The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 20, No. 1, pp.231-237, Feb. 29, 2020. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2020.20.1.231
JIIBC 2020-1-33

MQTT 기반 IoT 홈 시스템 구현

Implementation of IoT Home System based on MQTT

김우조*, 최진구**

U-zo Kim*, Jin-ku Choi**

요 약 본 논문에서는 가정에서의 환경 모니터링을 기반으로 하여 홈 IoT 시스템을 구현하였다. 구현한 시스템에서는 각종 센서들로부터 데이터들을 실시간으로 수집하고 서버로 전송하였다. 이들 데이터들의 기반으로 홈의 디바이스들을 자동 또는 수동으로 제어하는 것을 구현하였다. MQTT 프로토콜을 사용하여 토픽에 따라 실시간으로 받아들이는 센서들의 데이터의 값과 그 변화를 볼 수 있었다. 브로커(Broker)를 통해 실시간으로 받아지는 각 토픽별 데이터를 토대로 자동으로 환경을 제어하는 시스템의 구현하고 시험을 통하여 동작을 확인하였다. 이런 시스템은 향후 실시간으로 데이터를 추적 및 모니터링이 필요한 헬스케어, 산업용 IoT, 화재 예방 등에 유용하게 응용될 것이라 기대된다.

Abstract In this paper, we implemented a home IoT system based on MQTT protocol. In this system, data are collected from sensors in real time and transmitted to the server system. Based on collected data, home devices could be controlled automatically or manually. By using the MQTT protocol, we were able to see the data values of sensors collected in real time according to the topic setting. We implemented a system that automatically sets up home devices based on topic data, and it worked. The system is expected to be useful in applications that require monitoring and tracking of data in real time.

Key Words : IoT, MQTT, MQTT Protocol, Smart Home

Ⅰ. 서 론

스마트기기와 ICT 기술의 발전에 따라서 사물들을 인터넷에 연결하여 다양한 정보를 공유하는 사물인터넷(IoT) 서비스들이 확대되고 있다. 사물인터넷(IoT)에 연결되는 디바이스들의 수와 종류들도 증가되고 있으며, 이에 따른 시장수요도 지속적으로 확대하고 있는 추세이다. 사물인터넷(IoT)에서 사물간의 통신은 인터넷을 기반으로 정보를 상호 전송하는 방법들이 많이 제안되고 있으나, 통신방식 중에 하나인 MQTT(Message Queue Telemetry Transport)는 낮은 대역폭, 낮은 전력을 갖는 자원이 제한된 환경에서 비동기로 메시지 교환하는 경량 메시징 프로토콜이다. 이 프로토콜의 특징은 메모리 및 전력 이용을 효율화하기 위한 가벼운 패킷 구조를 채택하고 것이다. 특히 작은 오버헤드와 낮은 대역폭, 대기시간과 신뢰 할 수 없는 네트워크 환경에 최적화되어 있다.[3]

본 논문에서는 MQTT 프로토콜 기반으로 IoT 홈 시스템을 설계 및 구현하였다. 이 시스템은 각종 센서와 데이터를 획득하는 부분, 이들 데이터를 전송받아 데이터를

*비회원
**정회원, 한국산업기술대학교 컴퓨터공학부
접수일자: 2019년 12월 31일, 수정완료: 2020년 1월 30일
게재확정일자: 2020년 2월 7일

Received: 31 December, 2019/ Revised: 30 January, 2020 / Accepted: 7 February, 2020
**Corresponding Author: jkchey@kpu.ac.kr
School of Computer Engineering, Korea Polytechnic University, Korea.

가공하는 서버 부분, 데이터를 기반으로 DC 모터와 LED를 제어하는 제어 디바이스부분과 사용자가 실시간으로 제어 및 모니터링하는 영역으로 설계하였다.

2장에서 MQTT 프로토콜 기반으로 사물인터넷(IoT) 관련 기술들을 언급하고 3장에서는 시스템과 응용소프트웨어 설계를 바탕으로 구현과 시험을 기술하였으며 4장에서 결론으로 마무리하였다

II. 관련 연구

1. 관련 기술

사물인터넷(IoT) 플랫폼을 기반으로 스마트홈 시스템들은 일반적으로 데이터를 센싱, 데이터 습득 및 가공, 전송하는 부분으로 구성하고 있다.

사물인터넷(IoT)을 기반으로 홈 모니터링 시스템에서 온도, 습도, 조도 센서로부터 센싱 데이터를 무선통신인 WiFi를 이용하여 서버로 데이터를 전송하고 서버는 이들 데이터를 분석하고 홈 디바이스를 제어하는 시스템을 제안하였다.[1][2]

스마트홈에서 최적화된 프로토콜을 사용하여 구현하는 연구가 증가하였으며, 논문[4]에서는 홈오토메이션 시스템 구현에서 WiFi 모듈(ESP8266)을 사용하여 조도, LED, 부저 등의 디바이스를 사용하였으며, 서버와 통신은 MQTT 메시지 프로토콜을 사용하여 시스템을 구현하였다.

또한 홈 서버 및 제어기는 임베디드 보드, PC, 스마트기기, 라즈베리파이 등의 다양한 디바이스로 구성하며 통신 방식도 WiFi, Bluetooth, ZigBee, IrDA, Ethernet 등을 사용하고 있다. 게이트웨이도 각각 통신 모듈을 지원하는 하드웨어로 구성하고 있다.

논문[5]에서는 MQTT기반의 홈오토메이션 시스템으로 홈 서버용으로 라즈베리파이2, 통신용으로는 ESP8266 WiFi 모듈을 사용하여 WiFi와 ZigBee 통신을 지원하는 게이트웨이를 구성하였다.

홈 서버는 OpenHAB(Open Automation Bus)이란 오픈소스를 사용하여 구현하였고, 센서들은 온도, 습도, 조도, 모션, 전류 센서들을 사용하여 홈 디바이스들을 제어하였다.

2. MQTT(Message Queue Telemetry Transport)

MQTT 프로토콜은 제한된 통신 환경을 고려하여 설계된 것으로 모바일기기에 최적의 프로토콜로 알려진 표준이다.[3][6] 프로토콜이 차지하는 모든 면의 리소스 점유를 최소화, 경량화 하였으며, 낮은 대역폭과 낮은 품질의 네트워크에서 장애와 단절에 대비하였다.

또한 메시지 사이즈는 2바이트로 오버헤드를 최소화하고, 신뢰성 있는 메시징을 위한 QoS 옵션 제공하고 있는 기술이다.[3][6]

가. MQTT 아키텍처

MQTT 아키텍처는 한 개의 브로커(Broke)에 다수의 클라이언트(Client)들로 구성되어 있으며, 각 클라이언트(Client)는 브로커(Broke)에게 자신의 고유 ID를 제공하기 위해 연결된다.

브로커(Broke)는 클라이언트(Client)들의 연결을 관리하고, 그들 사이에 메시지를 전송한다. MQTT의 전반적인 아키텍처는 TCP/IP를 기반으로 하고 있으므로 장치들 사이 메시지 교환을 한다.[3][6]

나. MQTT 메시지

MQTT 프로토콜의 데이터 교환은 클라이언트(Client)들의 발행(Publish)과 구독(Subscribe) 방식의 메시지큐(Message queue) 구조로 이루어진다.

브로커(Broker)는 발행(Publish)로 받은 명령을 구독(Subscribe)하고 있는 클라이언트(Client)에게 전달해주는 역할만 한다.

발행자(Publisher) 및 구독자(Subscriber)는 자신이 원하는 메시지를 정확하게 받거나 전달하기 위해서 토픽(Topic)을 사용 메시지를 전송한다.

토픽(Topic)은 메시지 정보와 실제 데이터 값으로 구성되며, 메시지는 문자열로 표현하고 각 계층은 “/” 문자로 구분된 계층구조로 구성하고 있다.[3][6]

다. MQTT 패킷(Packet) 포맷

패킷(Packet) 전체 구조는 2 바이트의 Fixed header, 옵션에 따른 Variable Header, Packet의 마지막 부분인 Payload로 이루어져 있다.[3][6]

Fixed header는 Message Type(4비트), DUP(1비트), QoS(2비트), Retain(1비트)로 구성되며, Message Type의 4비트는 메시지 제어 유형을 정의하며 총 16개의 데이터 유형을 정의하고 있다. 나머지는 플래그 비트로 DUP는 PUBLISH 제어 패킷의 중복 전달을 나타내고 QoS는 PUBLISH 서비스 품질을 나타내며 RETAIN는 PUBLISH 플래그 유지를 나타낸다.

Ⅲ. 시스템 세부 설계

1. 개발 환경

홈 IoT 시스템을 구현은 표 1과 같은 환경에서 시스템을 구현하였다.

표 1. 개발 환경
Table 1. Development environment

개발 환경	소프트웨어 또는 디바이스
MQTT Broker	Mosquitto, Raspberry Pi3, Node-RED
IoT Sensors	Arduino & WiFi module, Arduino IDE, 온도센서, 습도센서, 조도센서, 화재감지센서
IoT Control Devices	Arduino & WiFi module, Arduino IDE, DC모터, LED 제어
DB Server	MariaDB
Push Server	Node Socket.io
Client(Mobile)	Android Studio, 안드로이드 스마트폰

2. 시스템 구성

스마트 홈 IoT 시스템 구성도는 아래 그림 1과 같이 환경을 센싱하는 센서들과 인터페이스하는 아두이노 부분, 라즈베리파이에 구축한 브로커(Broker) 서버, LED와 DC 모터로 구성한 제어 디바이스, Mobile(스마트폰) 부로 구성한다.

홈 내에서 환경 센서들의 값에 따라서 제어 디바이스를 제어하고, Mobile(스마트폰)으로 실시간 알림을 한다. 온도/습도의 설정 값에 따라서 DC 모터가 자동으로 동작하여 원하는 값으로 확보하고, 조도 값에 따라서 LED 밝기를 제어한다. 부재 중이 홈 환경을 실시간으로 스마트폰으로 확인하고 제어할 수 있다.

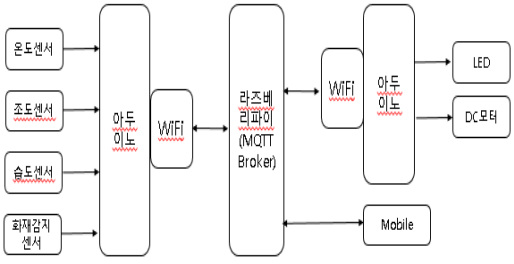

그림 1. 시스템 구성도
Fig. 1. System configuration diagram

클라이언트 발행자(Publisher)는 아두이노의 중심으로 온도, 화재감지, 조도, 습도 센서들로 데이터를 수집하였다. 구독자(Subscriber)로는 Mobile들로 구성하였다. 발행자(Publisher)에서 생성된 모든 메시지는 WiFi통신을 통해서 브로커(Broke)로 전송되며, 브로커(Broke)로는 라즈베리파이(Raspberry Pi)에 Mosquitto를 설치하여 MQTT 서버로 구축하였다.

서버에서 센서 데이터들을 분석하여 LED 점등 및 DC 모터를 제어하도록 하였다. 브로커(Broke)는 특정 메시지를 수신을 원하는 구독자에서 전송하게 된다. 구독자(Subscriber)인 Mobile은 스마트폰 응용 소프트웨어로 스마트홈의 실시간 Push 서비스를 적용하였으며 홈 내의 실시간으로 모니터링 및 제어할 수 있게 하였다.

제어 디바이스들은 수집되는 환경 데이터에 따라서 자동 또는 수동으로 동작할 수 있도록 하였다. 예를 들어 일정 온도 또는 습도 이상이 되면 DC모터 ON, 조도센서에서 일정 값 이상이면 LED ON, 화재감지센서가 검출되면 DC 모터 ON과 모바일로 알람을 제공하도록 하였다.

3. 센서와 제어 디바이스 구현

센서 및 제어 디바이스들과 인터페이스하기 위한 메인 컨트롤러는 WiFi 모듈이 내장된 아두이노 보드를 사용하였다. 이는 기존의 아두이노 우노(Uno)와 호환되는 보드로 Atmega328P와 WiFi 모듈인 WizFi250을 내장하고 있다.[7] 따라서 센서 값의 도출과 서버와 데이터를 송신을 쉽게 하였다. 센서로는 온도, 습도, 조도, 화재감지 센서로 구분되며 제어 디바이스는 DC모터와 LED 제어가 있다. 아두이노 중심으로 센서들과 구현한 것은 아래 그림 2와 같다.

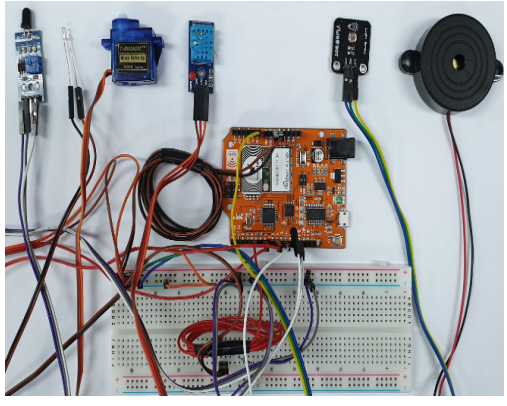

그림 2. 아두이노와 센서 디바이스
Fig. 2. Arduino and sensor devices

각 센서에서 센싱되는 데이터 값은 MQTT 토픽(Topic)에 맞게 주기적으로 송신한다. 또한 사용자가 MQTT 브로커(Broker)로 송신한 명령이 존재하면 해당 명령 토픽(Topic)을 구독(Subscribe)하여 디바이스를 제어한다.

4. MQTT 브로커(Broker) 구현

가. MQTT 브로커(Broker)

다양한 MQTT 브로커(Broker) 프로그램 중에서 Mosquitto를 사용하여 라즈베리파이에 설치하고 브로커(Broker) 역할을 수행할 수 있게 하였다. MQTT 브로커(Broke)에서는 각종 센서에서의 센싱 데이터 관리, 제어 디바이스의 명령 데이터 등 모든 데이터의 관리하며 발행자(Publisher)와 구독자(Subscriber) 사이에서 중개자 역할을 한다. 브로커(Broker)와 각종 기기를 내부 IP에 할당함으로써 N:M 통신을 구축할 수 있게 하였다. 각 기기는 자신의 정보를 브로커(Broker)에 전달하고 필요한 정보만 얻어올 수 있도록 토픽(Topic)을 지정함으로써 필요한 정보를 얻어올 수 있도록 하였다.

나. MQTT 토픽(Topic) 설계

MQTT 토픽(Topic)은 데이터 수집에서 주제를 분류하기 위하여 데이터는 Location, Device ID, Sensor Type, Subject 네 가지의 주제로 분류된다. 발행자(Publisher)와 구독자(Subscriber)는 기본적으로 “location/deviceID/sensor Type/subject” 형태의 토픽(Topic) 구조를 통하여 데이터를 전달한다. MQTT 토픽(Topic)는 그림 3과 같이 정의하였으며 필요한 정보만 갖고 올 수 있도록 하는 코드의 일부분을 보여준다. 구독자가 원하는 토픽(Topic)이 있다면 필요한 정보를 얻어올 수 있는 구조이며 구독한 토픽(Topic)의 정보를 토대로 각 디바이스의 활동을 제어할 수 있다.

Node-RED를 사용하여 그림 4와 같이 사물이나 서비스를 추상화한 노드를 와이어를 통해 연결하여 관리자만의 로직을 설계하여 각 토픽(Topic)들을 구독하고 있는 Subscriber들을 한 눈에 알아보기 쉽게 하였으며, MQTT 프로토콜을 이용하여 브로커(Broker)와의 연결 정보를 제공하여 노드들이 안정적으로 연결되어있음을 알 수 있다.

```
void callback(char* topic, byte* payload, unsigned int
 length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
// MQTT Topic
//LED원격제어
  if (strcmp("c4/light", topic) == 0) {
    ledOnOff(payload);
  }
//창문제어
  else if (strcmp("c4/window", topic) == 0) {
    setWindow(payload);
  }
// 자동 제어
  else if (strcmp("c4/auto", topic) == 0) {
    setAuto(payload);
  }
// 온도 측정
  else if (strcmp("c4/rtemp", topic) == 0) {
    rtemp = atof((char *)payload);
    Serial.println(rtemp);
  }
// 습도 측정
  else if (strcmp("c4/rhumi", topic) == 0) {
    rhumi = atof((char *)payload);
    Serial.println(rhumi);
  }
}
```

그림 3. 토픽의 정보와 디바이스 제어
Fig. 3. Topic information and control devices

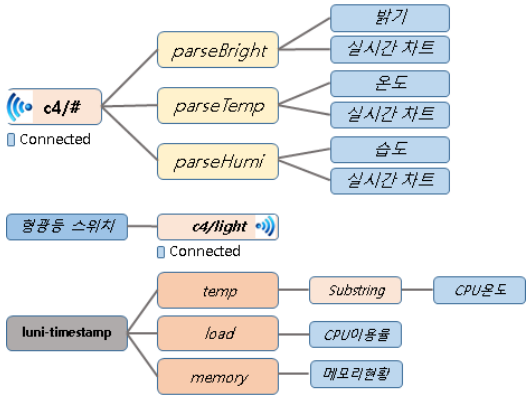

그림 4. 연결 정보
Fig. 4. Connection information

다. 실내외 환경을 고려한 자동 조정

사용자가 필요하다고 생각할 때 켜고 끌 수 있는 기능을 구현하였으며 일반적으로 사람에게 적합한 실내 환경을 기상청 정보를 기반을 두어 자동적으로 제어할 수 있

도록 설계를 하였다. 아래 그림 5은 자동화 옵션의 기기 제어 소스의 일부로 자동조정모드면 조도 값을 센싱하여 LED를 ON/OFF 또는 가정의 온도를 센싱하여 기준 값(t)보다 높고 낮음에 따라서 창문을 OPEN/CLOSED 하는 것이다.

```
void autoOpt() {
  if (autoState == HIGH) {
    if (cdsVal <= 50) { // 자동 점등
      client.publish("c4/light", "1");
    }
    else if (t > rtemp) {
      client.publish("c4/window", "1");
    }
    else if (t < rtemp) {
      client.publish("c4/window", "0");
    }
  }
```

그림 5. 디바이스 제어 소스
Fig. 5. Devices control source

라. 화재 발생 시 대응

언제 일어날지 모르는 내부 사항 문제 중 하나인 화재 발생 시 피해를 최대한 줄이기 위해 화재 감시센서를 이용해 사용자에게 즉각적 푸쉬 알람 또는 내부에 있을 시 경보가 울리도록 설계하였다. 센서가 화재를 감지하면 곧바로 화재 관련 토픽에 메시지를 보내고 실시간으로 파악하는 스마트 도어와 부저에서 각각 제어가 일어나도록 하였다. 아래 그림 6은 소스코드의 일부이다.

```
else {
    playTone(300, 160);
    if (fireState == HIGH) {
      Serial.println("화재 발생!");
      client.publish("c4/fire", "1");
// 화재 상황 발생
      myServo.attach(servoPin);
      angle = 90;
      myServo.write(angle);
      delay(500);
      myServo.detach();
      fireState = LOW;
    }
    playTone(300, 160);
    delay(300);
    playTone(300, 160);
  }
```

그림 6. 화재 감지 및 대응하는 소스
Fig. 6. Fire detection and corresponding source

5. 웹브라우저

브로커(Broker)로부터 수집된 데이터를 저장은 MQTT 토픽(Topic)과 일치하는 속성 값으로 DB를 구성하였다. DB에 저장된 데이터는 실시간으로 클라이언트로 서비스되게 하였다. Node-RED를 사용하여 따라 그림 7과 같이 사물이나 서비스를 추상화한 노드를 와이어를 통해 연결하여 관리자만의 로직을 설계하여 각 토픽(Topic)들을 구독하고 있는 구독자(Subscriber)들과 연결 정보들을 나타내었다.

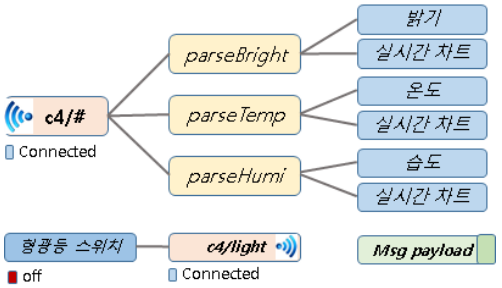

그림 7. 연결제어 흐름도
Fig. 7. Connection-control flowchart

그림 8은 브로커(Broker)를 통해 각 토픽(Topic)에 따른 실시간 데이터 값들을 도식화하여 사용자가 정보를 쉽게 알아볼 수 있도록 GUI를 구현하였다. 또한 브로커(Broker)의 하드웨어 정보를 간략하게 도식화하여 과부하 여부를 판단할 수 있게 하였고 스위치 모듈을 사용하여 특정 토픽(Topic)을 구독하는 구독자(Subscriber)를 직접 제어할 수 있도록 하였다.

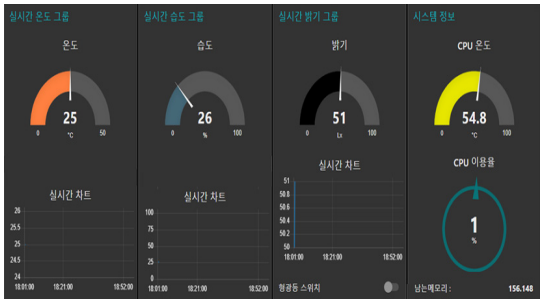

그림 8. 구현한 웹 브라우저
Fig. 8. Implemented Web-Browser

6. 스마트폰용 응용 어플리케이션 구현

앱은 최초 실행 시 그림 9와 같은 로그인 화면이 실행된다. 초기 로그인 화면에서는 WiFi 주소, ID, 비밀번호 입력하고, 이 정보를 통해 브로커 서버에 연결을 시도한다. 이후 그림 9에 화면처럼 실시간으로 온도, 습도, 조도 값을 알 수 있다. 또한 화재감지 센서를 통해서 현재 집에 화재가 발생하였는지 확인 할 수 있고 화재 발생 시 경고음과 스마트폰에 푸쉬 알람이 확인된다. 그림 10은

조명, 창문을 수동으로 작동 할 수 있고 자동화 스위치를 ON 했을 시 설정 값에 따라서 자동으로 제어할 수도 있다. 예를 들어 조도센서를 통해 값이 50이하일 때 자동 LED 점등이 되도록 설계하였다.

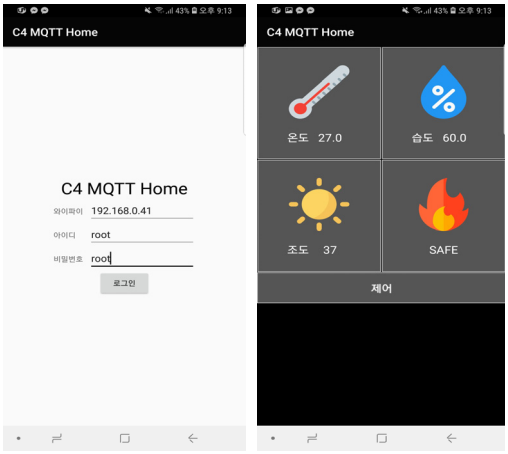

그림 9. 앱 초기와 모니터링 화면
Fig. 9. App. Initial and monitoring screen

그림 10. 디바이스 제어 메뉴
Fig. 10. devices control menu

7. 구현시스템 검증

구현한 시스템에서 그림 2와 그림 8에서 각종 센서들과 연결된 발행자에서 데이터 값들을 실시간으로 성공적으로 읽어오는 과정을 보여주고 있으며, 그림 9은 브로커(Broker)를 통해서 구독자(Subscriber)로 데이터 흐름을 보여주고 있다. 또한 구독자(Subscriber)는 원하는 토픽(Topic)을 받아 볼 수 있었다. 홈 환경 제어 조건에 따라서 디바이스를 제어하는 것은 그림 10에서와 같이 정상적으로 제어됨을 확인할 수 있었다.

Ⅳ. 결 론

최근 IoT에서 서비스와 시장이 확대되고 있는 상황에서 MQTT 프로토콜도 많은 관심이 집중되어 있다. MQTT의 장점이 가벼운 메시징 프로토콜을 이용하여 IoT 시스템에 적용한다면 실시간 데이터 수집과 다양한 센서 값들의 바탕으로 시스템의 제어, 알람 및 모니터링 등이 가능하다.

본 논문에서는 가정에서의 환경 모니터링을 기반으로 하여 홈 IoT 시스템을 구현하였다. 구현한 시제품에서 센서들로부터 데이터들을 실시간으로 수집하고 서버로 전송하였다. 센서 데이터들의 기반으로 자동 또는 수동으로 제어 디바이스를 제어를 구현하였다.

토픽(Topic)에 따라 실시간으로 받아들이는 센서들의 데이터의 값을 볼 수 있었다. 브로커(Broker)를 통해 실시간으로 받아지는 각 토픽(Topic)별 데이터를 토대로 자동으로 환경을 제어하는 시스템의 설계 및 구현하였다.

이런 시스템은 향후 실시간으로 데이터를 추적 및 모니터링이 필요한 헬스케어, 산업용 IoT, 화재 예방 등에 유용하게 응용될 것이라 기대된다.

향후 과제로 머신러닝 기반 사용자의 행동 패턴 분석 학습 및 동선 예측 알고리즘의 연구를 통해 가장 먼저 들를 장소를 파악하여 자동 점등 기능을 추가한다면 사용자의 심리적 쓸쓸함을 조금은 해소시켜 줄 것으로 기대된다.

References

[1] M. Al-Kuwari and A. Ramadan and Y. Ismael and L. AlSughair and A. Gastli and M. Benammar. “Smart-home automation using IoT-based sensing and monitoring platform”, In 2018 IEEE 12th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG 2018), pp. 1-6, Apr 2018. DOI:https://doi.org/10.1109/CPE.2018.8372548

[2] B. Anilkumar, N. Lakshmidevi, P. Choudary,“ Home Automation through Smart Phone Using Esp8266 WiFi Module By IOT”, International Journal of Current Trends in Engineering & Research (IJCTER) Vol. 3, no. 4, pp.17-21, Apr 2017.

[3] MQ Telemetry Transport, http://mqtt.org

[4] Kodali, R.K.; Soratkal, S. "MQTT based home automation system using ESP8266", In Proceedings of the IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Agra, India, pp. 1-5. Dec 2016.
https://doi.org/10.1109/R10-HTC.2016.7906845

[5] Froiz-Míguez, I.; Fernández-Caramés, T. M.; Fraga-Lamas, P.; Castedo, L. "Design, Implementation and Practical Evaluation of an IoT Home Automation System for Fog Computing Applications Based on MQTT and ZigBee-WiFi Sensor Nodes". Sensors 18, no. 8, 2018.
https://doi.org/10.3390/s18082660

[6] Mosquitto Web page. https://mosquitto.org/

[7] https://kocoafab.cc/product/orangeboard

저 자 소 개

김 우 조(비회원)

• U-zo Kim received the B.S. in computer engineering from Korea Polytechnic University in 2019. His research interests include MQTT protocol, IoT platform, etc.

최 진 구(정회원)

• Jin-Ku Choi received the Dr. Degree in electrical engineering from Waseda University, Japan in 2003. He is currently a Professor at the Department of Computer Engineering of Korea Polytechnic University. His research interests include Embedded system and design IoT platform, etc.

※ 본 연구는 2019년도 한국산업기술대학교 연구년에 의하여 연구되었음.