임베디드 소프트웨어 경진대회
The World Embedded Software Contest

제16회 임베디드SW경진대회
개발완료보고서 [자유공모]

산업통상자원부

KESSIA 임베디드소프트웨어·시스템산업협회
Korea Embedded Software and System Industry Association

1. 작성 시 주의사항

※ 제출 분량 : A4 용지 상세내용 포함 20 page 이내 (최대 폰트 크기 12pt)
※ 제출 포맷 : pdf

2. 팀 정보

팀명	황도복숭아	팀장	박혜진
팀원	김지수	팀원	김서율
팀원	이태윤	팀원	

목차

3. 본 개발완료보고서

0. 작품 제목

IoT 시스템의 대시보드 저작도구와 서버개발, YellowPeach

1. 개요

1.1 작품 개요

모든 IoT 시스템은 IoT 센서나 장치의 상태를 차트 또는 그래프의 모양으로 모니터링 하는 대시보드 응용 프로그램을 가진다. 대시보드란 한 화면에서 다양한 정보를 중앙 집중적으로 관리하고 찾을 수 있도록 하는 기능을 말하며 예시로 자동차 계기판 등이 있다. 본 팀도 경험한 바 있지만, 이 응용프로그램을 개발하는 것은 많은 시간과 노력이 필요하다.

따라서 본 팀은 쉽고 빠르게 대시보드 응용프로그램과 가상 IoT시스템을 만들 수 있는 저작도구와 서버, 테스트 IoT 응용시스템을 개발한다. 본 팀이 만든 전체 시스템을 YellowPeach라고 부른다.

본 팀은 현재 IoT시스템의 통신 프로토콜로 가장 많이 사용되는 MQTT 프로토콜로 만들어진 IoT시스템을 대상으로 한다. YellowPeach의 저작도구는 메뉴방식으로 대시보드 응용프로그램과 가상 IoT 시스템 를 저작한다. 서버에는 MQTT 브로커, 데이터베이스, 뷰어 프로그램, 가상 IoT 시스템 에뮬레이터 프로그램이 있으며 모두 라즈베리파이3 위에 개발하였다. 저작도구에서 만든 대시보드 응용프로그램은 JSON파일로 서버의 데이터베이스에 저장된다. 대시보드는 웹 브라우저를 통해 언제 어느 기기에서나 서버의 뷰어 프로그램을 실행시켜 볼 수 있다. 또한, YellowPeach는 IoT시스템이 구현되어 있지 않은 경우에도 대시보드를 만들 수 있도록 지원한다. 이 경우에는 저작도구에서 가상 IoT 시스템을 저작하여 서버의 가상 IoT 시스템 에뮬레이터 프로그램을 동작시킨다. 또한 테스트 IoT 응용시스템인 스마트 팜과 스마트 홈은 두 개의 라즈베리파이3 위에 임베디드 소프트웨어로 개발하였으며 YellowPeach의 전체 동작과정을 검정하였다.

따라서 YellowPeach는 단 몇 분 만에 다양한 종류의 차트와 컴포넌트로 구성된 대시보드를 만들 수 있으며, 대시보드의 수정 또한 간단하여 확장성을 갖추고 있다. 그 뿐만 아니라 서버는 라즈베리파이3 위에 개발하여 경량으로 이식성과 이동성을 겸비하였다.

1.2 필요성

▪ **대시보드 작성 시간과 노력단축**

본 팀이 실제 스마트 팜을 모니터링하는 대시보드 어플리케이션을 안드로이드 앱으로 만들었었는데, 안드로이드 프로그래밍 관련 공부와 개발을 포함해 한 달이 넘는 시간이 소요 되었다. 그러나 YellowPeach의 저작도구를 이용하면 코딩을 하지 않고, 10분 안에 만들어 뷰어로 볼 수 있다.

▪ **IoT 시스템 없이 대시보드 만들기 가능**

IoT 시스템의 구축과 대시보드 생성을 동시에 하거나, IoT 시스템 없이 대시보드를 만드는 경우도 많이 있다. YellowPeach는 가상 IoT 시스템을 구축하고 구동함으로써 이러한 경우를 지원한다.

1.3 개발 내용

① 저작도구

- 대시보드 응용프로그램을 저작하여 JSON 형식으로 서버의 데이터베이스에 저장하는 프로그램이다. JAVA, JAVA FX로 구현
- MQTT 브로커와 연결되어 IoT시스템의 정보를 얻어와 대시보드를 저작할 수 있음
- IoT시스템이 없이 대시보드를 만들 경우, 가상 IoT 시스템 를 만들 수 있는 에뮬레이션 메뉴 개발

② 서버

- 경량성, 이식성, 이동성을 고려하여 라즈베리파이3 위에 개발
- 오픈 소스인 MQTT 브로커, 데이터베이스, Apache Tomcat를 연동 되도록 설치
- 뷰어 프로그램 : 관리자가 모든 웹 브라우저로 서버에 접속하여 대시보드를 모니터링 하기위한 프로그램이다. HTML5, JSP로 구현
- 가상 IoT 시스템 에뮬레이터 : 가상의 센서 값을 발생하여 IoT시스템이 없어도 대시보드를 저작하고 모니터링하기 위해 제작 하였다. C로 구현

③ 테스트 IoT시스템

- YellowPeach의 전체 시스템을 검정하기 위해 직접 라즈베리파이3를 이용하여 개발
- 스마트 팜 : 토양습도센서, 조도센서, 온도센서, 카메라, 수중 모터 등으로 이루어져 있다. 자동 급수기능과, 자동 조도조절 기능으로 식물을 자동 재배한다.
- 스마트 홈 : 온도센서, 조도센서, 미세먼지센서, 불꽃 감지센서, 피에조 부저, 카메라 등으로 이루어져 있다. 모형집의 상태를 모니터링 할 수 있다.

2. 작품 설명

2.1 전체 시스템의 구조

YellowPeach의 전체 시스템의 이해를 돕기 위해 논리적인 구조와 실제 구현된 모습을 함께 나타내었다. YellowPeach의 논리적인 구조는 그림 1과 같으며, 구현된 실제 모습은 그림 2와 같다.

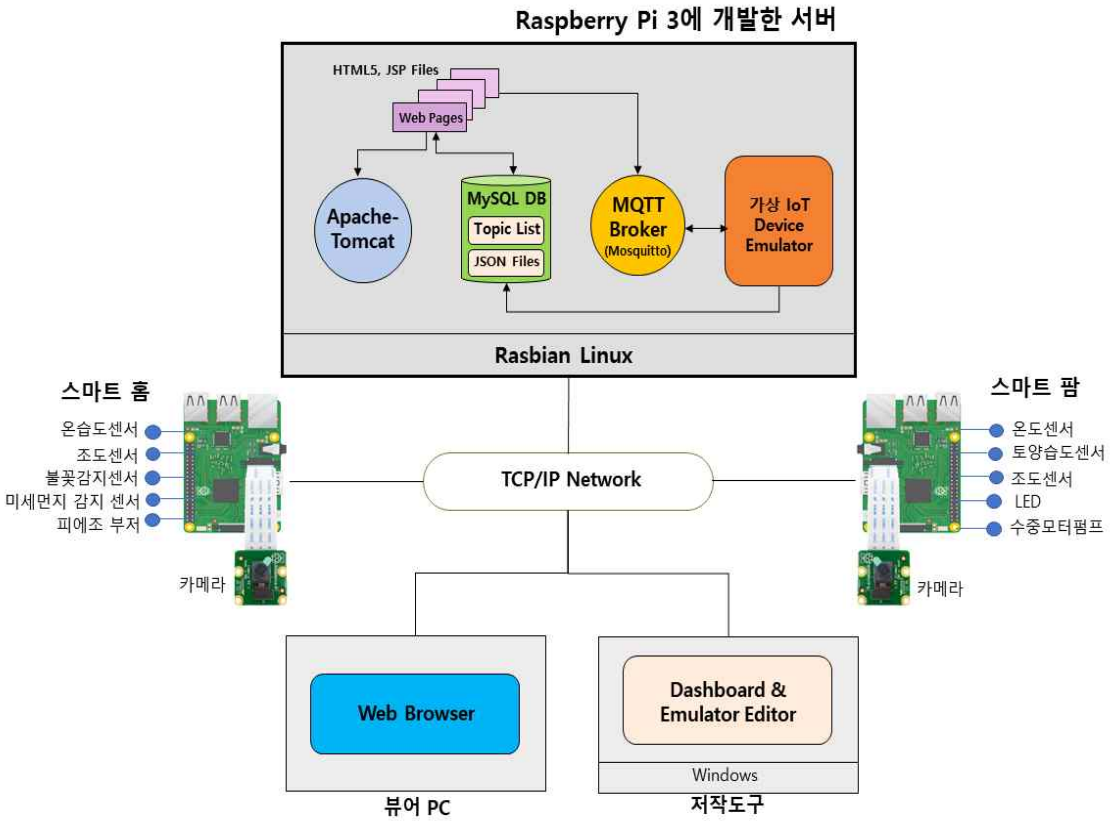


그림 1. YellowPeach의 논리 구조

그림 2. YellowPeach의 실제 구현 모습

YellowPeach의 전체 시스템은 서버, 저작도구, 2 개의 테스트 IoT시스템(스마트 팜, 스마트 홈)으로 구성된다.

① 서버

라즈베리파이3를 이용하여 경량의 서버를 개발하였으며, Apache Tomcat, MQTT 브로커, MySQL, 뷰어프로그램, 가상 IoT 시스템 에뮬레이터가 탑재되어있다.

- **Apache Tomcat**
  웹 기반의 뷰어 프로그램을 실행하기 위해 웹서버를 구축하였다.
- **MQTT 브로커(Mosquitto)**
  오픈소스 MQTT 브로커인 Mosquitto를 이용하였다.
- **MySQL 데이터베이스**
  데이터베이스에 저작도구에서 만든 대시보드 응용프로그램과 가상 IoT시스템이 JSON파일로 저장된다.
- **뷰어**
  모든 기기에서 웹 브라우저 만으로 대시보드 실행하기 위해 HTML5와 JSP로 개발하였다.
- **가상 IoT 시스템 에뮬레이터**
  C언어로 개발한 가상 IoT시스템 및 장치로 동작하는 임베디드 소프트웨어이다

② 저작도구

자바와 JavaFX를 이용하여 개발하였으며, 서버의 MQTT 브로커에 접속하고, 대시보드 응용 프로그램과 가상 IoT시스템을 제작하는 도구이다.

③ 테스트 IoT 응용시스템

라즈베리파이3를 이용하여 테스트 IoT 이용시스템을 개발하였다.

- **스마트 팜**
  자동식물재배가 가능한 임베디드 소프트웨어를 개발하였으며, 서버의 MQTT 브로커로 센서 값과 사진을 전송한다.
- **스마트 홈**
  모형 집을 센서들과 비디오 스트리밍으로 모니터링 하는 임베디드 소프트웨어를 개발하였으며, 모형 집의 상태를 서버의 MQTT 브로커로 전송한다.

2.2 서버의 구조

서버의 구조는 그림 3과 같다. YellowPeach의 서버는 리눅스 기반의 라즈베리파이3에 개발하였다. 서버에는 Apache Tomcat, MQTT 브로커, MySQL, 뷰어프로그램, 가상 IoT 시스템 에뮬레이터가 탑재되어있다.

① 웹 브라우저는 데이터베이스에 저장된 JSON파일을 불러와 대시보드를 실행한다.

② Apache Tomcat을 통해 HTML5와 JSP로 개발된 뷰어 프로그램을 실행시킨다.

③ 서버의 MQTT 브로커(Mosquitto)는 IoT 장치간의 메시지를 중개하고, 장치와 웹브라우저의 실행중인 대시보드로 센서 값을 전송하며, 명령도 중개한다.

④ 가상 IoT 시스템을 묘사한 JSON파일들은 데이터베이스에 저장된다. 이 JSON 파일은 저작도구에서 사용자에 의해 저작된다. 가상 IoT 시스템 에뮬레이터는 개수에 상관없이 JSON 파일에 묘사된 대로 가상 IoT 시스템처럼 행동하고 뷰어 프로그램의 요청이 오면 실행된다.

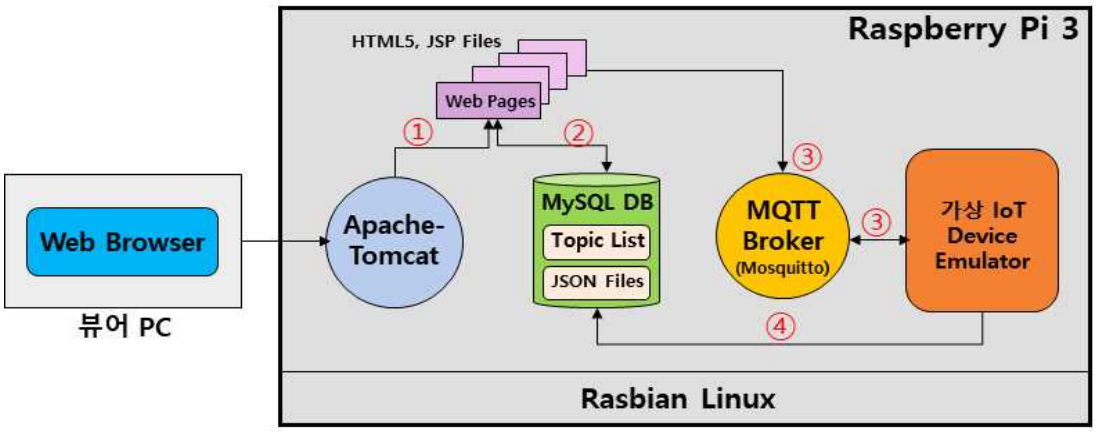


그림 3. 서버, Raspberry Pi3의 구조

2.3 테스트 IoT 시스템

2.3.1 테스트 IoT 시스템의 구조

테스트 IoT 시스템으로 스마트 팜과 스마트 홈 2 개의 IoT 응용시스템을 개발하였다. 라즈베리파이3를 이용하였고, 라즈비안 리눅스를 설치하였다.

그림 4와 같이 스마트 팜에는 온도 센서와 토양습도센서, 조도센서, LED, 수중모터펌프 총 다섯 개의 센서와 라즈베리 파이 전용 카메라, ADC컨버터를 연결하였다. 그림 5는 스마트 홈의 구조를 나타낸다. 스마트 홈에는 온습도 센서와 조도센서, 불꽃감지센서, 미세먼지 감지 센서, 피에조 부저 총 다섯 개의 센서와 라즈베리 파이 전용 카메라, ADC컨버터를 연결하였다. ADC 컨버터는 라즈베리파이에 아날로그 입력 핀이 없기 때문에 아날로그 값을 받기 위해 필수적으로 필요하며, 아날로그 값을 디지털로 바꾸어주는 컨버터를 이용하여 센서 값을 입력받았다.

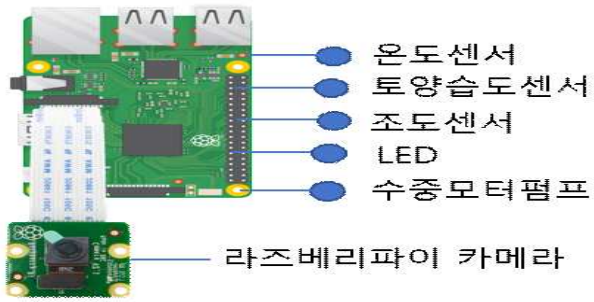


그림 4. 스마트 팜의 구조

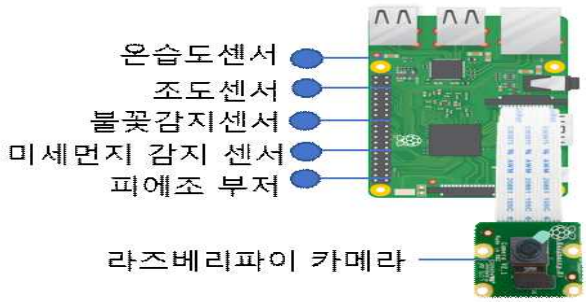


그림 5. 스마트 홈의 구조

2.3.2 스마트 팜의 제어 프로그램

스마트 팜의 제어 프로그램은 그림 6과 같이 동작한다.

- 스마트 팜은 센서 값을 측정하여 자동으로 식물을 재배한다.
- 조도에 따라 자동으로 LED를 제어한다.
- 토양습도에 따른 자동급수를 한다.
- 온도, 조도, 토양습도 값을 1초 주기로 관리자에게 전송한다.
- 관리자의 명령을 받아 사진을 전송한다.
- 센서 값과 사진은 MQTT 브로커(Mosquitto)를 거쳐 관리자 PC의 웹 브라우저에게 전송된다.

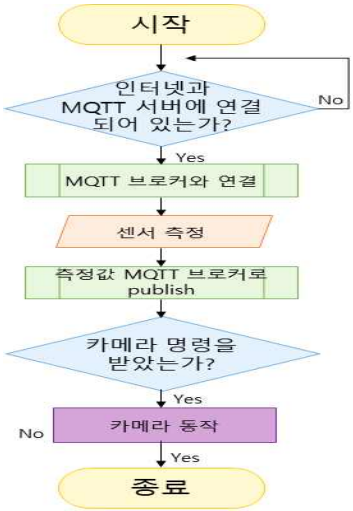


그림 6. 스마트 팜의 제어 프로그램

2.3.3 스마트 홈의 제어 프로그램

스마트 홈의 제어 프로그램은 그림 7과 같이 동작한다.

- 스마트 홈은 센서 값을 측정하여 모형 집 내부의 상태를 모니터링 한다.
- 온도, 습도, 불꽃감지, 조도, 미세먼지 값을 1초 주기로 관리자에게 전송한다.
- 스트리밍 서버가 구축되었으며 관리자의 명령을 받으면 비디오 스트리밍 전송을 한다.
- 화재 소화 명령을 받으면 부저를 울려 소화가 되었음을 알린다.
- 센서 값과 비디오 스트리밍은 MQTT 브로커(Mosquitto)를 거쳐 관리자 PC의 웹 브라우저에 전송된다.

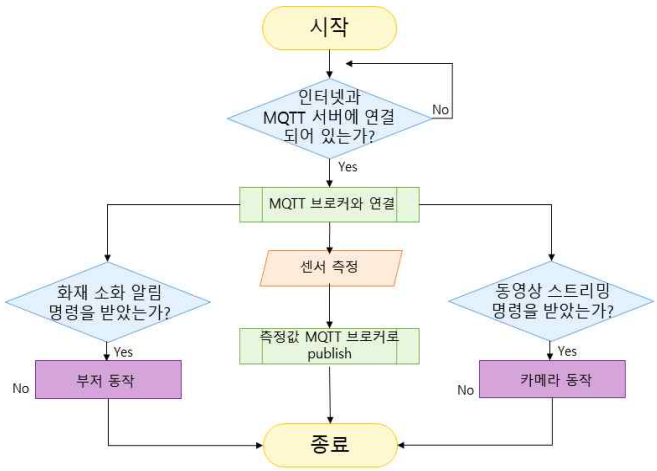


그림7. 스마트 홈의 제어 프로그램

2.4 YellowPeach의 저작도구

저작도구는 실제로 서버 라즈베리파이3와 연동하여 작동하며 저작도구의 구조는 그림8과 같다.

① 저작도구는 자바와 JavaFX로 만든 GUI가 실행된다.

② 서버의 MQTT 브로커로부터 전송받은 IoT 시스템의 정보를 GUI를 통해 사용자에게 보여주고, JDBC 드라이버를 통해 서버의 데이터베이스에 IoT 시스템 정보를 저장한다.

③ JSON 빌더는 JSON 파일을 저장한다. 저작도구에서 만든 대시보드 응용프로그램 및 가상 IoT 시스템의 JSON 파일을 그림 9는 스마트 홈을 모니터링 하기 위해 저작된 대시보드의 JSON 파일의 예시이다.

④ 생성된 JSON은 JDBC드라이버를 통해 서버의 데이터베이스에 저장된다.

⑤ 또한 JSON 파일은 로컬 저장소에도 저장할 수 있다.

⑥ JSON 파서는 서버의 데이터베이스 또는 로컬 저장소로부터 불러온 JSON 파일을 파싱하여 대시보드 및 가상 IoT 시스템을 사용자에게 보여준다.

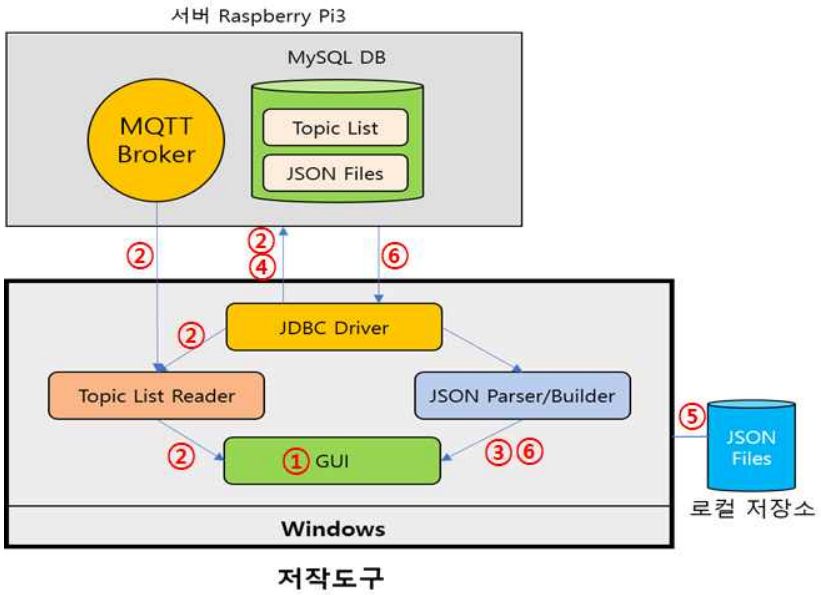


그림8. 저작도구의 구조

```
{
"systemName": "SmartHome",
  "isVirtual": false,
  "version": "PC",
  "content": [
    {
      "0": {
        "chart": "bar",
        "chartName": "Home-cds",
        "firstTopic": "SmartHome/miraeb1/cds",
        "firstName": "cds",
        "secondTopic": "null",
        "secondName": "null",
        "firstControl": "null",
        "secondControl": "null",
```

```
        "row": 0,
        "level": 1,
        "unit": "null"
      },
      "1": {
        "chart": "video",
        "chartName": "CCTV",
        "firstTopic": "192.168.0.42:8082",
        "firstName": "null",
        "secondTopic": "null",
        "secondName": "null",
        "firstControl": "null",
        "secondControl": "null",
        "row": 0,
        "level": 1,
        "unit": "null"
      },
      "2": {
        "chart": "text",
        "chartName": "memo",
        "firstTopic": "SmartHome Memo",
        "firstName": "null",
        "secondTopic": "null",
        "secondName": "null",
        "firstControl": "null",
        "secondControl": "null",
        "row": 0,
        "level": 2,
        "unit": "null"
      }
    }
  ],
  "dashboardName": "SmartHome-test"
}
```


그림 9. 저작도구에 의해 저작된 대시보드의 JSON 예시

2.5 YellowPeach의 뷰어 프로그램

뷰어는 웹 프로그램으로 작성되었고 HTM5와 JSP로 개발하였다. 프로그램의 구조는 그림 10과 같이 Apache Tomcat, MySQL 데이터베이스, MQTT 브로커와 연동하여 동작한다.

① 뷰어 프로그램은 Apache Tomcat을 통해 실행되며 관리자는 어느 기기에서나 웹 브라우저를 통해 서버에 접속하여 볼 수 있다.

② 관리자가 서버의 주소를 입력하여 뷰어를 실행시키면 Index.jsp가 실행되어 초기화면의 시작 버튼이 나타난다.

③ 시작버튼을 누르면 List.jsp가 실행되어 데이터베이스에 저장된 대시보드의 목록을 보여준다.

④ 관리자가 대시보드를 선택하면 View.jsp가 실행되고, 데이터베이스에 저장된 대시보드 JSON 파일을 파싱하여 대시보드를 볼 수 있다.

⑤ 관리자가 선택한 대시보드가 가상의 IoT 시스템으로 만들어진 대시보드일 때, MQTT 브로커에 가상 IoT 시스템 에뮬레이터가 동작하라는 명령을 전송한다.

⑥ 뷰어에서는 IoT장치 및 센서 값을 MQTT 브로커에서 실시간으로 받아 모니터링 할 수 있다.

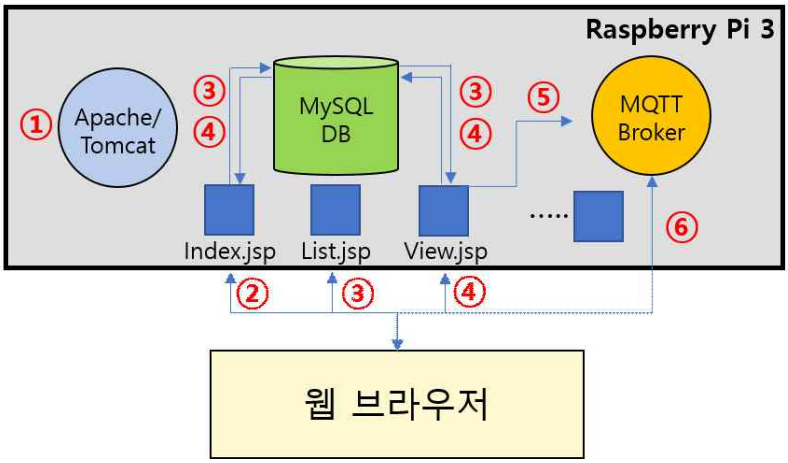


그림 10. 뷰어 프로그램의 구조

2.6 YellowPeach의 가상 IoT 시스템 에뮬레이터

가상 IoT 시스템 에뮬레이터는 그림 10과 같이 서버에 개발되었으며 MySQL, MQTT 브로커, 저작도구, 뷰어와 연동하여 동작한다.

① 저작도구에서 가상 IoT 시스템을 저작하면 서버의 데이터베이스에 JOSN 파일로 저장된다.

② 뷰어에서 대시모드의 목록에서 가상 IoT 시스템의 대시보드를 선택하여 MQTT 브로커에게 선택한 가상 IoT 시스템의 이름을 전송한다.

③ JDBC 드라이버를 이용하여 MQTT 브로커로부터 받은 가상 IoT 시스템 장치 이름을 데이터베이스에서 검색한다.

④ 데이터베이스에 저장된 JSON 파일을 불러와 JSON 파서가 실행된다.

⑤ 파싱된 가상 IoT 시스템의 정보를 MQTT 브로커에게 전송한다.

⑥ 뷰어에서는 실제 IoT 시스템이 있는 것과 같이 대시보드를 모니터링 할 수 있다.

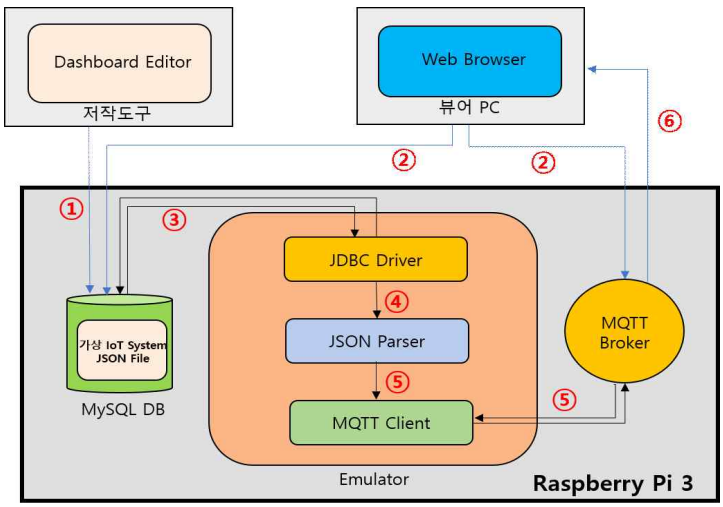
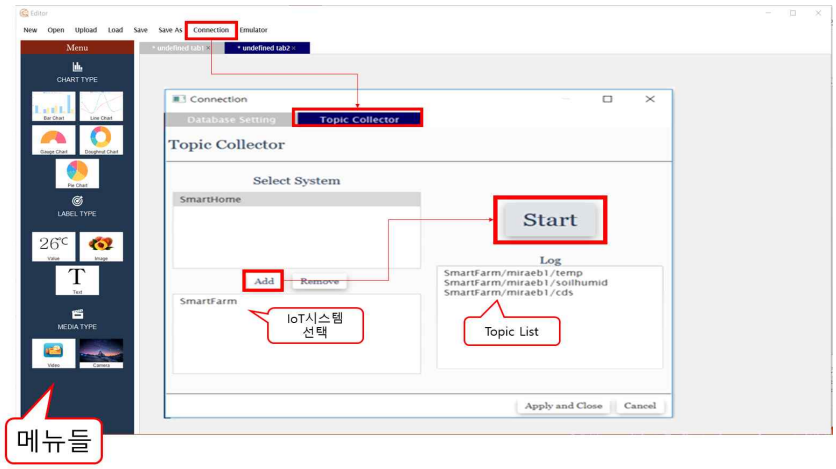


그림 11. 가상 IoT 시스템 에뮬레이터의 구조

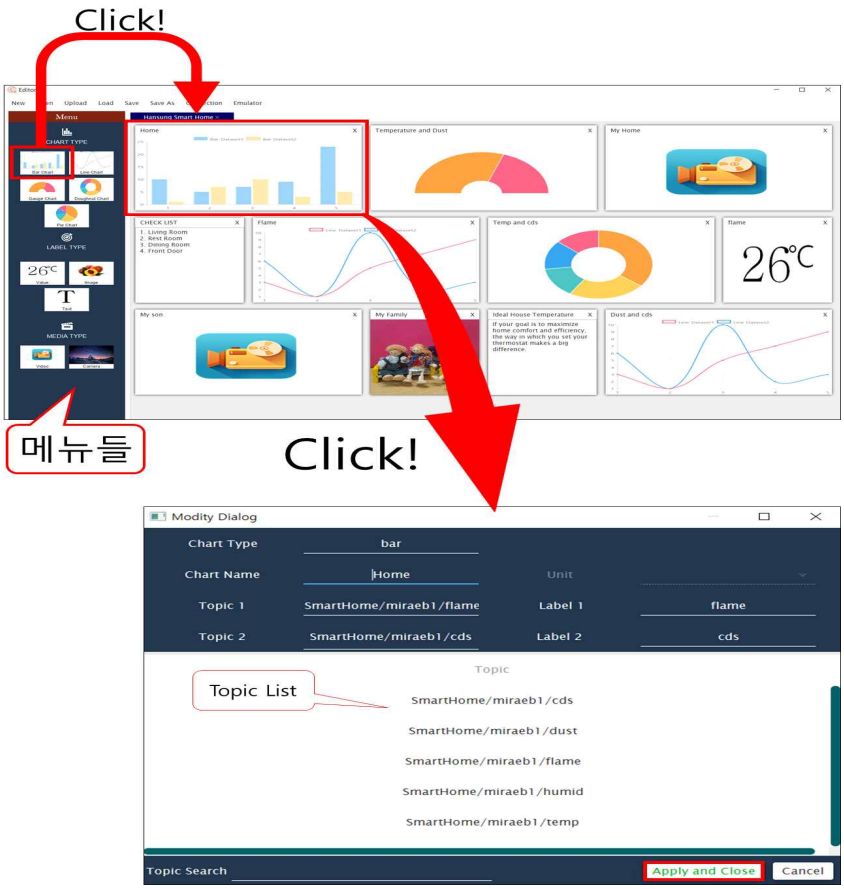
2.7 프로그램 사용법

2.7.1 IoT 시스템의 대시보드 만들기



Connection 메뉴의 Topic Collector 버튼을 눌러 서버의 MQTT 브로커로부터 IoT 시스템의 정보를 받아온다.

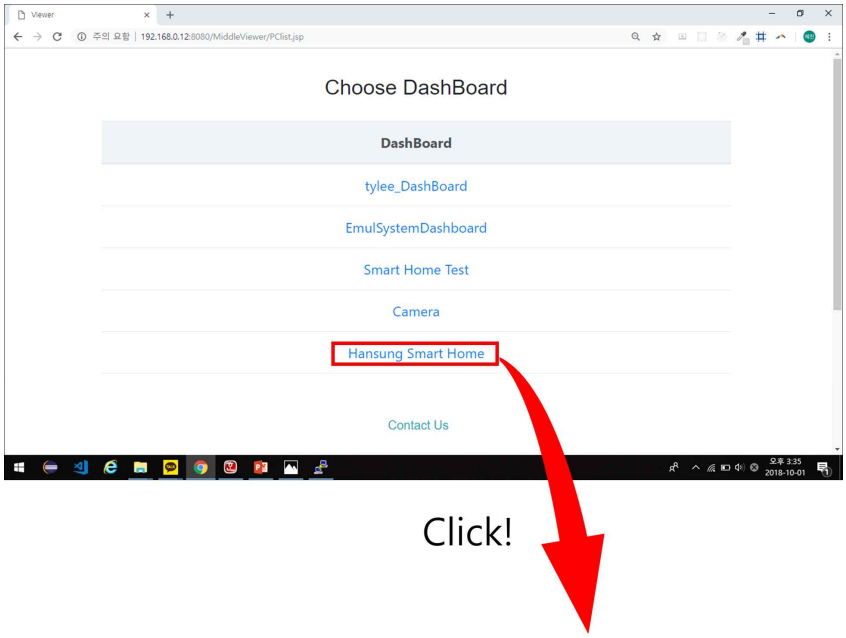
저작할 IoT 시스템을 선택하여 Topic 리스트를 확인한다.



대시보드를 구성할 컴포넌트를 클릭한다. 차트를 포함한 다양한 종류의 컴포넌트를 클릭하여 대시보드를 구성할 수 있다.

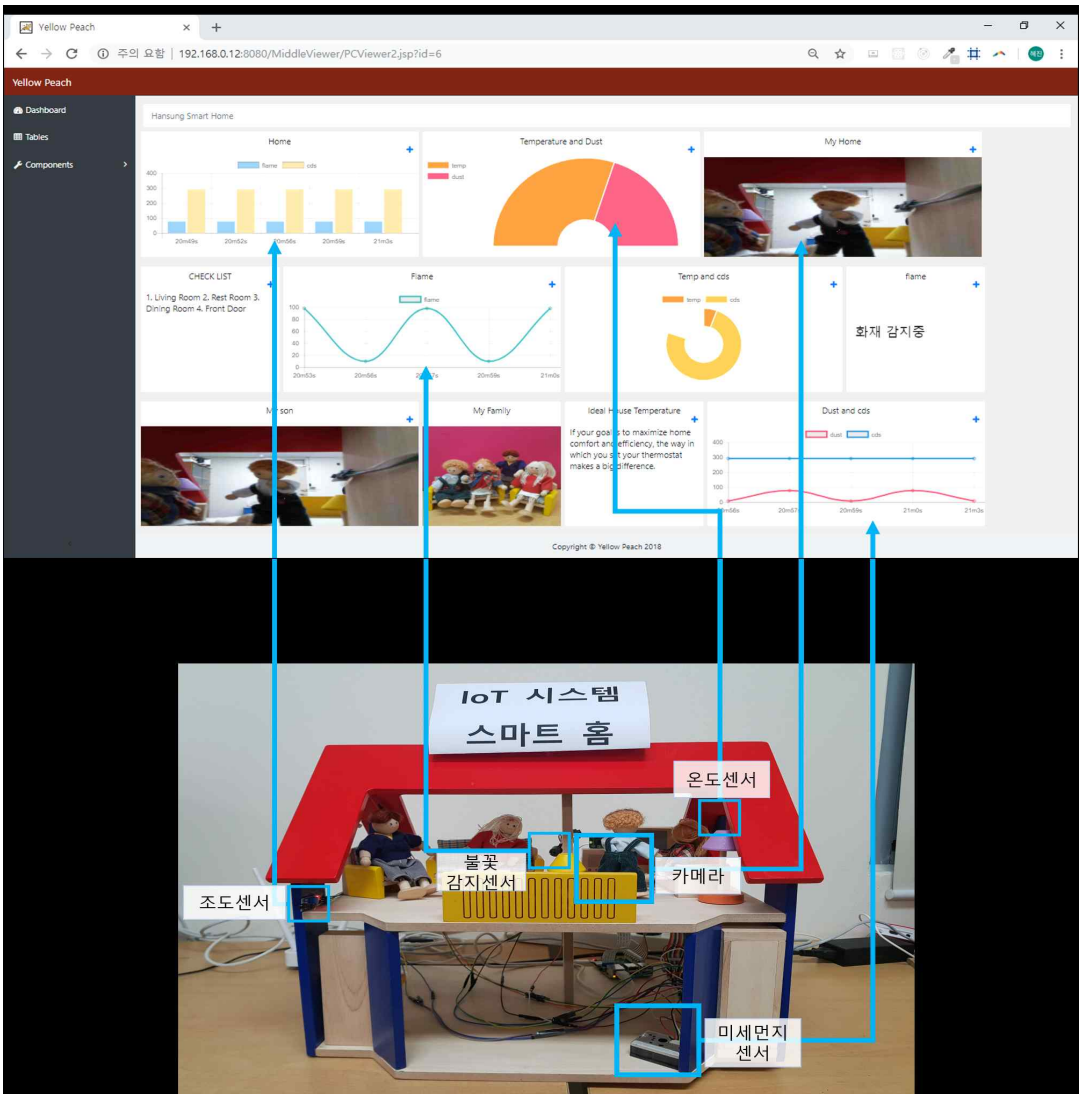
대시보드 컴포넌트를 클릭하여 차트의 정보를 입력한다.

2.7.2 대시보드 실행



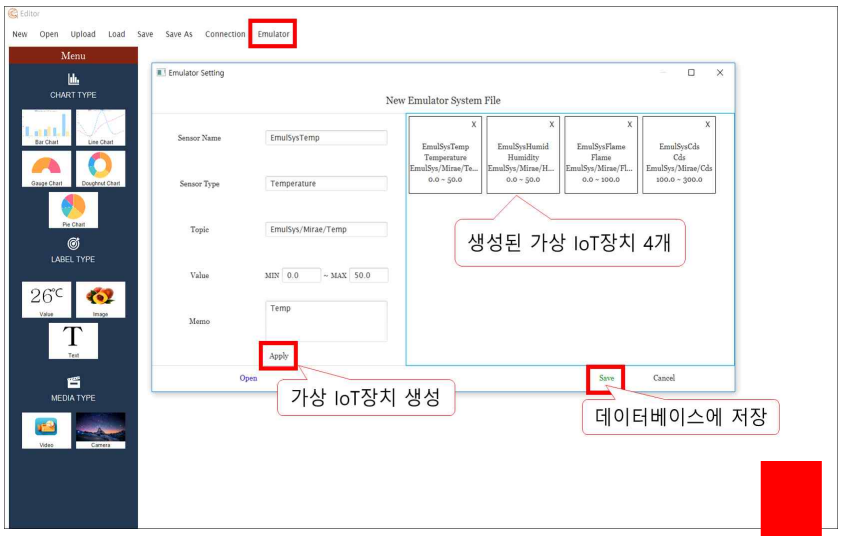
IoT 시스템 관리자는 웹 브라우저를 통해 서버에 접속한다. 대시보드 리스트에서 모니터링 할 대시보드를 선택한다.

선택된 대시보드(Hansung Samrt Home)는 관리자 pc에서 실행된다. 대시보드에는 스마트 홈에 연결된 센서 값과 동영상이 실시간으로 출력된다. 대시보드에서 IoT 장치를 관리 및 제어한다.



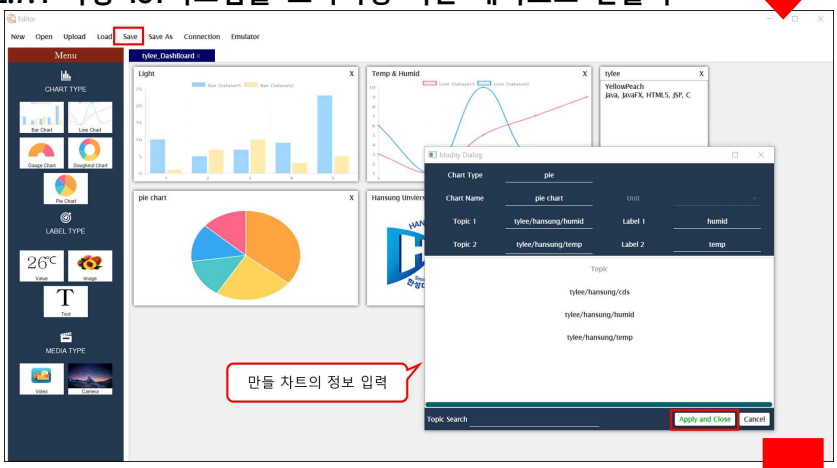
2.7.3 가상 IoT 시스템 만들기

YellowPeach는 IoT 시스템이 구축되어 있지 않은 경우, 가상의 IoT 장치를 만들고 서버에 설치된 에뮬레이터가 작동하게 함으로써 가상 IoT 시스템들로 이루어진 대시보드를 미리 만들어 볼 수 있다.



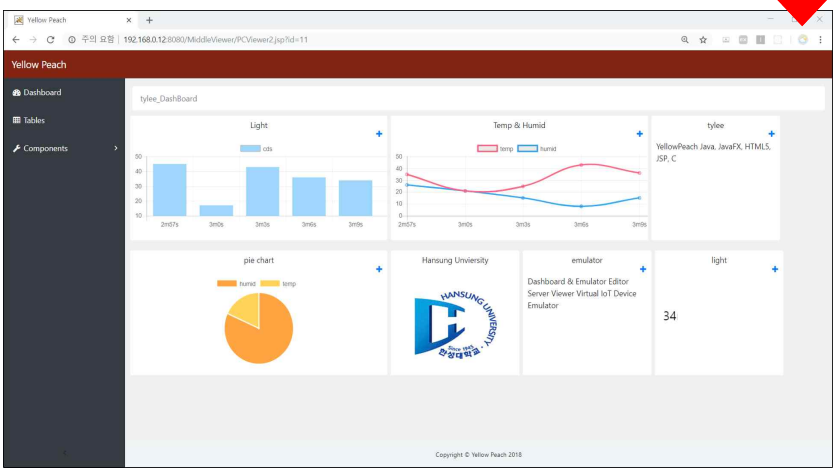
저작도구 메뉴에서 Emulator를 선택하여 가상 IoT 시스템을 저작할 수 있다. 가상 IoT 시스템의 정보를 입력하여 Apply버튼을 눌러 IoT 장치를 생성한다. 저작을 완료한 후 Save 버튼을 눌러 데이터베이스에 저장한다.

2.7.4 가상 IoT시스템을 모니터링 하는 대시보드 만들기



저작할 가상 IoT 시스템을 선택하고, 대시보드를 만든다.

2.7.5 가상 IoT 시스템을 모니터링하는 뷰어 실행



서버에 있는 에뮬레이터가 가상 IoT 시스템들을 흉내내어 대시보드로 센서 값들을 전송한다. 대시보드는 이 가상 값들을 관리자에게 실시간으로 출력한다.

2.8 개발환경

서버는 라즈베리파이 3 B 모델에서 개발하였다. 스마트 팜과 스마트 홈은 라즈베리파이 3 B 모델과 여러 센서로 개발하였다. 저작도구는 윈도우가 설치된 PC에서 JAVA, JAVA FX로 개발했다. 뷰어 프로그램은 HTML5, JSP로 개발 하였고 그 외 개발환경은 다음과 같다.

Windows10	Raspbian Linux	Eclipse	Scene Builder
Apache Tomcat	Mosquito MQTT Broker	MariaDB	Chrome Debugger

3. 프로그램 설명

3.1 파일 구성

3.1.1 서버 Raspberry Pi3

가상 IoT 시스템 에뮬레이터 파일	
Emulator.c	에뮬레이터 파일
뷰어 - WebContent 파일	
이름	내용
Index.jsp	기기를 판별하기위한 파일
MobileMain.jsp	모바일 기기의 메인페이지를 보여주는 파일
MobileList.jsp	모바일의 대시보드 리스트를 보여주는 파일
MobileViewer.jsp	모바일용 대시보드를 그려주고, 데이터를 제어할 수 있도록 하는 파일
MobileZoom.jsp	데이터를 확대해서 보여주기 위한 파일
TabletMain.jsp	태블릿 기기의 메인페이지를 보여주는 파일
TabletList.jsp	태블릿의 대시보드 리스트를 보여주는 파일
TabletViewer.jsp	태블릿용 대시보드를 그려주고, 데이터를 제어할 수 있도록 하는 파일
TabletZoom.jsp	데이터를 확대해서 보여주기 위한 파일
PCMain.jsp	PC의 메인페이지를 보여주는 파일
PCList.jsp	PC의 대시보드 리스트를 보여주는 파일
PCViewer.jsp	PC용 대시보드를 그려주고, 데이터를 제어할 수 있도록 하는 파일
PCZoom.jsp	데이터를 확대해서 보여주기 위한 파일
Viewer.css	뷰어의 스타일을 지정하는 CSS파일
custom.js	뷰어의 네비게이션 바를 작동하는 javascript 파일
뷰어 - src 파일	
이름	내용
ChartDAO.java	기기에 맞는 대시보드 리스트와, id에 해당하는 대시보드를 가져오기 위한 파일
ChartDTO.java	json데이터를 get하고 set하기 위한 파일
PoolManager.java	데이터베이스에 접속하기 위한 파일

3.1.2 저작도구

application 패키지	
이름	내용
Main.java	어플리케이션 구동을 위한 메인 클래스
Application.css	메인 css파일

application.fxml 패키지	
BoardLayout.fxml	대시보드가 그려지는 레이아웃
DatebaseOptionDialog.fxml	데이터베이스와 MQTT Broker 설정 화면의 다이얼로그
DraggableNode.fxml	대시보드 레이아웃에 올려지는 하나의 차트 레이아웃
EmulatorDialog.fxml	에뮬레이터 시스템 파일을 만드는 다이얼로그
LoadDialog.fxml	데이터베이스에서 대시보드 파일이나 에뮬레이터시스템 파일을 다운로드 하는 다이얼로그
ModifyDialog.fxml	차트의 데이터를 설정하는 다이얼로그
RootLayout.fxml	Toolbar와 BoardLayout이 올라가 있는 백 레이아웃
SelectVersionDialog	대시보드의 IoT시스템을 정하고 PC 혹은 Mobile버전은 설정하는 다이얼로그
SendDialog.fxml	로컬 저장소에있는 JSON파일을 데이터베이스에 보낼 수 있는 다이얼로그
StreammingDialog.fxml	Video나 Camera 차트의 설정을 변경하는 다이얼로그
VirtualNode.fxml	에뮬레이터 다이얼로그에서 사용자의 직관성을 위하여 나타내주는 UI
application.model 패키지	
Chart.java	하나의 차트 정보를 가진 객체 클래스
DraggableNode.java	하나의 차트를 가지고 대시보드 레이아웃에 시각적으로 하나의 차트로 나타나는 클래스
SystemInfo.java	시스템의 정보를 가지고 있는 객체 클래스
VirtualNode.java	에뮬레이터 시스템에서 하나의 토픽의 정보를 가진 객체 클래스
application.util 패키지	
Connect.java	MySQL과의 작업을 해주는 클래스
PoolManager.java	Connect클래스에서 connection pool을 관리하는 클래스
MqttSubscriber.java	선택한 IoT시스템의 디스커버리 토픽을 suubscrobe하는 클래스

3.1.3 테스트 IoT 시스템

스마트 팜	
runsensor.c	라즈베리파이에 부착된 센서들의 값을 읽고 MQTT 브로커로 publish하는 파일
simplest_web_server.c	이미지 파일을 업로드하는 웹서버 파일
pic.c	'IMG' 토픽을 subscribe하고 이미지를 촬영하여 웹서버에 업로드하는 파일
스마트 홈	
runsensor.c	라즈베리파이에 부착된 센서들의 값을 읽고 MQTT 브로커로 publish하는 파일
extinguish.c	'Extinguish'라는 토픽을 subscribe 하여 피에조 부저를 울리는 파일
mjpg_streamer.c	라즈베리파이 카메라 스트리밍을 하는 파일

3.2 함수별 기능

3.2.1 서버 Raspberry Pi 3

테스트 IoT장치 에뮬레이터 - Emulator.c	
MQTT_connect()	client 객체를 생성하여 브로커와 연결하는 함수
mysql_real_connection()	client 객체를 생성하여 데이터베이스 서버와 연결하는 함수
json_get_object()	JSON 객체를 파싱하는 함수
send_msg()	브로커로부터 토픽을 publish하는 함수
subscribe_msg()	브로커로부터 토픽을 subscribe하는 함수
뷰어 - Index.jsp	
함수명	설명
isMobile()	기기의 종류를 판별하여 기기에 맞는 JSP파일로 이동할 수 있게 해준다.
뷰어 - PCViewer.jsp, MobileViewer.jsp , TabletViewer.jsp	

함수명	설명
$(document).ready()	JSON데이터를 받아와 파싱한 후, 파싱한 데이터를 저장하는 함수
onFailure()	MQTT 브로커 연결에 실패 했을 때 다시 연결을 시도하는 함수
onMessageArrived()	MQTT 브로커로부터 메시지가 도착했을 때 차트를 그려주는 함수
onConnect()	MQTT 브로커와 연결했을 때, 파싱한 토픽을 구독하도록 하는 함수
MQTTconnect()	client 객체를 생성하여 브로커와 연결하는 함수
send_pic_message()	카메라에서 함수가 호출 되었을 경우 메시지를 publish하여 사진 찍고, 찍은 이미지를 보여주도록 하는 함수
send_message()	카메라를 제외하고 함수가 호출되었을 경우 메시지를 publish하여 control할 수 있도록 하는 함수
getList()	파싱한 데이터를 불러와 알맞게 데이터를 그려주는 함수
isVirtual()	선택한 대시보드가 가상 IoT 시스템 에뮬레이터의 대시보드일 경우 MQTT 브로커로 메시지 publish하는 함수

뷰어 - PCZoom.jsp, MobileZoom.jsp , TabletZoom.jsp	
함수명	설명
onFailure()	MQTT 브로커 연결에 실패 했을 때 다시 연결을 시도하는 함수
onMessageArrived()	MQTT 브로커로부터 메시지가 도착했을 때 차트를 그려주는 함수
onConnect()	MQTT 브로커와 연결했을 때, 파싱한 토픽을 구독하도록 하는 함수
MQTTconnect()	client 객체를 생성하여 브로커와 연결하는 함수
send_pic_message()	카메라에서 함수가 호출 되었을 경우 메시지를 publish하여 사진 찍고, 찍은 이미지를 보여주도록 하는 함수
send_message()	카메라를 제외하고 함수가 호출되었을 경우 메시지를 publish하여 control할 수 있도록 하는 함수
chartDraw()	확대하려고 하는 데이터에 맞게 확대하여 그려주는 함수

뷰어 - ChartDAO.java	
함수명	설명
ChartDAO()	PoolManager의 instance get 하기위한 생성자 함수
ChartDTO getPCDashBoard (int dashboardPcNumber)	ChartDTO를 반환하여 PC id에 해당하는 대시보드를 가져오도록 하는 함수
ChartDTO getMobileDashBoard (int dashboardMobileNumber)	ChartDTO를 반환하여 Mobile id에 해당하는 대시보드를 가져오도록 하는 함수
ArrayList<ChartDTO> getMobileList (int dashboardMobileNumber)	특정 페이지에 맞는 Mobile 대시보드의 리스트들이 담겨서 반환되어 가져오도록 하는 함수
ArrayList<ChartDTO> getPCList (int dashboarPcNumber)	특정 페이지에 맞는 PC 대시보드의 리스트들이 담겨서 반환되어 가져오도록 하는 함수

3.2.2 저작도구

Main.java	
함수명	설명
toJson()	현재 선택된 대시보드 탭의 정보를 JSON파일로 변환하는 함수
checkDBConnect()	데이터베이스의 연결상태를 확인하는 함수
Connect.java	
getIotSystemList()	데이터베이스에서 IoTSystem의 목록을 가져오는 함수
insertIoTSystemList()	IoTSystemList테이블에 새로운 IoTSystem을 추가하는 함수
updateIoTSystemList()	데이터베이스에 있는 IoTSystem을 수정하는 함수
isIoTSystemExist()	IoTSystemList에 선택된 IoTSystem이 이미 존재하는지 확인하는 함수
getDashboard()	데이터베이스에서 대시보드를 가져오는 함수
insertDashboard()	데이터베이스에 대시보드를 추가하는 함수
updateDahsvoard()	데이터베이스에 있는 대시보드를 수정하는 함수
isDashboardExist()	데이터베이스에 선택된 대시보드가 이미 존재하는 지 확인하는 함수
getEmulatorSystem()	데이터베이스에 있는 에뮬레이터 시스템을 가져오는 함수
insertEmulatorSystem()	데이터베이스에 에뮬레이터 시스템을 추가하는 함수

updateEmulatorSystem()	데이터베이스에 있는 에뮬레이터 시스템을 수정하는 함수
isEmulatorSystemExist()	데이터베이스에 선택된 대시보드가 이미 존재하는 지 확인하는 함수
createEmulatorTable()	생성된 에뮬레이터 토픽들이 담길 테이블을 만드는 함수
insertTopicToEmulatorTable()	에뮬레이터 토픽 테이블에 토픽들을 넣는 함수
dropEmulatorTopicTable()	에뮬레이터 토픽 테이블을 삭제하는 함수
RootLayoutController.java	
handleNew()	새로운 비어있는 대시보드를 만드는 함수
createNode()	DraggableNode 생성하는 함수
handleOpen()	대시보드 파일을 여는 함수
handleSave()	현재 열려있는 파일에 대시보드를 저장하는 함수
handleSaveAs()	FileChooser를 열어서 사용자가 저장할 파일을 선택하게 하는 함수
handleSend()	send다이얼로그를 생성하는 함수
handleLoad()	Load다이얼로그를 생성하는 함수
showModifyDialog()	차트 수정 다이얼로그를 여는 함수
showSelectVersionDialog()	대시보드의 IoT시스템을 정하고 PC 혹은 Mobile버전은 설정하는 다이얼로그를 생성하는 함수
handleDatabase()	데이터베이스와 MQTT Broker 설정 화면의 다이얼로그를 생성하는 함수
handleEmulator()	에뮬레이터 시스템 파일을 만드는 다이얼로그를 생성하는 함수
DatabaseOptionDialogController.java	
initialize()	dbData.json을 파싱하여 데이터베이스 정보를 가져와 뿌려주는 함수
setList()	데이터베이스에서 IoTSystem파일을 가져와 List안에 넣는 함수
handleStart()	MQTT Broker에서 선택한 시스템의 토픽들을 가져오는 함수
EmulatorController.java	
handleApply()	VirtualNode를 Field에 추가하는 함수
toEmulatorJson()	사용자가 만든 가상 IoTSystem을 JSON파일로 변환하는 함수
handleSave()	사용자가 만든 가상 IoTSystem JSON파일을 저장하는 함수

4. 개발 중 장애요인과 해결방안

▶ 개발 프로세스 결정의 어려움

▪ 프로토콜 선택의 어려움

기존의 많은 연구에서 MQTT가 HTTP보다 빠르게 전송한다고 알려져 있어, 본 개발에서는 IoT 시스템의 프로토콜로 MQTT를 사용하기로 결정하였다. 한 예로 'flespi'의 연구를 참고하면(flespi, *"HTTP vs MQTT performance tests"*, https://flespi.com/blog/http-vs-mqtt-performance-tests) MQTT가 HTTP보다 25배 빠른 전송을 한다.

▪ 저작도구 개발도구 선정의 어려움

네트워크에 연결이 되어있지 않아도 프로그램을 실행할 수 있도록 하기 위해 웹이 아닌 자바 어플리케이션으로 개발하였다. 그리고 자바 스윙과 비교했을 때, UI구성이 더 간편하고 고수준의 그래픽 작업을 수행할 수 있기 때문에 JavaFX를 이용하여 저작도구를 개발하였다.

▪ 뷰어 개발도구 선정의 어려움

기기 종류에 상관없이 데이터를 모니터링하기 위하여 부트스트랩 프레임워크를 사용하여 웹으로 개발하였고, 어플리케이션에 적합하고 손쉽게 차트를 그리기 위해 Chart.js 라이브러리를 사용하여 뷰어를 개발하였다.

▶ **개발환경의 불편함**

장소의 제약 없이 YellowPeach를 사용하기 위해 PC아닌 라즈베리파이3에 서버를 구축하여 언제 어디서나 사용할 수 있도록 하였다.

▶ **하드웨어 구성의 어려움**

하드웨어에 익숙한 전자정보공학과 친구와 팀을 구성하였다.

5. 개발결과물의 차별성

- **코딩 없이 대시보드 제작 가능**

  YellowPeach의 저작도구와 뷰어프로그램을 이용하면 코딩 없이 대시보드를 만들 수 있다.

- **단 5분 안에 대시보드 어플리케이션 저작가능**

  저작도구의 메뉴방식을 통해 대시보드 어플리케이션을 단 5분 안에 만들 수 있다.

- **다양한 차트와 컴포넌트로 구성된 대시보드**

  바, 라인, 게이지, 도넛, 파이 차트와 텍스트, 이미지, 카메라, 비디오 스트리밍 등의 다양한 컴포넌트로 대시보드를 만들 수 있다.

- **경량으로 개발된 서버**

  라즈베리파이3에 구현된 서버는 경량으로 개발되었고, 이식성과 이동성을 겸비하였다.

- **가상 IoT시스템 구축 가능**

  실제 IoT 시스템이 없는 상태에서도 가상의 IoT 장치를 만들고 서버에 설치된 에뮬레이터가 작동하게 하여 가상 IoT 시스템들로 이루어진 대시보드를 미리 만들어 볼 수 있다. 따라서 YellowPeach를 이용하면 사전에 대시보드를 만든 후에 IoT 시스템을 구축할 수 있다.

- **웹 브라우저를 통한 뷰어 실행**

  어느 기기에서나 웹 브라우저를 통해 서버에 접속하면 대시보드를 모니터링 할 수 있어 IoT 시스템 관리에 용이하다.

- **간편하고 빠른 대시보드의 수정**

  새로운 IoT 장치를 추가, 제거하여도 별도의 작업 없이 저작도구만으로 간편하게 대시보드를 수정할 수 있다.

6. 단계별 개발계획 및 실제 참여인원 및 업무 분장

6.1 제작자 정보

No.	성명	담당 업무
1	박혜진	Raspberry Pi 환경 구축, 테스트 IoT System 개발, Emulator System 개발
2	김지수	Dashboard Viewer 개발
3	김서율	DashBoard & Emulator Editor 개발
4	이태윤	Dashboard Viewer 개발, 테스트 IoT System 개발

6.2 단계별 개발계획 및 업무분장