치에 적합하다. 로컬 통신만 필요한 경우에는 MQTT 브로커를 로컬에서 운영하고 발행자와 구독자가 동일한 로컬 네트워크에 있지 않는 경우에는 인터넷에 MQTT 브로커를 배포할 수도 있다.

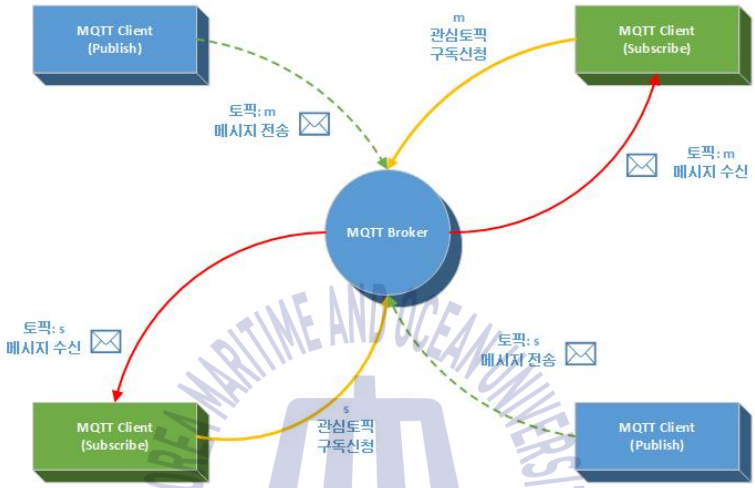


그림 2.5 MQTT 발행/구독 구조

Fig 2.5 The MQTT Publish/Subscribe Architecture

MQTT의 메시지는 계층 구조의 토픽과 관련 있다. 사물인터넷 환경에서는 사물과 사용자의 애플리케이션 모두 메시지를 송수신하는 클라이언트가 된다. 그래서 사물은 자신을 구분할 수 있는 구분자가 포함된 토픽으로 메시지를 전송할 것이고 애플리케이션은 해당 토픽을 관심 토픽으로 브로커에 등록해 사물의 상태 변경을 자동적으로 수신 받는다. 반대로 사물이 명령어를 수신하기 위해 미리 정한 토픽으로 애플리케이션이 제어 명령 메시지를 전송하면 사물이 이를 수신해서 제어 명령을 수행할 수도 있다.

토픽은 슬래시(/)를 이용해서 계층적으로 구성할 수 있어서 대량의 사물들을 효율적으로 관리 할 수 있다. 브로커가 메시지를 중계하고 있으므로 정보를 전송받기 위해서 네트워크에서 상대방의 인터넷 주소(IP)를 서로 몰라도 상관없이

토픽으로 원하는 정보를 받아볼 수 있다. 발행자와 구독자는 1:N의 관계가 가능하여 한 개의 토픽에 다수의 클라이언트가 구독할 수 있다.

메시지 수신을 원하는 클라이언트는 필터링(Filtering) 문자를 포함한 토픽을 관심 토픽으로 등록할 수 있다. 필터링 문자로는 '#'과 '+'가 있다. 또한, '#' 또는 '+' 기호를 사용하여 복수개의 토픽을 지정할 수도 있다. 표 2.6은 MQTT 필터링 문자에 대한 몇 가지 예를 보여준다.

표 2.6 MQTT 필터링 문자('+', '+')의 구독 예
Table 2.6 MQTT filtering character('+', '+') subscriptions example

Topic with Filtering	Matches	Does not Match
home/room/+	home/room/temperature home/room/humidity	home/kitchen/temperature office/kitchen/humidity
home/+/temperature	home/kitchen/temperature home/room/temperature	home/kitchen/humidity office/lobby
+	home office	home/kitchen home/kitchen/temperature business/lobby
home/#	home/ home/kitchen/humidity home/room/temperature	home office/lobby

MQTT는 표 2.7과 같이 다양한 종류의 브로커를 가지고 있다. 그 중에서도 가장 대중적으로 활용되는 것은 Mosquitto이다.

Mosquitto는 MQTT 전용 브로커로 경량이며 브로커가 가져야 할 대부분의 기능을 충실히 지원한다는 장점이 있다. 그렇기 때문에 소형 디바이스를 사용하는 사물인터넷 환경에서 사용하기에 적합하다. 또한 Mosquitto는 MQTT 프로토

콜 v3.1.1을 구현한 BSD 라이센스 기반의 오픈소스 메시지 브로커이다[37]. Mosquitto는 저전력 센서와 휴대전화 등의 모바일 기기와 임베디드 컴퓨터, 그리고 마이크로 컨트롤러와 같은 M2M 메시징에 적합하도록 설계되었으며, 마이크로소프트 윈도우와 애플 운영체제, 리눅스 계열 및 라즈베리 파이 등의 다양한 플랫폼을 지원한다.

표 2.7 MQTT 브로커 비교
Table 2.7 MQTT Broker

Broker	QoS0	QoS1	QoS2	SSL	dynamic topics	cluster	web sockets	plugin system
mosquitto	√	√	√	√	√	×	√	√
HiveMQ	√	√	√	√	√	√	√	√
Apache ActiveMQ	√	√	√	√	√	√	√	√
RabiitMQ	√	√	×	√	√	?	?	?
mosca	√	√	×	?	?	×	√	×

나. Web Socket

기존의 웹 방식은 웹 클라이언트인 웹 브라우저에서 웹 서버로 URL를 요청하고, 웹 서버는 해당 웹 클라이언트로 응답을 송신하는 구조이다. 웹 클라이언트에서는 수신된 응답을 이용하여 정보를 갱신하고, 수신된 정보를 화면에 나타내기 위하여 웹 브라우저 전체를 갱신하여야 한다. 이때 웹 브라우저의 깜빡임이 발생하게 되며, 결과적으로 사용자의 시각적인 피로감을 증가시키는 원인이 되기도 한다. 이를 해소하기 위하여 iFrame을 이용하여 숨겨진 프레임을 이용하거나 Long Polling, Stream 등의 방식이 사용되기도 한다. 그러나 이러한 방식은 네트워크 트래픽을 증가시키며, 전체적인 네트워크 부하를 증가시키는 단점이 있다[38-39]. IETF RFC6455에서 정의된 웹 소켓은 웹 브라우저상에서

웹 서버와 양방향 통신이 가능한 TCP 기반의 웹 소켓(Web Socket) 표준 프로토콜을 기술하고 있다[40].

그림 2.6은 웹 소켓 프로토콜을 통신 방법을 표현한 것으로, 웹 소켓이 웹 서버와 웹 클라이언트간의 양방향 통신을 지원하기 위하여 연결(Connection)을 설정하고 해제하는 나타내고 있다. 웹 서버와 웹 클라이언트가 양방향 통신을 수행하기 전에 먼저 Opening Handshake를 수행하여 연결을 설정한다. 연결이 설정된 웹 서버와 웹 클라이언트는 상시적으로 정보를 송수신 할 수 있다. 이러한 방법은 푸시 서버(Push Server)의 기본으로 네트워크의 트래픽을 줄일 수 있다는 장점이 있다.

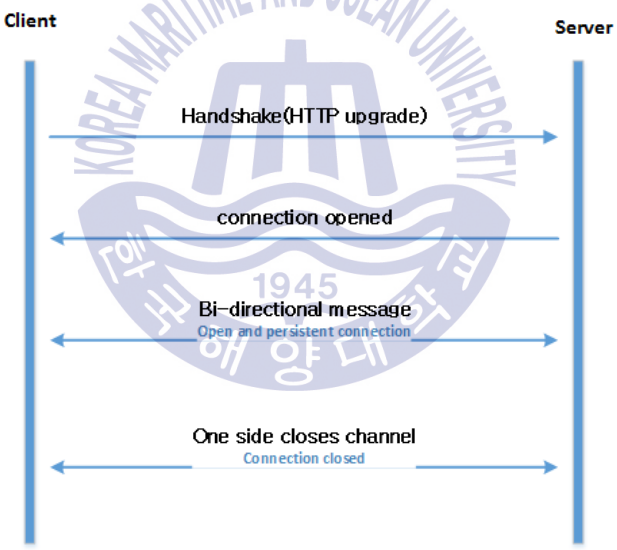


그림 2.6 웹 소켓 프로토콜
Fig 2.6 Web Socket protocol

그림 2.7은 기존 웹 방식에서의 Polling과 웹 소켓의 통신 방식을 비교하여 나타내고 있다. 기존 웹 방식의 Polling은 요청과 응답이 1:1로 구성된 방식으로 데이터를 수신하기 위하여 반드시 요청이 수행되어야 한다. 따라서 웹 화면의

일부를 갱신하기 위하여 전체 화면을 요청 후 응답을 수신해야 하는 네트워크 트래픽이 발생하는 구조이다. 반면, Web Socket 방식은 응답과 요청이 비동기적으로 운영되는 방식으로 기존 웹 Polling 방식을 개선함으로써 갱신이 필요한 데이터만 수신하는 방식으로 네트워크 트래픽에 효율적인 방식이다.

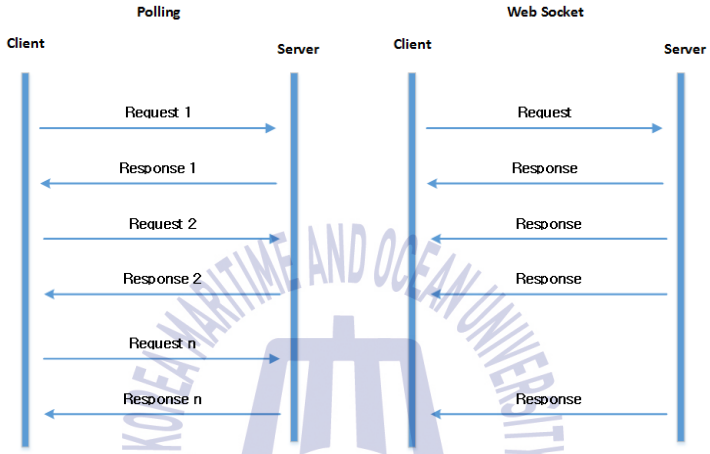


그림 2.7 Polling과 웹 소켓 통신 방식 비교
Fig 2.7 Polling vs. web socket communication

다. Node-RED

사물인터넷은 사물과 사물, 사물과 사람 사이에서 낮은 수준의 하드웨어와 온라인 서비스를 동시에 다루는 이벤트를 끊임없이 발생하여 수많은 데이터를 생산한다. 그림 2.8과 같이 사물인터넷을 위한 사물 연결 플랫폼은 인터넷에 연결된 사물에 대한 데이터 저장 및 처리, 데이터 시각화, 디바이스 제어 및 다른 서비스와의 연결 기능을 제공하는 플랫폼을 말한다[41]. 따라서 다양한 수준의 연결 흐름을 쉽게 다룰 수 있는 도구가 필요하게 되었고, 이런 목적으로 사용되고 있는 것 중의 하나가 Node-RED이다.

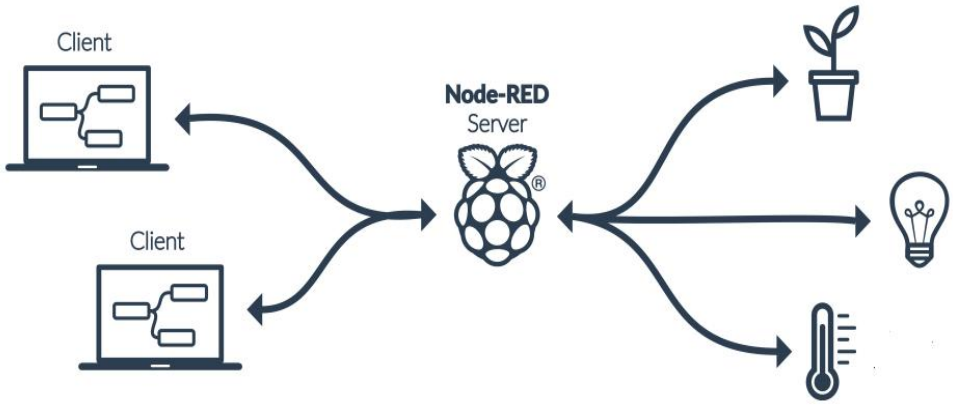


그림 2.8 Node-RED 개념
Fig 2.8 Concept of Node-RED

Node-RED는 IBM에서 개발하였으며 현재 오픈소스 비주얼 개발도구로서 하드웨어 장치, API 및 온라인 서비스를 포함하는 사물인터넷을 함께 연결하는 기능을 수행한다[42]. Node-RED는 그림 2.9처럼 각종 디바이스와 프로토콜이 입력과 출력을 가지는 노드로 추상화되어 있다. 다시 말해서 노드는 사물이나 서비스를 추상한 것이며 노드를 연결하는 와이어는 정보의 흐름을 추상화한 것이다. 따라서 Node-RED를 이용하여 노드와 와이어로 응용 프로그램을 쉽고 효과적으로 작성할 수 있는 것이다.

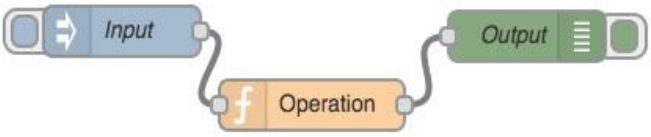


그림 2.9 Node-RED 노드 예
Fig 2.9 An example of Node-RED simple node

Node-RED의 특징은 다음과 같다.

- 사물인터넷 애플리케이션을 제작하는 비주얼 개발 도구이다.
- Node.js 위에 구축되며 거대한 노드 모듈들을 활용하여 다양한 시스템을 통합할 수 있는 도구이다.
- 간단한 런타임 배포 및 프로토타입 제작에 적합하다.
- 다양한 연동을 간단하게 작성할 수 있다.
- 자바스크립트 프로그래밍은 필요하지만 진입 장벽이 낮아 쉽게 배워서 활용할 수 있다.
- 클라우드 기반의 서비스화가 용이하다.
- 개방형 표준, 유연성, 확장성, 공유의 특징이 있다.

사물인터넷 애플리케이션 개발자는 Node-RED가 설치된 디바이스에 웹 브라우저를 사용하여 접속하고, 노드들을 서로 연결하여 새로운 서비스 또는 데이터 처리 루틴을 만들 수 있다. 예를 들어 온도센서 디바이스가 온도 측정값을 MQTT나 TCP로 전송한다고 가정하면 그림 2.10처럼 MQTT, TCP 노드를 데이터 노드에 연결하는 것만으로 데이터 전송 플로우를 완성할 수 있다. 이러한 방식은 개발자 입장에서 직관적이면서도 빠른 개발을 가능하게 하고, 다양한 단말 플랫폼이 존재하는 사물인터넷 환경에서 개발자들에게 응용 디바이스를 쉽고 빠르게 설계·개발할 수 있도록 해주는 장점을 제공한다. 그리고 Node-RED는 node.js[43] 기반으로 개발하였으므로 node.js의 많은 라이브러리를 사용할 수 있다는 것도 중요한 장점이다. 또한 개발자가 쉽게 새로운 노드를 추가할 수 있도록 개발자 참조 문서도 제공되고 있다.

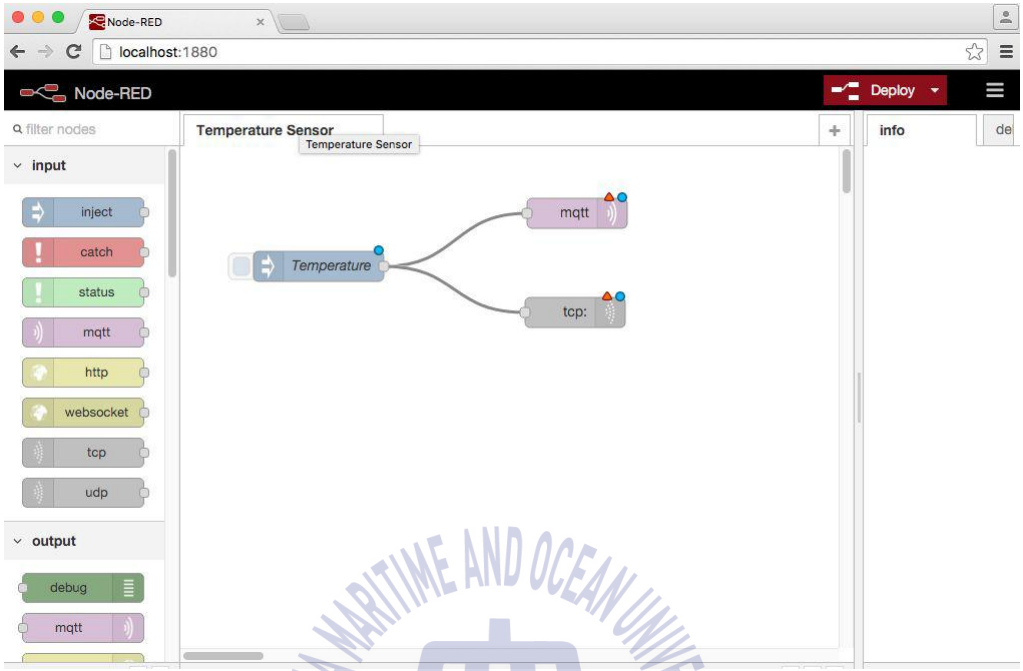


그림 2.10 Node-RED 인터페이스
Fig 2.10 Node-RED Interface

2.2.3 오픈소스 서비스 플랫폼

연동이 용이한 클라우드 서비스를 여건이나 환경에 맞게 구축하는 것은 쉬운 일이 아니다. 따라서 플랫폼을 구축할만한 역량이 모자란 독립 개발자나 중소기업 입장에서는 디바이스 개발에 역량을 집중하고, 정보의 확산은 개방형 클라우드 서비스를 이용하는 것이 자신의 디바이스의 효용을 최대한 높일 수 있는 방법일 것이다. 개방형 서비스 플랫폼을 이용하면, 일반 사용자도 사물인터넷의 구축에 일조를 할 수 있으며, 플랫폼에 축적된 데이터를 이용한 새로운 서비스의 개발도 용이해진다.

클라우드 기반 서비스에는 제공되는 최종 서비스 형태에 따라 SaaS(Software as a service), PaaS(Platform as a Service), 그리고 IaaS(Infrastructure as a service)가 있다. 즉, 이들 클라우드 기반 제품/서비스의 최종 형태는 각각 소프

트웨어(애플리케이션), 플랫폼, 또는 인프라(기반시설)로 제공된다. 또한 클라우드 서비스는 제공되는 환경이 기업 내부인지 또는 외부인지에 따라서 사설(Private) 클라우드, 공용(Public) 클라우드가 있고 이 둘을 혼합하거나, 여러 이종의 클라우드를 혼합하는 하이브리드(Hybrid) 클라우드 환경이 있다. 하이브리드 클라우드 환경은 기업 내외의 ICT 자원의 활용을 극대화하기 위해 전략적으로 선택되는 방법이다. PaaS는 통합 사물인터넷 소프트웨어 서비스이며, 애플리케이션과 서비스를 구축하고, 배치, 관리의 기반을 마련해 주는 클라우드 플랫폼이다.

Xively[44]는 개별 DIY 디바이스에서 데이터를 받아서 자신만의 애플리케이션을 만들 수 있도록 해주는 개방형 사물인터넷 플랫폼이다. Xively는 개별 DIY 디바이스가 특정 데이터 포맷을 준수하면, 클라우드 형태로 디바이스의 데이터 관리를 담당해주는 PaaS의 역할을 한다. 그림 2.11과 같이 Xively 플랫폼을 이용하면 디바이스는 종류에 상관없이 Xively API를 통해서 클라우드에 데이터를 저장하고, 이를 이용한 다양한 응용을 만들 수 있다. 따라서 특정 하드웨어의 구매 없이도 DIY 디바이스를 이용해서 다양한 서비스를 구축할 수 있는 개방형 플랫폼의 장점을 보여준다고 할 수 있다.

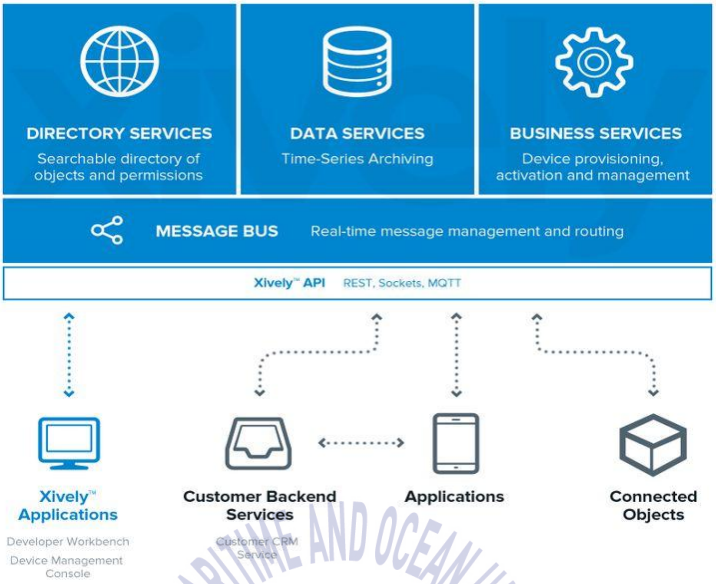


그림 2.11 Xively 서비스 개념
Fig 2.11 Xively Service concept

ThingSpeak는 인터넷 기반 모니터링과 컨트롤 하드웨어 제조사인 ioBridge를 통해서 2011년에 공개된 오픈소스 사물인터넷 클라우드 서비스 플랫폼이다[45]. ThingSpeak 플랫폼은 데이터 저장, 데이터 프로세싱, 데이터 전달, 위치 기반 서비스, 상태 업데이트, 소셜네트워크 통합 플러그인을 지원한다. 웹기반으로 HTTP 메시지를 통해서 데이터를 주고받을 수 있으며, 데이터에 대한 필터링을 지원하여 평균 및 합산과 같은 데이터 처리 옵션을 둘 수도 있다. ThingSpeak는 Twitter 기반의 API를 제공하여 ThingSpeak 채널에 저장된 데이터를 Twitter로 보낼 수 있고 데이터 공개여부 및 데이터에 접근하는 사용자에 대한 권한 설정 등 프라이버시를 위한 선택사항을 제공할 수도 있다.

ThingSpeak는 사물인터넷 디바이스에서 클라우드를 통해 데이터를 업로드, 검색 및 차트와 그래프로 데이터의 시각화를 제공하는 사물인터넷 서비스 공급자이다. 인터넷에 연결된 장치는 ThingSpeak 서비스를 사용하여 웹 데이터베이스에 데이터를 업로드 할 수 있다. 다른 클라이언트는 ThingSpeak 서비스를 사

용하여 데이터를 가져올 수 있다. 그림 2.12와 같이 ThingSpeak를 사용하여 사용자는 자신의 채널을 만들 수 있다. 사용자는 공개 또는 비공개 채널을 선택할 수 있다. 간단히 말해서 채널을 데이터베이스 테이블과 같이 시각화 할 수 있는 것이다. 모든 채널은 테이블의 필드로 시각화 할 수 있는 속성(최대 8 개)을 가질 수 있고, 모든 채널은 데이터 읽기 및 데이터 쓰기를 위한 API Key를 제공한다. 공개 채널 데이터는 키 없이 다른 모든 사용자에게 읽힐 수 있지만 데이터를 쓰려면 해당 채널에 해당하는 API 키가 있어야 한다.

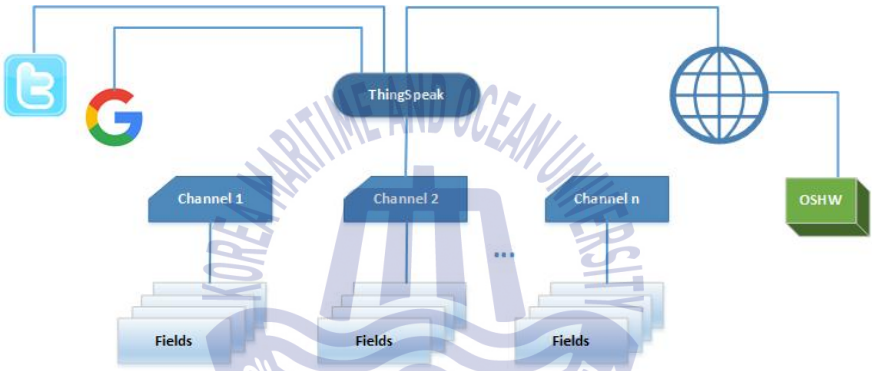


그림 2.12 ThingSpeak 서비스 개념
Fig 2.12 A concept of ThingSpeak Service

2.3 스마트 홈

2.3.1 사물인터넷 스마트 홈

스마트 홈은 그림 2.13과 같이 사물인터넷 기능이 포함된 가전제품 및 가정설비(조명, 냉난방, 가스 밸브 등)가 유무선 통신 네트워크 기반 주거환경에서 스스로 정보를 생산해서 다른 사물과 사람에게 전달하고, 사람의 수요를 파악하거나 예측하여 일정 수준의 자율적인 장치 제어 및 장치간의 협업을 통한 복합적인 제어를 지원함으로써 주거생활의 질을 높여주는 솔루션이나 서비스를 포함하는 개념이다[46].

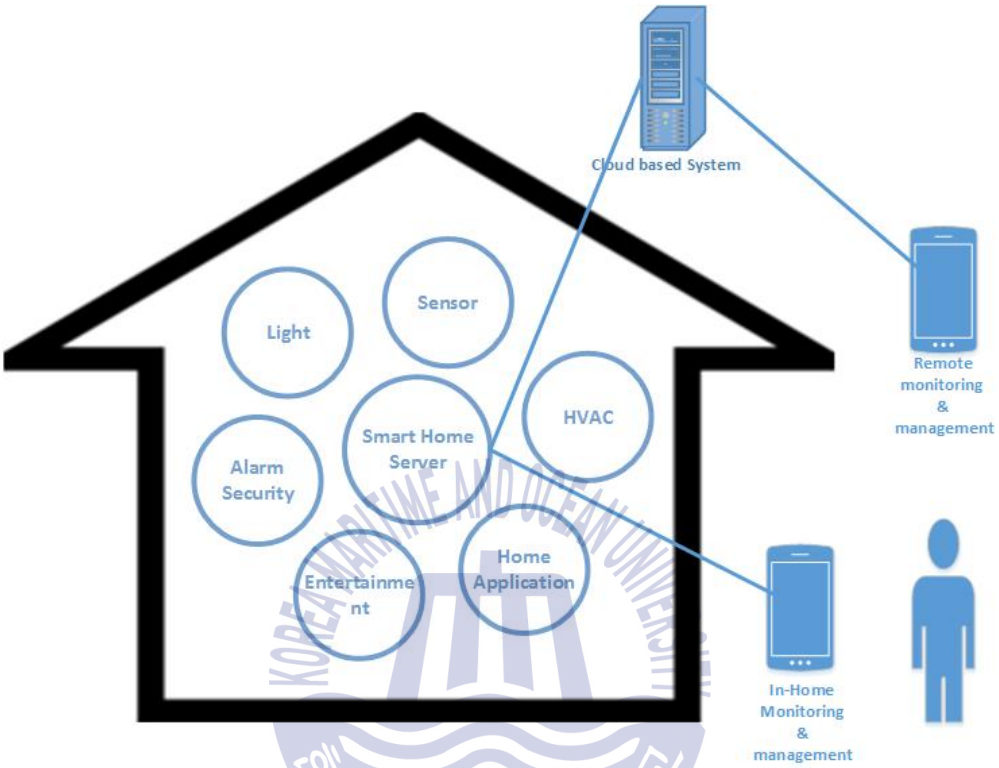


그림 2.13 스마트 홈의 개념
Fig 2.13 A concept of Smart home

사물인터넷 기술과 스마트 홈 기술의 융합은 기존의 서버-클라이언트 방식으로 제공되던 자원의 효율화 측면에서 스마트 홈서비스들이 비용이나 사물인터넷 기반의 서비스로 대체될 수 있도록 하며, 완전히 새로운 유형의 스마트 홈 서비스 창출의 가능성을 보여주고 있다[46].

스마트 홈은 일반적으로 가정자동화(Home Automation)와 원격 제어(Remote Control/Access)로 이루어진다. 대부분의 가정자동화는 그림 2.14처럼 이벤트 유형의 입력이 발생하면 정해진 조건에 따라 자동으로 동작을 수행하고, 원격 제어는 가정에서 사용되는 여러 장치를 허브와 같은 디바이스를 통해 제어하는 것이다.

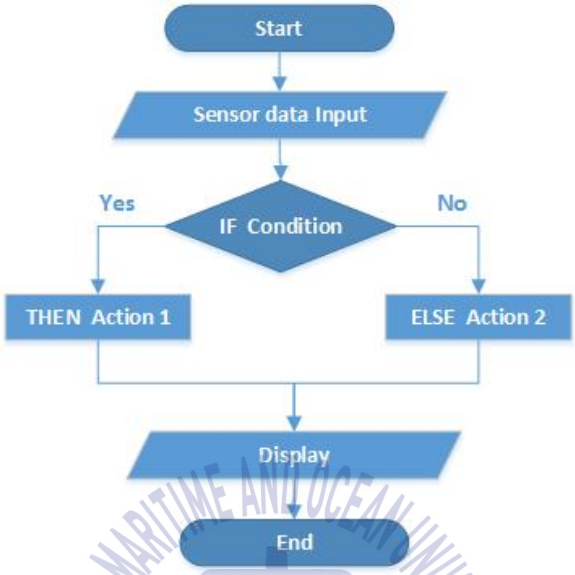


그림 2.14 홈 자동화 작업 흐름
Fig 2.14 Home Automation Flow

스마트 홈 기술은 생활의 편리함·즐거움·안락함 등 인간의 기본적인 욕구를 충족시킨다는 점이 스마트 홈 시장 성장의 원동력이다. 언제 어디서나 가정 내 환경에 대한 접근과 제어가 가능하며, 실내외의 어떠한 위험에 대해서도 확고한 보안성과 다양한 종류의 양방향 엔터테인먼트 서비스를 제공하는 것이 강점이다. 현재 기술 수요가 있는 부분은 가정용 전기·가스·수도 사용량 원격검침, 화재나 누전 경보기, 침입과 출입 감지 센서, 노인이나 환자의 긴급호출과 범죄나 사고 방지를 위한 보안 센서 등이다.

이동통신 3사(SKT, KT, LG U+)는 일반 소비자를 대상으로 스마트 단말과 가정 내 다양한 가전기기들의 연결을 통한 스마트 홈 기반의 사물인터넷 환경을 확대하려 하고 있다. 1980년대 후반부터 가정의 화재 및 보안 분야부터 상용화가 시도되었으며, 10년 전까지만 하더라도 가정자동화와 같이 초보적인 수준에 그쳤으나, 스마트 디바이스, 초고속 무선네트워크, 클라우드, 빅데이터 등 첨단

ICT 기술이 융합되면서 거대 산업으로 발전하고 있다[47]. 산업 범위도 스마트 융합가전, 가정자동화, 홈시큐리티, 스마트 헬스케어, 스마트 그린, 스마트 엔터테인먼트 등 다양한 산업으로 세분화되고 있다.

표 2.8 3대 통신사의 홈 IoT 서비스
Table 2.8 Home IoT service of three mobile telecomm companies

이통사	SK 텔레콤	KT	LGU+
서비스	스마트 홈	기가 IoT 홈메니저	홈 IoT
기능	스마트플러그, 가스밸스차단기, 도어락, 보일러, 청소기, 제습기	문열림감지, 도어락, 가스밸스	가스락, 열림감지, 에너지미터, 플러그
비용	각각 개별 구매 가능	월 15,000원(3년약정)	월 11,000원(3년약정)
향후 계획	음성제어 및 사용 패턴 분석	삼성전자 협력 IoT 홈매니저 가전제품 원격제어 추가	지능형 IoT 계획 (외부 환경 감지하여 자동 가동)

2.3.2 스마트 홈서비스 모델

스마트 홈서비스는 홈 네트워크 내의 스마트 기기 및 IoT 기기들의 상태를 조회하고 제어할 수 있는 서비스를 제공하고 있다. 이러한 서비스들 중 일부는 가정 내에서의 제어뿐 아니라, 사용자가 시간과 장소에 구애 받지 않고 홈 네트워크 내의 기기들을 제어할 수 있는 기능을 제공하고 있다. 이를 위해, 특정 스마트 기기들의 제어를 위한 클라우드 환경을 통해 인터페이스를 제공하는 서비스들이 출시되고 있다. 하지만, 스마트 기기들의 제어 기능을 제공하는 클라우드 서비스들은 특정 스마트 기기 제품군에 대한 제어만을 제공하고 있으며, 홈 네트워크 내에서 사용 가능한 다양한 종류의 스마트 기기들에 대한 서비스를 제공하지 못하고 있는 실정이다. 또한, UPnP/DLNA[48-49] 기능을 이용하여 홈 네트워크 내에서 기기간의 협업 서비스를 제공하지만, 클라우드를 통한 외

부 제어 기능을 제공하지 못하는 기존 가전 기기들을 이용한 스마트 홈서비스 제공에는 한계가 있다[50].

여기에서는 홈 네트워크 상의 다양한 기기들을 이용하여 외부 제어 기능을 제공할 수 있도록 구성이 가능한 스마트 홈서비스 모델의 종류를 분류한다.

가. 스마트 게이트웨이 모델

홈 네트워크 내에는 다양한 종류의 스마트 기기들이 존재한다. TV, 냉장고, 에어컨 등과 같이 고유의 기능을 제공하는 백색 가전에 스마트 기능을 추가한 스마트 가전군과 온도, 습도, 먼지, 움직임 등을 감지하는 각종 센서 군이 존재하며, 카메라와 같이 상황을 인지할 수 있는 상황인지 모듈들이 존재한다.

이러한 다양한 스마트 기기들은 네트워크를 통해 기기 상태 정보 제공 및 제어 기능을 제공하고 있다. 그러나 이러한 기능들은 홈 네트워크라는 제한된 환경에서만 제공될 수 있으며, 홈 외부의 네트워크를 통한 제어 기능을 제공하지 못하고 있다[51]. 이러한 한계를 극복하기 위해 스마트 게이트웨이(Smart Gateway)와 스마트 홈 플랫폼을 이용하여 홈 내·외부의 상이한 통신 프로토콜을 변환하여 외부에서 홈 기기를 제어할 수 있는 방법을 제공할 수 있다.

그림 2.15는 스마트 게이트웨이 기반으로 스마트 홈서비스를 구성하는 모델을 보여준다.

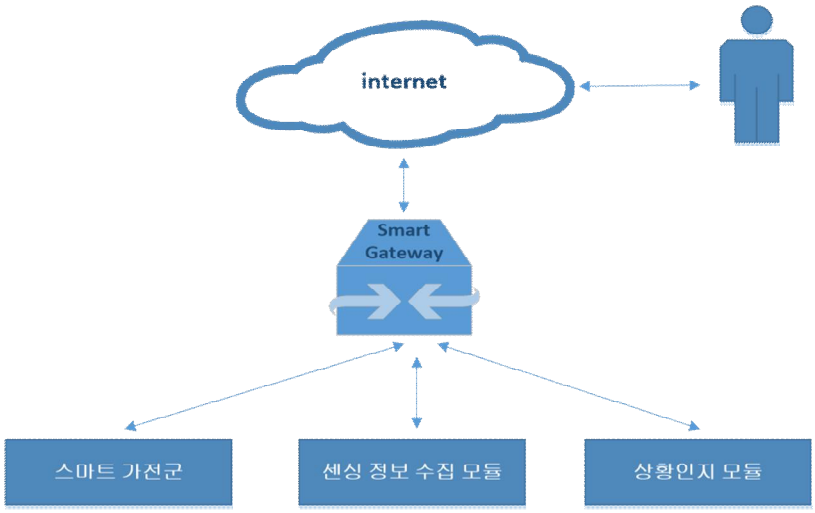


그림 2.15 게이트웨이 기반 스마트 홈서비스 모델
Fig 2.15 Smart Home Service Model under Gateway

홈 네트워크 내에 위치하는 스마트 게이트웨이는 홈 로컬 네트워크를 통해 스마트 가전, 센서, 상황인지 모듈 등의 스마트 기기 정보와 기기 상태 정보를 수신하여 스마트 홈 플랫폼으로 전송하고, 스마트 홈 플랫폼으로부터 송신한 제어신호를 해당 기기로 재전송하는 기능을 수행한다. 스마트 홈 플랫폼은 스마트 게이트웨이로부터 수신한 스마트 기기 정보와 기기 상태 정보를 저장하고, 스마트 기기들이 제공하는 기능별로 자원을 가상화하여 RESTful[52] API형태로 외부에 제공하게 된다. 이렇게 제공된 RESTful API를 이용하여 서비스 제공자는 다양한 종류의 스마트 홈서비스를 개발할 수 있다.

홈 네트워크 상에서 스마트 게이트웨이와 스마트기기간의 통신 프로토콜과 외부 네트워크에서 사용되는 통신 프로토콜이 서로 상이하다. 일반적으로, 홈 네트워크 상에서 동작하는 기존 스마트 가전들은 UPnP/DLNA 프로토콜을 주로 사용하며, 스마트 홈 플랫폼과 외부와의 연결은 RESTful 방식을 주로 사용한다. 이와 같이, 홈 내·외부의 상이한 통신 프로토콜 문제를 해결하기 위해 프로토콜 변환이 이루어져야 한다.

나. IoT Cloud 모델

홈 네트워크 내에는 스마트 홈서비스를 구성하기 위해 사용되는 다양한 종류의 스마트 기기 및 IoT기기들이 존재할 수 있다. 특정 스마트 기기들은 홈 내·외부에서 기기 상태 모니터링 및 제어가 가능하도록 별도의 IoT 클라우드를 통해 각각의 인터페이스를 제공하고 있다. 이러한 IoT 클라우드는 각각 지원하는 IoT기기의 상태 모니터링 및 제어만을 제공하고 있다. 따라서, 홈 네트워크 상의 서로 다른 IoT 클라우드에 종속된 스마트 기기 및 IoT 기기간의 협업 서비스에는 제한이 있다. 이러한 한계를 극복하기 위해, 각 IoT 클라우드에서 스마트 기기 상태 모니터링 및 제어를 위해 제공하는 인터페이스를 스마트 홈 플랫폼을 이용하여 단일화된 인터페이스를 제공하도록 하여 스마트 홈서비스 구성이 가능하다. 그림 2.16는 별도의 인터페이스를 제공하는 클라우드 서비스 기반으로 스마트 홈서비스를 구성하는 모델을 보여준다.

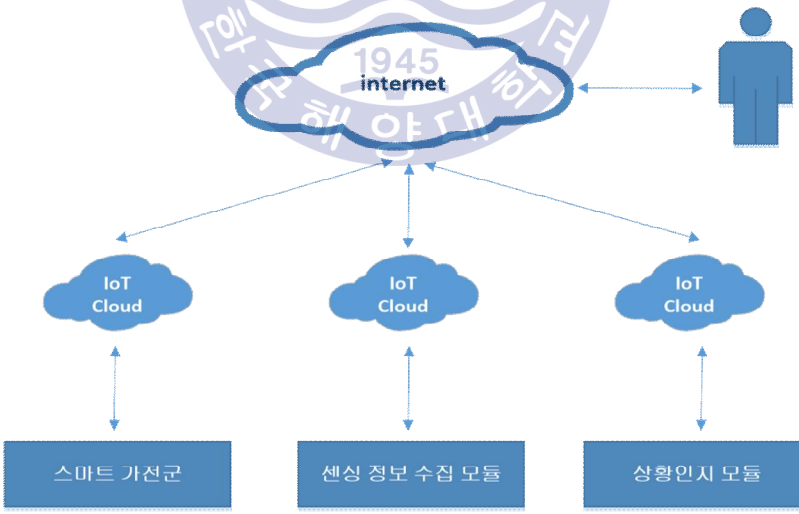


그림 2.16 IoT 클라우드 기반 스마트 홈서비스 모델
Fig 2.16 Smart Home Service Model under IoT Cloud

홈 네트워크 내의 스마트 기기 및 IoT기기들은 각각 해당 IoT 클라우드에 기기 상태 정보를 제공한다. 스마트 홈 플랫폼은 각 IoT 클라우드를 통해 홈 네트워크 내의 기기들의 상태 정보를 수신하고, 각 IoT 클라우드를 통해 해당 기기를 제어할 수 있다. 스마트 홈 플랫폼은 각각의 IoT 클라우드에서 제공하는 인터페이스를 통해 제공된 기기의 정보를 저장하고, 기기가 제공하는 기능별 자원을 가상화하여 단일화된 RESTful API 형태로 외부에 제공하게 된다. 이렇게 제공되는 API를 이용하여 서비스 제공자는 스마트 기기의 종류에 관계없이 다종의 스마트 홈서비스 개발이 가능하다.

이 경우에는 스마트 홈 플랫폼을 통해 서로 다른 IoT 클라우드에서 제공하는 기기의 정보와 인터페이스를 단일화된 RESTful API 형태로 제공하기 위한 인터페이스 변환 기술이 필요하다.

다. Hybrid 모델

하이브리드 스마트 홈서비스 모델은 앞서 기술한 스마트 게이트웨이 모델과 IoT 클라우드 기반 스마트 홈서비스 모델을 결합한 형태의 모델이다. 이 경우, 스마트 게이트웨이를 통해 상태정보 모니터링과 제어가 가능한 기기들은 스마트 게이트웨이를 통해 이용하고, 각각의 IoT 클라우드를 통해 이용이 가능한 기기들은 해당 IoT 클라우드를 통해 제어할 수 있도록 구성된다. 이를 위해 스마트 홈 플랫폼은 스마트 게이트웨이와 각 IoT 클라우드를 통해 전달되는 기기 정보들을 관리하고 있으며, 스마트 홈서비스에게는 홈 내 기기의 상태정보 이용 및 제어를 위해 동일한 형태의 RESTful API를 제공한다.

하이브리드 스마트 홈서비스 모델은 스마트 게이트웨이를 통한 UPnP/DLNA를 지원하는 기존 스마트 기기와 IoT 클라우드를 통한 IoT 기기를 제어를 위해 단일화된 RESTful API를 제공하기 때문에 상이한 제조사 및 프로토콜을 사용하는 기기간의 연동 및 제어가 가능하다.

그림 2.17은 하이브리드 형태의 스마트 홈서비스를 구성하는 모델을 보여준다.

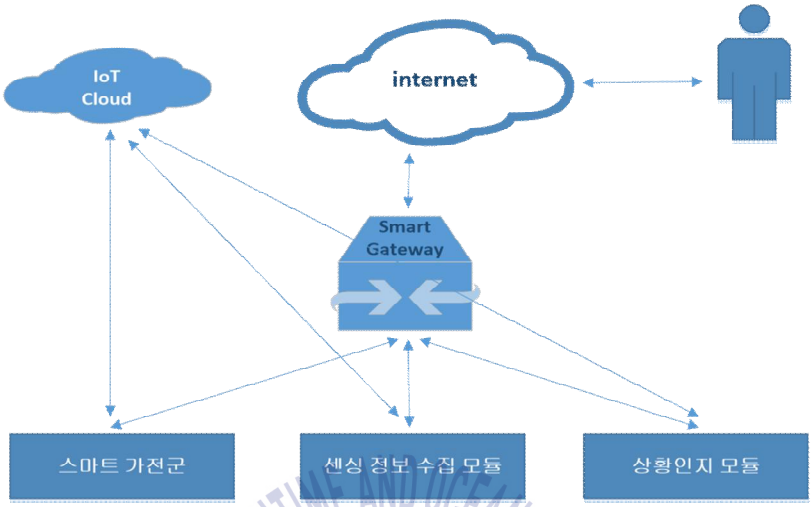


그림 2.17 하이브리드 스마트 홈서비스 모델

Fig 2.17 Hybrid Smart Home Service Model

제 3 장 스마트 홈 시스템 설계

3.1 시스템 요구 사항

이번 장에서는 가정에서의 주변 환경 정보를 수집하여 송수신하고 저장 및 피드백을 수행하는 사물인터넷 환경의 스마트 홈 시스템을 설계한다. 설계 내용으로는 기능명세와 상세 요구사항 명세이다.

3.1.1 시스템 기능 명세

일반적으로 사람들은 집에서 많은 시간을 보내기 때문에 주거 환경 정보를 획득하고 판단하여 최소한의 개입으로 대처할 수 있다면, 삶의 질이 좀 더 쾌적하고 안락한 생활 패턴으로 향상될 것이다. 이를 위하여 실내에 설치되는 시스템은, 센서 디바이스를 통해서 감지되는 환경 데이터로는 온습도, 조도, 가스, 불꽃, 수분, 동작 감지 데이터로서 주기적이거나 또는 정해진 이벤트 발생 시각에 해당 센서 모듈을 통해서 측정되고 수집된다. 수집된 데이터는 저장 및 분석과 피드백을 위해서 게이트웨이로 무선 네트워크를 통해 경량의 메시지 프로토콜을 사용하여 송수신할 수 있도록 한다. 게이트웨이에 수신된 데이터는 데스크톱이나 모바일 기기에 실시간으로 표현되고, 메신저를 통하여 위험 상황이나 관심정보를 알려주며, 개방형 인터넷 클라우드 서비스와 연동하여 데이터를 기록하고, 상황 정보에 따라서 부저 및 카메라와 같은 액추에이터를 자동적으로 동작하게 하는 일련의 피드백을 수행한다.

스마트 홈 시스템 서비스를 위한 피드백을 위해 필요한 기능을 서술하면 다음과 같다. 첫 번째는 정보의 가시화(visualization) 서비스이다. 가시화는 가장 기본적인 피드백의 형태라고 할 수 있다. 사물인터넷 환경을 통하여 수집한 정보를 데스크톱 또는 모바일 기기를 통해 실시간으로 정보를 보여주는 것이다. 예를 들어 현재 실내·외 온도, 습도, 대기상태를 텍스트 혹은 그래픽으로 가시화하는 것만으로도 사용자는 해당 정보를 이용하여 환경 최적화에 반영할 수

있을 것이다. 또는 위험한 상황의 화재나 외부침입을 즉시 알 수 있게 되어 피해를 최소화할 수 있을 것이다.

두 번째는 모니터링을 통한 알람(alarm) 서비스이다. 가시화 서비스는 사람이 상시적으로 주목하지 않는다면 필요 정보를 놓칠 가능성이 있지만 알람은 사물의 상태를 직관적으로 표현할 수 있는 방법이다. 예를 들어 사용자의 시각이나 청각을 자극할 수 있는 LED 또는 부저를 통하여 사람이 멀지 않은 장소에 있는 경우 사물의 상태를 사용자에게 전달하는 서비스이다. 혹은 사용자가 집에 없을 경우를 대비해서 이메일이나 트위터(twitter) 같은 인터넷 서비스를 활용하여 사용자의 스마트 폰으로 알람 메시지를 전송할 수도 있다.

세 번째는 자동제어서비스이다. 사람의 개입 없이 사물인터넷 환경의 스마트 홈서비스를 통해 액추에이터가 자동적으로 동작하거나 알림 서비스를 받은 사용자가 원격으로 조작할 수 있는 서비스이다. 예를 들어 주기적으로 화분에 물을 주는 것을 잊어 버렸을 경우 화분 토양의 수분을 계측하여 수분이 부족한 경우 워터펌프를 직접 동작시켜 물을 공급하는 것이다. 사람이 없는 경우 화재나 가스 누출이 발생한 경우 가스를 차단할 수 있다.

3.1.2 세부 요구사항 명세

본 논문은 스마트 홈서비스를 위해 가정내 환경 정보를 수집하여 사용한다. 각 센서의 감지 정보에 따른 상세 요구사항 명세는 다음과 같다.

첫째, 적절한 냉난방 온도 설정이나 생활에 필요한 정보를 위해서 실내/실외의 온도/습도와 날씨 및 불쾌지수 측정하여 가시화한다. 사람이 가장 쾌적하게 느낄 수 있고 건강한 환경을 유지하기 위한 계절별 온도와 습도는 표 3.1과 같다[53].

표 3.1 최적의 온도와 습도

Table 3.1 Optimum inside temperature and humidity

구분	여름	봄/가을	겨울
온도(°C)	24 ~ 28	19 ~ 23	18 ~ 20
습도(%)	60	50	40

온도와 습도를 통해 산출되는 불쾌지수 지표는 개인에 따라 차이가 있겠지만 표 3.2의 기준으로 온습도의 조절을 통하여 대처할 수 있게 한다. 불쾌지수의 산출 방법은 식 (1)과 같다[54].

표 3.2 불쾌지수 지표

Table 3.2 Discomfort Index Phase

단계	지수범위	설명
매우높음	80이상	전원 불쾌감을 느낌
높음	75~80 미만	50%가 불쾌감을 느낌
보통	68~75 미만	불쾌감을 나타내기 시작함
낮음	68미만	전원 쾌적함을 느낌

$$\text{불쾌지수} = \frac{9}{5}T - 0.55(1-RH)(\frac{9}{5}T - 26) + 32 \quad (T\text{:기온},\ RH\text{: 상대습도}) \qquad (1)$$

둘째, 식물 관리를 위한 내부 온도 및 수분 상태를 측정하고 알림 서비스하고 수분 공급을 원격으로 조작할 수 있다. 집 안팎에 화분을 돌보는 많은 사람들이 가장 어려워하는 것이 적절한 때 화분에 물을 주는 것이다. 이 서비스는 물주는 시기를 정확하게 사람이 알 수 있도록 하는 것이 목적이다. 화분의 수분을 측정하여 알려주면 집안에서는 사람이 직접 물을 주거나 먼 곳에 있는 경

우 외부에서 원격으로 적절한 수분을 공급할 수 있다. 고온 또는 영하의 날씨에 추워서 식물이 죽는 것을 막을 수 있도록 식물 주변 온도도 측정하여 관리할 수 있다.

셋째, 외부침입을 감시하기 위한 인체를 감지하고 모바일 메시지로 전송하고 웹 카메라를 통하여 집안의 상황을 모니터링 할 수 있다. 아파트 복도나 현관문 등에 부착되어 사람이나 움직이는 물체가 감지되면 조명을 켜주고 꺼주는 인체감지 센서를 많이 사용하는 것을 확인할 수 있을 것이다. 인체감지센서가 필요한 곳은 보안, 방문객통보, 조명관리, 그리고 최근의 사물인터넷 영역까지 확장되고 있다. 본 시스템에서는 부재중의 무단 침입자를 감지하여 주인에게 알람 서비스를 제공하는 것이 목적이다.

넷째, 화재 발생이나 가스 누출과 같은 위험한 상황에 대비하기 위한 불꽃 및 가스 감지하고 경고 알람 및 모바일 메시지를 전송하고 웹 카메라를 통하여 집안의 상황을 모니터링 할 수 있다. 화재 모니터링은 중요 문화재를 비롯하여 공공장소, 주택, 공장, 상가 등에서 많이 사용하고 있다. 최근에는 홈 사물인터넷에서도 집안의 화재감지를 측정하기 위해 많이 사용된다. 불꽃 감지 센서는 근접 화재경보 센서 기능으로 활용할 수 있고, 산업용 장비 또는 민간용 장비의 고전압 방전불꽃 또는 스파크 불꽃을 감시하고 발화체 및 점화장치의 점화 확인이 가능하다. 센서를 통하여 실시간으로 불꽃과 가스를 감지하다가 화재나 가스 누출 발생을 인식하게 되면 경고 알람 및 메시지를 전송한다.

다섯째, 전력낭비를 방지하기 위해서 집안 밝기의 상태와 인체를 감지 및 상황에 맞는 동작을 수행한다. 가정 내부에 사람이 없는 상황에서 전등이 켜져 있는 상황을 대비한 것이다. 일정 시간동안 사람의 동작을 감지 후 실내의 전등을 자동적으로 끄거나, 서비스 이용자에게 알람 서비스를 제공하여 직접 방에 있는 전등 스위치를 사용하지 않더라도 원격으로도 소등을 할 수 있다.

3.2 스마트 홈 시스템 구성요소

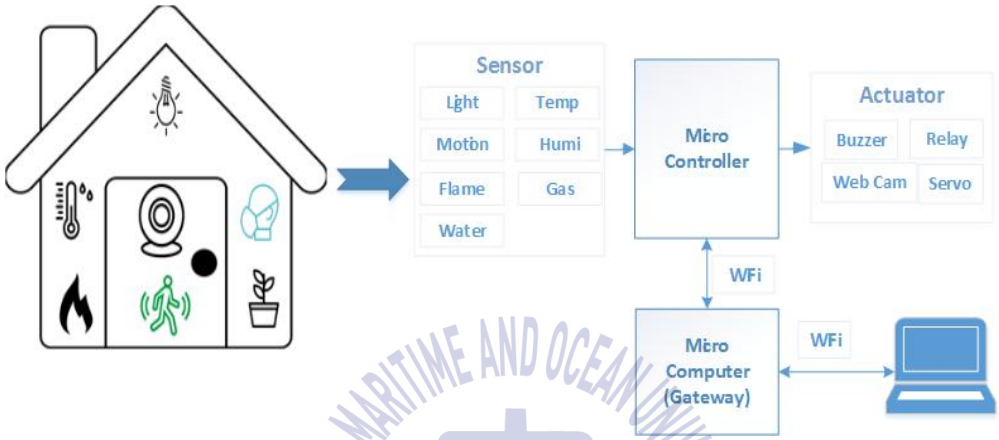


그림 3.1 제안 시스템 구성 요소
Fig 3.1 Home Control System Component

그림 3.1은 본 논문에서 제안하는 스마트 홈 시스템의 전체 구성요소를 나타내고 있다. 각 구성요소를 정리해보면 우선 센서 디바이스가 있다. 사물인터넷이 화제가 되는 배경은 마이크로 컨트롤러 보드의 변화와 관련이 있다. 이전에 마이크로 컨트롤러 보드를 네트워드에 연결하기 위해서는 개발자가 직접 인터페이스를 구현해야 했다. 하지만 최근에는 네트워크 연결 기능을 외부 접속 모듈로 제공하는 유형과 표준으로 내장된 유형의 오픈소스 하드웨어 플랫폼이 많아지면서, 디바이스를 쉽게 네트워크에 직접 연결되거나 사물인터넷 게이트웨이를 통해서 데이터를 전송할 수 있게 되어 훨씬 신속하고 저렴하게 시스템을 구축할 수 있다. 사물인터넷 환경에서 사물에 해당하는 센서 디바이스는 하나 이상의 센서 모듈과 액추에이터가 연결되어 있는 마이크로 컨트롤러로, 집에서 발생하는 온도, 습도, 조도, 수분, 가스, 불꽃과 같은 환경 데이터를 수집하고, 액추에이터는 센싱된 정보에 따라 상황에 맞는 알람을 동작이나 원격에서 사용

자의 명령에 따른 전원을 켜고 끄는 동작을 수행하는 부저와 릴레이 등이 있다.

두 번째 구성요소는 마이크로컴퓨터로 표현된 게이트웨이이다. 스마트 홈 시스템에서 게이트웨이는 사물인터넷 환경에서 사물과의 연결, 데이터의 공유와 정보의 활용 측면에서 중요한 역할을 담당한다. 센서 디바이스와 게이트웨이는 와이파이를 지원하여 유선 디바이스가 갖는 설치의 불편함을 없애고 사물의 배치나 위치 변경이 용이하여 데이터의 계획적인 수집, 분석, 저장과 효율적이고 안정적인 사물의 연결성을 제공한다. 즉, 센서 디바이스로부터 수집된 환경 정보는 실시간으로 무선 통신을 사용하여 사물인터넷 게이트웨이로 전송되는 것이다. 또한, 게이트웨이는 센서 디바이스에 비해 컴퓨팅 능력, 메모리, 운영체제, 고급 언어 사용으로 메인 처리 장치로 리눅스 운영체제 기반의 마이크로컴퓨터를 사용한다[55]. 게이트웨이의 주요 구성요소로는 소프트웨어와 하드웨어, 외부 네트워크 인터페이스, 디바이스 연결 인터페이스로 구성된다. 사물인터넷에서 이용되는 디바이스는 인터넷에 직접 접속할 수 없는 것도 있다. 이와 같은 경우, 디바이스와 인터넷 간의 중계 역할을 게이트웨이가 담당한다. 중계 및 서버 역할을 수행하는 스마트 게이트웨이로서 데스크톱 또는 서버 컴퓨터를 포함한 여러 가지 선택 사항이 있지만 사물인터넷 솔루션은 단일 보드와 같은 소형 컴퓨팅 장치를 사용하는 경향으로 인해 운영체제가 사용 가능한 라즈베리 파이가 게이트웨이로 사용되었다. 라즈베리 파이는 메모리의 확장 용량이 부족하고 CD, DVD 및 하드 드라이브와 같은 온보드(on-board) 장치는 수용할 수는 없지만 USB, 블루투스, 와이파이 등과 같은 디바이스 연결 인터페이스를 지원하고, 외부 네트워크 인터페이스로는 이더넷, 와이파이를 지원한다. 하드웨어는 게이트웨이의 목적에 맞게 성능을 고려할 필요가 있는데 현재 최근까지 출시된 라즈베리 파이는 높은 성능과 낮은 비용, 저전력과 소형화를 요구하는 사물인터넷에서 사용하기 매우 적합하다. 운영 소프트웨어는 라즈비안 리눅스를 기반으로 최근까지 지속적으로 여러 차례 버전 업이 되었다. 라즈비안은 게이트웨이 제어를 위한 라이브러리와 여러 프로그래밍 언어를 지원하고, 게이트웨이 운영에 필요한 추가적인 언어와 데이터베이스서버, 중계서버, 개발도구 소프트

웨어를 설치하여 사용한다.

마지막 구성요소는 인터페이스이다. 인터페이스는 수집 정보를 사용자에게 전달하는 구성 요소이다. 사용자는 일반적으로 데스크톱이나 모바일 기기와 같은 스마트 디바이스를 통해서 홈의 주변 정보를 확인할 수 있다. 본 시스템에서는 게이트웨이에 구현되어 있는 가시화 서비스나 개방형 클라우드 서비스와 같은 인터페이스를 통해서 사물인터넷 환경의 정보에 접근할 수 있다.

이 장에서 설계하고 있는 시스템은 장소와 시간에 구애받지 않고 실시간으로 가정에서 생산되는 다양한 센싱 정보를 효율적으로 생산, 중계 및 소비할 수 있는 사물인터넷 기반의 시스템이다. 단순한 단방향 통신이 아닌 사물과 사물, 사물과 사람의 실시간 양방향 통신을 구현하기 때문에 사물 모니터링과 사용자로부터 명령을 주고받기 위한 인터페이스와 게이트웨이 그리고 센서 디바이스가 요구되고 구성 요소와 주요 기능은 표 3.3과 같다.

표 3.3 스마트 홈 시스템 구성요소와 주요 기능
Table 3.3 Smart Home system components and features

구성요소	주요기능
센서 디바이스	• 센서 모듈을 통한 환경 정보 수집·전송 • 액추에이터 제어
게이트웨이	• 사물들을 유연하게 연결하는 기능으로 메시지에 대한 중계, 공유 및 활용 • 추가적인 개발 환경 제공 (개발도구, 프로그래밍 언어 등) • 데이터베이스 서버 및 중계 서버 수행
인터페이스	• 웹브라우저 및 모바일 인터페이스 • 개방형 연결 인터넷 서비스

3.3 스마트 홈 시스템 설계

시스템 구성 요소와 각 기능을 종합한 스마트 홈 시스템의 구조는 그림 3.2와 같다.

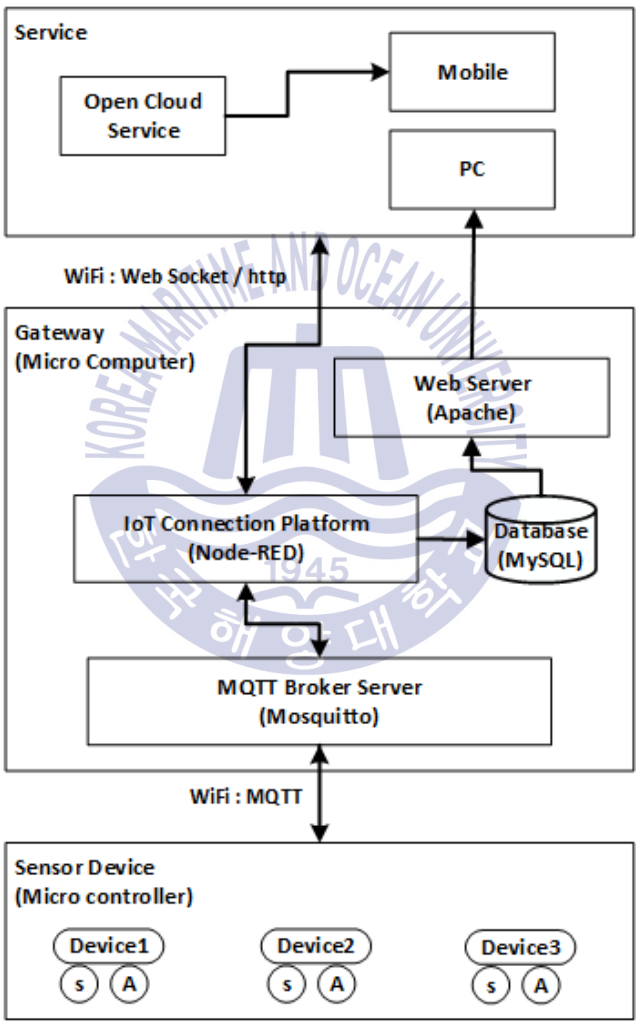


그림 3.2 제안 시스템 구조

Fig 3.2 Architecture of proposed system

게이트웨이는 외부 인터넷망과 무선 홈 네트워크를 경제적이고 안정적으로 연결하여 집 주변의 환경 정보를 공유하고 활용하는 환경을 제공한다.

게이트웨이 내부의 브로커 서버는 연결된 센서 디바이스들의 중개 서버로서 메시지 버스를 이용해 사용자와 사물 간, 사물과 사물간의 통신을 지원한다. MQTT의 메시지 전송 통로인 메시지 버스는 topic을 만들어 일대일, 일대다, 다대다의 메시지 전송을 브로커 서버가 수행한다. 또한, MQTT와 웹 소켓 프로토콜간의 실시간 양방향 메시지 중개 역시 담당하고 있다. 본 연구의 시스템에서는 Mosquitto라는 오픈소스 브로커 서버를 사용한다.

게이트웨이에 설치되어있는 Node-RED는 사물에 대한 데이터 관리, 데이터 시각화, 디바이스 제어, 매시업 서비스를 제공하는 오픈소스 소프트웨어로 사물 연결 미들웨어의 역할을 수행한다. 사물과 사용자의 다양한 연결에 응답하기 위해 사물의 연결성을 제공하고 브로커와 송수신되는 데이터를 데이터베이스에 저장하고 시각화하여 정보를 표현한다. 또한 개발자에게는 사물인터넷 애플리케이션을 개발 도구의 기능을 제공하기도 하는데, 다양한 디바이스 및 프로토콜과의 연동을 간단하게 작성할 수 있게 하는 강점이 있다. 수집된 많은 데이터의 가공 및 그 결과의 시각화가 Node-RED의 주요 기능이다.

인터페이스는 로컬이나 원격에서 사물에 대한 정보를 접근하여 활용하는 방법으로 인터넷 브라우저와 모바일 디바이스를 사용하여 사물인터넷 서비스를 이용할 수 있게 한다.

3.3.1 센서 디바이스 설계

그림 3.3, 3.5, 3.7, 3.9는 제안 시스템에서 사용하는 센서 디바이스의 하드웨어 설계를 보여준다. 센서 디바이스는 무선 통신이 가능한 디바이스들로서 정보를 수집하여 전송하는 발행자 역할을 수행한다. 센서 디바이스의 공통적인 주요 기능은 다음과 같다.

- 센서 디바이스에서 센서 모듈을 연결하는 컨트롤러는 아두이노로 무선 모듈이 장착되어 무선통신인 와이파이 통신을 사용할 수 있다.

- 각 디바이스에 연결된 센서 모듈은 집의 주요 장소에 배치되어 실시간으로 주변의 온도, 습도, 모션, 조도, 수분, 화염, 가스 정보를 획득하여 수집된 정보를 게이트웨이로 MQTT 프로토콜을 통해 전송한다.
- 디바이스에 연결된 부저, 서보모터, 릴레이와 같은 액추에이터는 정보의 상황이나 사용자의 요구에 따라 인간의 개입 없이 자동으로 동작하거나 원격 명령으로 동작을 수행한다.

센서 디바이스 1의 하드웨어 설계는 그림 3.3과 같고 수집정보는 온도와 습도, 조도, 모션 탐지 정보이고 액추에이터는 릴레이를 갖도록 설계하였다. 릴레이는 사람이 없는 상황에서의 조명 상태를 제어하고자 할 때 사용한다.

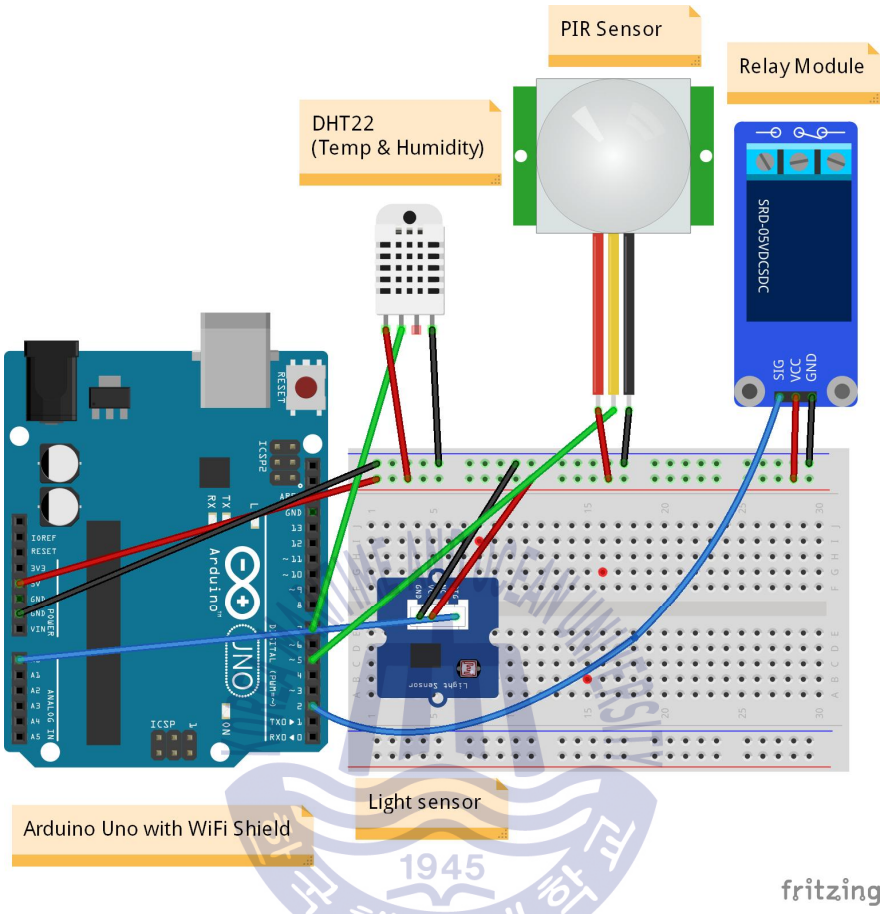


그림 3.3 센서 디바이스 1 설계

Fig 3.3 Design of Sensor Device 1

센서 디바이스 1의 메인 프로그램은 그림 3.4와 같이 동작한다.

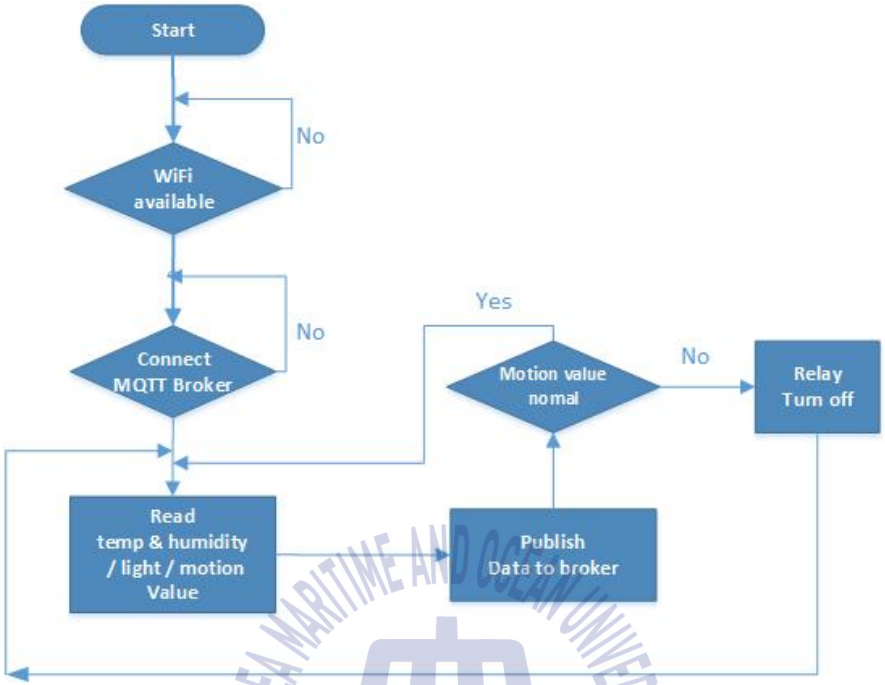


그림 3.4 센서 디바이스 1 프로그램 순서도

Fig 3.4 Flow chart of program for Sensor Device 1

센서 디바이스 2의 하드웨어 설계는 그림 3.5와 같고 수집정보는 온도와 습도, 수분 측정값이고, 액추에이터로는 릴레이를 사용하여 가정내의 설비를 On/Off할 수 있는 모듈로 설계하였다.

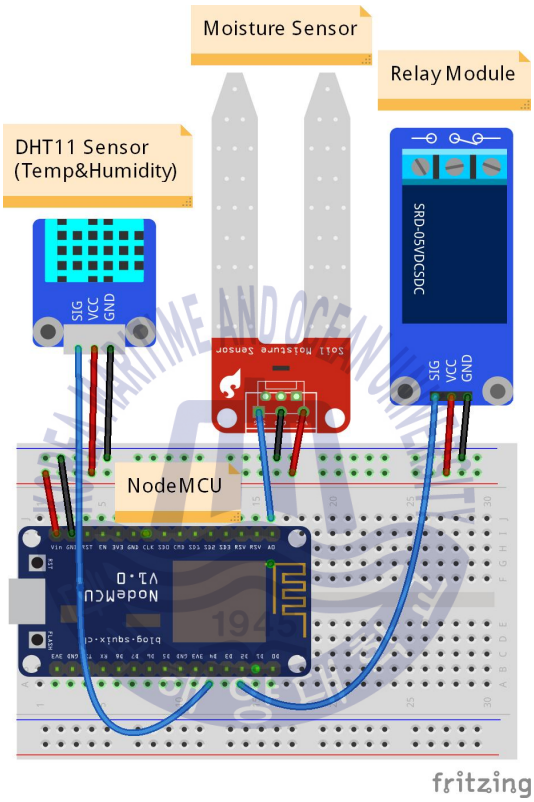


그림 3.5 센서 디바이스 2 설계

Fig 3.5 Design of Sensor Device 2

센서 디바이스 2에 설치된 메인 프로그램은 그림 3.6과 같이 동작한다.

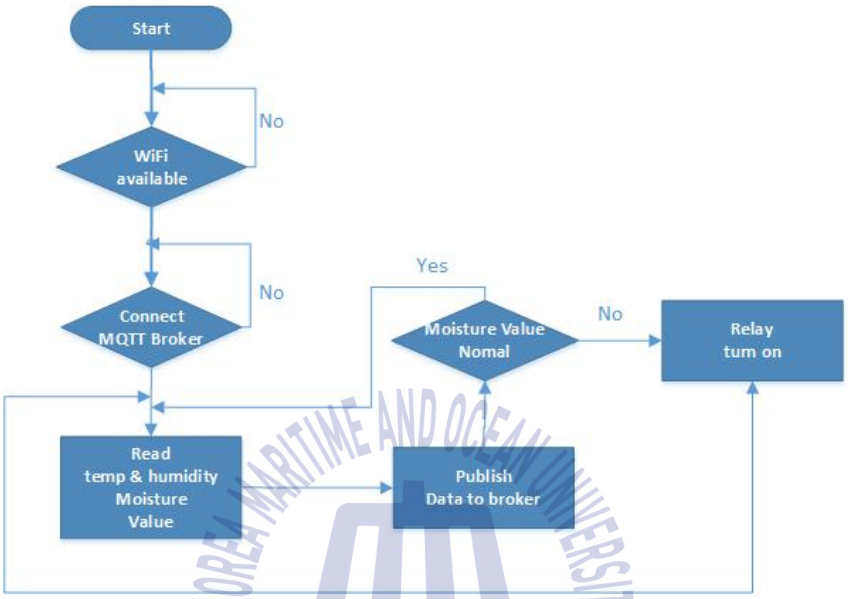


그림 3.6 센서 디바이스 2 프로그램 순서도

Fig 3.6 Flow chart of program for Sensor Device 2

센서 디바이스 3의 하드웨어 설계는 그림 3.7과 같고 수집정보로는 적외선 인체감지 정보, 불꽃감지 정보와 가스탐지 정보이며, 화재 발생이나 무단침입의 상황에 대처하기 위해 부저가 동작하도록 설계하였다.

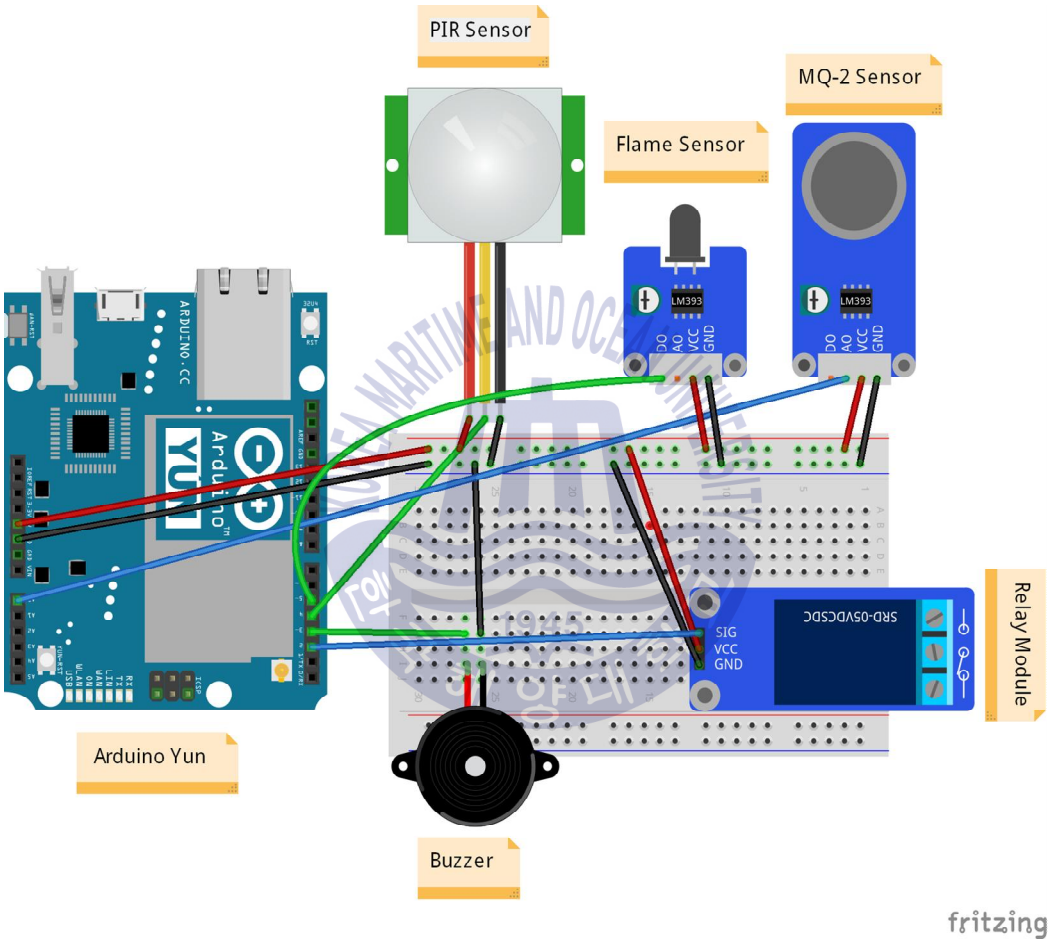


그림 3.7 센서 디바이스 3 설계

Fig 3.7 Design of Sensor Device 3

센서 디바이스 3의 메인 프로그램의 동작은 그림 3.8과 같다.

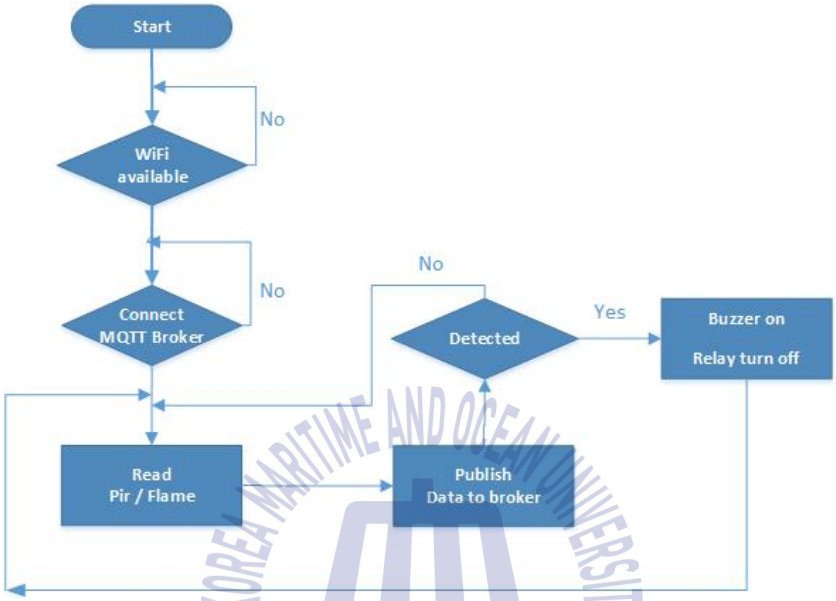


그림 3.8 센서 디바이스 3 프로그램 순서도
Fig 3.8 Flow chart of program for Sensor Device 3

3.3.2 게이트웨이 설계

게이트웨이는 사물인터넷 기반의 스마트 홈 시스템에서 센서 관리, 실시간 데이터 처리, 응용서비스 연계 그리고 보안과 같은 중요한 역할을 담당한다. 게이트웨이는 마이크로 컴퓨터와 통신 모듈로 구성된 하드웨어, 하드웨어를 효율적으로 운영하기 위한 운영체제, 센서 및 자원을 관리하기 위한 미들웨어와 센서로부터 수집된 정보를 가시화 단계로 전송하기 위한 애플리케이션으로 구분된다. 사물들을 유연하게 연결하는 기능으로 메시지에 대한 중계, 공유 및 활용한다. 여기에서는 사물인터넷 기술을 접목한 시스템을 구현하기에 최적화되어 있고 저가형이지만 다양한 입출력 주변 장치를 지원하고 소형 컴퓨터 기능을 모두 갖추고 있는 라즈베리 파이 2를 사용한다.

게이트웨이의 주요 기능은 다음과 같다

- 게이트웨이의 메시지 송수신을 위해서 MQTT를 사용하는데 MQTT 자체는 메시지를 어떻게 보낼 것인지를 정의하는 규약일 뿐, 실제 MQTT를 동작시키기 위해서는 서버 역할을 해주는 프로그램이 필요하다. 따라서 MQTT 프로토콜을 지원하는 메시지 브로커 서버인 Mosquitto를 설치하여 운영한다. Mosquitto 오픈소스는 효율적이고 경량의 통신으로 양방향 pub/sub 메시징 서비스가 가능하고 이기종 플랫폼 간에 개발이 용이하다. 또한 웹브라우저에 정보를 제공하기 위해 HTTP 프로토콜 대신에 웹 소켓을 지원하여 Mosquitto의 웹 소켓 서버를 통해 웹 브라우저를 이용해서 실시간 정보를 가시화한다.
- 사물인터넷 하드웨어 플랫폼과 온라인 서비스를 동시에 다루기 위해서 응용개발 도구인 Node-RED를 사용하여 기본적으로 게이트웨이를 통하여 수집된 정보는 그대로 또는 재해석하여 화면에 텍스트나 그래프형태로 시각화한다. 해당 정보는 스마트폰 앱을 통해서 정보를 확인하거나 경우에 따라 액추에이터를 구동할 수도 있다. Node-RED는 게이트웨이에서 사물 연결 플랫폼 역할을 수행한다. 게이트웨이에 연결된 센서, 통신, 자원 등 모든 것이 추상화된 사물 연결 플랫폼인 Node-RED를 통해서 서비스가 가능하다. 또한 MySQL 데이터베이스 서버와 개방형 클라우드 서비스에 정보를 저장하고, 중요한 정보는 트위터를 통해 사용자에게 알람 서비스를 전달한다. 카메라를 통해서 수집된 영상 정보는 외부에서 이메일과 메신저로 실시간 확인할 수도 있다.

MQTT는 아두이노나 라즈베리 파이 같은 임베디드 장치간 통신을 위한 가벼운 메시징 프로토콜이다. 본 시스템을 구축하는데 안정적인 데이터 수집을 위해서 센서 디바이스와 게이트웨이 사이의 통신은 사물인터넷 표준 프로토콜인 MQTT를 적용하여 통신량을 줄일 수 있도록 하였다.

본 시스템에서는 다른 상용제품에 비해 성능이 우수하고 오픈소스로 활용할

수 있는 Mosquitto MQTT broker를 사용한다. 그림 3.9은 제안한 시스템에서 사물인터넷 센서 디바이스와 사물인터넷 게이트웨이 사이에 데이터를 송·수신하기 위한 MQTT의 발행/구독의 모델 구조를 보여준다. 앞서 언급하였듯, MQTT 프로토콜은 토픽 구조로 정보를 전송하게 된다. 라즈베리 파이에 설치된 MQTT 브로커에 수신을 희망하는 정보의 구독 토픽을 등록하면, 아두이노 센서 디바이스에서는 센싱된 정보를 정해진 토픽으로 MQTT 브로커로 발행한다. MQTT 브로커는 구독을 신청한 곳에 정보를 중계한다. 여기에서는 Node-RED에서 정보 구독을 신청하고 발행된 정보를 받아서 관리 및 모니터링을 수행한다.

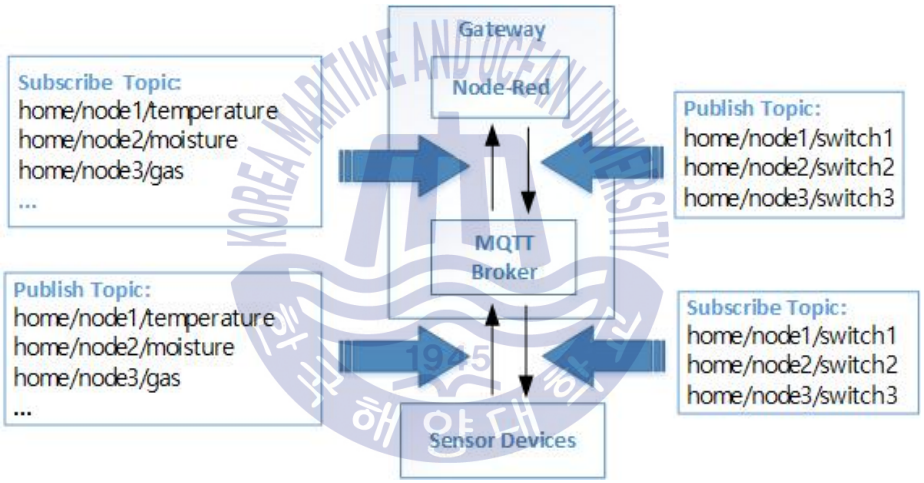


그림 3.9 발행/구조 모델 구조
Fig 3.9 Publish/Scribe model structure

안정적인 사물인터넷 데이터 수집을 위해서 주제를 분류하여 MQTT 토픽을 설계하였다. 각 송수신하는 항목별로 주제를 분석하여 원하는 동작을 수행한다. 표 3.4처럼 크게 Publish는 각 항목별로 건물/디바이스/주제로 분류하였고 Subscribe는 건물/디바이스/액추에이터 종류의 주제로 하였다.

표 3.4 MQTT 토픽 설계

Table 3.4 Publish/Subscribe Sensing Value per Topic

Pub/Sub	Topic	Sensing Value	Sensor
Pub	home/node1/temperature	실내 온도	센서 모듈
	home/node1/humidity	실내 습도	
	home/node1/light	실내 조도	
	home/node2/temperature	화분 주변 온도	
	home/node2/humidity	화분 주변 습도	
	home/node2/moisture	화분 수분	
	home/node3/flame	화재	
	home/node3/gas	가스	
	home/node3/motion	모션	
Sub	home/node1/switch1	전원 스위치	액추에이터 모듈
	home/node2/switch2	펌프 스위치	
	home/node3/switch3	가스 스위치	

그림 3.10은 전체 시스템에서 데이터를 전송하거나 전송받는 메시지의 흐름을 나타낸다. 기본적으로 MQTT 프로토콜은 메시지를 발행하고 특정 토픽을 구독하는 구조이다. 발행자나 구독자는 모두 브로커에 연결되어 동작한다. 하나 이상의 발행자와 구독자가 동시에 메시지를 보내고 받을 수 있다. 따라서 MQTT 브로커를 중심으로 발행자와 구독자가 동시에 동작되는 모델을 설계해야 한다.

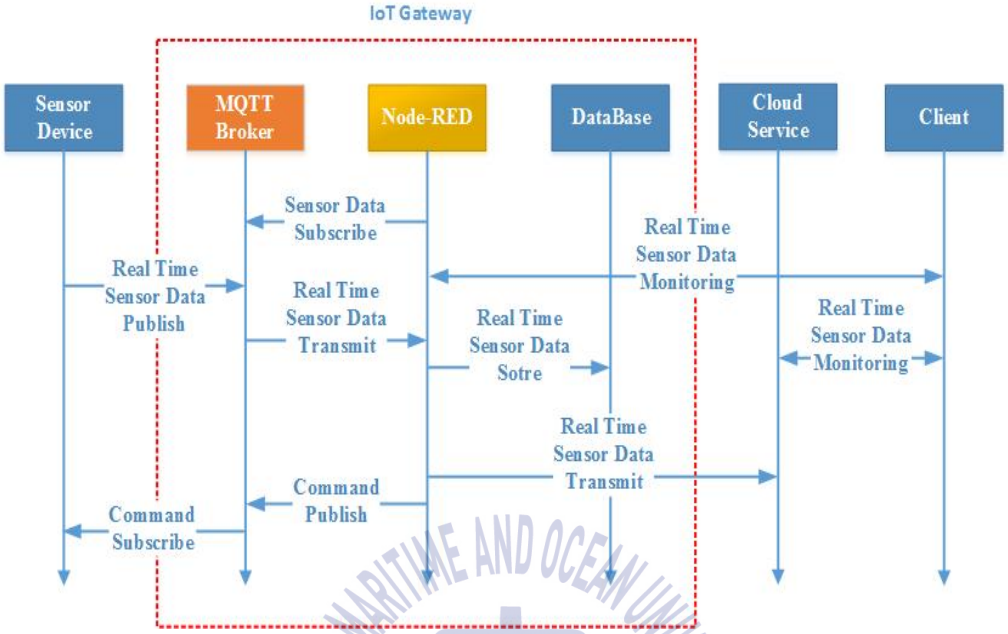


그림 3.10 시스템의 메시지 흐름
Fig 3.10 Message flow of the system

MQTT 브로커를 중계자로 한 클라이언트(Sensor Device, Node-RED) 간의 통신은 다음과 같이 4단계로 이루어진다.

첫 번째 단계는 연결 단계로서 클라이언트와 브로커 간의 연결을 맺는 것이다. 여기에서는 계정 관리에 필요한 아이디와 패스워드는 클라이언트 연결시에 요구하지 않도록 한다.

두 번째는 토픽 등록 단계로서 메시지 수신을 위한 클라이언트에 대한 것으로 수신하고자 하는 메시지의 토픽을 등록하는 것이다. 하나 이상의 토픽을 브로커에 등록해 해당하는 토픽의 메시지를 수신할 수 있다.

세 번째는 송수신 단계로 송신 클라이언트(Sensor Device)는 토픽과 메시지를 전송하면 앞서 토픽을 등록한 수신 클라이언트(Node-RED)에게 메시지가 전달된다. 이때 메시지의 송수신을 보장하기 위해 QoS 레벨을 함께 전송함으로

써 송신 클라이언트, 브로커, 수신 클라이언트 간에 트랜잭션 보장을 준수하게 한다.

네 번째는 연결 해지 단계로서 더이상의 메시지 송수신이 필요 없을 경우 토픽 등록을 해지하고 클라이언트와 서버 간의 연결을 종료하는 것이다. 구현된 시스템에서는 계속적으로 모니터링을 수행하는 것으로 네 번째 단계는 수행하지 않도록 한다.

3.3.3 인터페이스 설계

제안하는 스마트 홈 시스템의 서비스 마지막 단계는 데이터의 소비 단계로 정보를 확인하는 것이다. Node-RED에서 제공하는 시각화를 통한 데이터의 소비 외에도 클라우드 서비스를 통해서 데스크톱이나 스마트폰을 통한 시각화가 가능하다. 해당 서비스는 인터페이스 설계에 의해 수행하는데, 이 서비스의 구현은 개방형 클라우드 플랫폼인 ThingSpeak를 이용한다. ThingSpeak는 인터넷을 통해 실시간으로 센서의 데이터를 저장하고, 이를 웹 서비스와 연동, 사용자에게 데이터를 전송, 시각화하는 기능을 제공한다. 센서는 데이터를 클라우드에 저장하고, 해당 데이터는 지원 가능한 서비스를 이용하여 웹 서비스와 쉽게 연동할 수 있고, 앱을 구성하여 자신만의 애플리케이션을 구축할 수도 있다. 현재 개발자 버전은 무료로 지원하며, 다양한 외부 채널과의 결합이 장점이다. 정보를 제공할 수 있도록 전체 시스템을 프로그램하고 관리하는 관리자와 환경 정보를 조회하는 일반 사용자의 접근이 가능하다.

제 4 장 스마트 홈 시스템 구현

4.1 사물인터넷 센서 디바이스 구현

이번 장에서는 본 논문에서 제안한 사물인터넷 환경의 스마트 홈 시스템을 구현한다. 앞서 설계한 내용을 기반으로 구현된 스마트 홈 시스템의 센서 디바이스는 집 내부에 배치되어서 디바이스별로 연결된 센서 모듈과 프로그램에 의해서 정해진 주변 데이터를 수집하고 데이터의 종류에 따라 정해진 토픽으로 MQTT 브로커로 실시간 전송한다. 센서 디바이스는 아두이노, 센서 모듈, 액추에이터 모듈, 프로그램으로 구성된다.

4.1.1 하드웨어 구현

하드웨어는 오픈소스 하드웨어 플랫폼인 아두이노를 사용하여 구현하였다. 별도의 운영체제 없이 구동할 수 있는 아두이노는 생산성과 소형화가 용이하므로 사물인터넷의 사물로 적용하기 적합하다.

본 논문에서는 사물인터넷 환경에서의 실시간 사물 연결과 공유 및 활용을 테스트하기 위해 하드웨어를 스마트 홈에서 일반적이라 할 수 있는 대표적인 센서 모듈들로 구성하여 테스트베드를 만들어서 실험하였다.

첫 번째 센서 디바이스는 그림 4.1처럼 구성하였고, 구현된 하드웨어의 사양은 표 4.1과 같다. 센서 디바이스의 MCU는 아두이노 우노 보드에 와이파이 쉴드를 적층해 인터넷에 무선으로 연결한다. 센서 모듈은 DHT22 온습도 센서를 사용하여 실시간으로 데이터를 수집하고 온도와 습도를 활용하여 불쾌지수를 계산하여 온도, 습도, 불쾌지수를 브로커로 전송한다. 조도 및 모션 센서 모듈은 실내의 밝기와 사람의 움직임을 감지하여 실내 조도의 데이터를 해당 토픽으로 브로커로 전송하고 사람이 없는 상태에서 불이 켜져 있는 것을 확인하고 릴레이 모듈을 통해 실내 조명을 오프시킨다.

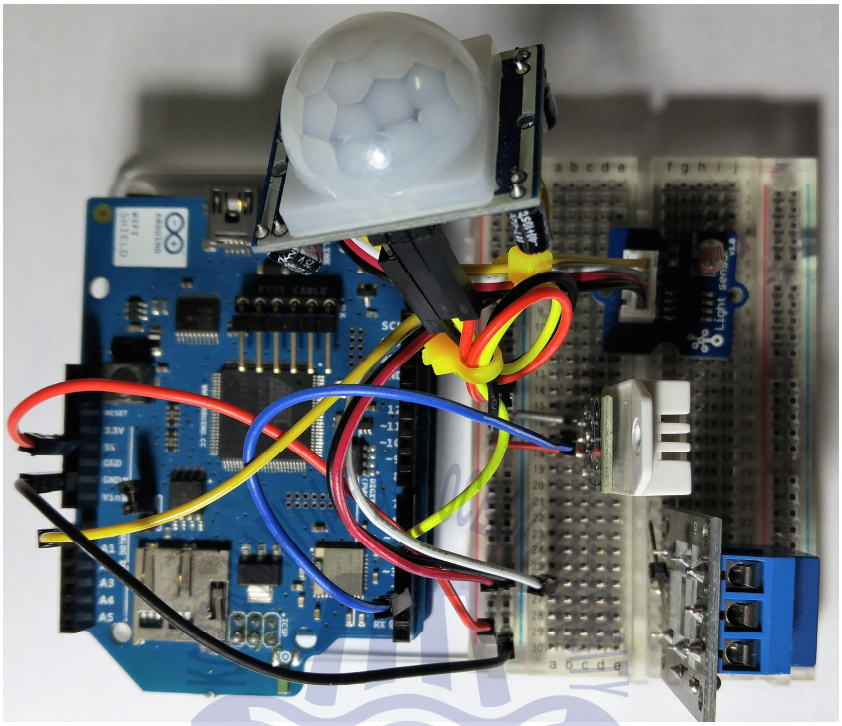
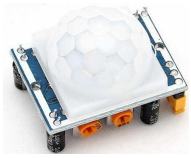


그림 4.1 센서 디바이스 1
Fig 4.1 Sensor device 1

표 4.1 센서 디바이스 1의 센서/액추에이터 모듈 상세
Table 4.1 Specification of Sensor Device 1

구분	기능	이미지
온·습도 (DHT22)	온도와 습도 정보를 동시에 얻을 수 있는 다기능 센서 모듈로 주변의 조건이 -40 ~ 80° C 사이의 온도와 0 ~ 100% 사이의 습도 일 때 정확한 값이 측정한다. 측정 온도 오차는 0.5° C 측정 습도 오차는 2% 이다. 여기에서는 실내의 온도와 습도를 측정한다.	
조도	빛 자체 또는 빛에 포함되는 정보를 전기신호로 변환하여 감지하는 센서 모듈로 특징은 비접촉, 비파괴, 고속도, 주변에 잡음의 영향을 주지 않고 할 수 측정한다.	
적외선	인체는 36.5도의 열을 가지고 있으므로 적외선이 약 10㎛의 파장이 나온다. 이것을 감지하여 인체의 움직임이 감지하는 모듈이다.	
Relay	릴레이 모듈은 평범한 디지털 On/Off 스위치로, 높은 전압과 낮은 전압의 5V 컨트롤러의 회로를 제어할 수 있다.	

두 번째 센서 디바이스의 구성은 그림 4.2와 같고, 구현된 하드웨어 사양은 표 4.2와 같다. 센서 디바이스의 MCU는 아두이노 윤 보드를 사용하였다. 아두이노 윤 보드는 아두이노 우노보다 성능면에서 우수하다. 내부적으로 아두이노 아키텍처와 리눅스를 결합한 유·무선 통신 모듈이 결합된 보드로 USB 통신 뿐만 아니라 이더넷, 와이파이를 통해 프로그램이 가능하다. 이 디바이스에서는 아두이노 윤 보드의 기능을 최대한 활용하여 리눅스에 영상 서비스가 가능하도록 설정하고 USB 인터페이스를 통해 웹캠을 연결하여 정지 영상 및 실시간 영상 스트리밍 서비스를 제공한다.

MQ2 가스 센서는 주변의 가스 누출에 대한 감지하여 해당 토픽으로 브로커로 전송한다. 가스 누출이 감지되면 부저를 통하여 알람 서비스를 제공한다.

침입 감시와 화재 감시를 위해 모션 및 불꽃 센서 모듈을 통해서 해당 정보가 감지되었을 경우 영상 정보를 파일로 저장하고 사용자에게 알람 서비스를 제공한다. 뿐만 아니라 원격에서 집안 내부를 웹 브라우저나 모바일폰 브라우저 통해서 직접 스트리밍 서비스가 제공된다.

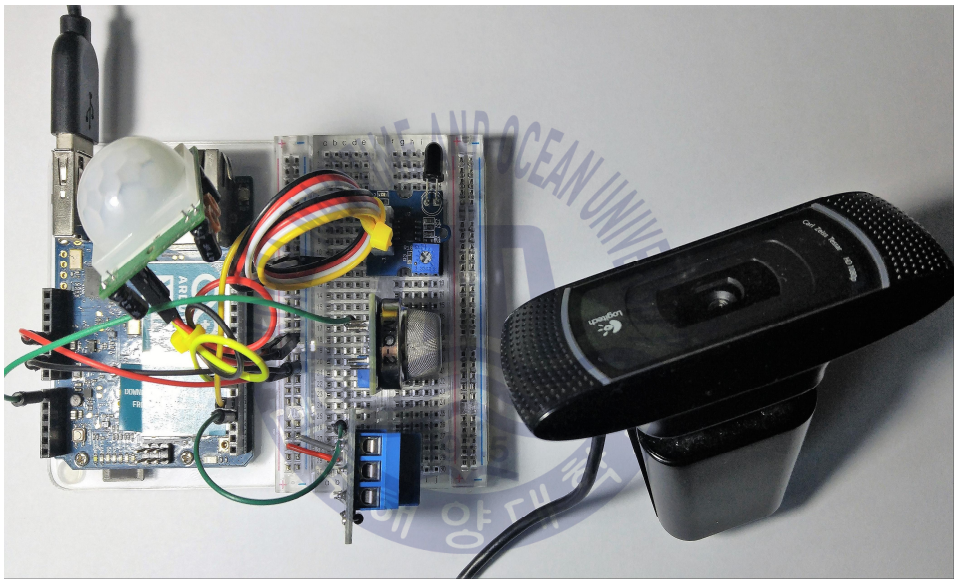
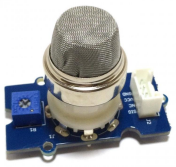


그림 4.2 센서 디바이스 2
Fig 4.2 Sensor device 2

표 4.2 센서 디바이스 2의 센서/액추에이터 모듈 상세

Table 4.2 Specification of Sensor Device 2

구분	기능	이미지
불꽃	화재의 근원 또는 760nm - 1100nm 범위의 파장을 갖는 빛의 근원을 검출할 수 있다. 그림에 보이는 검은색 부분은 적외선에 민감하다. 모듈의 가변저항을 이용하여 감도를 조절할 수 있다.	
가스 (MQ2)	가스 센서는 통상적으로 한가지 이상의 가스를 탐지할 수 있는데, MQ2 모듈은 내부에 포함된 히터와 센서로 H2, LPG, CH4, CO, Alcohol, Smoke or Propane을 검출해낼 수 있는 모듈이다.	
적외선	인체는 36.5도의 열을 가지고 있으므로 적외선이 약 10㎛의 파장이 나온다. 이것을 감지하여 인체의 움직임이 감지하는 모듈이다.	
Buzzer	디지털 및 아날로그 PWM 출력으로 소리를 낼 수 있다. 여기에서는 가스 센서의 값에 따라 소리를 출력하여 내부에서 경보 기능을 제공한다.	
Relay	릴레이 모듈은 평범한 디지털 On/Off 스위치로, 높은 전압과 낮은 전압의 5V 컨트롤러의 회로를 제어할 수 있다.	

세 번째 센서 디바이스는 그림 4.3처럼 구성하였고, 구현된 하드웨어 사양은 표 4.3과 같다. 센서 디바이스의 MCU는 NodeMCU로 사물을 소형화하기 위해 디자인된 ESP8266 WiFi 기반의 개발보드이다. 센서 모듈은 화분 주변의 온습도 정보와 화분의 흙에 수분 상태를 실시간으로 데이터를 수집하여 브로커로 전송한다. 만약 수분이 부족한 경우 트위터로 정보를 보내주고 원격에서 릴레이를 통해서 모터 펌프를 On/Off 할 수 있다.

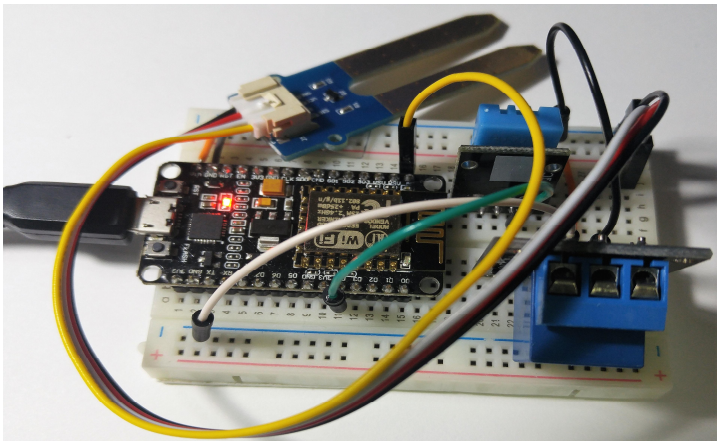
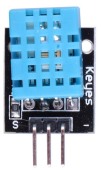
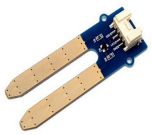


그림 4.3 센서 디바이스 3
Fig 4.3 Sensor device 3

표 4.3 센서 디바이스 3의 센서/액추에이터 모듈 상세
Table 4.3 Specification of Sensor Device 3

구분	기능	이미지
온•습도 (DHT11)	온도와 습도 정보를 동시에 얻을 수 있는 다기능 센서로 주변의 조건이 0 ~ 50° C 사이의 온도와 20 ~ 90% 사이의 습도 일 때 정확한 값이 측정한다. 측정 온도 오차는 2° C 측정 습도 오차는 5%이다.	
수분	토양에 직접 삽입하여 토양의 수분과 온도를 측정하는 센서 (센서는 물에 잠기지 않아야 한다.)	
Relay	릴레이 모듈은 평범한 디지털 On/Off 스위치로, 높은 전압과 낮은 전압의 5V 컨트롤러의 회로를 제어할 수 있다.	

4.1.2 소프트웨어 구현

아두이노에서 동작하는 소프트웨어는 아두이노의 공식사이트[31]에서 제공하는 통합 개발 환경(IDE: Integrated Development Environment)을 활용하여 구현하였다. 스마트 홈 시스템을 구성하는 센서 디바이스의 공통적인 동작의 흐름은 그림 4.4와 같다.

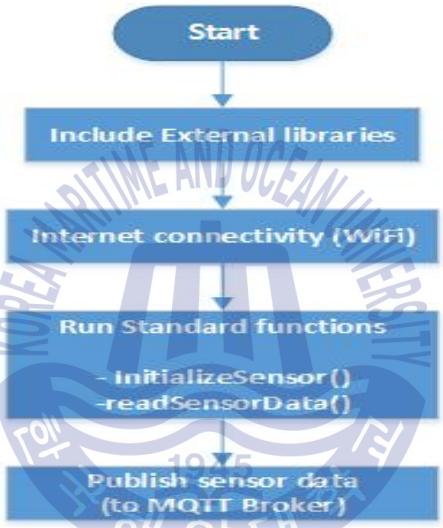


그림 4.4 소프트웨어 플로우 차트
Fig 4.4 Software Flowchart

통신 준비와 센서를 통한 정보 수집 후 통신 모듈을 통해서 정보를 게이트웨이로 전송한다. 정확한 측정값을 위해서는 약간의 보정(Calibration) 작업이 필요하다. 일반적으로는, 시스템이 부팅된 후 센서의 값을 읽어오기 시작하면 해당 값을 기준값으로 사용하는 것을 권장한다. 하지만, 주변 환경에 따라 측정 오차가 발생할 수 있으므로 이렇게 측정된 초기값을 미리 기억해서 이후 측정값들을 보정하는 것이 필요하다. 본 구현에서는 센서 초기화 함수를 사용하여 30초 정도 센서 보정작업을 수행하였다.

센서에서 감지한 데이터를 MQTT 브로커에게 전송하기 위한 전체적인 동작은 아래의 6가지 단계로 나눌 수 있다. 해당 프로그램은 와이파이 통신을 사용, 인터넷에 연결하여 메시지를 전송한다.

1) Include External Libraries : 센서 모듈에 따라서 요구되는 라이브러리를 추가한다.

2) Internet Connectivity Setup(WiFi) : 와이파이 연결을 위한 접속 정보 및 통신 가능 상태 확인을 위하여 와이파이 모듈의 초기화하고 연결을 시도한다.

3) Initialize Sensor : 아두이노에 연결된 센서에 대한 핀 번호 설정과 정확한 측정을 위한 보정(Calibration) 작업을 수행한다.

4) Read Sensor Data : 센서를 통하여 외부 환경 정보를 읽는다.

5) Publish Sensor Data : 감지된 정보를 MQTT 브로커로 전송한다.

6) Standard Arduino Functions : 아두이노에서 사용하는 기본 함수인 setup() 함수에서 2)~ 3)을 수행하고 loop() 함수에서 4)~5)번의 기능을 수행한다.

4.2 게이트웨이 구현

4.2.1 하드웨어 구현

게이트웨이 개념을 적용해 사물인터넷 환경의 게이트웨이는 PC를 대신할 수 있는 소형화가 용이한 라즈베리 파이를 사용한다. 라즈베리 파이는 컴퓨팅 능력, 메모리, 운영체제 사용, 고급 언어 사용 등의 장점이 있어 전체 시스템에서 메인 처리 장치로 사용할 수 있다.

표 4.4는 구현 시스템의 기능에 대한 설명이다.

표 4.4 게이트웨이 상세 사양

Table 4.4 Details of gateway

구분	기능	이미지
Micro Computer	Model : Raspberry Pi 2 Model B+ CPU : A 900MHz quad-core ARM Cortex-A7 Memory : 1GB RAM OS : Linux (RASPBIAN JESSIE(based on desktop)) Power : Micro USB socket 5V, 2A 4 USB ports / 40 GPIO pins Full HDMI port / Ethernet port Combined 3.5mm audio jack / composite video Camera interface (CSI) / Display interface (DSI) Micro SD card slot	
RPi Camera	고품질 8메가픽셀의 Sony IMX219 image sensor를 사용하여 라즈베리파이 CSI 인터페이스 에 맞는 라즈베리파이 맞춤형 모듈	

4.2.2 소프트웨어 구현

본 논문에서 다른 제품에 비해 성능이 우수하고 오픈소스로 활용이 가능한 Mosquitto MQTT 브로커를 사용한다. Mosquitto MQTT 브로커는 메시지를 중계하는 핵심적인 요소로서 MQTT프로토콜과 Web Socket 프로토콜을 모두 지원하고 사물인터넷과 같은 임베디드 디바이스에 최적화되어 매우 가볍게 동작한다. Mosquitto는 공식 웹 사이트에서 자신의 서버 환경에 맞는 버전을 다운로드 받아서 설치할 수 있으며, 본 시스템에서 사용하는 라즈베리 파이에는 인터넷이 연결된 경우 그림 4.5와 같은 명령을 사용하여 설치할 수 있다. 설치 후 명령 실행 창에서 mosquitto -v 명령을 입력하면 브로커의 실행 화면을 확인할 수 있다.

```
$ sudo apt-get update

$ sudo apt-cache search mosquitto

$ sudo apt-get install mosquitto mosquitto-clients
```

그림 4.5 Mosquitto 브로커 설치
Fig 4.5 Mosquitto Broker installation in Raspberry Pi

설치 후 브로커가 정상적으로 동작하는지 확인해야 한다. 두 개의 다른 커맨드 창을 연 후, 하나의 창에서 그림 4.6과 같이 "hello/world" 라는 토픽으로 브로커 서버에 구독 신청한다. 만약 브로커 서버에게 "hello/world" 토픽으로 메시지가 오면 구독자에게 메시지를 자동으로 전송한다.

그림 4.6 MQTT 동작 확인
Fig 4.6 MQTT Subscribe Test

그림 4.7과 같이 다른 커맨드 창에서 브로커 서버로 "hello/world" 라는 토픽으로 "Hello from MQTT" 라는 메시지를 발행한다. 브로커로 전송된 메시지는 그림 4.7과 같이 수신된 메시지를 확인할 수 있다.

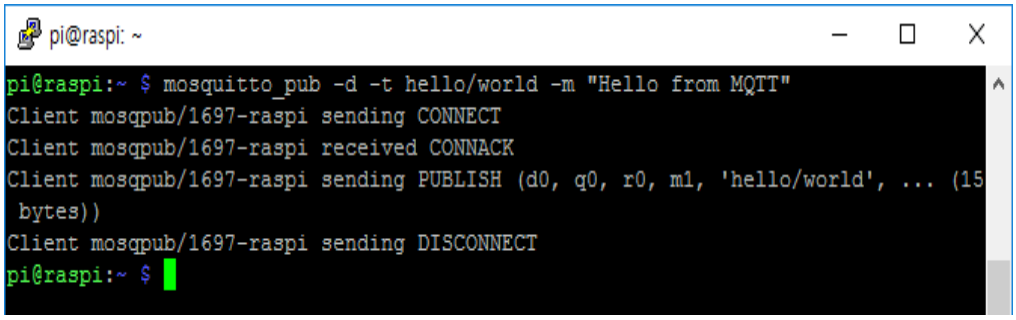


그림 4.7 MQTT 발행 확인
Fig 4.7 MQTT Publish Test

그림 4.6, 그림 4.7에서처럼 MQTT에서 이러한 단계별 통신을 제어하기 위한 제어 패킷이 표 4.5와 같이 정의되어 있다. 제어 패킷은 연결, 토픽등록, QoS에 따른 송수신, 연결 상태 확인, 연결해지에 대한 것으로 구분된다. 현재의 테스트는 공유기를 통한 인터넷 내부 IP를 이용하여 로컬 네트워크에서 수행하였으나 공인 IP를 활용하여 인터넷 외부 망에서도 데이터를 발행하거나 구독하는 것이 가능하다.

표 4.5 MQTT 제어 패킷의 종류

Table 4.5 Details of MQTT control packet

제어 패킷	방향	설명
CONNECT	클라이언트 → 브로커	클라이언트가 브로커에 연결을 요청 필요한 경우 id과 passwd 요구 가능
CONNBACK	브로커 → 클라이언트	서버의 연결 성공에 대한 응답
PUBLISH	클라이언트 → 브로커	메시지 전송
PUBACK	클라이언트 → 브로커	메시지 전송에 대한 응답(QoS 1레벨)
PUBREC	클라이언트 → 브로커	전송된 메시지가 수신됨(QoS 2레벨)
PUBREL	클라이언트 → 브로커	메시지 송수신이 릴리즈됨(QoS 2레벨)
PUBCOMP	클라이언트 → 브로커	메시지가 완전히 송수신됨(QoS 2레벨)
SUBSCRIBE	클라이언트 → 브로커	토픽에 대한 등록 요청
SUBACK	브로커 → 클라이언트	토픽 등록에 대한 응답
UNSUBSCRIBE	클라이언트 → 브로커	토픽 등록 해지 요청
UNSUBACK	브로커 → 클라이언트	토픽 등록 해지에 대한 요청
DISCONNECT	클라이언트 → 브로커	클라이언트 연결 종료

MQTT 브로커에게 구독 신청하고 송수신되는 데이터를 처리하는 애플리케이션을 개발하기 위하여 Node.js를 이용한 경량의 자바스크립트 런타임을 이용하고 웹 브라우저에서 개발이 가능한 Node-RED 개발도구로 사물인터넷 애플리케이션을 개발한다. Node-RED는 라즈비안 Jessie부터 기본 프로그램으로 설치되어 있으며 추가되는 블록은 별도로 설치해야 한다. 본 논문에서는 최신 버전을 설치하여 개발하였다.

그림 4.8은 Node-RED에서 센서 디바이스 1의 센서 모듈 중에서 MQTT 브로커에 구독 신청한 온도, 습도, 조도 센서 메시지를 전송받아서 그래픽 화면으로 보여주는 상황을 프로그래밍한 것이다.

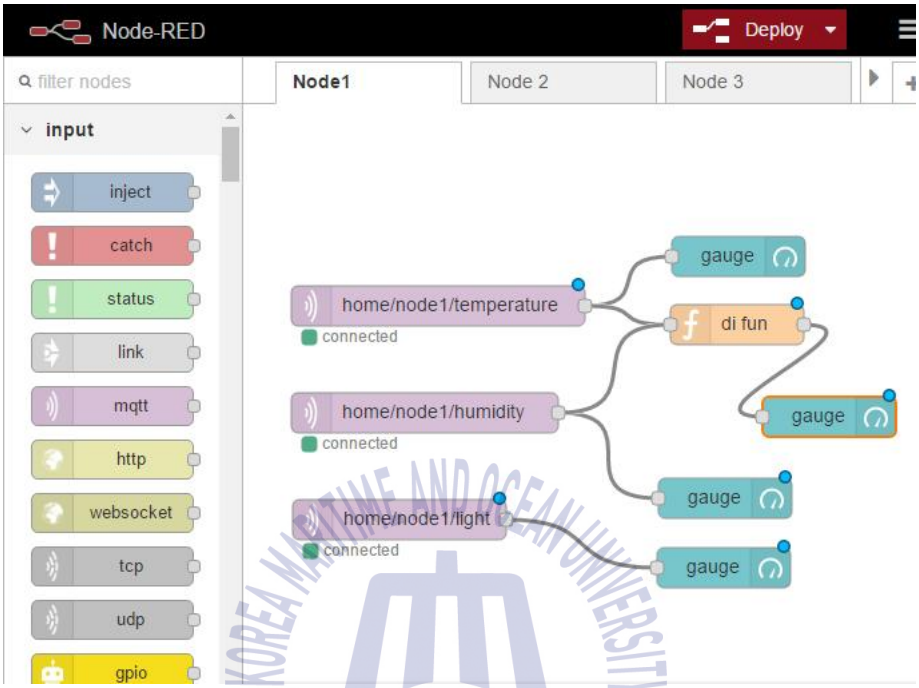


그림 4.8 센서 디바이스 1의 Node-RED 흐름
Fig 4.8 Node-RED flow of sensor device 1

MQTT 노드의 작성은 그림 4.9와 같이 Node-RED의 mqtt 노드에 MQTT 브로커 정보를 등록하고 Topic에는 브로커 서버에 구독할 정보의 토픽을 정확하게 입력한다.

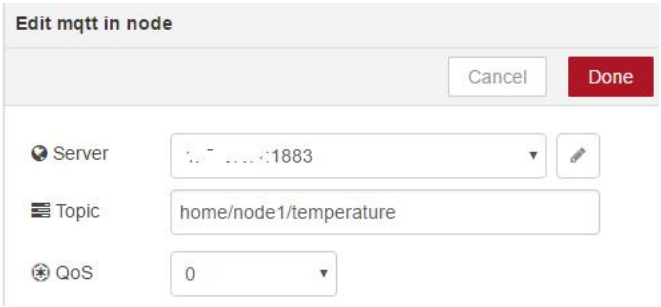


그림 4.9 MQTT 노드 편집 : 온도 센서
Fig 4.9 Edit of MQTT Node : temperature sensor

gauge 노드는 Node-RED의 기본 노드가 아니기 때문에 추가적으로 설치해서 사용한다. 그림 4.10은 gauge 노드 설정을 위한 인터페이스 화면이다. UI 노드는 가시화를 위한 대시보드 역할을 하는 노드로 먼저 대시보드 이름에 해당하는 Tab을 설정한다. Title에는 대시보드에 표시될 텍스트를 입력한다. Group은 그룹으로 표시될 노드들과 같은 이름으로 설정한다. Value format는 표시될 데이터 형식과 단위를 기록하고 Min과 Max에 표시할 데이터의 하한 값과 상한 값을 입력한다.

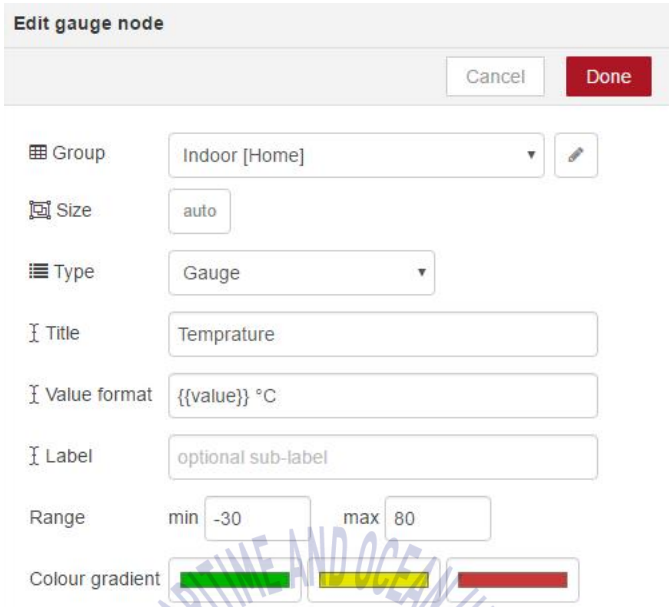


그림 4.10 gauge 노드의 Edit
Fig 4.10 Edit of gauge node

그림 4.11은 Node-RED에서 센서 디바이스 2가 화분 주변의 온습도와 토양의 수분 상태를 모니터링하기 위한 Node-RED 흐름을 보여준다. 센서 디바이스 2의 센서 모듈 중에서 MQTT 브로커에 구독 신청한 온도, 습도, 수분 센서 메시지를 전송받아서 그래픽 화면으로 보여주는 상황이다. 수분의 상태를 모니터링하여 수분이 많거나 적은 상태인 경우 트위터로 메시지를 전송하는 알림서비스가 가능하다.

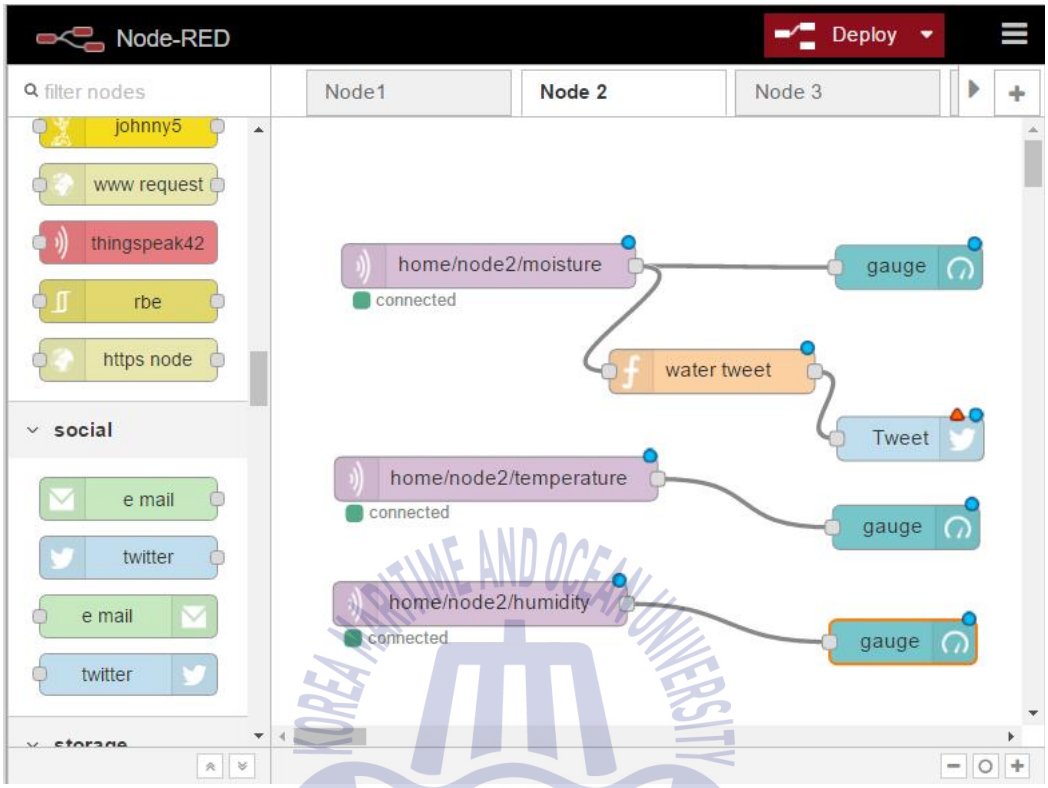


그림 4.11 센서 디바이스 2의 Node-RED 흐름
Fig 4.11 Node-RED flow of sensor device 2

식물의 종류에 따라서 수분의 상태가 다를 수 있으므로 임계 수치를 그림 4.12와 같이 'water tweet' function 노드의 스크립트에서 조정하면 된다. 실험을 위해서 임의적으로 수행해 보았고 수분의 상태를 트위터를 통해 그림 4.13과 같이 전송 받을 수 있다.

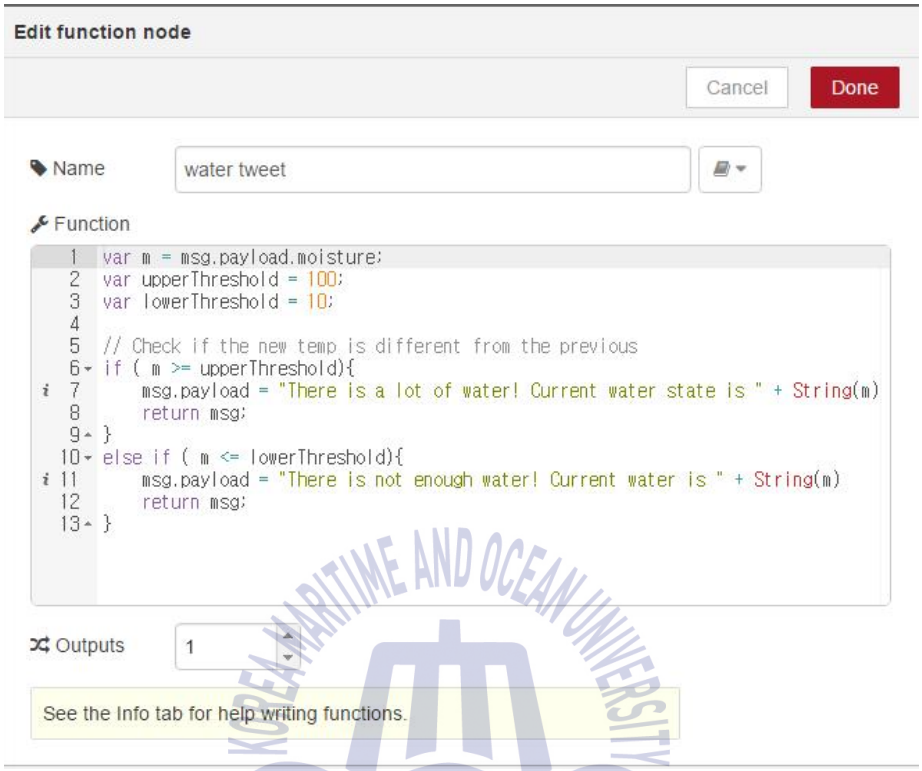


그림 4.12 함수 편집 : 수분 감지 트윗

Fig 4.12 Edit function node : water tweet

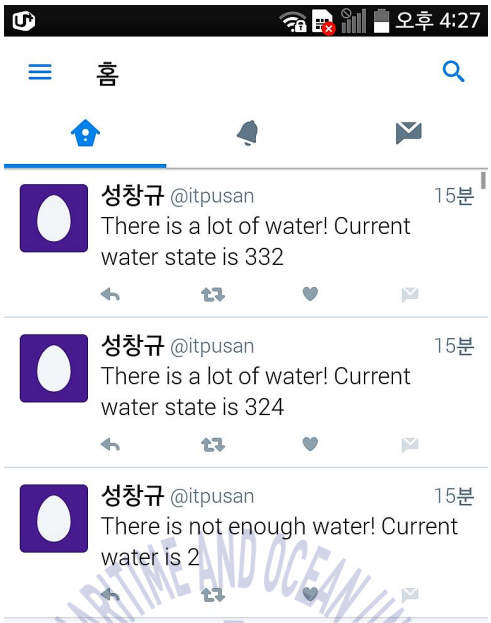


그림 4.13 습도 트윗

Fig 4.13 Moisture Value Tweet

그림 4.14는 외부 날씨 관련 서비스를 위해 API 서비스가 제공되는 openweathermap과 연동하여 서비스를 제공하고 일기예보와 사용자 주변의 온도의 차이를 확인할 수 있다. 해당 사이트에서 제공되는 일기 예보 서비스 중 바람, 기압 등 다른 종류를 더 추가하여 여러 가지 예보를 시각적으로 표현할 수 있다.

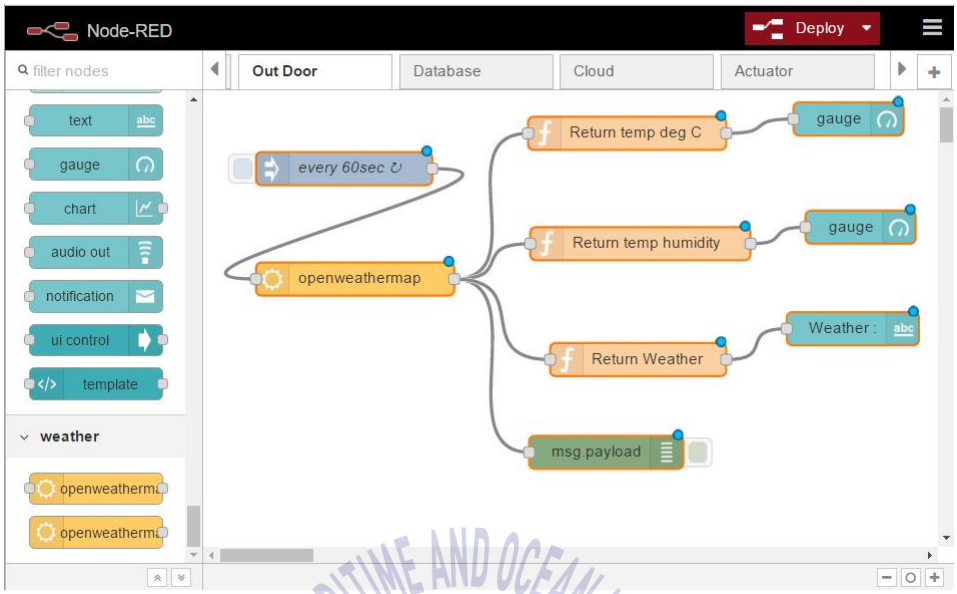


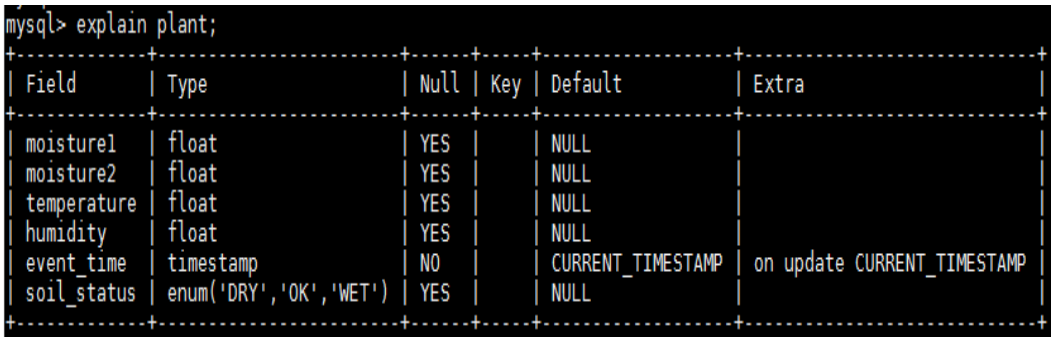
그림 4.14 Node-RED 흐름 : 실외
Fig 4.14 Node-RED flow : Outdoor

해당 웹사이트에서 API Key를 발급받아서 지역을 설정 후 연결하면 지역에 관련되는 데이터가 아래와 같은 JSON으로 처리된 정보가 전송되는데 필요한 데이터만 파싱하여 처리하였다.

{ “weather“: “Clouds“, “detail“: “scattered clouds“, “tempk“: 276.15, “tempc“: 2.9, “humidity“: 51, “maxtemp“: 276.15, “mintemp“: 276.15, “windspeed“: 2.6, “winddirection“: 40, “location“: “Pusan-gwangyŏksi“, “sunrise“: 1481408539, “sunset“: 1481443938, “clouds“: 40, “description“: “The weather in Pusan-gwangyŏksi at coordinates: 35.13, 129.05 is Clouds (scattered clouds).“ }

브로커로 발행되는 데이터 중에서 화분의 온·습도와 수분을 주기적으로 MySQL데이터베이스에 저장한다. 데이터를 저장하기 위한 테이블 구조는 그림

4.15와 같다.



```
mysql> explain plant;
+------------+----------------------+------+-----+-------------------+-----------------------------+
| Field      | Type                 | Null | Key | Default           | Extra                       |
+------------+----------------------+------+-----+-------------------+-----------------------------+
| moisture1  | float                | YES  |     | NULL              |                             |
| moisture2  | float                | YES  |     | NULL              |                             |
| temperature | float               | YES  |     | NULL              |                             |
| humidity   | float                | YES  |     | NULL              |                             |
| event_time | timestamp            | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| soil_status | enum('DRY','OK','WET') | YES |     | NULL              |                             |
+------------+----------------------+------+-----+-------------------+-----------------------------+
```

그림 4.15 데이터베이스 테이블 구조
Fig 4.15 Database Table Description

MySQL 데이터베이스에 수집된 데이터를 저장하기 위해서 MySQL 노드를 추가로 설치하고 해당 노드에서 그림 4.16 화면과 같이 사용할 데이터베이스의 연결 설정을 한 후 function 노드에 SQL 수행 스크립트를 작성한다. 여기에서는 게이트웨이인 라즈베리 파이에 MySQL을 운영하므로 localhost(127.0.0.1)로 접속이 가능하다.

그림 4.17은 토픽에 따라 전송받은 데이터를 데이터베이스 서버에 저장하는 SQL 쿼리가 작성된 스크립트이다.

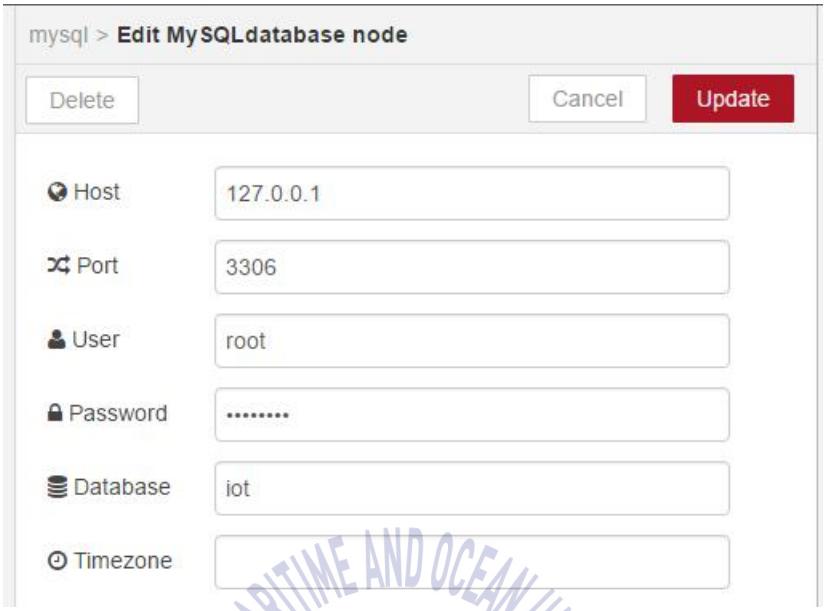


그림 4.16 Node-RED에서의 mysql 노드 구성

Fig 4.16 mysql node configuration in Node-RED

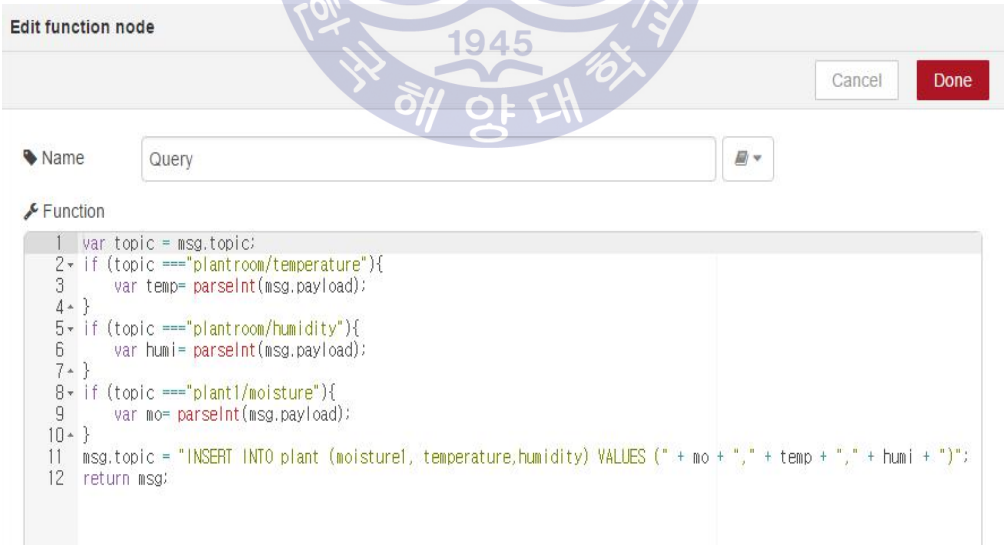


그림 4.17 함수 편집 : 쿼리

Fig 4.17 Edit function node : query

그림 4.18은 브로커로 발행되는 수분, 온도, 습도 정보를 개방형 사물인터넷 클라우드인 Thingspeak에 전송하는 Node-RED 프로그래밍을 나타내었다. Thingspeak을 Node-RED에서 접속하기 위해서 앞서 mysql 노드와 같이 Thingspeak 노드를 Node-RED에 추가 설치하고 Thingspeak에 가입된 정보로 클라우드 서버에 접속 가능하도록 설정한다. 브로커로 데이터가 발행되면 클라우드 서비스에 중계하도록 하였다.

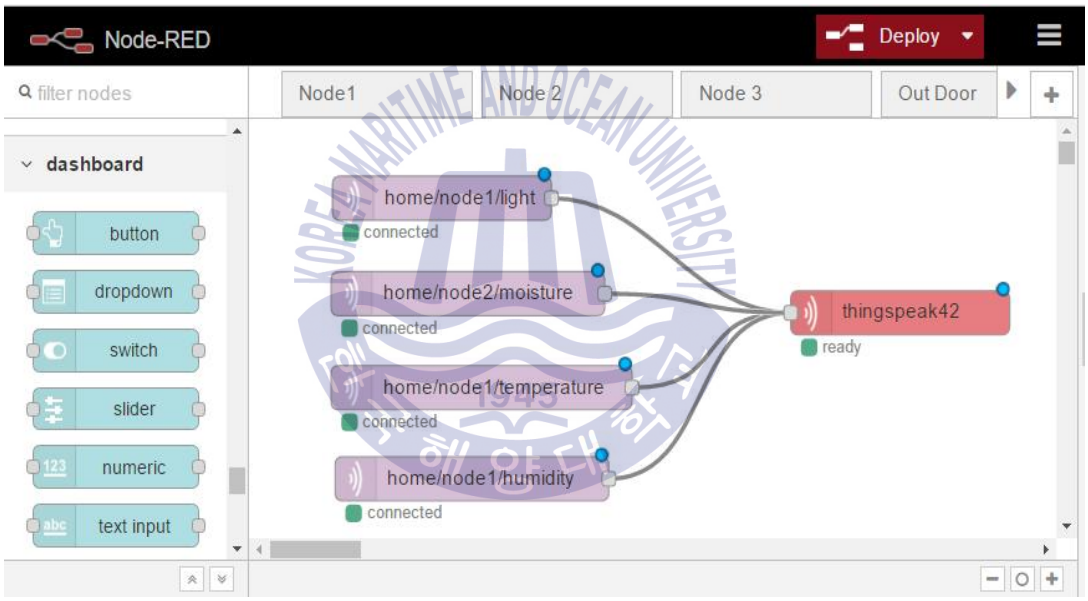


그림 4.18 Node-RED 흐름 : 클라우드 서비스로 데이터 전송
Fig 4.18 Node-RED flow :Send data to Cloud Service

인터넷 기반 인터페이스를 제공하는 개방형 사물인터넷 서비스 플랫폼인 Thingspeak를 활용하여 모바일 서비스를 통해서 언제 어디서나 확인 가능하도록 구현하였으며 이를 그림 4.19에서 확인할 수 있다.

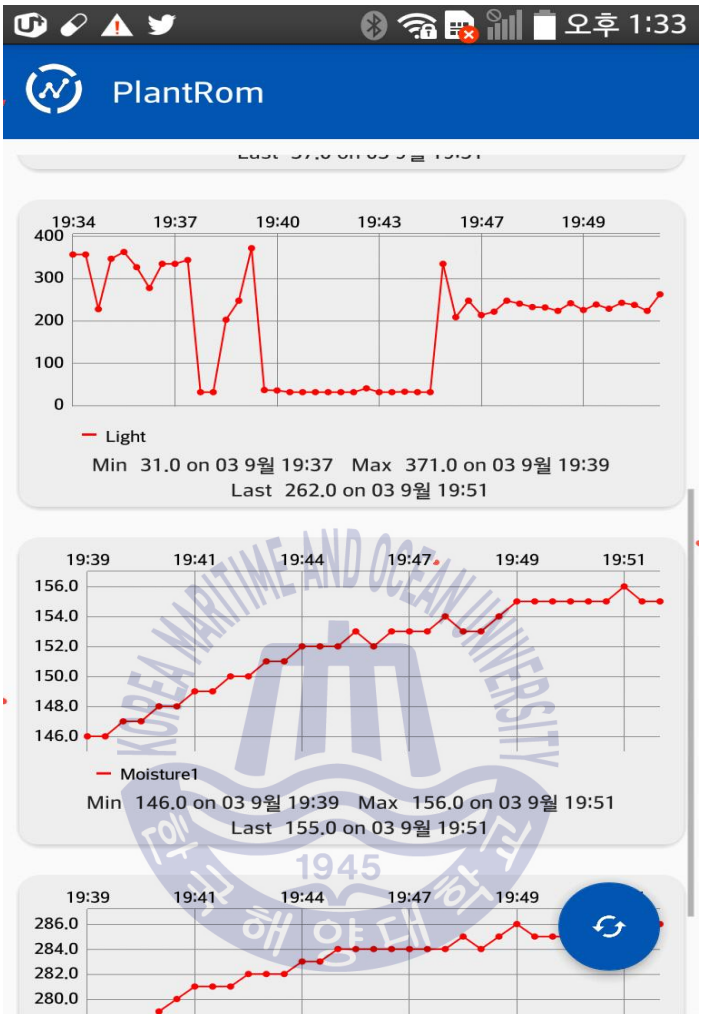


그림 4.19 모바일 모니터링 : ThingSpeak
Fig 4.19 Mobile Monitoring : ThingSpeak

그림 4.20은 Node-Red에서 브로커로 정보를 발행하는 것으로 각종 액추에이터를 구동하기 위한 기능을 구현한 것이다.

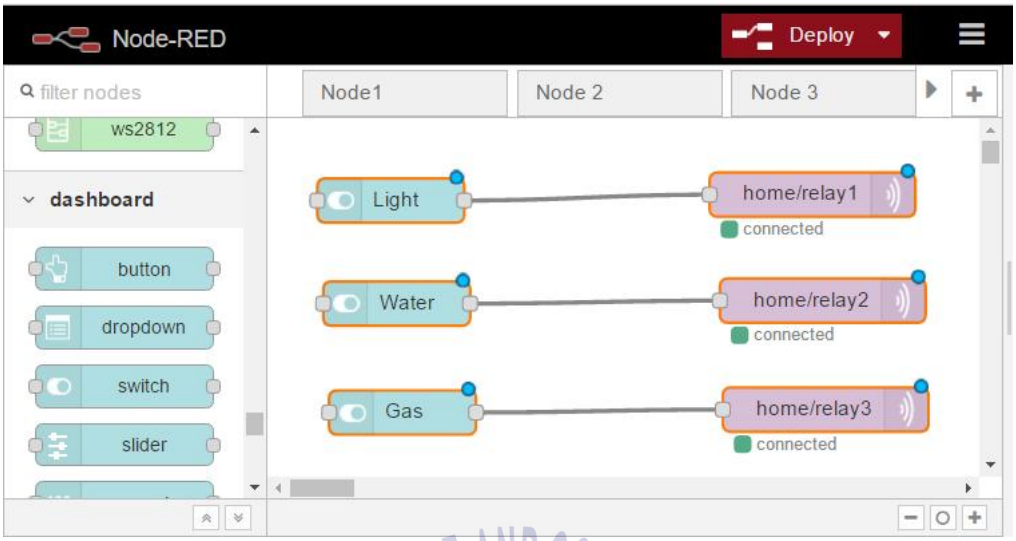


그림 4.20 Node-RED 흐름 : 액추에이터 동작
Fig 4.20 Node-RED flow : Actuator Operation

웹 브라우저를 이용해서 http://게이트웨이IP:1880/ui로 접속하면 그림 4.21과 같이 측정된 값을 게이지나 그래프와 같은 시각화된 화면을 볼 수 있다. 브로커로부터 전송되는 데이터형이 수치형일 경우 UI 노드들 중에서 gauge 노드를 그대로 연결하여 사용하였다. 전송되는 데이터의 형식은 JSON 형태로 전송되는 경우가 있는데 출력에서 출력 노드인 Debug 노드를 활용하여 데이터 형식을 확인하고 function 노드에서 데이터 파싱을 위한 스크립트를 추가적으로 작성하여 gauge 노드와 연결한다. UI 노드즐 중에서 스위치 노드로 센서 디바이스에 연결된 릴레이와 같은 액추에이터를 작동할 수 있도록 인터페이스도 작성하였다.

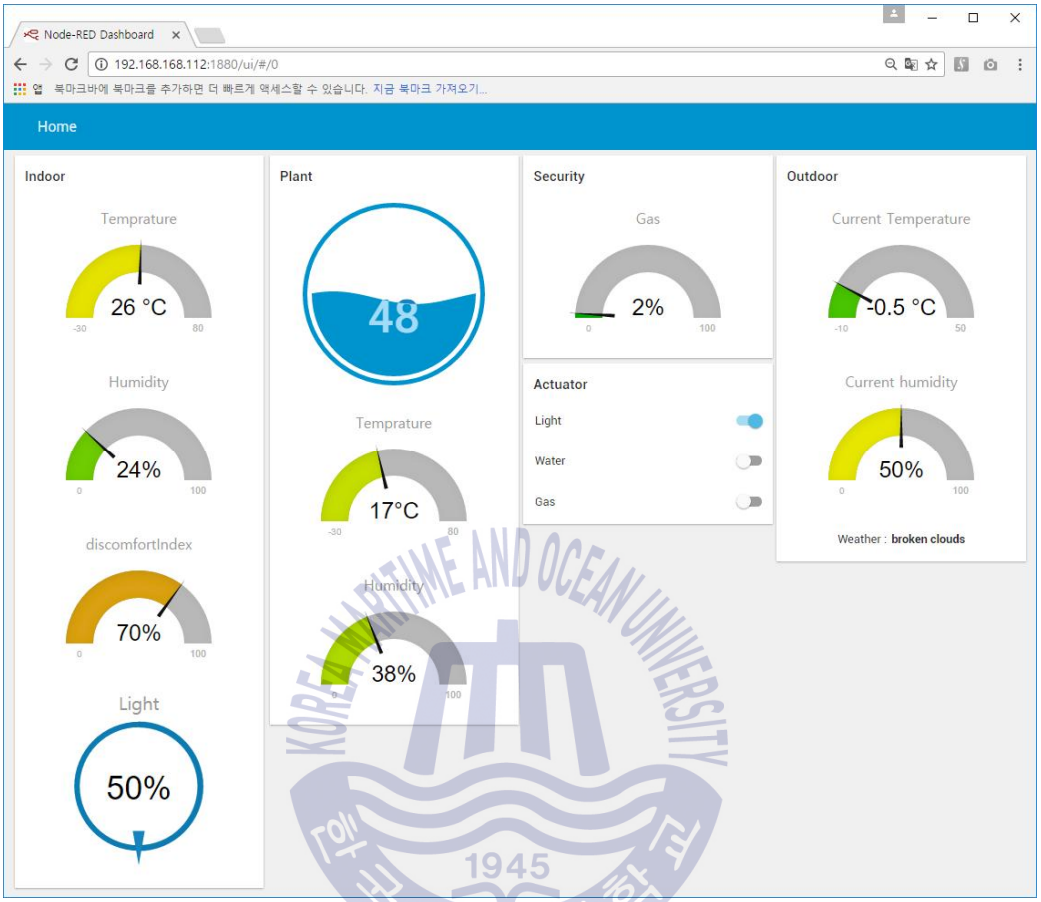


그림 4.21 Node-RED로부터 실시간 센서 데이터 모니터링

Fig 4.21 Real Time Sensor Data Monitoring from Node-RED

스마트폰의 앱을 통해서 브로커 서버와 연결되어 브로커가 배포하는 데이터를 받아서 다양한 정보를 표시할 수 있고 전원을 켜고 끄는 등의 모니터링 기능을 수행할 수 있다. 그림 4.22는 브로커가 배포하는 데이터를 스마트폰에서 전달받아 가시화한 화면이고, 그림 4.23은 스마트폰 앱을 통해 웹캠으로 원격 화상 감시를 실행하고 있는 화면이다.

그림 4.22 모바일 홈 환경 모니터링 :
Fig 4.22 Mobile Home Environment Monitoring

그림 4.23 모바일 홈 보안 모니터링

Fig 4.23 Mobile Home Security Monitoring

웹 소켓 통신 프로토콜을 활용하여 웹 브라우저와 웹 서버간의 데이터를 양방향으로 실시간 송수신할 수 있다. MQTT 통신을 위해 설치한 Mosquitto 브로커는 웹 소켓 연결을 지원하며, 자바스크립트와 HTML로 작성된 웹 애플리케이션을 실행할 수 있다. 이를 위해서는 우선, 그림 4.24와 같이 Mosquitto 브로커에 웹 소켓 환경을 설정해 주어야 한다. 웹 소켓 포트를 별도로 설정한 후 Mosquitto 서버를 재 시작해야 정상적으로 동작한다.

```
$ sudo nano /etc/mosquitto/mosquitto.conf #설정 수정
--아래 내용 추가--
listener 8080
protocol websockets
--------------------
$ sudo service mosquitto restart  #재시작
$ sudo netstat -nlp ¦ grep mosquitto #프로세스 확인
```

그림 4.24 웹 소켓 설정
Fig 4.24 Setup for Web Socket

HTTP 프로토콜은 데이터를 전송할 때마다 연결한 후 전송이 끝나면 연결이 끊어져 클라이언트의 요청이 없으면 통신이 불가능하다. 따라서 실시간으로 변화하는 데이터를 지속적으로 볼 수 없다.

반면에 웹 소켓은 순수 웹 환경에서 실시간 양방향 통신이 가능하므로 클라이언트로부터 최초의 연결이 되면 계속적으로 데이터 송수신이 가능하므로 클라이언트의 요청이 없어도 서버로부터 데이터를 전송받을 수 있으므로 서버에서 지속적으로 갱신되는 데이터를 받아볼 수 있는 장점이 있다. 그림 4.25는 이두이노에서 MQTT 프로토콜로 전송된 온·습도센서 입력 값을 구축된 웹서비스로 출력한 결과이다. 웹 서비스는 웹 소켓 프로토콜을 활용하여 웹브라우저에서 온·습도 정보를 실시간으로 출력하고 있으며 전송된 센서 값에 따라 게이지의 변화를 확인 할 수 있다.

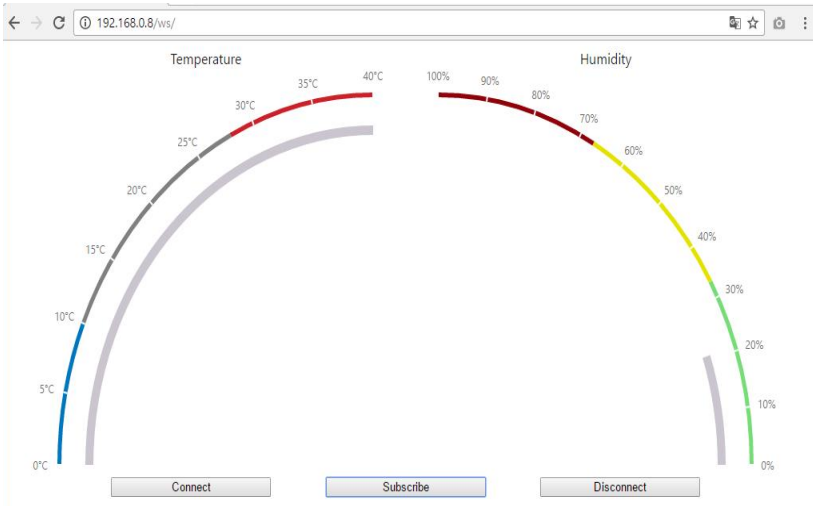


그림 4.25 웹 소켓 프로토콜을 이용한 연결
Fig 4.25 Connection using Web Socket Protocol

4.3 서비스 평가

표 4.6은 구현된 시스템에서 사용한 MQTT 프로토콜과 HTTP 프로토콜의 성능 테스트한 것이다.

아래의 성능 테스트에서 Mosquitto Broker와 HTTP Server의 평균 응답시간과 처리량에 대하여 분석하였다. Mosquitto Broker는 HTTP Server보다 초당 처리량이 약 23배 우수하고 응답시간이 약 445배 빨라서 사물인터넷 환경에서 센싱 데이터를 처리하는데 높은 효율성을 보였다. 해당 실험을 통해서 기존에 저 사양의 PC에서 사용하던 HTTP 프로토콜보다 라즈베리파이에서 적용된 MQTT 프로토콜이 높은 안정성을 보이는 것을 알 수 있다.

표 4.6 MQTT vs HTTPS

Table 4.6 MQTT vs HTTPS

	Mosquitto Broker	HTTP Server
The number of the processed sensor data(count)	168653	6453
Average response time(ms)	4	1780
throughput (The number of jobs/sec)	4312	187
Battery (Hour)	3.45%	4.23%

제 5 장 결론 및 향후 연구

사물인터넷은 모든 사물이 인터넷에 연결되는 사물간의 연결 기술이다. 이와 같은 사물인터넷 서비스에서 경제적이고 안정적인 사물 연결성에 대한 요구가 커지고 있다. 작은 규모의 사물인터넷 서비스라고 하더라도 서버에서 동작하는 애플리케이션을 포함하여 디바이스를 구성하는 하드웨어, 임베디드 소프트웨어, 디바이스와 센서를 연결하는 게이트웨이, 무선 통신 기술, 네트워크 등 필요로 하는 지식이 매우 다양하다.

본 논문에서는 사물인터넷 환경의 홈 서비스 구조를 제안하고 오픈소스를 활용하여 스마트 홈 시스템을 설계 및 구현하였다. 사물인터넷 서비스를 구축하는 경우 오픈소스를 사용하면 다음과 같은 장점이 있다.

첫째, 일반적으로 기존의 폐쇄형 시스템보다 오픈소스 시스템의 기술적 진보가 훨씬 빠르다.

둘째, 기존의 시스템구축 비용보다 오픈소스 솔루션의 총 소유 비용이 적게 소요된다. 오픈소스 하드웨어는 개인이나 전문 기업들이 개발해 놓은 소스코드, 부속 모듈 등이 존재하기 때문에 처음부터 모든 것을 개발할 필요가 없다.

셋째, 분야별 요구사항이나 아이디어를 추가하여 더욱 혁신적이고 참신한 제품을 생산하기 용이하므로 개발 속도가 현저하게 빠르다는 점이다.

IT 분야에서 오픈소스는 단순한 일시적인 유행이 아니라 지속적으로 발전하고 성장해가는 흐름이라 할 수 있다. 시스템 구축 과정을 통하여 스마트 홈 시스템에서 요구되는 실시간성, 양방향성, 경량성과 소형화 및 확장성이 용이하고, 저비용으로 유연한 시스템을 빠르게 개발할 수 이음을 확인하였다.

따라서 본 논문에서는 오픈소스 사물인터넷 플랫폼과 사물인터넷 홈 게이트웨이로 구성되는 사물인터넷 홈서비스 환경 구조를 제안 다양한 디바이스에서 생산되는 데이터와 명세를 토대로 사물인터넷 기반의 스마트 홈 시스템을 설계하고 구현하였다. 사물인터넷 환경은 사물의 연결과 데이터의 공유, 정보의 활

용이 중요하다. 사물간의 메시지 전달의 신뢰도와 경량성이 있는 양방향 메시지 프로토콜인 MQTT를 활용하여 시스템을 개발하였다. 사물인터넷 환경에서 디바이스를 위한 연결성을 제공하고 사물인터넷 개방형 클라우드 서비스 플랫폼과 연동하기 위한 브로커 서버를 구축하였다. 사물 연결 및 사물인터넷 개발 도구인 Node-RED를 활용하여 홈에서 사용할 수 있는 애플리케이션을 개발하였다. Node-RED는 실시간으로 발생하는 데이터를 가져와서 Node.js 스크립트 프로그램을 통하여 각 서비스에 맞게 처리하였다. 그래서 Node-RED는 사용자에게 확장성, 유연성, 신속성이 다른 개발도구에 비해서 우수하고 간단한 인터페이스로 쉬운 개발환경을 제공하는 하고 오픈소스로 저비용으로 신속하게 사물인터넷 서비스를 구축하는 애플리케이션으로 개발할 수 있음을 확인하였다.

본 시스템은 사물인터넷 환경에서 영역별 맞춤 센서 디바이스와 로컬이나 원격 디바이스와 연동하여 서비스 중심적인 애플리케이션으로 확장할 수 있어 가정뿐 아니라 공장, 사무실, 공공 서비스 등 다양한 분야에서 활용이 가능할 것이다. 향후 연구로 본 논문에서 다루지 못한 다양한 사물인터넷 솔루션 적용을 통해, 시스템의 성능과 안정성을 향상시켜 분야별 맞춤형 서비스에 적합한 시스템에 대한 연구가 필요하다.

참고문헌

[1] 이보겸, “국내외 주요 통신사업자의 스마트홈서비스 동향,” 정보통신방송정책, vol. 27, no. 5. pp. 27-35, 2015.

[2] 김성민, 최환석, 이우섭, "오픈소스 하드웨어를 활용한 ACOME 기반의 IoT 홈 게이트웨이 환경 개발,“ 한국콘텐츠학회논문지, vol. 16, no. 3, pp. 296-304, 2016.

[3] C. Frolund, J. H. Jeon, Y. K. Lee, S. Y. Lee, H. J. Kim, "Study for strengthening the ICT DIY ecosystem," Information and Communication Technology Convergence (ICTC), pp. 269-274, 2014.

[4] 장홍석, “오픈소스 SW의 글로벌 동향과 우리 기업의 해외 진출 방안,” 한국무역협회 국제무역연구원, Issue Papers 2016, no. 4, 2016. 7.

[5] 박종현, 방효찬, 김세한, 김말희, 이인환, 최병철, 이강복, 강성수, 김호원, 사물인터넷의 미래, 전자신문사, 2014.

[6] 주대영, 김종기, 초연결시대 사물인터넷(IoT)의 창조적 융합 활성화 방안, 산업연구원, 2014.

[7] K. Ashton, “That ‘internet of things’ Thing,” RFID Journal, vol. 22, no. 7, pp. 97-114, 2009.

[8] 서준오, IoT환경에서 MQTT와 WebSocket을 활용한 실시간 사물제어 시스템 설계 및 구현, 호남대학교 대학원 박사학위논문, 2016.

[9] 이상홍, IoT 현황 및 주요 이슈, 정보통신기술진흥센터, 2014.

[10] H. Shen, Z. Li, L. Yu and C. Qiu, “Efficient data collection for large-scale mobile monitoring applications,” IEEE Transactions on Parallel and

Distributed Systems, vol. 25, no. 6, pp. 1424-1436, 2014.

[11] 3S. K. Datta, C. Bonnet, and N. Nikaein, “An IoT gateway centric architecture to provide novel M2M services,” Internet of Things (WF-IoT), 2014 IEEE World Forum on 2014, pp. 514-519, 2014.

[12] H. J. La, C. W. Park and S. D. Kim, “A framework for effectively managing dynamism of IoT devices,” Journal of Korean Institute of Information Scientists and Engineers : Software and Applications, Vol. 41, No. 8, pp. 545-556, Aug, 2014.

[13] 박재현, 임정선, M2M 시장 현황과 국내외 성장 전망, KT경제경영연구소 DIGIECO, 2013.

[14] 전종홍, 차홍기, 이원석, 김형준, “오픈소스 사물인터넷(OSIoT) 동향 및 전망,” 한국통신학회지(정보와통신), vol. 32, no 5, pp. 23-30, 2015.

[15] Morgan Stanley, The Mobile Internet Report, 2009.

[16] IoT 산업에서 오픈소스의 활용방안. 숭실대교육교재(김형채), [Internet].

Available: http://www.slideshare.net/chaeya/iot-38294115.

[17] 김경원, 박종빈, 금승우, 임태범, 윤경로, “사물인터넷 기반 스마트 홈서비스 프레임워크,” 방송과 미디어, vol 20, no. 3, pp 54-65, 2015.

[18] SmartThings, [Internet]. Available: http://www.smartthings.com/developers/

[19] Withings, [Internet]. Available: http://www.witjthings.com/

[20] The Electronic Times, “SKT Smart Farm, creation of agricultural farms realize the dream of smart”, 2013. 05.

[21] 융합진흥부, 국내외 사물인터넷 정책 및 시장동향과 주요 서비스 사례, 한

국방송통신전파진흥원, 2015.

[22] The Electronic Times, LG U+, 2012 Yeosu EXPO LTE Building vehicle control system, 2012.05

[23] G. Chrisman, IBM Innovations and Smarter Planet projects, 2012.

[24] CISCO, Smart+Connected Communities-Changing a City, a Country, the World, 2013.

[25] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "Contemporary Internet of Things platforms," arXiv preprint arXiv:1501.07438, 2015.

[26] EVRYTHNG Homepage, [Internet]. Available at: https://evrythng.com/platform/ [Accessed 25 Semtember 2016].

[27] Chris Dibona, OPENSOURCES, 한빛미디어, p 15, 2000.

[28] OSHW Homepage, [Internet]. Available at: http://freedomdefined.org/OSHW [Accessed 3 October 2016].

[29] Open Source Hardware Association Homepage, [Internet]. Available at: http://www.oshwa.org/ [Accessed 3 October 2016].

[30] M. J. Martínez Abascal, Open Technologies for Prototyping the Internet, Escuela Universitaria de Ingeniería Técnica de Telecomunicación, 2013.

[31] Arduino Homepage, [Internet]. Available at: https://www.arduino.cc [Accessed 3 October 2016].

[32] Raspberry Pi Homepage, [Internet]. Available at: https://www.rasapberrypi.org/ [Accessed 5 October 2016].

[33] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things",

Transaction on IoT and Cloud Computing, vol. 3, no. 1, pp. 9-18, 2015.

[34] K. Tang, Y. Wang, H. Liu, Y. Sheng, X. Wang, and Z. Wei, “Design and implementation of push notification system based on the MQTT protocol,” 2013 International Conference on Information Science and Computer Applications (ISCA 2013), pp. 116-119, 2013.

[35] M. Prihodko, Energy consumption in location sharing protocols for Android applications, Master’s Thesis of Linköopings University, 2012.

[36] 심승현, 김학범, “사물인터넷과 MQTT 기술,” 정보보호학회지, vol. 24, no. 6, pp. 37-47, 2014.

[37] Mosquitto Homepage, [Internet]. Available at: http://mosquitto.org/, [Accessed October 11. 2016].

[38] Y. H. Kang, "Design of a seamless messaging application using WebSocket in IoT environment,“ Proceedings of Korean Institute of Information Technology Summer Conference, pp. 245-247, May. 2014.

[39] J. T. Park, H. S. Hwang and I. Y. Moon, "Implementation of Smart TV Application using HTML5 and Health Bicycle,“ Journal of Advanced Navigation Technology, vol. 18, no. 1, pp. 101-106, Feb. 2014.

[40] RFC6455, [Internet]. Available at: http://www.websocket.org/

[41] 최성찬, 류민우, 진남, 김재호, “사물인터넷 플랫폼 및 서비스 동향,” 한국통신학회지, vol 31, no. 4, pp. 20-27, 2014.

[42] Node-RED, A visual tool for wiring the Internet of Things, [Internet]. Available at: http://nodered.org/ [Accessed 5 October 2016].

[43] S. Tikov, S. Vinoski, “Node.js:Using JavaScript to Build High-Performance

Network Programs," Internet Computing, vol. 14, no. 66, pp. 80-83, Dec. 2010.

[44] 황신범, 대기질 모니터링 시스템과 빅데이터 플랫폼의 연동을 위한 DMS 설계 및 구현, 충남대학교 대학원 석사학위논문, 2014.

[45] M. A. G. Maureira, D. Oldenhof and L. Teernstra, "ThingSpeak – an API and Web Service for the Internet of Things," World WideWeb, 2015.

[46] N. Dickey, D. Banks, S. Sukittanon, "Home automation using Cloud Network and mobile devices", Proceedings of IEEE, pp. 1~4, 15~18, 2012.

[47] K. Suzuki, M. Inoue, "Home network with cloud computing for home management," Consumer Electronics, 2011 IEEE 15th International Symposium, pp. 421-425, 2011.

[48] UPnP Forum, UPnP AV Architecture : 1, UPnP Forum, 2006.

[49] DLNA, The DLNA Networked Device Interoperability Guidelines Expanded Digital Living Nerwork Alliance, Digital Living Nerwork Alliance, 2006.

[50] B. A. Miller, T. Nixon, C. Tai, M. D. Wood, "Home networking with Universal Plug and Play," IEEE Communications Magazine, vol. 39, no. 12, pp. 104-109, 2001.

[51] 김국세, 민혜란, 이금분, 조범준, 이준, "인터넷상의 홈 네트워크 가전제어를 위한 UPnP 확장," 한국멀티미디어학회 학술 논문 발표집, vol. 2006, pp. 483-486, 2006.

[52] R. T. Fielding, R. N. Taylor, "Principled design of the modern Web architecture," ACM Transactions on Internet Technology, vol. 2, issue 2,

pp. 115-150, 2002.

[53] 기상청, [Internet]. Available at: http://www.kma.go.kr/HELP/basic/help_01_05.jsp [Accessed 5 October 2016].

[54] 에어코리아, [Internet]. Available at:http://www.airkorea.or.kr/dustForecast [Accessed 5 October 2016].

[55] 성창규, 류길수, "오픈소스 하드웨어를 활용한 사물인터넷 응용 서비스 시스템," 한국마린엔지니어링학회지, vol. 40, no. 6, pp. 542-547, 2016.