기상정보와 주변 정보의 매시업을 통한 상품추천시스템 구현

이주은*, 김유진*, 김채연*, 이은솔O, 장재석*, 최재홍*, 이준동*
강릉원주대학교, 멀티미디어공학과O
강릉원주대학교, 멀티미디어공학과*
e-mail: dlwndms7308@naver.com*, you_j_k@naver.com*, kcy1060@naver.com*, dldmsthf2929@naver.comO,
wowx1001@naver.com*, inform1@gwnu.ac.kr, jlee@gwnu.ac.kr*

Implementation of product recommendation system through mashup of weather information and peripheral information

Ju-Eun Lee*, You-Jin Kim*, Chae-Yeon Kim*, Eun-Sol LeeO,
Jae Suk-Jang*,Sung-Jin Kim*, Jae-Hong Choi*, Jun-Dong Lee*
Dept. of Multimedia Engineering, GangNeungWonju UniversityO
Dept. of Multimedia Engineering, GangNeungWonju University*

● 요 약 ●

본 논문에서는 다양한 아두이노 무선센서 모듈과 Raspberry Pi, 웹서버를 이용한 IOT 기반 환경정보 수집시스템과 기상청 API를 통한 기상정보, 상점 서비스를 매시업하여 상품추천시스템을 구현하였다. 이 시스템은 사용자가 주변 환경의 데이터를 정확하게 확인하고 그에 맞는 상품을 추천받을 수 있도록 한다. 상품추천시스템에서는 상점 외부에 부착된 환경정보 수집시스템에서 측정한 데이터와 기상청 API 데이터를 DB에 저장하고 DB에 저장된 데이터를 이용하여 상황에 맞는 기후화면디자인과 환경정보 데이터를 html로 구성하여 보여준다. Raspverry Pi에 연결된 모니터를 통해 실시간으로 정보를 보여주며 일정 시간 간격으로 관련 상품 광고를 보여주며 필요한 물건을 추천해준다.

키워드: 사물인터넷(IOT), 아두이노(Arduino), 라즈베리파이(Raspberry Pi), 센서(Sensor), API(Applic ation Programming Interface), 상품추천(Merchandise Recommendation)

I. Introduction

매시업(Mashup)이란, 다양한 정보(콘텐츠)와 서비스를 혼합하여 새로운 서비스나 융합 애플리케이션을 개발하는 것을 의미하는 단어이다. 구글 지도와 부동산 정보사이트인 크레이그 리스트를 결합시킨 '하우징맵' 사이트가 대표적인 매시업의 예이며 지도 정보에서 특정 지역을 선택하면 해당 지역의 부동산 매물정보를 보여주는 서비스를 제공한다. 매시업의 장점은 기존의 자원을 활용하여 만들기 때문에 새로운 서비스를 구축하기 위하여 투여되는 비용이 매우 적다는 점이다. 본 논문에서는 매시업을 활용하여 주변 환경데이터를 측정하는 데이터 환경정보 시스템과 상점 서비스를 결합하고 추가로 기상청 API에서 가져온 데이터를 이용하여 사용자에게 주변의 환경정보와 필요한 상품을 추천해주는 IOT 기반 상품추천시스템을 구현하였다.

II. Preliminaries

1. 시스템 제안

IOT 기반 상품 추천시스템은 소비자가 이용하는 상점 주변의 환경정보 데이터를 쉽게 확인할 수 있도록 상황에 따른 기후 상태창을 모니터에 띄워주며 일정 시간 간격으로 관련 상품 광고를 보여준다. 광고 창에서는 기후에 따라 상점 내 마스크, 자외선 차단제, 우산, 음료, 안공눈물 등 필요한 물건들을 추천해주는 구조로 설계하였다. IOT 기반 상품 추천시스템은 상점 외부, 내부에 설치하여 이용자가 본 시스템을 효율적으로 활용할 수 있도록 한다.

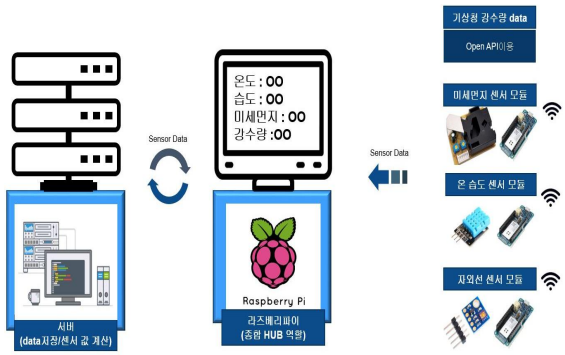

Fig. 1. Merchandise Recommendation System Architecture

2. 시스템 구성

2.1 환경정보 수집시스템

환경정보 수집 시스템은 아두이노와 UV센서, 미세먼지센서, 온습도센서, 라즈베리파이를 이용하여 만든 시스템이다. 환경정보 데이터 측정을 위하여 각각의 아두이노와 센서들을 연결하고 아두이노 IDE를 활용하여 센서 데이터 측정 수치 및 시간을 설정한다. 환경데이터를 측정하여 값을 얻으면 아두이노에 웹서버를 만들어 센서값이 30초 간격으로 웹서버에 뜰 수 있도록 작업하였다. 라즈베리파이에서는 30초 간격으로 아두이노 웹서버에 올라간 센서값을 가져와서 DB에 저장하도록 작업하였다.

```
MariaDB [test]> select fdust,UV,TEMP,WET,WFKOR,DATE_FORMAT(test_s.DATE,'%Y-%m-%d %H:%i') AS DATE,day from test_s,test_w
    -> where day=0 and
    -> DATE(test_s.DATE)=test_w.DATE and
    -> test_w.HOUR-HOUR(test_s.DATE)>=0 and
    -> test_w.HOUR-HOUR(test_s.DATE)<=3 and
    -> test_w.g_code='4213037000 ' and
    -> test_s.fdust!=0 order by DATE DESC limit 1;
+-------+------+------+------+-------+------------------+-----+
| fdust | UV   | TEMP | WET  | WFKOR | DATE             | day |
+-------+------+------+------+-------+------------------+-----+
|    45 |  4   |   29 |   30 | 맑음  | 2019-06-04 16:17 |   0 |
+-------+------+------+------+-------+------------------+-----+
1 row in set (0.07 sec)
```

Fig. 2. System Architecture_Sensor value transfer result to DB

2.2 기상청 API

환경정보 수집시스템에서 사용된 미세먼지센서, 온습도센서, UV센서로 수집할 수 없는 강수상태를 추가하기 위하여 기상청 API를 사용하였다. 기상청에서 제공하는 xml파일의 구문을 분석하여 기상청 데이터를 파싱한 후 테이블에 저장하였다. 기상청 데이터를 파싱할 때, 특정 장소의 데이터를 얻어야 하므로 위도 경도를 그대로 입력하는 것이 아니라 기상청이 지정한 특수한 값(grid x, y)을 변환해야 한다. 읍/면/동 단위로 이 값이 존재하며, 사용자의 편의성을 위해 서버 데이터베이스 Login테이블에 자바코딩을 통해 읍/면/동 이름과 지역코드 및 gridx, y를 저장해 두었다.

<hour>**24**</hour>	동네예보 3시간단위(21시~24시까지)
<day>**0**</day>	1번째날(오늘/내일/모레 중 오늘)
<temp>**3.1**</temp>	현재시간온도(21시~24시)
<tmx>**-999.0**</tmx>	최고온도 missing(값이 없을 경우)
<tmn>**-999.0**</tmn>	최저온도 missing(값이 없을 경우)
<sky>**4**</sky>	하늘상태코드 [맑음(1), 구름조금(2), 구름많음(3), 흐림(4)]
<pty>**0**</pty>	강수상태코드 [없음(0), 비(1), 비/눈(2), 눈(3)]
<wfKor>**흐림**</wfKor>	날씨한국어
<wfEn>**Cloudy**</wfEn>	날씨영어
<pop>**23**</pop>	강수확률%
<r12>**0**</r12>	12시간 예상강수량
<s12>**0**</s12>	12시간 예상적설량
<ws>**1.7000000000000002**</ws>	풍속(m/s)
<wd>**2**</wd>	풍향
<wdKor>**동**</wdKor>	풍향한국어
<wdEn>**E**</wdEn>	풍향영어
<reh>**68**</reh>	습도%

Fig. 3. Meteorological Agency Data Form

이 중 날씨 상태를 문자로 나타내는 wfKor부분과 시각을 나타내는 hour, day를 자바코딩을 통해 주기적으로 파싱한다. hour는 예보 3시간 단위로, 태그 값이 12라면 9시부터 12시까지 라는 뜻이다. day는 0,1,2로 구성되어있으며 당일은 0, 내일은 1, 내일모레는 2이다. 부가적으로 파싱한 날짜를 알아야 하므로 데이터 삽입날짜(DATE)를 서버컴퓨터 데이터 베이스안의 test_w테이블에 추가 한다.

```
MariaDB [test]> select * from test_w;
+------+------+---------------+------------+------+------------+
| HOUR | TEMP | WFKOR         | DATE       | day  | g_code     |
+------+------+---------------+------------+------+------------+
| 21   | 12.0 | 맑음          | 2019-05-20 |    0 | 4213037000 |
| 24   | 11.0 | 맑음          | 2019-05-20 |    0 | 4213037000 |
| 3    | 10.0 | 맑음          | 2019-05-20 |    1 | 4213037000 |
| 6    | 10.0 | 맑음          | 2019-05-20 |    1 | 4213037000 |
| 9    | 13.0 | 맑음          | 2019-05-20 |    1 | 4213037000 |
| 12   | 18.0 | 구름 조금     | 2019-05-20 |    1 | 4213037000 |
| 15   | 20.0 | 구름 많음     | 2019-05-20 |    1 | 4213037000 |
| 18   | 19.0 | 맑음          | 2019-05-20 |    1 | 4213037000 |
| 21   | 16.0 | 맑음          | 2019-05-20 |    1 | 4213037000 |
| 24   | 14.0 | 맑음          | 2019-05-20 |    1 | 4213037000 |
+------+------+---------------+------------+------+------------+
10 rows in set (0.00 sec)
```

Fig. 4. Save to database after API parsing

2.3 DB 테이블 설계 및 가공

데이터베이스 안에 테이블은 총 3개이며, Login, test_w, test_d(센서 종합 데이터 테이블)로 구성되어 있다. 이 3개를 조인 할 것이며 html의 자바스크립트는 조인된 테이블의 데이터를 꺼내 분석하여 사용자에게 보여준다. 조인이 필요한 이유는 센서 데이터 수집 시간과 기상청 외부 데이터 초기화 주기, 위치가 다르므로 시간과 위치를 기준으로 조인을 한다.

test_w, test_d는 각각 Login테이블의 지역코드와 서로의 HOUR, DATE를 기준으로 조인한다. 이를 통해 Login의 테이블의 위치명, 주소 값은 참조하되, test_d, test_w 데이터를 변경 시 Login테이블 데이터를 연쇄적으로 제거, 변경하지 못하게 한다. 또, 모듈의 특성상 예보를 통해 상품을 추천하므로 매일 데이터를 갱신하고 제거해야 한다. 그러므로 test_w 또는 test_d의 하루 지난 데이터들은 이용할 가치가 없으므로 데이터 생성 날짜를 기준으로 주기적으로 제거한다.

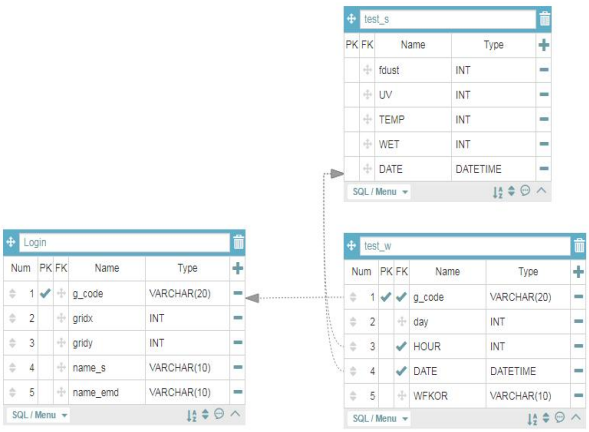

Fig. 5. database ERD

3. 실험

환경정보수집시스템에서 얻어온 환경정보데이터와 기상청 API에서 파싱한 데이터가 DB에 저장되고, 저장된 데이터 값을 이용하여 상황에 따른 기후 상태 창과 일정 시간 간격으로 관련 상품 광고를 잘 보여주는지 실험해보았다. 실험 결과 아래 그림 6처럼 기후 상황에 따라 상태 창이 잘 나타나는 것을 확인할 수 있으며 기후 상황에 적절한 광고 영상이 잘 실행되는 것을 알 수 있었다.

Fig. 6. climatic status screen

III. Conclusions

본 논문에서는 기상청 API와 환경정보수집시스템 데이터를 종합하여 이용자에게 정확한 기후정보를 전달하고 기후정보에 따라 관련 상품을 추천한다. 실험을 통해 환경정보 수집 시스템이 정상적으로 동작하는 것을 확인하였고, 기상청 API에서 파싱한 데이터가 DB에 잘 저장되는 것을 확인하였으며, DB에 저장된 데이터에 따라 모니터 기후정보 화면과 광고창이 바뀌는 것을 확인하였다. 이후 연구에서는 상품추천시스템과 상점 DB를 연동하고, 재고량 및 할인정보를 이용하여 더 편하고 좋은 서비스를 제공하도록 할 것이다.

ACKNOWLEDGEMENT

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단 -현장맞춤형 이공계 인재양성 지원사업의 지원을 받아 수행된 연구임 (No. NRF-2017H1D8A1031020).

REFERENCES

[1] 레이몬드 이.(2009). 프로 웹 2.0 매쉬업. 경기: 위키북스
[2] 김진경, 라상용, 최재홍, 이준동. "IoT 허브 구현." 한국컴퓨터정보학회 학술발표논문집 , 25.2 (2017.7): 157-158.
[3] 김진경, 라상용, 최재홍, 이준동. (2018). R.Box에서의 센서 네트워크와 CMS 서버 구현, 한국컴퓨터정보학회 학술발표논문집 ,26(1), 77-78
[4] 다카모토 다카요리.(2016). 모두의 아두이노 (장진희, 옮김). 서울: 길벗
[5] 우재남.(2017). 이것이 우분투 리눅스다. 서울: 한빛미디어
[6] 천인국.(2015). 어서와 Java는 처음이지!. 경기: 인피니티북스
[7] 기상청 데이터 xml, http://www.kma.go.kr/wid/queryD FS.jsp?gridx=(위도변환값)&gridy=(경도변환값)