Capstone Design 과제 결과보고서 요약

과 제 명	도서관 미발권 방지 시스템
팀 구성원 및 역할	정경석 - 팀장, 앱 제작 및 웹서버 통신 임성택 - 아두이노 및 라즈베리파이 통신 김하영 - 아두이노 및 라즈베리파이 통신 조은영 - ARTIK 코딩 & 통신 박성연 - ARTIK 코딩 & 통신
개발동기 목적 및 필요성	최근에 도서관에서 좌석 발권을 하지 않고 무단으로 자리에 앉는 사람들이 증가함에 따라 불편함을 느끼는 도서관 이용자들이 매우 많다. 이러한 문제를 해결해 쾌적한 도서관 환경을 만들고 도서관 질서를 확립하는데 도움을 주기 위해 미발권인들에게 경고를 보내는 시스템을 만들고자 한다.
과제 해결 방안 및 과정	IR sensor를 사용해 좌석에 사람이 있는지 확인하고 어플리케이션에서 얻은 발권 여부 정보와 종합해 발권하지 않은 사람의 좌석에 LED등을 켜 경고를 주고, 전광판에 그 좌석 번호를 크게 띄운다. 이를 통해 미발권한 사람에게 발권의 필요성을 알리고 경고하여 앞으로의 도서관 질서에 기여한다. 우선 실제로 도서관에서 좌석을 발권할 때와 유사한 환경을 만들기 위해 좌석발권을 위한 앱을 간단히 만든다. 이 앱은 좌석 발권 여부를 서버에 보내고 서버는 ARTIK board에 그 정보를 보낸다. ARTIK에 연결된 IR sensor에서 거리 측정을 통해 책상 아래에 사람의 다리가 있는지 확인하고 사람의 존재 여부를 판단한다. 사람이 있는데 좌석 발권을 하지 않았다면 LED를 켜 경고하고 socket 통신을 통해 전광판에 그 좌석 번호를 띄운다.
★대표적 결과물 (도면, 사진 등)	1. 블록도

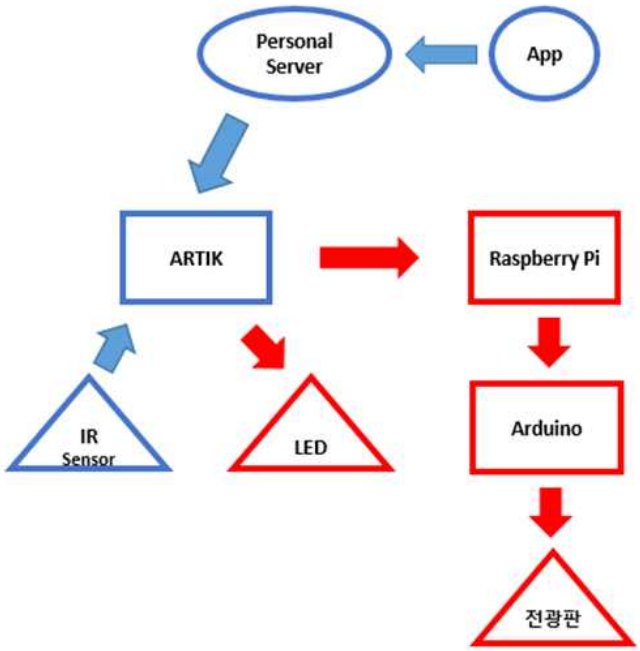
<table>
<tr>
<td></td>
<td>


그림 1. 미발권 시스템 블록도
2. 실제 연결 사진

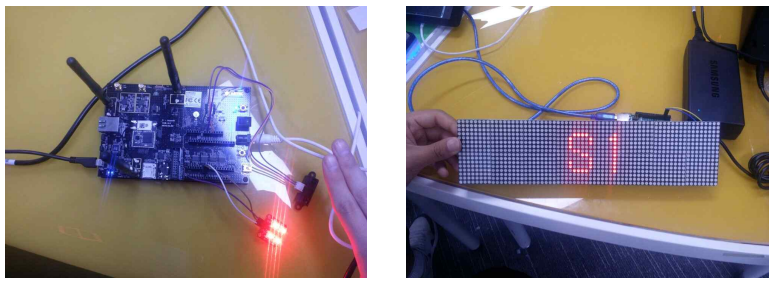



그림 2. 미발권 시스템 모듈
</td>
</tr>
<tr>
<td>기업연계형
프로젝트</td>
<td>
<table>
<tr>
<td>☐ 주제 제공</td>
<td>☐ 멘토링</td>
<td>☐ 샘플(부품) 제공</td>
<td>☐ 기타(현장견학 등)</td>
</tr>
<tr>
<td colspan="4"></td>
</tr>
</table>
</td>
</tr>
</table>

[서식11-3]

1. 과제 개요
(과제명, 참여자, 지도교수, 역할 담당, 참여기업 등)

- 과제명: 도서관 미발권 방지 시스템
- 지도교수 : 국태용 교수님
- 팀 구성원 및 역할

구분	이름	역할
팀 장	정경석	앱 제작 및 웹서버 통신
팀 원	임성택	전광판, 아두이노 및 라즈베리파이 통신
팀 원	김하영	전광판, 아두이노 및 라즈베리파이 통신
팀 원	조은영	ARTIK 코딩 & 통신
팀 원	박성연	ARTIK 코딩 & 통신

2. 개발동기 및 목적, 필요성

1) 개발동기
현재 도서관 발권 시스템은 다음과 같다. 발권 기계에서 원하는 좌석을 고르고 학생증을 가져다 대서 신원확인을 하면 발권이 완료된다. 이 과정에서 발권하지 않은 채 자리에 앉는 미발권자들을 걸러낼 방안이 없다. 이에 따라 많은 도서관 이용자들이 불편함을 겪고 있고, 특히 가장 좌석 이용이 활발한 시험기간에 미발권이 과열되는 양상을 보였다. 조원 모두가 실제로 불편함을 경험한 적이 있어 이를 방지하기 위한 시스템을 개발하고자 한다.

2) 목적
이번 프로젝트를 통해 실습 때 배운 내용들을 바탕으로 ARTIK, 라즈베리파이와 아두이노 등의 보드 사용법과 무선 통신을 숙지해 미발권 시스템을 구축하는데 활용하고자 한다. 클라우드(이 경우 개인 서버)와 ARTIK 보드 간 통신을 통해 데이터를 주고받고, 주고받은 데이터를 다시 다른 보드로 WiFi 소켓 통신을 통해 전송한다. 구체적으로 IR센서와 웹에서 받은 정보를 종합해 ARTIK에서 LED와 라즈베리파이 - 아두이노에 연결된 전광판에 표시할 데이터를 전달한다. 이를 통해 실시간으로 경고를 줌으로써 사용자가 창피함을 느끼게 해 미발권 재시도를 방지하고 쾌적한 도서관 환경을 만들고자 한다.

3) 필요성

주위 사람들뿐만 아니라, 시험기간만 되면 온라인에 도서관 발권 관련 불편함을 호소하는 글들이 많이 올라온다. 아래의 그림들은 우리 학교 학생들이 사용하는 커뮤니티에 올라온 미발권에 대한 불만 글이다.

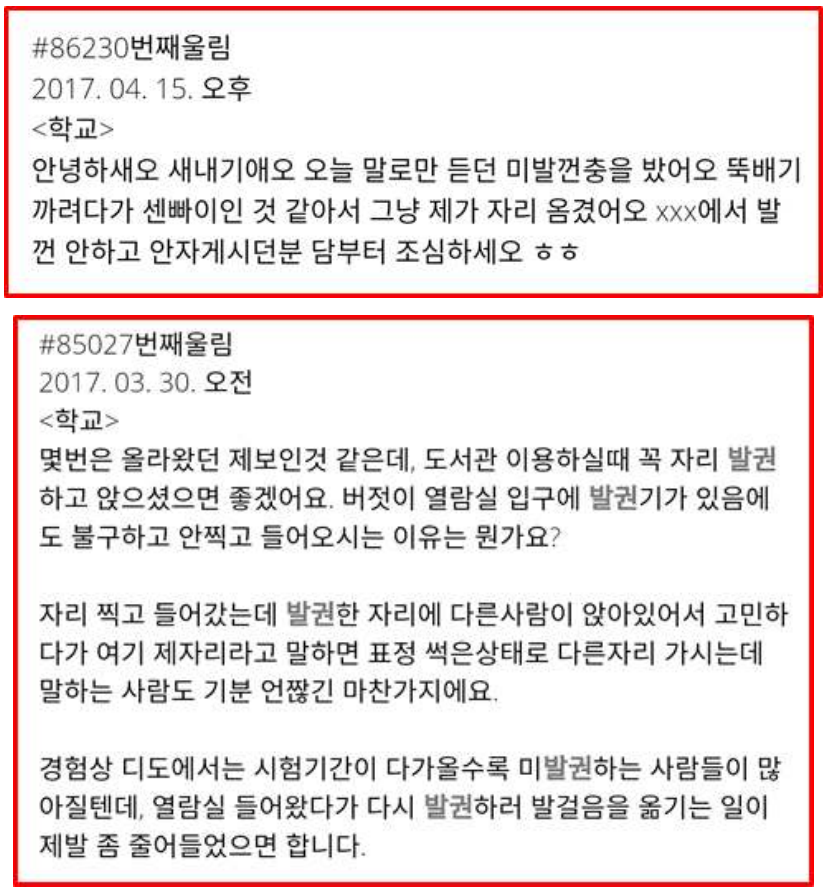


그림 3. 미발권자에 대한 불만 사항

이러한 미발권에 대한 불만들이 많을 뿐만 아니라 우리 조원들도 모두 도서관 이용 시 가장 불편한 점을 미발권으로 꼽았기 때문에 이 문제를 해결할 필요성을 느꼈고, 그 방법으로 미발권자를 시도하는 사람들에게 창피함을 주기로 결정했다.

3. 과제 해결 방안 및 과정

1) ARTIK Cloud

가장 처음에 만들기 시작한 앱 제작은 큰 어려움은 없었지만, 구현이 완료된 후에 ARTIK cloud로 정보를 보내는데 실패해서 개인 웹 서버를 이용하기로 했다.
이번 설계 프로젝트에서 사용하는 ARTIK 보드가 최근에 나온 것이라 사용방법에 관한 자료가 많이 없었다. 웹 서버를 사용하기 이전에 클라우드와의 연결을 위해 MQTT코드를 이용한 클라우드 통신 코드를 아두이노IDE에서 컴파일 한 후 아틱 보드에 업로드하려고 했는데 IDE쪽에서 업로드가 됐다는 메시지가 나옴에도 불구

하고 putty 모니터에서 연결이 되지 않았다는 알림이 떴다. 이것 역시 개인 서버를 이용하는 방향의 원인 중 하나가 됐다.

2) Raspberry pi3 보드

라즈베리파이를 실행시키기 위해선 SD카드 구입하고 라즈비안OS를 설치해야 한다는 사실을 늦게 알아서 처음에 고생을 하였다. SD카드 인식 문제로 어려움을 겪다가 인터넷 검색을 통해 문제점을 해결하였다. 이후 다른 문제없이 잘 진행되고 있었는데 하룻밤 사이에 갑자기 라즈베리파이에서 와이파이 검색을 못하고 뜨거워졌다. 그 원인을 생각해보니 라즈베리파이를 보관했던 장소에 햇빛이 강하게 들어와서 CPU가 뜨거워졌기 때문인 것 같았다. CPU과열 현상을 자체적으로 해결할 방법이 없었기 때문에 새로운 보드를 구입해 사용하였다.

3) Arduino D1 (호환 보드 ESP8266)

처음 구상했던 프로젝트에서 아두이노를 와이파이에 연결해야 했기 때문에 아두이노와 와이파이 쉴드가 한 보드에 들어있는 아두이노 D1 보드를 구매해 사용할 계획이었다. 하지만 그 이전에 구매한 전광판이 아두이노 UNO 전용이라 아두이노 D1 보드와 통신이 되지 않았다. 이에 대한 해결책으로 아두이노 D1 보드와 UNO 보드를 시리얼 통신으로 연결하려고 했는데 연결되지 않았다. 그래서 최종적으로 아두이노 UNO와 라즈베리 파이 보드를 시리얼 연결했다.

4) ARTIK과 Raspberry pi 간 소켓 통신

아틱과 라즈베리파이를 소켓 통신으로 연결하는 과정에서 코드를 작성하는 것이 다소 복잡했다. 라즈베리 파이에서 아두이노로 연결하는 코드와 소켓통신 코드를 적절하게 합쳐 전광판을 제어 가능하게 하는 코드를 작성해야 했기 때문이다. 하지만 실행될 함수의 순서를 생각하여 합쳐야했기 때문에 이 과정에서 약간의 어려움이 있었다. 적절한 헤더파일을 추가하고 변수를 통일 및 구분하여 소켓통신을 가능하게 할 수 있었다.

소켓 통신을 하던 와중에 LED가 말썽을 부렸다. 코드에 따라 LED 점멸등이 잘 일어나다가도 갑자기 ON OFF에 맞춰지지 않았다. 처음에는 그 원인을 접촉 불량으로 생각해서 여분의 새로운 LED와 선을 사용하거나 다른 핀에다가 꽂아보는 등 다양한 시도를 했는데 해결이 안됐다. 알고 보니 코드에서 #define HIGH 1 또는 #define LOW 0를 적지 않아서 HIGH와 LOW에 대한 정의가 이루어지지 않아서 LED가 잘 켜지지 않았던 것이다.

5) Raspberry pi와 아두이노 간의 serial 통신

라즈베리파이 보드와 아두이노 UNO보드를 연결하는데에는 여러 방법이 있는데 이 중 간편하게 USB를 이용한 serial 통신을 이용하였다. ARTIK에서 보내는 데이터

를 와이파이를 통해 Raspberry pi에서 받은 뒤 이를 아두이노로 보내기만 하면 되었기 때문이었다. 여기서는 ARTIK에서 받은 데이터를 아두이노로 바로 보내기 위해 변수 type를 설정하는 데에서 어려움을 겪었다.

6) ARTIK과 웹서버 간 와이파이 통신

보드와 웹서버 통신을 위해 시도한 다양한 방법 중 MQTT 프로토콜을 이용한 시도를 가장 먼저 했다. 이 때 putty에서 직접 코딩을 하려면 라이브러리를 따로 추가해야 돼서 Arduino IDE에 코드를 작성해 ARTIK 보드로 업로드 하고자 했다. 우선 아틱 클라우드에서 새로운 디바이스를 추가하고, IDE의 클라이언트 코드에 이 웹서버 주소와 디바이스 이름, 비밀번호, 토큰 등의 정보를 작성해 넣어서 데이터를 주고받기 위한 준비를 했다. Arduino IDE와 아틱 보드를 같은 와이파이에 연결하고 업로드를 눌렀는데 IDE에서 컴파일과 업로드가 모두 됐음에도 불구하고 ARTIK과 연결된 putty쪽에서는 연결 확인 메시지가 뜨지 않았다. 코드도 고쳐보고 라이브러리도 바꿔가면서 연결해봤지만 안돼서 다음 방법인 노드 레드로 넘어갔다.

노드 레드를 설치하기 위해 강의 자료에 나와 있는 과정을 그대로 따라 putty에 dnf update를 하고 npm –g install node-red명령어로 노드 레드를 설치하려고 했는데 중간에 warning이 여러 개 떴다. 우선 넘어간 후 인터넷 창을 열어 주소 입력창에 http://192.168.100.10*:1880을 입력해서 노드 레드에 들어갔는데, ARTIK input이 보이지 않아서 노드레드와 보드가 연결이 안됐음을 알 수 있었다. 그 원인이 설치 과정의 오류에 있어서 해결방법이 없어 다음 방법으로 넘어갔다.

마지막으로 MySQL 컴파일을 이용한 통신을 시도했다. 이것 역시 putty에서 직접 코딩은 라이브러리 경로를 찾지 못해 MySQL 관련 커맨드들을 인식하지 못하는 문제가 있었다. 다른 방법으로 IDE에서 코딩을 하고 업로드를 해보았으나, 여전히 mysql.h를 이용할 수 없었다.

MySQL 커맨드를 직접 사용해서 C언어로 컴파일 하는 것이 불가했기에 php를 이용한 방법을 택하기로 하였다. php를 사용하기 위해서는 다시 php 컴파일러가 필요했기에 이를 설치하려 시도했으나 아틱에 남는 공간이 부족하여 설치가 진행되지 않는 현상이 발생하였다. 아틱 내부의 불필요한 파일들을 지워가며 재설치 하려 했으나 같은 오류가 반복되어 끝내 서버와 아틱 간의 연결은 완성하지 못하였다.

4. 출품과제의 기술

1) 어플리케이션 및 서버

(1) 어플리케이션 및 서버 화면

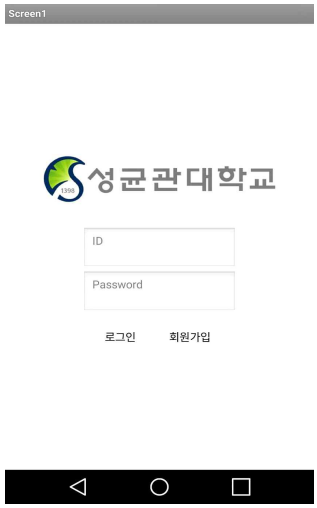


그림 4-1. 로그인

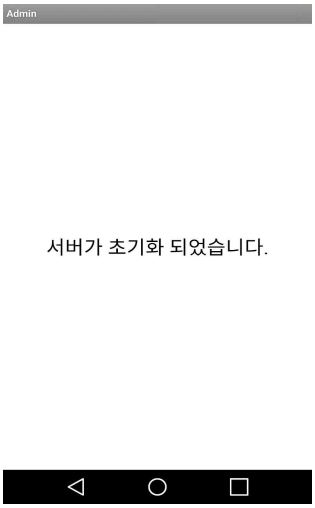


그림 4-2. 관리자 모드

그림 4-3. 회원가입

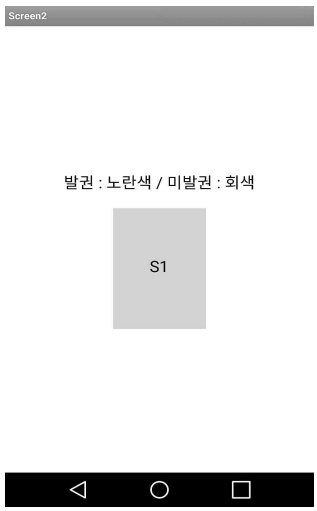


그림 4-4. 좌석 발권 (전)

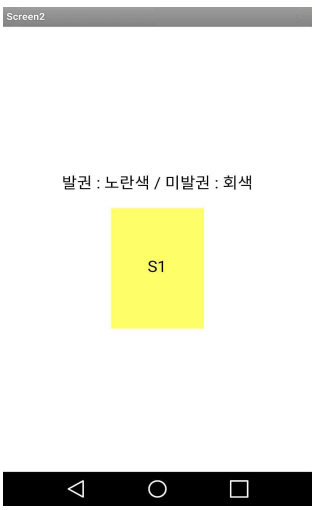


그림 4-5. 좌석 발권 (후)

–

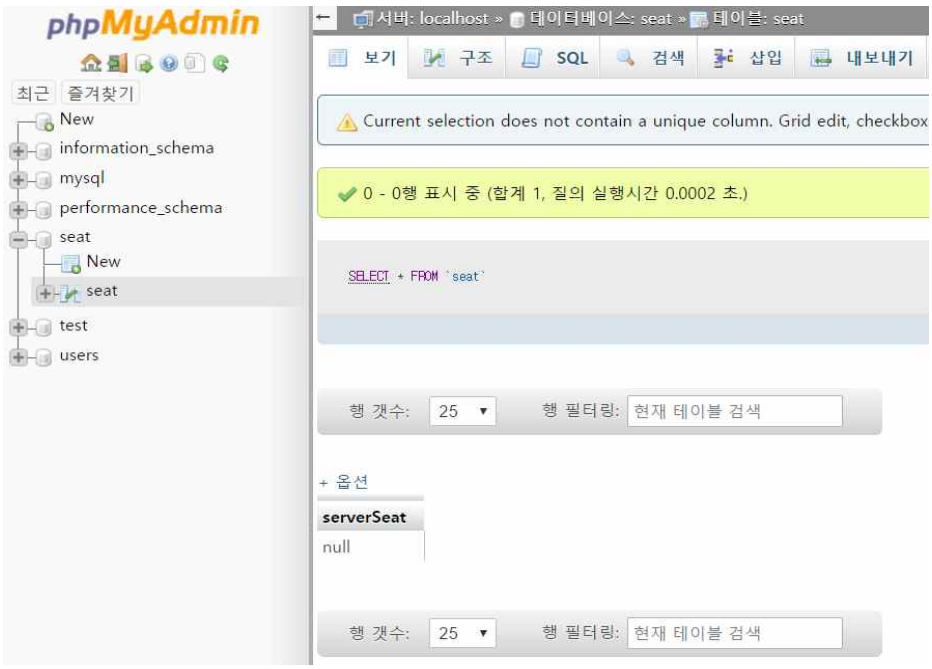


그림 4-6. 좌석 발권이 이뤄지지 않은 경우 서버 화면

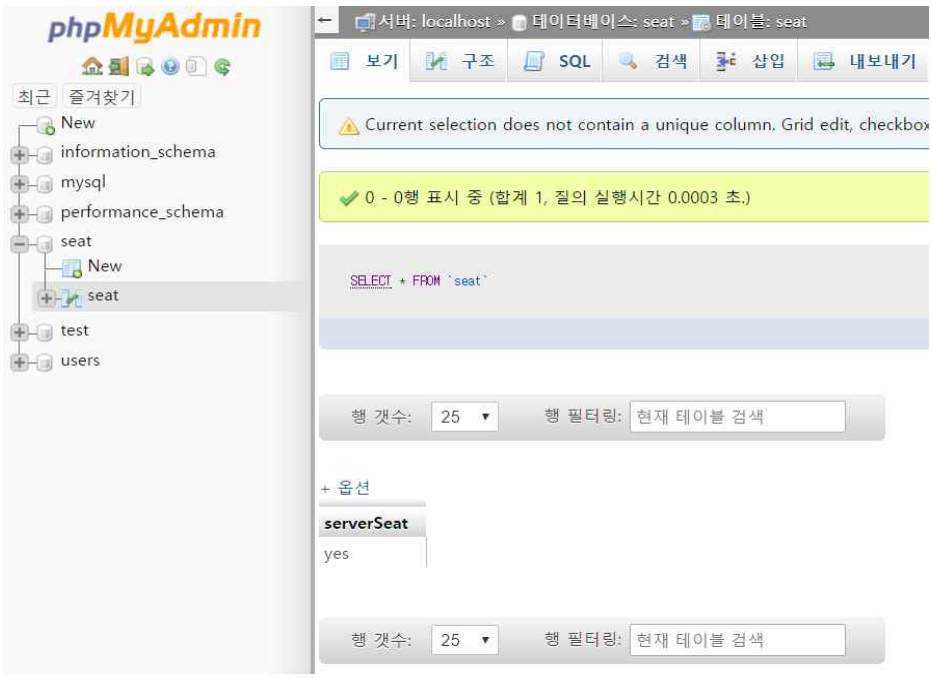


그림 4-7. 좌석 발권이 이뤄진 경우 서버 화면

(2) 어플리케이션 코드

① Screen1 - 로그인

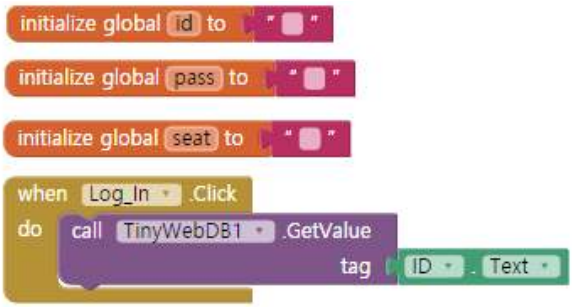


그림 5-1. 로그인 코드 1

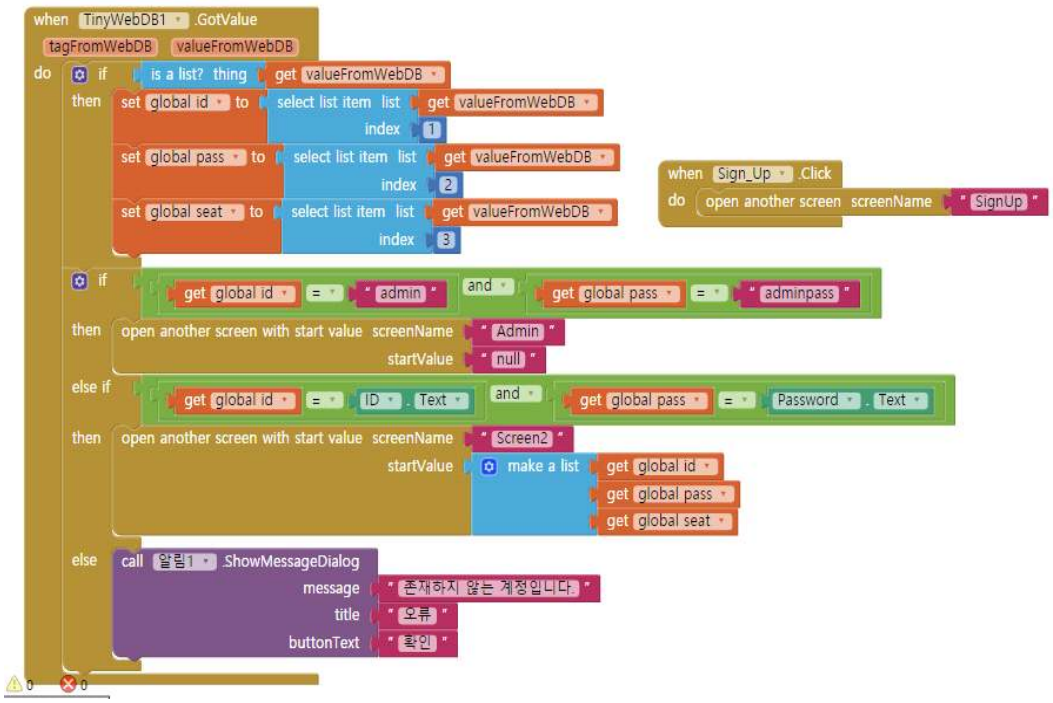


그림 5-2. 로그인 코드 2

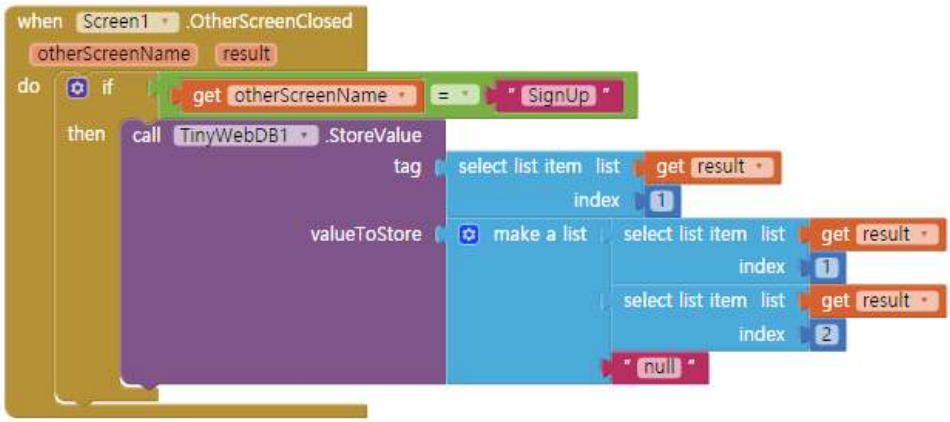


그림 5-3. 로그인 코드 3

· 그림 5-1 : 변수 선언, Log_In 버튼 클릭 시 입력한 ID로 태그된 데이터를 웹 DB로부터 불러옴

· 그림 5-2 : 웹 DB로부터 데이터를 받아왔을 경우 선언한 변수에 데이터를 저장하고 입력한 아이디 및 비밀번호와 일치하는 가를 확인한다. 관리자의 경우 관리자 화면으로, 옳은 회원이라면 좌석 발권 화면으로 넘어간다. Sign_Up 버튼을 클릭했을 경우 회원가입 화면으로 넘어간다.

· 그림 5-3 : 회원가입 화면에서 넘어왔을 경우 해당 정보를 웹 DB에 저장한다.

② Admin - 관리자 모드

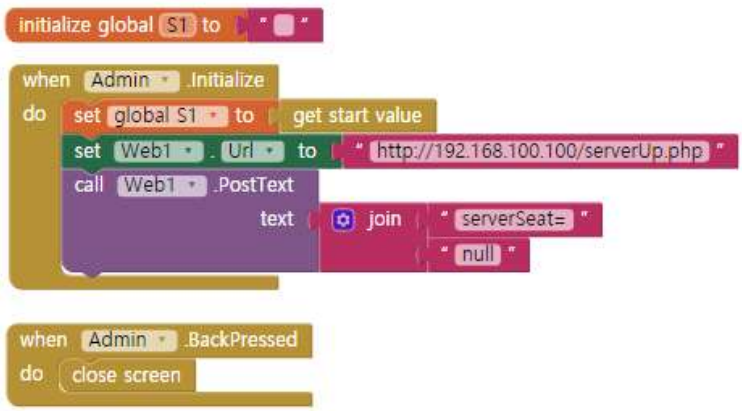


그림 5-4. 관리자 모드 코드 1

[serverUp.php]

```
<?php
        $con = mysqli_connect("localhost","root","autoset","seat");

        $serverSeat = $_POST["serverSeat"];

$statement = mysqli_prepare($con, "UPDATE seat.seat SET serverSeat=?");

mysqli_stmt_bind_param($statement,"s",$serverSeat);
mysqli_stmt_execute($statement);

$response = array();
$response["success"] =true;

echo json_encode($response);
?>
```

· 그림 5-4 : 관리자 모드로 접속했을 시 좌석을 미발권 상태로 초기화한다. 구축한

웹 서버의 serverSeat가 null로 설정되도록 업데이트한다. serverUp.php은 이를 위한 php 코드이다.

③ Sign Up - 회원가입

그림 5-5. 회원가입 코드

· 그림 5-5 : 가입 버튼을 클릭하면 가입이 완료되었다는 알림을 띄운다. 이후 회원가입화면을 닫으면서 입력한 아이디와 비밀번호로 리스트를 만들어 result에 저장하여 보낸다.

④ Screen2 - 좌석 발권

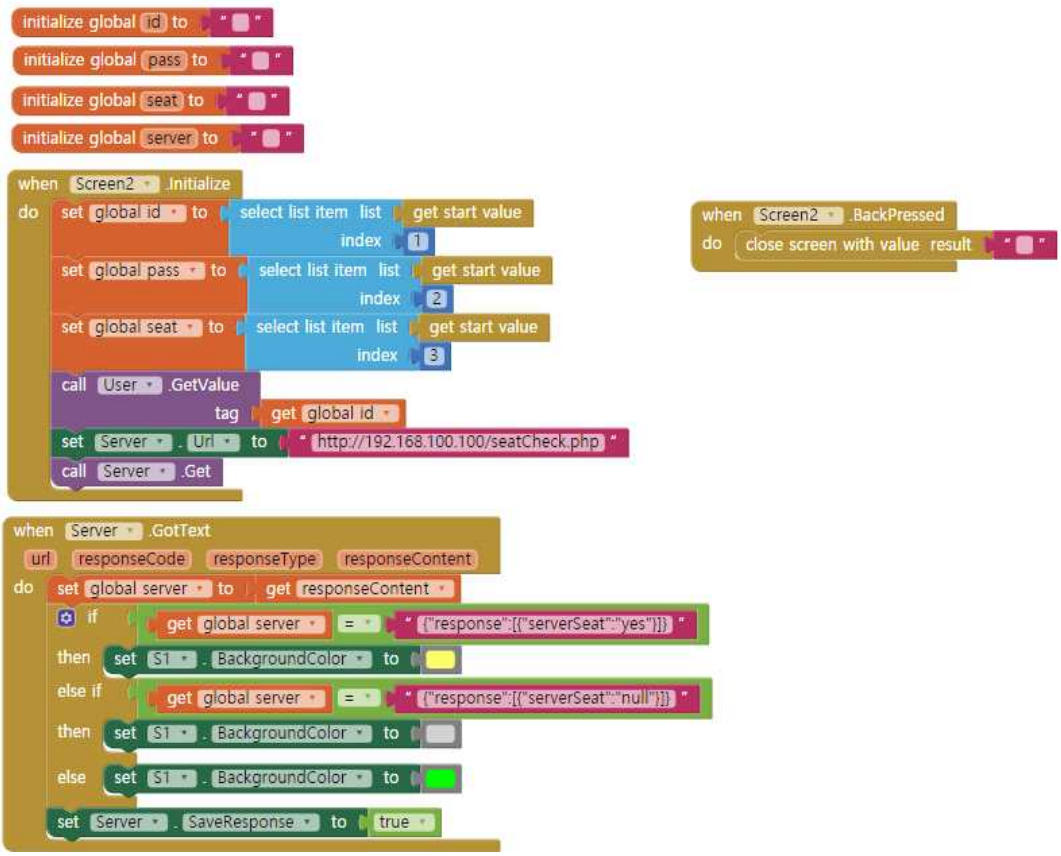


그림 5-6. 좌석 발권 코드 1

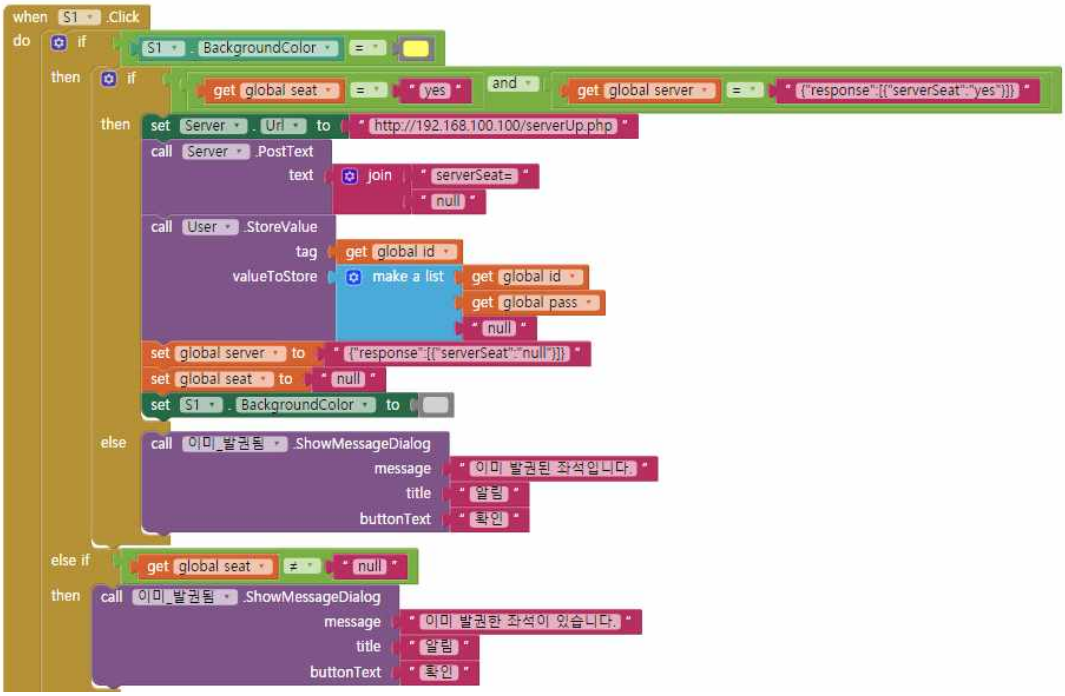


그림 5-7. 좌석 발권 코드 2

그림 5-8. 좌석 발권 코드 3

[seatCheck.php]

```
<?php
        $con = mysqli_connect("localhost", "root","autoset","seat");
        $result = mysqli_query($con,"SELECT * FROM seat;");
        $response = array();
```

```
        while($row = mysqli_fetch_array($result)){
                array_push($response, array("serverSeat" => $row[0]));
        }

echo json_encode(array("response"=>$response));
mysqli_close($con);
?>
```

· 그림 5-6 : 변수 선언 및 초기화를 한다. Screen2가 처음 실행될 시 변수에 회원정보를 저장하고 웹 DB와 서버를 호출한다. 서버가 호출되면 좌석 발권 여부를 판단하여 좌석의 색깔을 지정한다. seatCheck.php는 좌석 발권 여부 판단을 위해 서버와 앱을 연결하는 php이다. Screen2에서 뒤로 가기를 누를 시 화면을 닫고 빈 값을 반환한다.

· 그림 5-7 : 좌석을 클릭했을 때 경우에 따라 다음 단계를 진행한다. 이미 발권된 좌석인 경우 (노란색) 본인이 해당 좌석을 발권했다면 좌석의 발권을 해제하게 된다. 이 때 해당 좌석의 발권이 해제됐다는 정보를 서버로 보내 업데이트한다. 그렇지 않다면 이미 누군가가 발권한 좌석이라는 알림을 띄운다. 클릭한 좌석이 발권되지 않았지만, 본인이 다른 좌석을 발권한 상황이라면 이미 발권한 좌석이 있다는 알림을 띄운다.

· 그림 5-8 : 앞의 두 경우가 아니라면 클릭한 좌석은 미발권 상태이고 본인도 발권한 좌석이 없기 때문에, 해당 좌석의 발권이 이뤄진다. 이 때 해당 좌석의 발권이 이뤄졌다는 정보를 서버로 보내 업데이트한다.

2) 서버와 아틱 간 통신

앞서 말한 바와 같이, 아틱과 서버 간의 통신은 명령어 인식 불가와 아틱 내부 용량 부족으로 인한 컴파일러 설치 실패로 구성하지 못하였다.

3) Raspberry pi와 Arduino 간의 serial 통신

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2,3);
int cur=0;
char charBuf[50];

char inData[3];
char inChar=-1;

void setup(){
```

```
  mySerial.begin(9600);
  Serial.begin(9600);
  delay(3000);

  init_dotmatrix();
  display_fix("Warn");
  delay(5000);
}

void loop(){
  if(Serial.available()>0){
    inChar=Serial.read();
    inData[0]='S';
    inData[1]=inChar;
    inData[2]='\0';
    if(inData[1] == 'q'){
      display_shift("End");
    }
    else if(inData[1] != '0'){
      display_shift(inData);
    }
    else{
      display_shift("Good");
    }
  }
}

void init_dotmatrix(){
  mySerial.println("'<2: '>");
  delay(1000);
}

void display_fix(String arg_msg){
  String str_result = "'<1:"+arg_msg+"'>";
  mySerial.println(str_result);
}

void display_shift(String arg_msg){
  String str_result = "'<0:"+arg_msg+"'>";
  mySerial.println(str_result);
  delay(100);
```

```
}
```

4) ARTIK과 Raspberry pi간 socket 통신

(1) Raspberry pi (server)

```
#ifdef RaspberryPi
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <assert.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>
char device[]= "/dev/ttyACM0";
#define MYPORT 3490
#define BACKLOG 10
#define MAXDATASIZE 100

int fd;
unsigned long baud = 9600;
unsigned long time = 0;



void setup(){
 printf("%s \n", "Raspberry Startup!");
 fflush(stdout);

 if((fd=serialOpen(device, baud))<0){
  fprintf(stderr, "Unable to open serial device: %s\n", strerror(errno));
  exit(1);
 }

 if(wiringPiSetup() == -2){
 fprintf(stdout, "Unable to start wiringPi: %s\n", strerror(errno));
 exit(1);
 }
```

```
}

int main() {
  int sockfd, new_fd;
  struct sockaddr_in my_addr;
  struct sockaddr_in their_addr;
  int sin_size;
  int on = 1;

  char *recv_data;
  char recv_seat;

  setup();


  if ((sockfd = socket(AF_INET, SOCK_STREAM,0)) == -1) {
    perror("socket");
    exit(1);
  }
  my_addr.sin_family = AF_INET;
  my_addr.sin_port = htons(MYPORT);

  my_addr.sin_addr.s_addr = INADDR_ANY;
  bzero(&(my_addr.sin_zero), 8);

  setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on));

  if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr))  == -1$
    perror("bind");
    exit(1);
  }
  if (listen(sockfd, BACKLOG) == -1){
   perror("listen");
    exit(1);
  }
  printf("========= [PORT] : %d =========\n", MYPORT);
  printf("Server : waiting client\n");
  while(1){
    sin_size = sizeof(struct sockaddr_in);

    if((new_fd = accept(sockfd, (struct sockaddr *)&their_addr, &sin_size)) == $
```

```
        perror("accept");

        continue;
    }
    if (!fork()){
        while(1){
          recv_data = (char *)malloc(MAXDATASIZE*sizeof(char));
          assert(recv_data);
          if (recv(new_fd, recv_data, MAXDATASIZE, 0) == -1){
            perror("recv");
            exit(1);
          }
         printf("receive = %s", recv_data);

          recv_seat = recv_data[0];

          serialPutchar(fd, recv_seat);
          serialFlush(fd);
          fflush(stdout);


          int length = strlen(recv_data);
          if ((length == 2) && (recv_data[0] == 'q')){
            printf("====== Server is closing  ======\n");
            printf("====== Waiting new client ======\n");
            free(recv_data);
            break;
          }
          free(recv_data);
        }
        close(new_fd);
        while(waitpid(-1, NULL, WNOHANG) > 0);
    }
  }
  return 0;
}

#endif
```

(2) ARTIK (client)

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <assert.h>
#include "wiringARTIK.h"
/* 전처리기 선언 */

#define PORT 3490
#define MAXDATASIZE 100
#define OUTPUT_PIN1 124
#define ANALOG_PIN1 0
#define HIGH 1
#define LOW 0
/* 상수 값 설정 */

int main(int argc, char *argv[])
{
  int sensorVal;
  int sockfd, numbytes;
  char *send_data;                // 송신 data
  char reserve;
  struct hostent *he;             // host의 주소 저장
  struct sockaddr_in their_addr; // 서버의 소켓
  FILE *decision;
  FILE *text;

  /* host 이름 입력 */
  if (argc != 2){
    fprintf(stderr, "usage: ./client <host-name>\n");
    exit(1);
  }

  /* 입력된 host의 주소 저장 */
  if ((he = gethostbyname(argv[1])) == NULL){
    herror("gethostbyname");
```

```
    exit(1);
  }

  /* IPv4 소켓 생성 */
  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1){
    perror("socket");
    exit(1);
  }

  /* 서버 소켓 설정 */
  their_addr.sin_family = AF_INET;
  their_addr.sin_port = htons(PORT);
  their_addr.sin_addr = *((struct in_addr *)he->h_addr);
  bzero(&(their_addr.sin_zero), 8);

  /* 서버와 연결 */
  if (connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr)) ==
-1){
    perror("connect");
    exit(1);
  }

  /* 해당 서버에 연결 중이라는 문구 출력 */
  printf("============ [PORT] : %d ============\n", PORT);
  printf("======Connecting at %s======\n", argv[1]);

  while(1)
  {
    // 읽기 버퍼를 위한 메모리 할당
    send_data = (char *)malloc(MAXDATASIZE*sizeof(char));
    assert(send_data);

    printf("send[q : exit] : ");

        if((decision = fopen("decision.txt","w")) == NULL){
          printf("file open error\n");
          return 2;
        }
        setup();
        sensorVal = analogRead(ANALOG_PIN1);
```

```
        if (sensorVal >= 800){
          reserve = '1';
          digitalWrite(OUTPUT_PIN1,HIGH);
        }
    else{
          reserve = '0';
          digitalWrite(OUTPUT_PIN1,LOW);
        }
        printf("\nsensorIR = %d\n", sensorVal);
        fprintf(decision, "%c", reserve);
        fclose(decision);

        text = fopen("decision.txt", "r");

        fgets(send_data, MAXDATASIZE, text);

    int length = strlen(send_data);

    // 메세지 전송 => 에러가 있으면 중단
    if (send(sockfd, send_data, length+1, 0) == -1){
      perror("send");
      free(send_data);
      break;
    }

        // 인풋이 q이면 중단
    if ((length == 2) && (send_data[0] == 'q')){
      printf("========= Exiting client !! =========\n");
      free(send_data);
      break;
    }

        free(send_data);
        fclose(text);
        sleep(5);

  }
  close(sockfd);

  return 0;
}
```

```
int setup()
{
  digitalPinMode(OUTPUT_PIN1, OUTPUT);

  digitalWrite(OUTPUT_PIN1,LOW);

  return 0;
}
```

5) ARTIK - IR sensor,LED / Arduino - 전광판 동작

수업시간에 했던 실습을 바탕으로 IR sensor와 LED를 각각 아틱에 연결했다. 우리 팀은 IR sensor로부터 analog 신호를 받고 앱에서 발권 여부에 대한 정보를 받아 그 두 정보에 대한 조건을 만들어 LED를 켜고 끄는 것을 시도했다.

(1) Arduino - IR sensor, LED

IR sensor는 ARTIK Analog sensor에 연결해 센서에 물체를 점차 가까이 하며 원하는 거리에서의 analog 입력 값을 측정했다. 우리는 IR 센서를 책상 밑에 부착하여 책상으로부터 사람 다리까지의 거리를 측정하고자 했는데 여러 차례 실험한 결과 그 입력 값이 1000 인 지점에서 우리가 원하는 거리를 찾을 수 있었다.

(socket only version) 따라서 LED를 켤 때 IR sensor에서 1000보다 작은 값이 입력 값으로 들어왔을 때 LED가 점등되도록 했다.

(socket + php version) 따라서 LED를 켤 때 IR sensor에서 1000보다 작은 값이 입력 값으로 들어왔을 때 앱에서 발권 여부에 대한 정보를 받아 발권이 되어 있지 않으면 LED를 점등하도록 한다. (구현 실패)

(2) Arduino 전광판

전광판 동작은 아두이노와 연결해 동작시켰다. 아틱에서 오는 데이터가 1인 경우 S1을 전광판에 띄워 좌석 S1에 미발권자가 앉아있다는 정보를 알린다. 또한 0을 받은 경우 Good을 띄워 미발권자가 있는 좌석이 없다는 정보를 알린다.

5. 개념설계 및 상세설계(계산식, 설계도면 등)

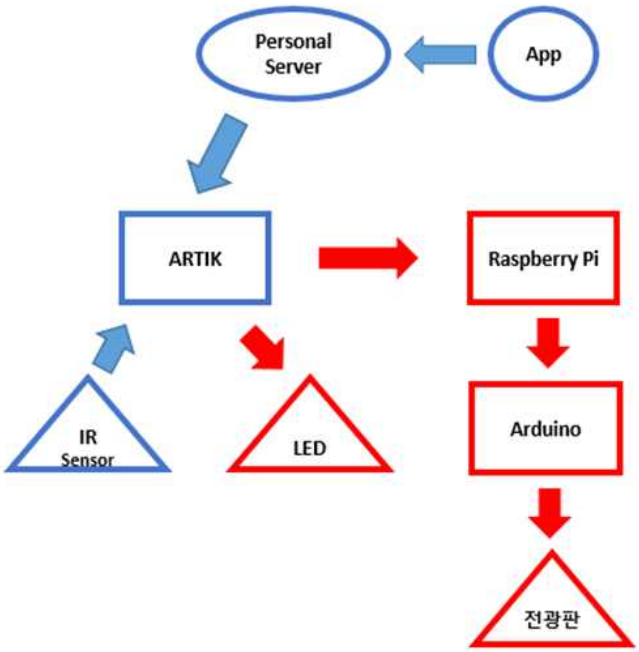


그림 6. 미발권 시스템 블록도

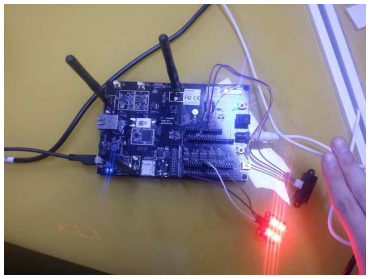
6. 기타(소요비용 등)

● 진행절차

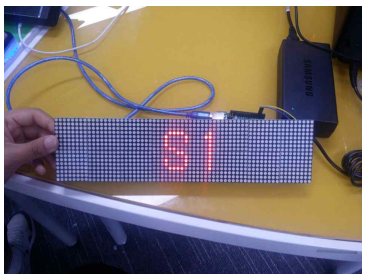
No.	수행내용	추진일정															
		1주	2주	3주	4주	5주	6주	7주	8주	9주	10주	11주	12주	13주	14주	15주	16주
1	아틱 보드 실습																
2	제안서 작성																
3	부품 구입																
4	초기 모델 제작 및 테스트																
5	중간 보고																
6	통신 프로그래밍																
7	초기 모델 완성																
8	문제점 개선																
9	최종 보고서 작성 및 발표																

● 진행과정

앱 & IR 프로그래밍



LED



소켓 통신 및 전광판

7. 참고문헌

· App Inventor – appinventor.mit.edu

· MySQL 설치 – www.atblog.co.kr/?p=6305

· 소켓 통신 관련 함수

- socket, htons, connect 등 : http://mintnlatte.tistory.com/40
- bzero : https://www.joinc.co.kr/w/man/3/bzero
- setsockopt :

  https://www.joinc.co.kr/w/Site/Network_Programing/AdvancedComm/SocketOption
- waitpid : http://forum.falinux.com/zbxe/?mid=C_LIB&page=3&document_srl=408548
- gethostbyname, hostent:

  http://forum.falinux.com/zbxe/index.php?document_srl=518686&mid=C_LIB
- argc, argv :

  http://mwultong.blogspot.com/2006/12/c-argc-argv-main-function-parameter.html