DU-도전학기 결과보고서

성 명		학 번	
단과대학		학과(전공)	
도전학기 과제명	(국문) 라즈베리파이를 활용한 스마트 원예 IoT 프로젝트 설계 (영문) Smart Garden IoT Project design using Raspberry Pi		
지도교수 의견	목표하였던 기능들을 수행하였다 판단됨. 과제를 통해 많은 성과를 내었다 생각함. (성명) (소속)		

1. 도전 과제의 목표

Open API를 통해 실시간 날씨 데이터를 받아 자동으로 화단과 화분을 관리하는 스마트 원예 시스템을 구축하는 것이다. HTML 웹 페이지를 제작하여 API 데이터가 정상적으로 넘어오는지 확인하며 아두이노와 Wi-Fi 모듈을 통해 데이터를 받아 화단과 화분을 자동으로 제어하는 시스템이다. 또한 자동으로 관리되는 것에 이어 모바일 애플리케이션을 개발하여 사용자가 즉각적으로 제어할 수 있도록 하는 것이 이 프로젝트를 진행한다. 목표한 부분까지 성과를 내면서 단순히 구현하는 것이 끝이 아닌 내가 직접 이해하고 공부해서 내 것으로 만들어야 의미 있는 목표라 생각한다.

2. 도전 과제 내용

[붙임 1] 참고

3. 도전 과제의 성과

프로젝트를 완성시키고 나서 교내외 경진대회 참가 및 수상, 특허 출원 진행 중, 논문 작성을 하였다. 교내에서 진행된 창업아이디어 경진대회에서 장려상을 수상하였고, 캡스톤 디자인 경진대회(공학제)에 참가하여 입상하였다. 뿐만 아니라 '마산 로봇랜드' 에서 진행된 전국지능로봇경진대회에 참가하여 장려상을 수상하였다. 그리고 Linc+사업단의 지원으로 프로젝트 아이템에 대해 특허 출원을 진행 중이다. 또한, '대구대학교 정보통신연구소' 소속으로 캡스톤 디자인 출품작으로 논문을 작성하였다.

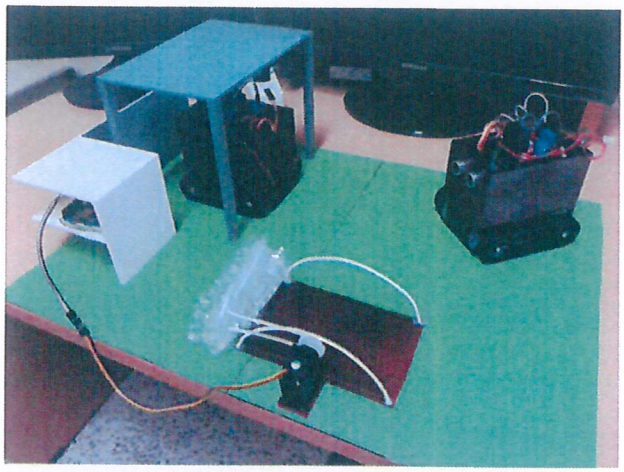
[붙임 2] 참고

4. 자기평가

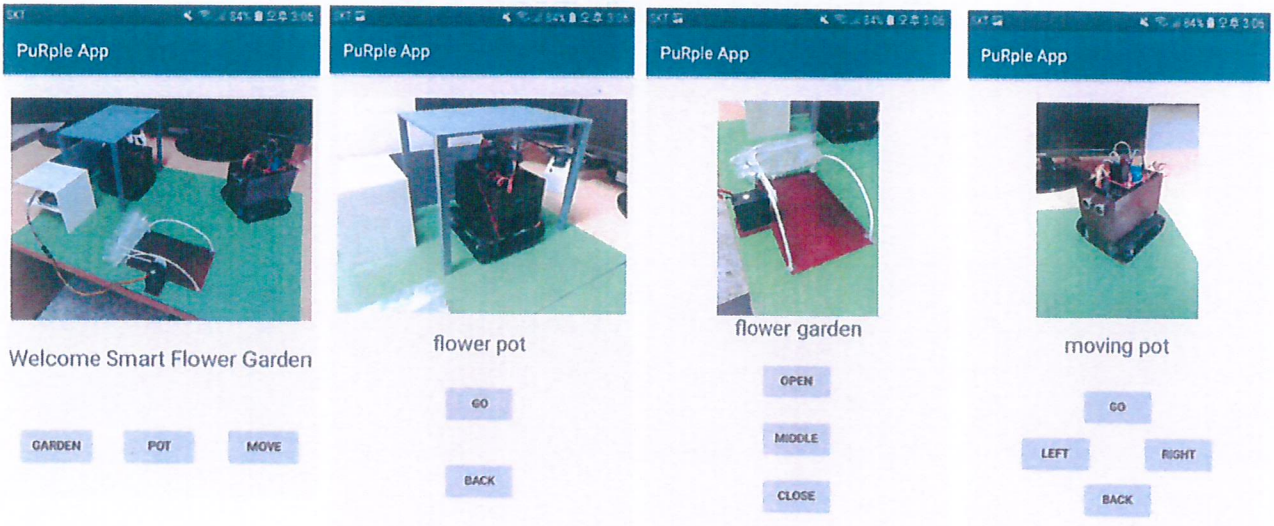
2019년도 2학기 DU-도전학기를 진행하면서 전자제어공학과 AI응용전공이 융합될 수 있는 프로젝트를 진행하며 새로운 시도를 하였고, 이를 통해 오픈소스 활용 및 서버 & 클라이언트 개념과 작동 원리 등 새로운 공부를 할 수 있었다. 다양한 기술들을 사용할 수 있었던 좋은 기회였지만 전문성이 부족하다 생각한다. 한번 경험해보았기에 다음번엔 좀 더 전문적으로 프로젝트를 진행할 수 있을 것이다.

5. 최종 결과물

(최종 HTML 웹 페이지)



(최종 외형 모습)



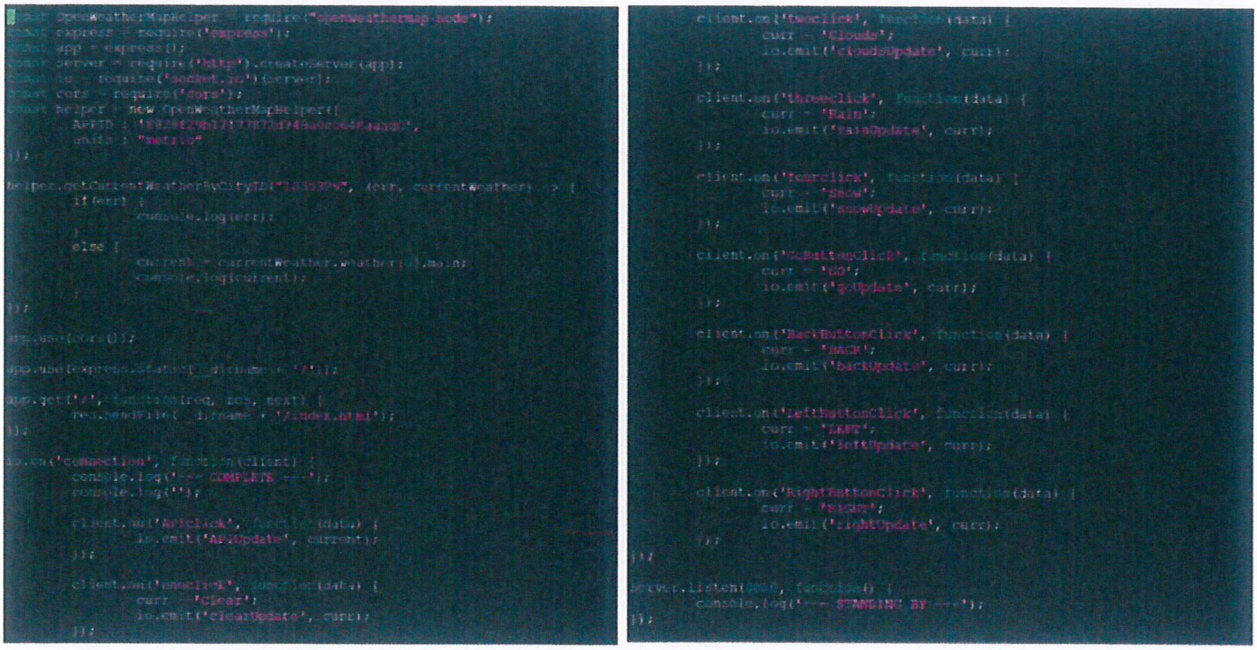
(애플리케이션 모습)

[붙임 1]

1. 서버 구축 & Open API 데이터 & HTML 웹 페이지

1.1 Node.js 서버 구축

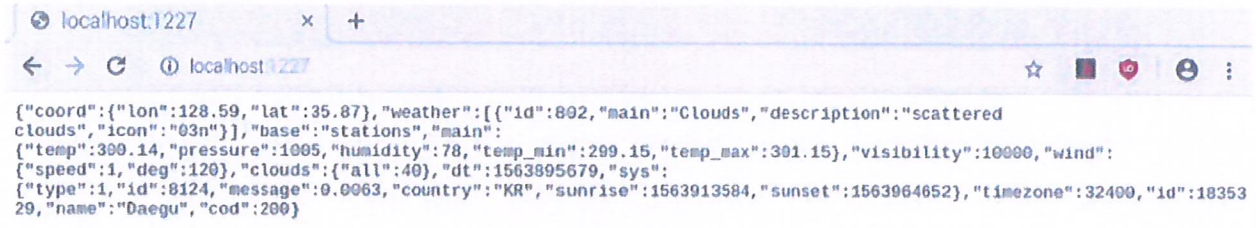
라즈베리파이 보드에 Node.js를 사용하여 서버를 구축하였다. 라즈비안 OS를 먼저 설치하고, 시스템을 최신으로 업데이트 & 업그레이드를 한다. 이후 명령어를 통해 라즈베리파이에 Node.js를 설치한다.



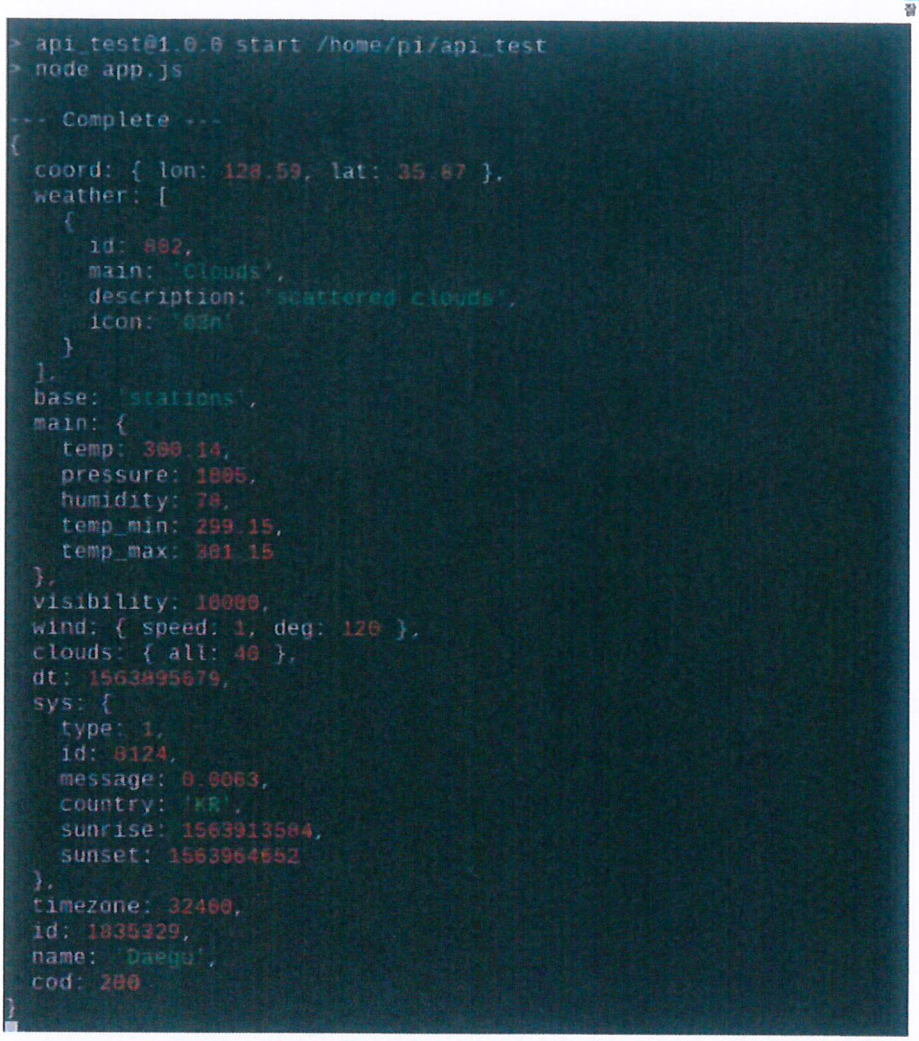
(서버 소스 코드)

1.2 Open API 사용

실시간 날씨정보를 얻기 위해 OpneWeatherMap 사이트에서 API key를 발급받아 API를 사용한다. OpenWeatherMap에서 제공하는 API는 Node.js 안에서 npm을 통해 사용할 수 있다. npm(Node Package Manager)란, 자바스크립트 언어를 위한 패키지 관리자이며 Node.js의 기본 패키지 관리자이다. 해당하는 npm을 설치하여 서버 소스코드에 추가하고, 실행시켜 JSON 형태로 출력하는 것을 확인할 수 있다.



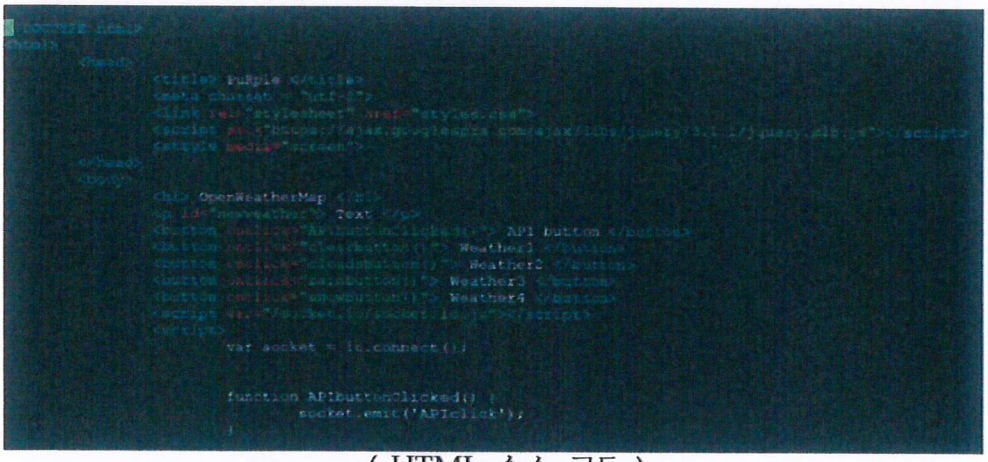
(웹 페이지에 나타나는 JSON 형태의 데이터)



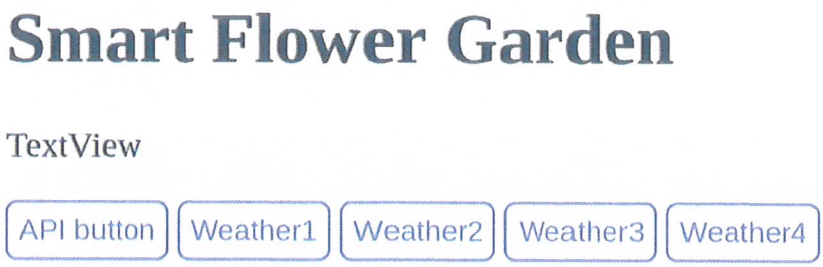
(터미널 콘솔에 나타나는 JSON 형태의 데이터)

1.3 HTML 웹 페이지 제작

서버가 작동되는 것을 확인하면 서버 접속 요청을 보냈을 때 응답하는 테스트 html 웹 페이지를 제작한다. 임의의 html 파일을 만들고 서버 소스 코드에서 html 파일이 위치한 경로를 작성하여 서버 IP주소로 접속 요청이 들어왔을 때 응답하여 html 웹 페이지를 띄울 수 있도록 작성한다.



(HTML 소스 코드)



(HTML 웹 페이지 모습)

1.4 Socket.IO 사용

Socket.IO는 실시간 웹 애플리케이션을 위한 자바스크립트 라이브러리이다. 이 라이브러리를 통해 클라이언트 라이브러리와 Node.js 서버 라이브러리로 구분되어 사용할 수 있다. 이 Socket.IO를 사용하여 작품내의 Node.js 서버와 여러 클라이언트간의 데이터 통신을 한다. 위의 서버 소스 코드에서는 이 Socket.IO가 사용되어있다.

2. 아두이노 + Wi-Fi 모듈

2.1 Wi-Fi 모듈 사용

아두이노에 ESP-8266 Wi-Fi 모듈을 사용하여 앞서 구축하였던 서버와 데이터 통신을 할 것이다. 펌웨어 업데이트를 하고 각 핀에 알맞게 연결하여 소스코드를 업로드 하여 AT 커맨드를 통해 정상적으로 작동되는지 확인한다. <ESP8266.h>를 통해 각 보드들이 정상적으로 작동되는 것을 확인하였다.

2.2 SocketIoClient.h 라이브러리 사용

아두이노 라이브러리 중 Socket.IO의 클라이언트 버전, <SocketIoClient.h>라는 라이브러리 파일이 있다. 이 라이브러리로 Node.js 서버에서 소켓 통신을 구현할 수 있다. 실시간 날씨 데이터를 소켓 통신으로 받아서 DC모터 & 서보모터 등 필요한 동작들을 실행할 수 있다.

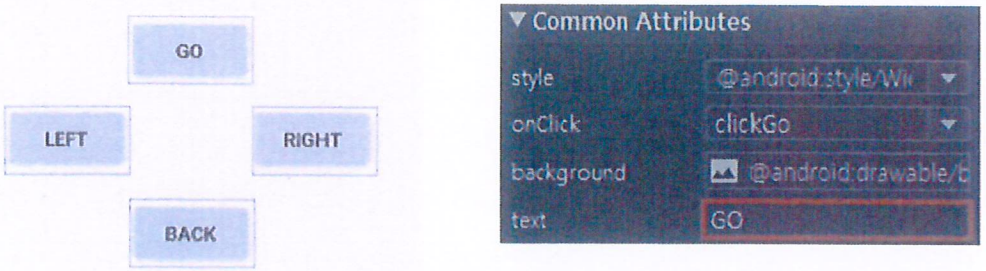
```
socket_io_test§
#include <Arduino.h>

#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>

#include <SocketIoClient.h>

#define USE_SERIAL Serial

ESP8266WiFiMulti WiFiMulti;
SocketIoClient webSocket;

void clearUpdate(const char * payload, size_t length) {
  USE_SERIAL.printf("got message: %s\n", payload);
  digitalWrite(2, HIGH);
}

void cloudsUpdate(const char * payload, size_t length) {
  USE_SERIAL.printf("got message: %s\n", payload);
  digitalWrite(2, LOW);
}

void setup() {
    pinMode(2, OUTPUT);
    USE_SERIAL.begin(115200);

    USE_SERIAL.setDebugOutput(true);

    USE_SERIAL.println();
    USE_SERIAL.println();
    USE_SERIAL.println();

      for(uint8_t t = 4; t > 0; t--) {
          USE_SERIAL.printf("[SETUP] BOOT WAIT %d...\n", t);
          USE_SERIAL.flush();
          delay(1000);
      }

    WiFiMulti.addAP("AIS5401[2.4G]", "aisrhd6b!");

    while(WiFiMulti.run() != WL_CONNECTED) {
        delay(100);
    }

    webSocket.on("clearUpdate", clearUpdate);
    webSocket.on("cloudsUpdate", cloudsUpdate);
    webSocket.begin("192.168.0.6", 8080);
}

void loop() {
    webSocket.loop();
}
```

(아두이노 제어 소스 코드)

3. 안드로이드 애플리케이션

3.1 레이아웃 구성

안드로이드 스튜디오를 사용하여 애플리케이션을 제작할 것이다. 간단하게 버튼들을 제작하고 그 버튼들 클릭을 하였을 때 onClick에 지정 변수를 등록하여 기능 구현 소스코드에서 함수로 작동시킬 것이다.



(안드로이드 레이아웃 버튼 & onClick 변수 선언)

3.2 socket.io-client 사용

socket.io-client를 통해 소켓 통신을 위해 안드로이드 스튜디오에서 일부 권한 부여 및 별도의 설정이 필요하다. 데이터 통신을 위해선 인터넷 연결이 필수이다. 이 인터넷 연결을 위해 Manifest에 들어가서 인터넷 허용 퍼미션을 추가하였다.

```
<uses-permission android:name="android.permission.INTERNET" />
```

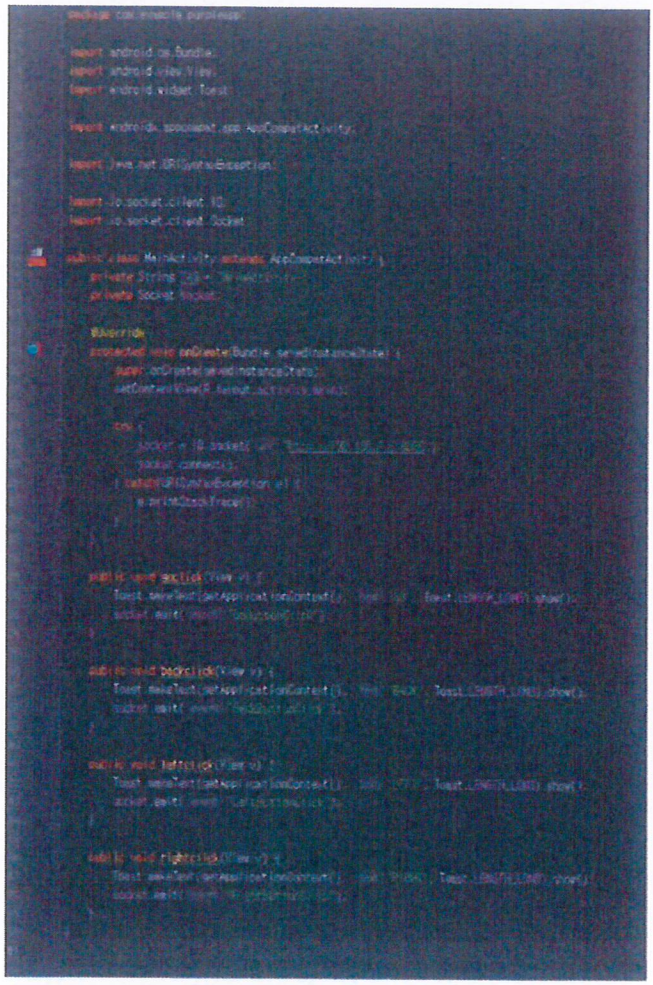
(Manifest에 인터넷 퍼미션 권한 부여)

그다음 build.gradle에 들어가서 Socket.IO 사용을 위해 의존성를 추가하였다.

```
implementation('io.socket:socket.io-client:1.0.0') {
    exclude group: 'org.json', module: 'json'
}
```

(build.gradle에 Socket.IO 의존성 추가)

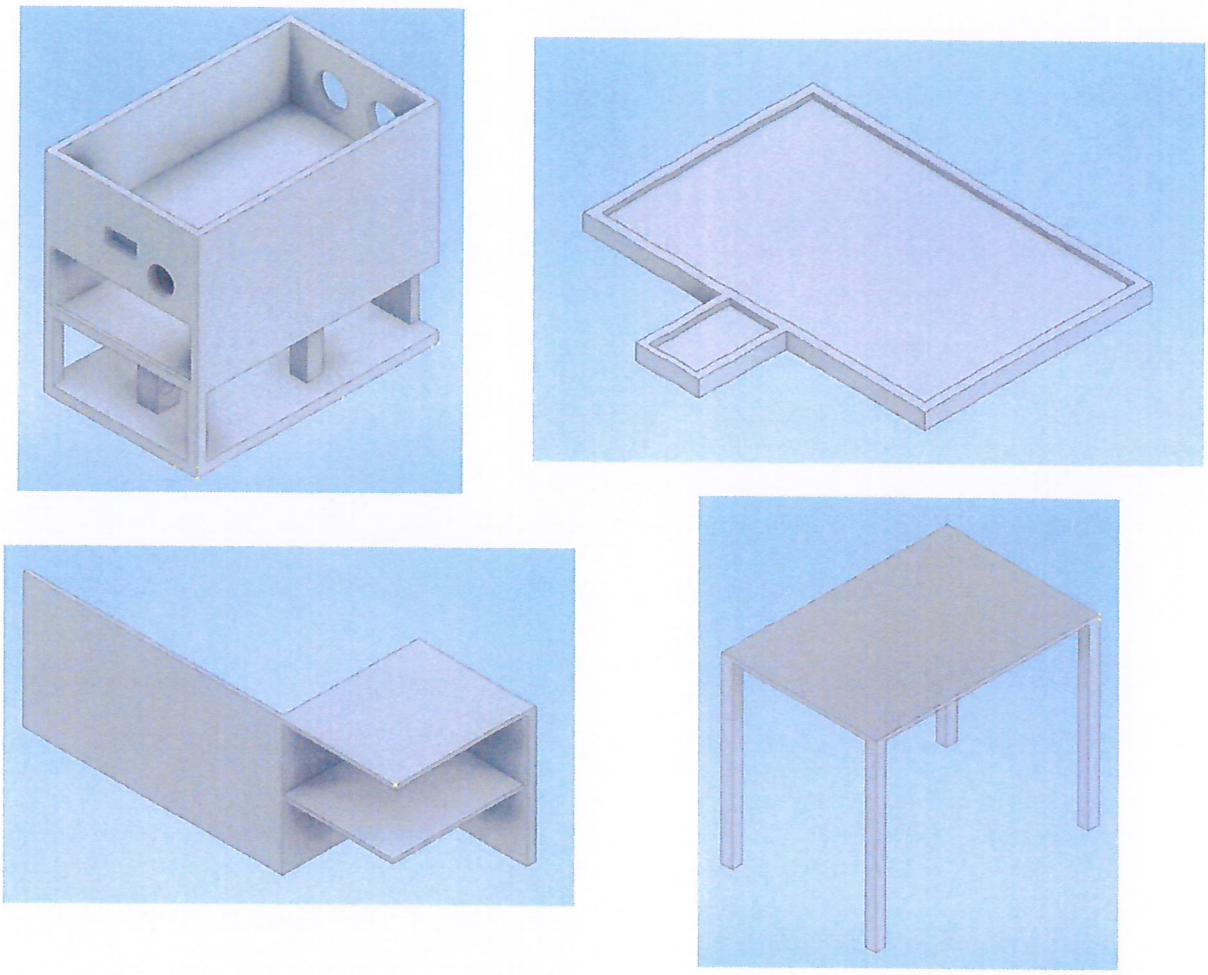
이제 모바일 애플리케이션에서 정상적으로 작동할 수 있도록 소스코드를 작성한다. Socket.IO 통신을 위해 서버의 IP주소를 입력하고 데이터를 전송하도록 소스코드를 작성한다.



(애플리케이션 제어 소스 코드)

4. 3D Printing 외형 프레임

프로젝트의 기술을 시뮬레이션으로 시연하기 위해 필요 품목들을 3D Printing으로 제작하기로 하였다. 화단 및 화분 모형, 가림막, 집모양의 형태와 벽 이 필요했었고, Inventor 프로그램을 사용하여 제작 후 출력하였다.



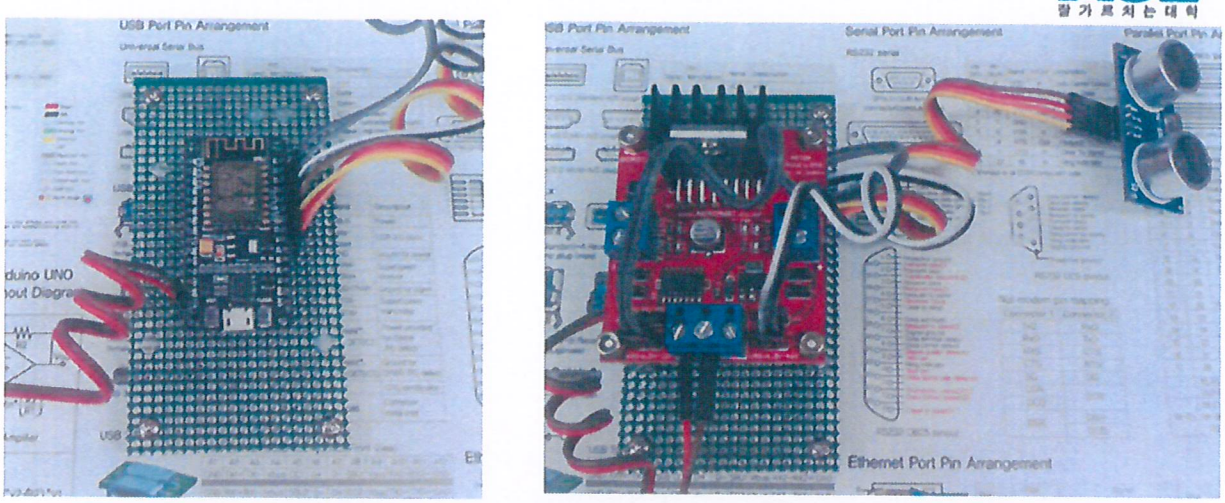
(3D Print 도면)

5. 출력물 & 센서 결합

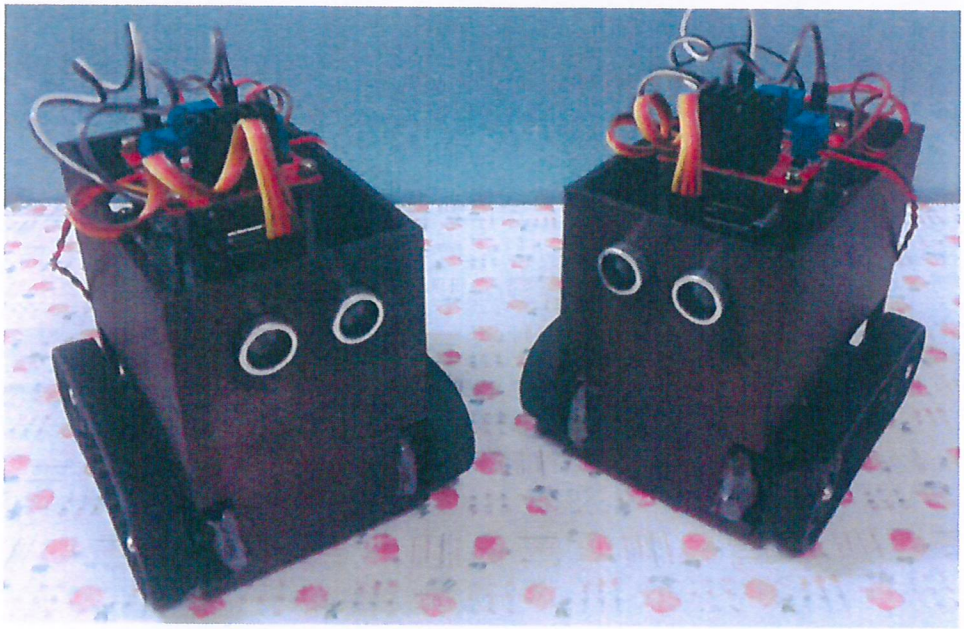
사용한 보드들과 센서들을 연결하였다. 이후 고정하기 위하여 PCB 기판에 소켓을 꽂아 납땜하였고 서포터를 통해 다른 부품들을 고정시켰다.

3D Print를 통해 제작한 화분 모형에 락카로 색을 입히고, 모터와 바퀴, 그리고 보드와 센서를 연결하였다.

화단 모형 또한 락카로 색을 입히고, 서보모터와 기타 부품을 통해 비닐하우스 모형을 나타내었다.



(보드 및 센서 연결 모습)

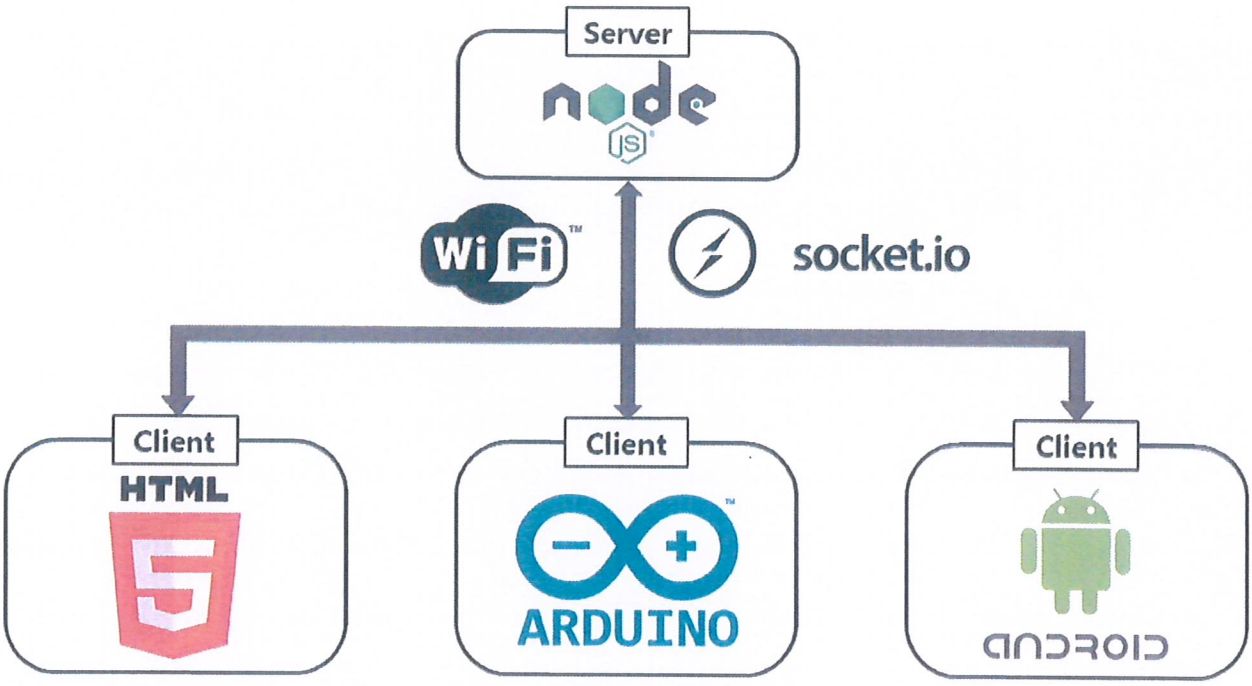


(완성된 화분 모형)

(완성된 화단 모형)

아크릴 판 위에 얇은 부직포를 부착하였다. 연두색의 부직포를 부착하여 풀색을 나타내었고, 이 위에 3D Print로 출력한 물품들을 비치하였다.

6. 시스템 종합



(프로젝트 시스템 구성도)

앞서 진행하였던 과정들을 종합하여 위와 같이 시스템을 구성하였다. Node.js로 구축한 서버가 각 클라이언트들 간의 Wi-Fi 통신을 사용하고, 소켓 통신으로 데이터 Request / Response (요청/응답) 구조를 만들어내었다.

참고 문헌

(1) 정재곤 저 / 2016년 / Do it! Node.js 프로그래밍

(2) 이소 히로시 저 / 2018년 / 모던 자바스크립트 입문

참고 사이트

(1) npm 라이브러리 - https://npmjs.com

(2) Socket.IO 사용방법 - https://socket.io

[붙임 2]

- 상장 : 스캔하여 별도 제출

- 논문 : 출력하여 별도 제출