아두이노 소켓통신 관련 스터디 문서

2015. 3. 29 ~2015. 4. 20  
작성자 : 엄승욱, 홍도영

2015. 4. 20 ~   
작성자 : 엄승욱

개발 1주차 : 소켓 통신

1. 소켓 통신이란

- 서로 떨어진 호스트간의 프로세스 통신을 하려면 상대방의 주소를 알아야 함

- 주소는 보통 IP로 사용하는데 프로세스는 여러 개일 수 있음

- 프로세스를 구분 짓기 위해 포트번호를 사용하여 여러 개의 프로세스와 통신 할 수 있게 함

- 보통 IPv4 : Port# 로 구성(ex : 192.16.8.5 : 2925)

- 1:1 통신(서버-클라이언트 통신)을 기반으로 하나 1:다, 다:다 통신도 가능하게 할 수 있음

(스레드 이용)

2. 소켓 통신하는 방법

- 서버 및 클라이언트의 기준에 따라 설계가 달라짐

- 서버 : socket()->bind()->listen()->accept()->{송/수신}+->close()

- 클라이언트 : socket->connect()->{송/수신}+->close()

- 송/수신 : write(), send() / read(), recv()

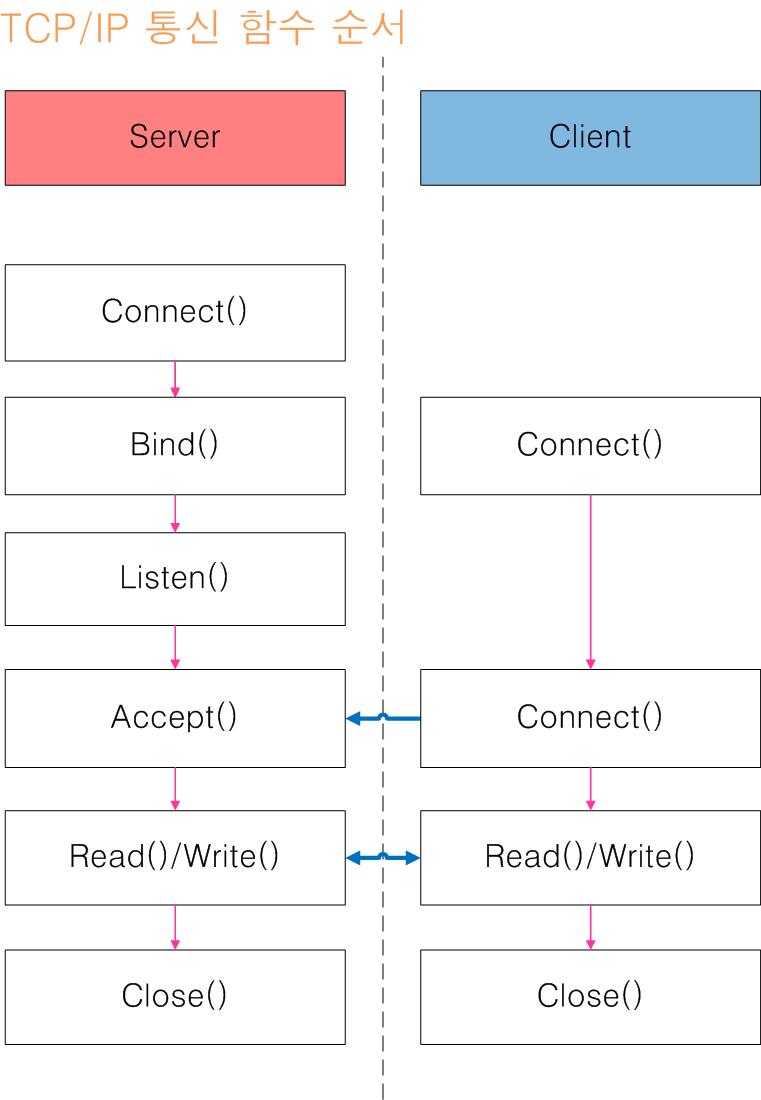
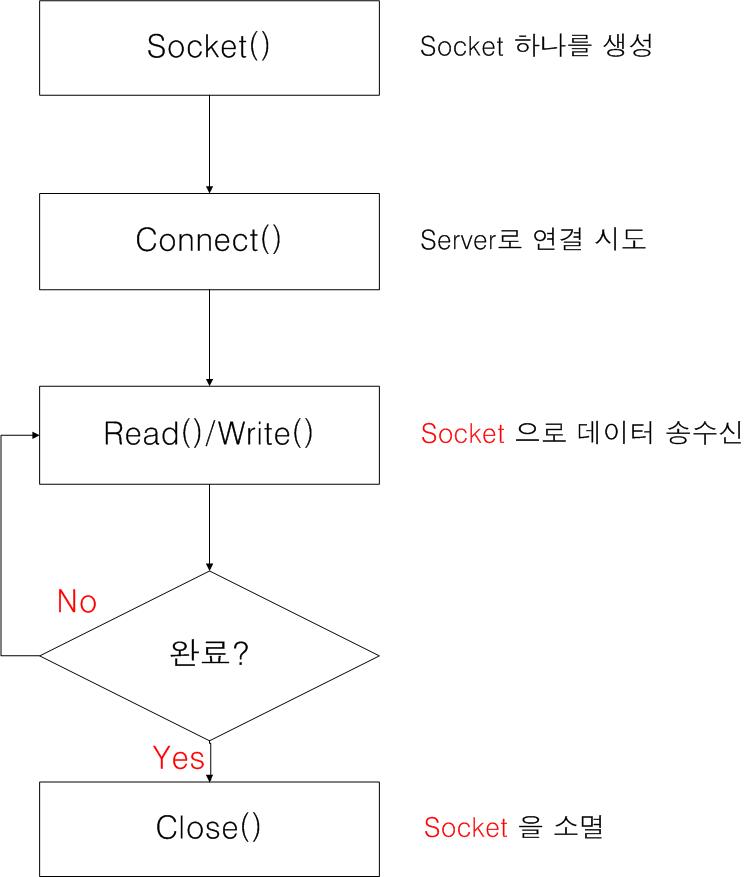
 

그림 1. 소켓-클라이언트 통신 구조 그림 2. 클라이언트 진행 순서도

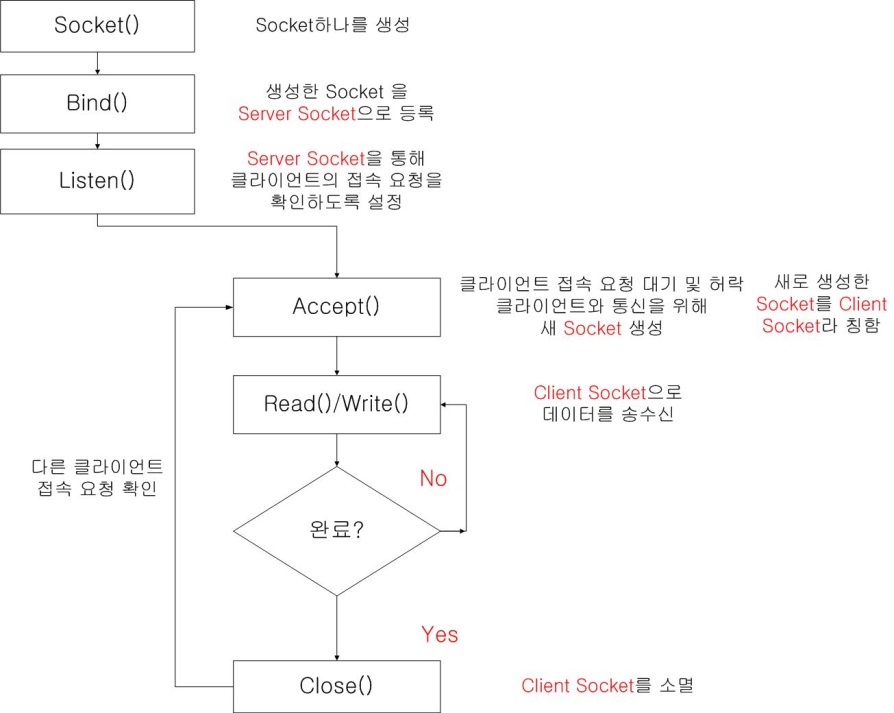


그림 3. 서버 진행 순서도

3. 서버 코드

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <errno.h>

#define BUF\_SIZE 1024

int main(void){

int server\_socket;

int client\_socket;

int client\_addr\_len;

int retval;

struct sockaddr\_in server\_addr;

struct sockaddr\_in client\_addr;

char buf\_rcv[BUF\_SIZE+5];

char buf\_snd[BUF\_SIZE+5];

server\_socket = socket(PF\_INET, SOCK\_STREAM, 0);

if(server\_socket == -1){

printf("server : server socket creation fail\n");

exit(1);

}

memset(&server\_addr, 0, sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(9000);

server\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

retval = bind(server\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

if(retval == -1){

printf("server : bind error\n");

exit(1);

}

retval = listen(server\_socket, SOMAXCONN);

if(retval == -1){

printf("server : listen error\n");

exit(1);

}

while(1){

client\_addr\_len = sizeof(client\_addr);

client\_socket = accept(server\_socket, (struct sockaddr\*)&client\_addr, &client\_addr\_len);

if(client\_socket == -1){

printf("server : accept fail\n");

printf("Error %d : %s \n", errno, strerror(errno));

exit(1);

}

read(client\_socket, buf\_rcv, BUF\_SIZE);

printf("receive : %s\n", buf\_rcv);

sprintf(buf\_snd, "%d : %s", strlen(buf\_rcv), buf\_rcv);

write(client\_socket, buf\_snd, strlen(buf\_snd)+1);

close(client\_socket);

}

return 0;

}

4. 클라이언트 코드

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/types.h>

#include <sys/socket.h>

#define BUF\_SIZE 1024

int main(int argc, char \*\*argv)

{

int client\_socket;

int retval;

struct sockaddr\_in server\_addr;

char buf\_rcv[BUF\_SIZE+5];

client\_socket = socket(PF\_INET, SOCK\_STREAM, 0);

if(client\_socket == -1){

printf("client : client socket creation fail\n");

exit(1);

}

memset(&server\_addr, 0, sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(9000);

server\_addr.sin\_addr.s\_addr = inet\_addr("210.125.29.109");

retval = connect(client\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

if(retval==-1){

printf("client : connection fail\n");

exit(1);

}

write(client\_socket, argv[1], strlen(argv[1])+1);

read(client\_socket, buf\_rcv, BUF\_SIZE);

printf("%s\n", buf\_rcv);

close(client\_socket);

return 0;

}

개발 2주차 : 소켓 통신 심화

1. 과제 목적

- 로그인 기능 구현{ 계정(id, pw) 생성, 계정 로그인, 계정 암호 변경, 명령어 1개 }

2. 프로토콜 및 기능 설계

- 클라이언트에서는 명령번호 입력 후 받은 번호로 스위칭하여 각 번호에 따른 기능을 수행

- 서버에서는 받은 명령번호를 되돌려주고 번호에 따른 기능을 수행 함

- 조건 : 유저 정보를 저장한 파일이 반드시 존재하여야 함(user.txt)

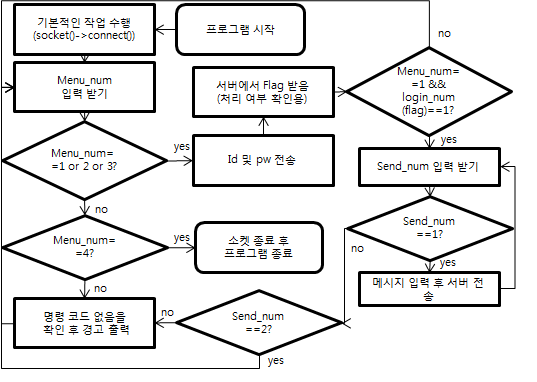


그림 1. 클라이언트 작동 순서도

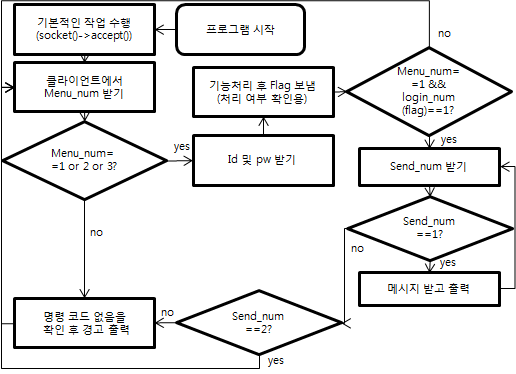


그림 2. 서버 작동 순서도

3. 서버 코드

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <errno.h>

#define BUF\_SIZE 1024

//사용자 구조체(아이디, 비밀번호)

struct user\_info{

char id[BUF\_SIZE/4];

char pw[BUF\_SIZE/4];

};

int main(void){

//서버, 클라이언트 소켓

//클라이언트 주소 크기

//명령번호, 로그인 플래그, 로그인 후 명령번호

//생성 플래그, 변경 플래그, 찾기 플래그, 종료 플래그

//아이디 길이, 비밀번호 길이

//반환값, 반복변수들

int server\_socket, client\_socket;

int client\_addr\_len;

int instr\_num, login\_num, send\_num;

int crt\_flag, change\_flag, find\_flag, exit\_flag;

int id\_len, pw\_len;

int retval, i=0, j=0;

//서버, 클라이언트 주소 구조체

struct sockaddr\_in server\_addr;

struct sockaddr\_in client\_addr;

//리시브, 센드 버퍼, 아이디 비교 변수

char buf\_rcv[BUF\_SIZE+5];

char buf\_snd[BUF\_SIZE+5];

char cmp\_id[BUF\_SIZE/4];

//저장된 유저 정보, 입력된 현재 유저 정보

struct user\_info reg\_user[99];

struct user\_info con\_user;

//저장된 유저 정보 수

int maxCnt=0;

//파일 제어 변수

FILE \*file;

//저장된 유저 정보 변수 초기화

memset(reg\_user, 0, sizeof(reg\_user));

//서버 소켓 생성

server\_socket = socket(PF\_INET, SOCK\_STREAM, 0);

if(server\_socket == -1){

printf("server : server socket creation fail\n");

exit(1);

}

//파일 읽기 모드 열기

file = fopen("user.txt", "r");

if(file==NULL){

printf("server : file open error\n");

exit(1);

}

//파일에 저장된 id, pw 읽어오기

while(!feof(file)){

fgets(reg\_user[i].id, sizeof(reg\_user[i].id), file);

id\_len = strlen(reg\_user[i].id);

if(reg\_user[i].id[id\_len-1]=='\n')

reg\_user[i].id[id\_len-1]='\0';

fgets(reg\_user[i].pw, sizeof(reg\_user[i].pw), file);

pw\_len = strlen(reg\_user[i].pw);

if(reg\_user[i].pw[pw\_len-1]=='\n')

reg\_user[i].pw[pw\_len-1]='\0';

i++;

}

//저장된 유저 정보 수 대입

maxCnt = i;

//파일 종료

fclose(file);

//서버 초기화 및 프로토콜, 포트, 주소 설정

memset(&server\_addr, 0, sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(9000);

server\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

//바인딩

retval = bind(server\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

if(retval == -1){

printf("server : bind error\n");

exit(1);

}

//리스닝

retval = listen(server\_socket, SOMAXCONN);

if(retval == -1){

printf("server : listen error\n");

exit(1);

}

while(1){

//클라이언트 연결 받기

client\_addr\_len = sizeof(client\_addr);

client\_socket = accept(server\_socket, (struct sockaddr\*)&client\_addr, &client\_addr\_len);

if(client\_socket == -1){

printf("server : accept fail\n");

printf("Error %d : %s \n", errno, strerror(errno));

exit(1);

}

while(1){

//현재 유저 정보 초기화

memset(&con\_user, 0, sizeof(con\_user));

//클라이언트에서 메뉴 번호(명령 번호) 받아오기

retval = recv(client\_socket, &instr\_num, sizeof(int), 0);

if(retval == -1){

printf("server : recv(instr\_num) error\n");

}

//명령 번호에 따른 기능 수행

//1. 로그인

//2. 계정 생성

//3. 비밀번호 변경

//4. 종료

switch(instr\_num){

case 1:

//명령번호 클라이언트로 보내기(확인)

retval = send(client\_socket, &instr\_num, sizeof(int), 0);

if(retval==-1){

printf("server : send(instr\_num) error\n");

break;

}

//아이디, 비밀번호 받아오기

retval = recv(client\_socket, con\_user.id, sizeof(con\_user.id), 0);

if(retval==-1){

printf("server : recv(id) error\n");

break;

}

retval = recv(client\_socket, con\_user.pw, sizeof(con\_user.pw), 0);

if(retval==-1){

printf("server : recv(pw) error\n");

break;

}

//저장된 유저 정보와 비교

for(i=0;i<maxCnt;i++){

//비밀 번호 맞으면 플래그로 확인 신호 보냄

if(strcmp(con\_user.id, reg\_user[i].id)==0 && strcmp(con\_user.pw,

reg\_user[i].pw)==0){

login\_num = 1;

retval = send(client\_socket, &login\_num, sizeof(int), 0);

if(retval==-1){

printf("server : send(ok) error\n");

break;

}

break;

}

}

//정보 일치한 것이 없으면 없음 신호를 보냄

if(i>=maxCnt){

login\_num = 0;

retval = send(client\_socket, &login\_num, sizeof(int), 0);

if(retval==-1){

printf("server : send(not) error\n");

break;

}

}

break;

case 2:

retval = send(client\_socket, &instr\_num, sizeof(int), 0);

if(retval==-1){

printf("server : send(instr\_num) error\n");

break;

}

retval = recv(client\_socket, con\_user.id, sizeof(con\_user.id), 0);

if(retval==-1){

printf("server : recv(id) error\n");

break;

}

retval = recv(client\_socket, con\_user.pw, sizeof(con\_user.pw), 0);

if(retval==-1){

printf("server : recv(pw) error\n");

break;

}

for(i=0;i<maxCnt;i++){

//유저 정보와 일치하다면 중복 신호 보냄

if(strcmp(con\_user.id, reg\_user[i].id)==0 && strcmp(con\_user.pw,

reg\_user[i].pw)==0){

crt\_flag = 0;

retval = send(client\_socket, &crt\_flag, sizeof(int), 0);

if(retval==-1){

printf("server : send(ok) error\n");

break;

}

break;

}

}

//일치하는 정보 없으면 확인 신호 보냄

if(i>=maxCnt){

crt\_flag = 1;

//파일 추가 모드 열기

file = fopen("user.txt", "a");

if(file!=NULL){

//파일에 생성된 계정 정보 추가

fprintf(file, "%s\n%s", con\_user.id, con\_user.pw);

}

retval = send(client\_socket, &crt\_flag, sizeof(int), 0);

if(retval==-1){

printf("server : send(not) error\n");

break;

}

fclose(file);

//파일 읽기 모드 열기

file = fopen("user.txt", "r");

i=0;

if(file==NULL){

printf("server : file open error\n");

exit(1);

}

//정보 갱신

while(!feof(file)){

fgets(reg\_user[i].id, sizeof(reg\_user[i].id), file);

id\_len = strlen(reg\_user[i].id);

if(reg\_user[i].id[id\_len-1]=='\n')

reg\_user[i].id[id\_len-1]='\0';

fgets(reg\_user[i].pw, sizeof(reg\_user[i].pw), file);

pw\_len = strlen(reg\_user[i].pw);

if(reg\_user[i].pw[pw\_len-1]=='\n')

reg\_user[i].pw[pw\_len-1]='\0';

i++;

}

maxCnt = i;

fclose(file);

}

break;

case 3:

retval = send(client\_socket, &instr\_num, sizeof(int), 0);

if(retval==-1){

printf("server : send(instr\_num) error\n");

break;

}

retval = recv(client\_socket, con\_user.id, sizeof(con\_user.id), 0);

if(retval==-1){

printf("server : recv(id) error\n");

break;

}

retval = recv(client\_socket, con\_user.pw, sizeof(con\_user.pw), 0);

if(retval==-1){

printf("server : recv(pw) error\n");

break;

}

for(i=0;i<maxCnt;i++){

//아이디가 일치한 정보 존재 시 확인신호 보냄

if(strcmp(con\_user.id, reg\_user[i].id)==0){

change\_flag = 1;

//등록된 정보의 비밀번호 변경

strcpy(reg\_user[i].pw, con\_user.pw);

//파일 쓰기 모드 열기

file = fopen("user.txt", "w");

if(file!=NULL){

//등록된 정보를 파일에 쓰기

for(j=0;j<maxCnt;j++){

fprintf(file, "%s\n%s", reg\_user[j].id,

reg\_user[j].pw);

if(j<(maxCnt-1))

fprintf(file, "\n");

}

maxCnt = j;

fclose(file);

}

retval = send(client\_socket, &change\_flag, sizeof(int), 0);

if(retval==-1){

printf("server : send(ok) error\n");

break;

}

break;

}

}

//일치하는 정보 없다면 없음 신호 보냄

if(i>=maxCnt){

change\_flag = 0;

retval = send(client\_socket, &change\_flag, sizeof(int), 0);

if(retval==-1){

printf("server : send(not) error\n");

break;

}

}

break;

case 4:

//종료 신호로 변경

retval = send(client\_socket, &instr\_num, sizeof(int), 0);

if(retval == -1){

printf("server : send(instr\_num) error\n");

break;

}

exit\_flag=1;

printf("server : client connect cut\n");

break;

default:

break;

}

//로그인이 되었다면

if(login\_num==1){

printf("server : login success\n");

//확인 차 유저 아이디 보냄

retval = send(client\_socket, con\_user.id, strlen(con\_user.id), 0);

if(retval == -1){

printf("server : send(check\_id) error\n");

break;

}

while(1){

//클라이언트에서 로그인 후 기능 번호 받음

retval = recv(client\_socket, &send\_num, sizeof(int), 0);

if(retval == -1){

printf("server : recv(send\_num) error\n");

break;

}

//받은 번호로 기능 수행

switch(send\_num){

case 1:

//리시브 버퍼 초기화

memset(buf\_rcv, 0, sizeof(buf\_rcv));

//클라이언트에서 입력한 내용 버퍼에 저장

retval = recv(client\_socket, buf\_rcv, sizeof(buf\_rcv), 0);

if(retval == -1){

printf("server : recv(msg) error\n");

break;

}

//받은 내용 출력

printf("server : %s \n", buf\_rcv);

break;

case 2:

//클라이언트에서 로그아웃 번호 받음

retval = recv(client\_socket, &login\_num, sizeof(int), 0);

if(retval == -1){

printf("server : recv(login\_num) \n");

break;

}

//로그아웃 번호 다시 보냄

retval = send(client\_socket, &login\_num, sizeof(int), 0);

if(retval == -1){

printf("server : sed(login\_num) \n");

break;

}

break;

default:

break;

}

//로그아웃 확인 시 로그아웃 출력

if(login\_num==0){

printf("server : logout\n");

break;

}

}

}

//종료 플래그 온이면 연결 종료

if(exit\_flag==1){

break;

}

}

close(client\_socket);

}

close(server\_socket);

return 0;

}

4. 클라이언트 코드

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio\_ext.h>

#define BUF\_SIZE 1024

int main(void)

{

//클라이언트 소켓

//메뉴 번호, 정지 플래그, 확인 번호

//로그인 플래그, 로그인 후 기능 번호

//로그아웃 플래그, 생성 플래그, 변경 플래그

//아이디, 비밀번호 길이

//반환값

int client\_socket;

int menu\_num, break\_flag, check\_num;

int login\_num, send\_num;

int logout\_flag, crt\_flag, change\_flag;

int id\_len, pw\_len;

int retval;

//서버주소

struct sockaddr\_in server\_addr;

//리시브, 센드 버퍼

//아이디, 비밀번호 변수

//서버에서 받는 아이디 변수(확인용)

char buf\_rcv[BUF\_SIZE+5];

char buf\_snd[BUF\_SIZE+5];

char id[BUF\_SIZE/4];

char pw[BUF\_SIZE/4];

char check\_id[BUF\_SIZE/4];

//소켓 생성

client\_socket = socket(PF\_INET, SOCK\_STREAM, 0);

if(client\_socket == -1){

printf("client : client socket creation fail\n");

exit(1);

}

//서버 주소 초기화 및 프로토콜, 포트번호, 주소 지정

memset(&server\_addr, 0, sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(9000);

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

//서버로 연결

retval = connect(client\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

if(retval==-1){

printf("client : connection fail\n");

exit(1);

}

printf("client : connection Success!\n");

\_\_fpurge(stdin);

while(1){

//UI 출력

printf("------Menu------\n");

printf("1. Login\n");

printf("2. Create Account\n");

printf("3. Change Password\n");

printf("4. Exit\n");

printf("> ");

//번호 입력

scanf("%d", &menu\_num);

//입력 버퍼 초기화

\_\_fpurge(stdin);

//fflush(stdin); -> not operate(linux)

//서버로 메뉴번호(기능번호) 전송

retval = send(client\_socket, &menu\_num, sizeof(int), 0);

if(retval == -1){

printf("client : send(menu\_num) error\n");

break;

}

//서버에서 보낸 번호 받기

retval = recv(client\_socket, &check\_num, sizeof(int), 0);

if(retval == -1){

printf("client : recv(check\_num) error\n");

break;

}

//아이디, 비밀번호 초기화

memset(id, 0, sizeof(id));

memset(pw, 0, sizeof(pw));

switch(check\_num){

case 1:

//아이디, 비밀번호 입력

printf("client : input id\n");

fgets(id, sizeof(id), stdin);

\_\_fpurge(stdin);

//fflush(stdin);

printf("client : input pw\n");

fgets(pw, sizeof(pw), stdin);

\_\_fpurge(stdin);

//fflush(stdin);

//전송용 데이터로 만들기(문자열의 끝 표현)

id\_len = strlen(id);

if(id[id\_len-1]=='\n')

id[id\_len-1]='\0';

pw\_len = strlen(pw);

if(pw[pw\_len-1]=='\n')

pw[pw\_len-1]='\0';

//아이디, 비밀번호 전송

retval = send(client\_socket, id, strlen(id), 0);

if(retval == -1){

printf("client : send(id) error\n");

break;

}

retval = send(client\_socket, pw, strlen(pw), 0);

if(retval == -1){

printf("client : send(pw) error\n");

break;

}

//서버에서 로그인 번호 받기

retval = recv(client\_socket, &login\_num, sizeof(int), 0);

if(retval == -1){

printf("client : recv(login\_num) error\n");

break;

}

//로그인 번호 0이면 정보 일치하지 않음을 표시

if(login\_num==0){

printf("client : no match id or password\n");

break;

}

break;

case 2:

printf("client : input id\n");

fgets(id, BUF\_SIZE/4, stdin);

\_\_fpurge(stdin);

//fflush(stdin);

printf("client : input pw\n");

fgets(pw, BUF\_SIZE/4, stdin);

\_\_fpurge(stdin);

//fflush(stdin);

id\_len = strlen(id);

if(id[id\_len-1]=='\n')

id[id\_len-1]='\0';

pw\_len = strlen(pw);

if(pw[pw\_len-1]=='\n')

pw[pw\_len-1]='\0';

retval = send(client\_socket, id, strlen(id), 0);

if(retval == -1){

printf("client : send(id) error\n");

break;

}

retval = send(client\_socket, pw, strlen(pw), 0);

if(retval == -1){

printf("client : send(pw) error\n");

break;

}

//생성 플래그 받기

retval = recv(client\_socket, &crt\_flag, sizeof(int), 0);

if(retval == -1){

printf("client : recv(crt\_flag) error\n");

break;

}

//생성 실패 시 실패 메세지 출력

if(crt\_flag==0)

printf("client : creation fail \n");

//생성 성공 시 성공 메세지 출력

else

printf("client : creation success \n");

break;

case 3:

printf("client : input id\n");

fgets(id, BUF\_SIZE/4, stdin);

\_\_fpurge(stdin);

//fflush(stdin);

printf("client : input pw\n");

fgets(pw, BUF\_SIZE/4, stdin);

\_\_fpurge(stdin);

//fflush(stdin);

id\_len = strlen(id);

if(id[id\_len-1]=='\n')

id[id\_len-1]='\0';

pw\_len = strlen(pw);

if(pw[pw\_len-1]=='\n')

pw[pw\_len-1]='\0';

retval = send(client\_socket, id, strlen(id), 0);

if(retval == -1){

printf("client : send(id) error\n");

break;

}

retval = send(client\_socket, pw, strlen(pw), 0);

if(retval == -1){

printf("client : send(pw) error\n");

break;

}

//변경 플래그 받기

retval = recv(client\_socket, &change\_flag, sizeof(int), 0);

if(retval == -1){

printf("client : recv(chagne\_flag) error\n");

break;

}

//실패하면 실패 메세지 출력

if(change\_flag==0)

printf("client : change fail \n");

//성공하면 성공 메세지 출력

else

printf("client : change success \n");

break;

case 4:

//종료 메세지 출력 후 플래그 1로 변경

printf("client : exit\n");

break\_flag=1;

break;

default:

printf("Not exist number\n");

break;

}

//종료 메세지가 1이면 종료

if(break\_flag==1)

break;

//로그인이 되었다면

if(login\_num==1){

//서버에서 받을 확인용 아이디 변수 초기화

memset(check\_id, 0, sizeof(check\_id));

//확인용 아이디 받기 및 출력

retval = recv(client\_socket, check\_id, sizeof(check\_id), 0);

if(retval == -1){

printf("client : recv(check\_id) error\n");

break;

}

//타 에러로 아이디가 다르다면 경고 출력

if(strcmp(check\_id, id)!=0){

printf("client : no match id\n");

break;

}

//제대로 왔다면 로그인 후 기능 수행

else{

//확인용 아이디 출력

printf("client : %s login success\n", check\_id);

while(1){

//로그인 후 UI 출력

printf("==function menu==\n");

printf("1. send message\n");

printf("2. logout\n");

printf("> ");

//기능번호 입력 받기

scanf("%d", &send\_num);

\_\_fpurge(stdin);

//fflush(stdin);

//기능번호 보내기

retval = send(client\_socket, &send\_num, sizeof(int), 0);

if(retval == -1){

printf("client : send(menu\_num) error\n");

break;

}

//기능 번호에 따른 기능 수행

switch(send\_num){

case 1:

//리시브 버퍼 초기화(사실 센드 변수를 사용해야함)

memset(buf\_rcv, 0, sizeof(buf\_rcv));

printf("input the message : ");

fgets(buf\_rcv, BUF\_SIZE/4, stdin);

if(buf\_rcv[strlen(buf\_rcv)-1]=='\n')

buf\_rcv[strlen(buf\_rcv)-1]='\0';

\_\_fpurge(stdin);

//fflush(stdin);

//메세지 전송

retval = send(client\_socket, buf\_rcv, strlen(buf\_rcv), 0);

if(retval == -1){

printf("client : send(msg) error\n");

break;

}

break;

case 2:

//로그인 해제 번호 서버로 보내기

login\_num = 0;

retval = send(client\_socket, &login\_num, sizeof(int), 0);

if(retval == -1){

printf("client : send(login\_num) error\n");

break;

}

//보낸 번호 다시 받기

retval = recv(client\_socket, &login\_num, sizeof(int), 0);

if(retval == -1){

printf("client : recv(login\_num) error\n");

break;

}

break;

default:

break;

}

//로그아웃 되었다면 로그아웃 메세지 출력 후 원래 기능 수행

if(login\_num==0){

printf("client : logout\n");

break;

}

}

}

}

}

close(client\_socket);

return 0;

}

개발 3주차 : 아두이노 입문

1. 아두이노 시작 하기 전에

- 스케치 설치 : <http://arduino.cc/en/Main/Software>

- 설치는 다른 응용프로그램과 동일

- 스케치는 아두이노 제품군에 코드를 올려 작동할 수 있게 해줌 => 펌웨어

- 와이파이 쉴드 업데이트 : <http://www.arduino.cc/en/Hacking/WiFiShieldFirmwareUpgrading>

- 와이파이 쉴드 펌웨어 업데이트를 하지 않으면 통신 코드 실행 시 오작동을 일으킬 수 있음

- 장치 연결 시 아날로그-디지털 구분을 잘 해주어야 하며 DAC/ADC 장착 여부도 알아야 함

2. 아두이노 스케치를 통한 온도 측정

- 온/습도 측정 : DHT22 사용

- 레퍼런스   
<http://www.dfrobot.com/wiki/index.php/DHT22_Temperature_and_humidity_module_SKU:SEN0137>

- 샘플 코드 : <http://arduino.cc/playground/Main/DHTLib>

- 주요 코드 설명

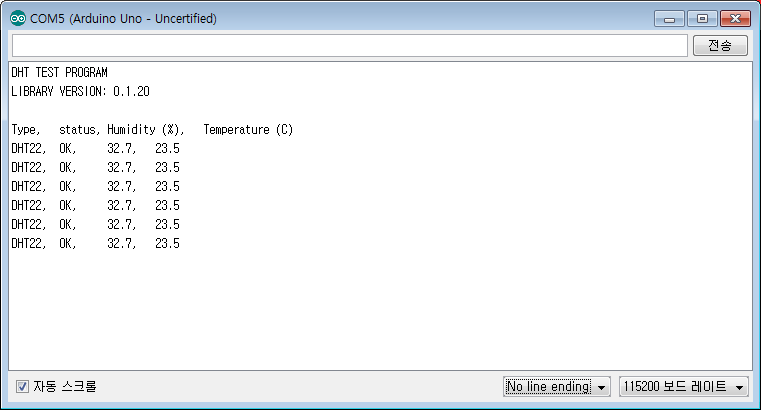
1) #define DHT22\_PIN 6 : 6번 핀을 온도의 데이터 전송으로 사용(S를 6번핀과 연결)

2) Serial.begin(115200); : 전송률 설정(시리얼 모니터 확인 시, 보드 레이트를 이것과 동일하게)

3) delay(10000); : 일정 시간의 지연을 줌(괄호안의 단위는 ms, 10000ms == 10s)

4) 나머지 코드는 대다수 dht라이브러리에서 가져 옴(라이브러리에서 함수 확인 가능)

- 실행 화면



3. 아두이노 와이파이 쉴드 사용하기

- 와이파이 레퍼런스 : <http://www.arduino.cc/en/Reference/WiFi>

- 라이브러리에 있는 클래스 및 함수들을 확인할 수 있으며 예제 코드 확인 가능

- 예제 코드 실행 : ConnectWithWPA(WPA기반의 와이파이 네트워크 망 접속)

(접속하고 접속한 네트워크망의 정보를 10초마다 출력)

#include <SPI.h>

#include <WiFi.h>

char ssid[] = "iptime"; // your network SSID (name)

char pass[] = "esw76012489@!"; // your network password

int status = WL\_IDLE\_STATUS; // the Wifi radio's status

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for Leonardo only

}

// check for the presence of the shield:

if (WiFi.status() == WL\_NO\_SHIELD) {

Serial.println("WiFi shield not present");

// don't continue:

while(true);

}

// attempt to connect to Wifi network:

while ( status != WL\_CONNECTED) {

Serial.print("Attempting to connect to WPA SSID: ");

Serial.println(ssid);

// Connect to WPA/WPA2 network:

status = WiFi.begin(ssid, pass);

// wait 10 seconds for connection:

delay(10000);

}

// you're connected now, so print out the data:

Serial.print("You're connected to the network");

printCurrentNet();

printWifiData();

}

void loop() {

// check the network connection once every 10 seconds:

delay(10000);

printCurrentNet();

}

void printWifiData() {

// print your WiFi shield's IP address:

IPAddress ip = WiFi.localIP();

Serial.print("IP Address: ");

Serial.println(ip);

Serial.println(ip);

// print your MAC address:

byte mac[6];

WiFi.macAddress(mac);

Serial.print("MAC address: ");

Serial.print(mac[5],HEX);

Serial.print(":");

Serial.print(mac[4],HEX);

Serial.print(":");

Serial.print(mac[3],HEX);

Serial.print(":");

Serial.print(mac[2],HEX);

Serial.print(":");

Serial.print(mac[1],HEX);

Serial.print(":");

Serial.println(mac[0],HEX);

}

void printCurrentNet() {

// print the SSID of the network you're attached to:

Serial.print("SSID: ");

Serial.println(WiFi.SSID());

// print the MAC address of the router you're attached to:

byte bssid[6];

WiFi.BSSID(bssid);

Serial.print("BSSID: ");

Serial.print(bssid[5],HEX);

Serial.print(":");

Serial.print(bssid[4],HEX);

Serial.print(":");

Serial.print(bssid[3],HEX);

Serial.print(":");

Serial.print(bssid[2],HEX);

Serial.print(":");

Serial.print(bssid[1],HEX);

Serial.print(":");

Serial.println(bssid[0],HEX);

// print the received signal strength:

long rssi = WiFi.RSSI();

Serial.print("signal strength (RSSI):");

Serial.println(rssi);

// print the encryption type:

byte encryption = WiFi.encryptionType();

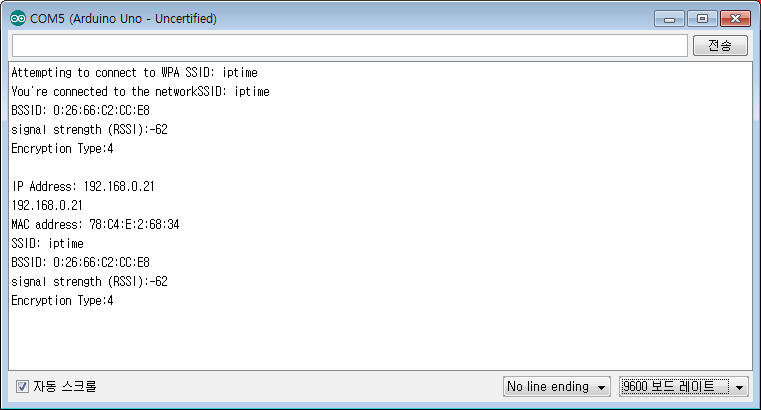
Serial.print("Encryption Type:");

Serial.println(encryption,HEX);

Serial.println();

}

- 실행화면



4. 아두이노 라이브러리 추가하기

- dht22의 경우 따로 라이브러리를 추가해주어야 함

- 라이브러리를 구했으면 C:\Program Files (x86)\Arduino\libraries(보통의 경우) 폴더에 받은

라이브러리 폴더를 복사해 넣으면 자동으로 추가가 됨

개발 4주차 : 아두이노 심화

1. 아두이노 웹소켓 연결

- 아두이노 제어는 라즈베리파이의 웹서버에서 이루어지므로 웹서버에 접근 가능한 클라이언트를 제작해야 함

- 웹서버는 클라이언트에서 먼저 연결요청을 하여 웹서버와의 Handshake 과정을 거쳐야 함

- 이후 통상적인 웹서버는 클라이언트가 request한 내용에 대해서만 response가 가능함

- 이것을 막기 위해 websocket(웹소켓)을 이용함

- 웹소켓은 Handshake 과정만 거치면 서버가 먼저 클라이언트에게 요청이 가능함

- 아두이노에서 웹소켓을 사용하기 위해서는 라이브러리 추가가 필요함  
 (<https://github.com/brandenhall/Arduino-Websocket>)

- 웹소켓 라이브러리는 WiFLY, Ethernet을 기반으로 하고 있기 때문에 라이브러리에 있는 헤더 및 소스코드에서 Client부분을 WiFiClient으로 수정해야 WiFi 환경에서 사용이 가능함

2. 아두이노 웹소켓 클라이언트 및 자바스크립트 웹서버

- 웹 서버와의 연결을 테스트 하기 위해 에코 서버 및 에코 클라이언트를 준비함

- 웹 서버에서는 node.js와 내부 모듈인 websocket을 이용하여 연결을 준비함

- 기본 프로젝트 생성 후 웹 소켓을 생성하고 연결되었을 때 하는 작동(프로토콜) 제작

- 프로토콜 내용은 클라이언트에서 받은 자료를 콘솔로 출력하고 받은 내용을 다시 클라이언트에 전달함

- 클라이언트에서는 시리얼 모니터가 가능하게 준비한 후 주변 WiFi 네트워크를 찾아 접속함

- WiFi 사용이 가능해지면 서버와 웹소켓에 연결함

- 정상적으로 연결되었다면 웹서버에 자료를 보내고 웹서버에서 보낸 자료를 받음

3. 아두이노 웹소켓 클라이언트 소스코드

/\* 암호(key) 관련 헤더 \*/

#include <Base64.h>

#include <global.h>

#include <MD5.h>

#include <sha1.h>

/\* 웹 소켓 사용 관련 헤더 \*/

#include <WebSocketClient.h>

/\* WiFi 사용 관련 헤더 \*/

#include <WiFi.h>

#include <SPI.h>

/\* 비밀 번호 \*/

char pass[] = "esw76012489@!";

/\* ssid 개수 \*/

int cntSSID;

/\* WiFi 접속 상태 변수 \*/

int status = WL\_IDLE\_STATUS;

/\* 서버 IP \*/

char server[] = "192.168.0.12";

/\* WiFi 및 웹소켓 클라이언트 변수 \*/

WiFiClient wifiClient;

WebSocketClient webSocketClient;

void setup() {

/\* 시리얼 보드레이트(baud rate) 설정

Serial.begin(9600);

/\* WiFi Shield 유무 확인 \*/

if (WiFi.status() == WL\_NO\_SHIELD) {

Serial.println("WiFi shield not present");

// don't continue:

while(true);

}

/\* 주변 WiFi 네트워크 검색 \*/

cntSSID = WiFi.scanNetworks();

if(cntSSID == -1){

Serial.println("Couldn't get a wifi connection");

while(true);

}

int i=0;

/\* 존재하는 WiFi에 연결

WiFi의 암호화 경우에 따라 접속 방법 구분 \*/

while (status != WL\_CONNECTED && i<cntSSID) {

Serial.print("Attempting to connect to SSID: ");

Serial.println(WiFi.SSID(i));

switch(WiFi.encryptionType(i)){

case ENC\_TYPE\_NONE: //WiFi 암호화가 걸리지 않은 경우

status = WiFi.begin(WiFi.SSID(i));

break;

case ENC\_TYPE\_CCMP: //WiFi 암호화 중 WPA/WPA2의 경우

status = WiFi.begin(WiFi.SSID(i), pass);

break;

default:

break;

}

// wait 3 seconds for connection:

delay(3000);

}

Serial.println("WiFi Connection Success!!");

//서버로의 연결

if(wifiClient.connect(server, 80)){

Serial.println("Connection success");

}

else

Serial.println("Connection fail");

/\* websocket 연결에 필요한 handshake 작업을 위한 정보 입력 \*/

webSocketClient.path = "/"; //websocket이 작동되는 경로

webSocketClient.host = "192.168.0.12"; //websocket이 작동하는 웹 서버 주소

webSocketClient.protocol = "example-echo"; //websocket이 사용하는 프로토콜

/\* websocket 연결을 위한 handshake 작업 \*/

if(webSocketClient.handshake(wifiClient)){

Serial.println("Handshake success");

delay(1000);

}

else

Serial.println("Handshake fail");

}

void loop() {

String data;

/\* websocket으로 데이터 전송 \*/

webSocketClient.sendData("Hello World!");

delay(500);

/\* websocket에서 보내는 데이터 받기 \*/

if(wifiClient.connected()){

if(webSocketClient.getData(data)){

Serial.print("Received data: ");

Serial.println(data);

}

}

delay(3000);

}

4. 자바스크립트 웹서버 소스코드

/\*\*

\* Module dependencies.

\*/

**var** express = require('express')

, routes = require('./routes')

, user = require('./routes/user')

, http = require('http')

, path = require('path')

, server = http.createServer(app)

, WebSocketServer = require('websocket').server;

**var** app = express();

// all environments

app.set('port', process.env.PORT || 80);

app.set('views', \_\_dirname + '/views');

app.set('view engine', 'ejs');

app.use(express.favicon());

app.use(express.logger('dev'));

app.use(express.bodyParser());

app.use(express.methodOverride());

app.use(app.router);

app.use(express.**static**(path.join(\_\_dirname, 'public')));

// development only

**if** ('development' == app.get('env')) {

app.use(express.errorHandler());

}

app.get('/', routes.index);

app.get('/users', user.list);

/\* 서버의 리스닝 \*/

server.listen(80, **function** () {

console.log('Server is listening on port 80');

});

/\* 웹소켓 생성 \*/

wsServer = **new** WebSocketServer({

httpServer: server,

autoAcceptConnections: **false**

});

/\* 서버에 요청 발생 시 여기로 들어감 \*/

wsServer.on('request', **function** (request) {

/\* 연결 객체에 example-echo 프로토콜 정의 \*/

**var** connection = request.accept('example-echo', request.origin);

/\* 정의된 프로토콜 사용에 메시지가 오면 콘솔 출력 후 다시 보냄 \*/

connection.on('message', **function** (message) {

**if** (message.type === 'utf8') {

console.log('Received message: ' + message.utf8Data);

connection.sendUTF(message.utf8Data);

}

**else** **if** (message.type === 'binary') {

connection.sendBytes(message.binaryData);

}

/\* close가 오면 연결 끊고 종료 \*/

connection.on('close', **function** (reasonCode, description) {

console.log('Peer ' + connection.remoteAddress + ' disconnected.');

});

});

});

5. 참고 내용

1. 웹서버 구축 및 웹소켓 사용 방법 : <http://pyrasis.com/nodejs/nodejs-HOWTO>

개발 5~6주차 : 아두이노 선풍기 전용 클라이언트

1. 전용 클라이언트를 제작하기 위한 준비(하드웨어)

- 선풍기 제어를 위한 릴레이 쉴드, 통신을 위한 와이파이 쉴드를 모두 장착함

- 기본적으로 쉴드에서 쉴드-보드 간 데이터 전송을 위한 0, 1번 핀(RX, TX)을 제외한 모든 핀을 쓸 수 있지만 와이파이 쉴드의 경우는 4, 7, 10, 11, 12, 13번 핀을 못씀(기본적으로 이 핀들은 무언가를 위해 설정이 되어있음)

- 릴레이 쉴드는 기본 설정으로 2, 7, 8, 10번의 핀이 릴레이와 연결되어 있음

- 릴레이 쉴드와 와이파이 쉴드의 핀 번호(7, 10번)가 충돌되기 때문에 오작동의 우려가 있으므로 릴레이 쉴드의 핀 번호를 바꾸어 주어야 함

- 바꾸는 방법은 릴레이 쉴드 사용법에도 나와있으며 실제 사용방법은 아래의 사진처럼 이용

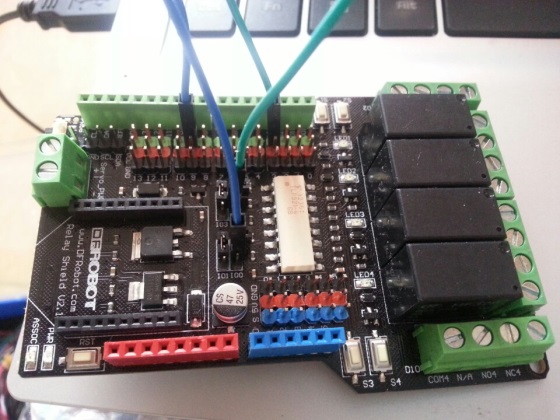


사진 1. 릴레이 쉴드 기본 핀 설정을 변경하는 연결

- 온/습도 체크를 위한 센서까지 장착 후 최종 모습은 아래와 같음

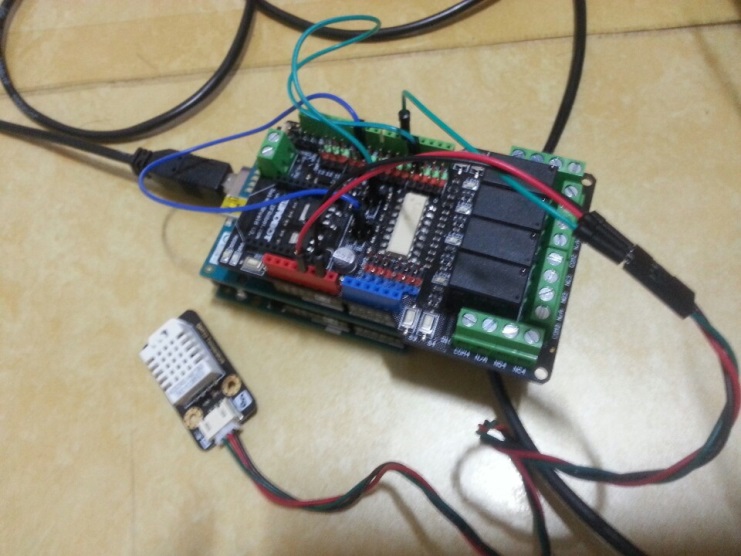


사진 2. 선풍기에 부착할 최종적인 아두이노의 모습

- 릴레이 쉴드에는 선풍기의 미풍, 약풍, 강풍, 회전과 연결된 선을 잘라 릴레이에 부착하여 신호를 통해 작동하게 함

2. 전용 클라이언트를 제작하기 위한 준비(소프트웨어 및 소스코드)

- 소프트웨어로는 펌웨어를 올릴 수 있는 스케치와 온도센서를 조절할 수 있는 DHT 라이브러리가 필요함

- 스케치 및 라이브러리에 관한 설명은 개발 3주차에 되어있음

- 소스코드의 전반적인 흐름  
 전원 On -> 와이파이 쉴드 유무 확인 -> 와이파이 쉴드 버전 확인 -> 와이파이 네트워크 확인  
 -> 주변에 존재하는 와이파이 네트워크 망 접속 -> 맥 주소 확인 -> 서버 접속 ->  
 {서버에서 메시지 받기 -> 프로토콜에 기반한 메시지 작성 -> 메시지 서버로 전송}+(루프)

- 추후 먼저 서버에게 맥 주소 정보와 기기의 정보를 전송할 코드 추가 예정

3. 클라이언트 전용 프로토콜

- 프로젝트에 필요한 프로토콜은 차영록 팀장이 작성한 오픈소스 기반 저가 스마트홈 개발 문서를 기반으로 하여 작성함

- 프로토콜 흐름은 아래의 그림들과 같음

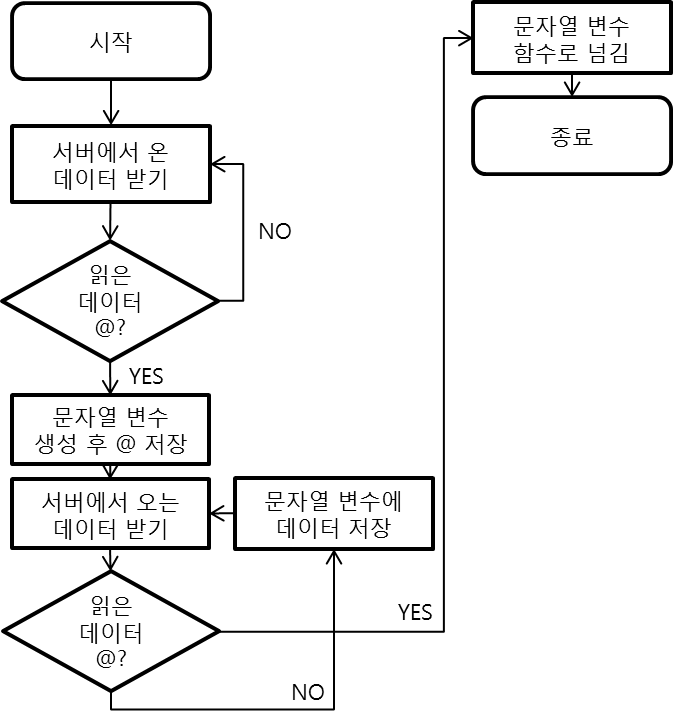


그림 1. 프로토콜 전용 문자열 수신을 위한 알고리즘 순서도

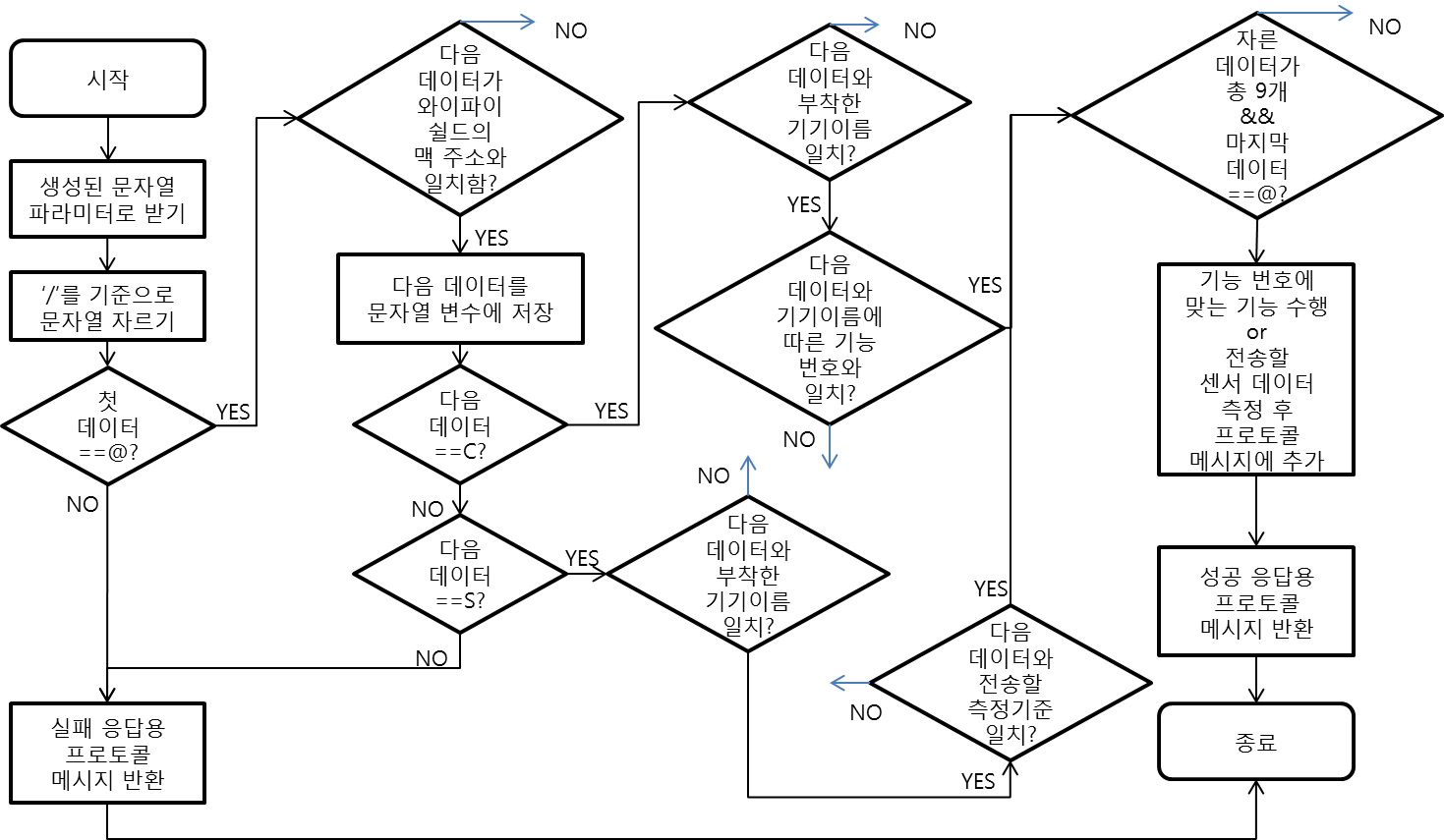


그림 2. 프로토콜 함수 알고리즘 순서도

- NO의 파란색 선으로 가는 경우는 모드 실패 응답용 프로토콜 메시지 반환으로 이어짐

4. 클라이언트 소스 코드

#include <WiFi.h>

#include <SPI.h>

#include <dht.h>

/\* About pin number

\* if you wanna another pin number,

\* you could change define number

\* NOTE : pin number must equal the connection pin number

\* Used pin

\* 0, 1 : RX, TX

\* 4 : SS for SD card

\* 7 : Handshake between shield and Arduino

\* 10 : SS for WiFi

\* 11 : MOSI

\* 12 : MISO

\* 13 : SCK

\* 2, 3, 5, 6, 8, 9 : Not used pin number

\*/

#define RELAY1 2

#define RELAY2 3

#define RELAY3 8

#define RELAY4 9

#define DHT\_PIN 5

char hexval[16] = {'0', '1', '2', '3', '4', '5', '6',

'7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};

char pass[] = "esw76012489@!";

int cntSSID, flagStr = 0;

int status = WL\_IDLE\_STATUS;

char server[] = "192.168.0.22";

WiFiClient wifiClient;

String buffer, macAddr;

byte mac[6];

char mac\_string[13];

dht DHT;

void setup() {

Serial.begin(9600);

pinMode(RELAY1, OUTPUT);

pinMode(RELAY2, OUTPUT);

pinMode(RELAY3, OUTPUT);

pinMode(RELAY4, OUTPUT);

if (WiFi.status() == WL\_NO\_SHIELD) {

Serial.println("WiFi shield not present");

// don't continue:

while(true);

}

String fv = WiFi.firmwareVersion();

Serial.println("Firmware Version = " + fv);

cntSSID = WiFi.scanNetworks();

if(cntSSID == -1){

Serial.println("Couldn't get a wifi connection");

while(true);

}

int i=0;

while (status != WL\_CONNECTED && i<cntSSID) {

Serial.print("Attempting to connect to SSID: ");

Serial.println(WiFi.SSID(i));

switch(WiFi.encryptionType(i)){

case ENC\_TYPE\_NONE:

status = WiFi.begin(WiFi.SSID(i));

break;

case ENC\_TYPE\_CCMP:

status = WiFi.begin(WiFi.SSID(i), pass);

break;

default:

break;

}

i++;

if(i>9)

i=0;

// wait 3 seconds for connection:

delay(1000);

}

Serial.println("WiFi Connection Success!!");

WiFi.macAddress(mac);

for(int j=0;j<6;j++){

mac\_string[j\*2] = hexval[((mac[5-j]>>4) & 0xF)];

mac\_string[(j\*2)+1] = hexval[((mac[5-j]) & 0x0F)];

}

mac\_string[13] = 0;

macAddr = mac\_string;

Serial.println(macAddr);

if(wifiClient.connect(server, 3000)){

Serial.println("Server Connection Success");

delay(1000);

}

else{

Serial.println("Server Connection fail");

while(true);

}

}

void loop() {

String data;

if(wifiClient.connected()){

if(wifiClient.available() > 0){

char c = wifiClient.read();

Serial.print(c);

if(c == '@' && flagStr == 0){

buffer = "@";

flagStr = 1;

}

if(flagStr == 1){

delay(500);

while(wifiClient.available()>0){

char ch = wifiClient.read();

Serial.print(ch);

buffer += ch;

if(ch == '@' && flagStr == 1){

data = protocol(buffer);

Serial.println(data);

wifiClient.print(data);

flagStr = 0;

break;

}

}

}

}

}

delay(1000);

}

/\* Arduino Protocol

\* Format : @/<RX\_NAME>/<TX\_NAME>/<INSTR1>/<INSTR2>/arg1/arg2/arg3/@

\* if arg==NULL then insert ^

\* <RX\_NAME> : source node

\* <TX\_NAME> : destination node

\* <INSTR1> : C, S, P(control, Sensor, resPonse)

\* <INSTR2> : item or judgement

\* (C, S : F(fan) //Appliance)

\* (P : S, F, M(success, fail, mesasurement))

\* arg1 : function or item

\* (F : 0 == off, 1 == on, 2 == low, 3 == mid, 4 == high, 5 == r\_on

\* 6 == r\_off)

\* (S : T == temperature, H == humidity, E == electricity)

\* return : string (@/<RX\_NAME>/mac\_string/P/<INSTR2>/^/^/^/@)

\*/

String protocol(String msg){

char temp[128];

String source, m;

char \*token;

int flag = 0, flag\_arg = 0;

int fan\_function = -1;

int sensor\_function = -1;

strcpy(temp, msg.c\_str());

//@

token = strtok(temp, "/");

Serial.print("Token : ");

Serial.println(token);

if(strcmp(token, "@")==0){

//<RX\_NAME>

token = strtok(NULL, "/");

Serial.print("Token : ");

Serial.println(token);

if(strcmp(token, mac\_string)==0){

//<TX\_NAME>

token = strtok(NULL, "/");

Serial.print("Token : ");

Serial.println(token);

source = token;

//<INSTR1>

token = strtok(NULL, "/");

Serial.print("Token : ");

Serial.println(token);

if(strcmp(token, "C")==0){

//<INSTR2>

token = strtok(NULL, "/");

Serial.print("Token : ");

Serial.println(token);

if(strcmp(token, "F")==0){

//<arg1>

token = strtok(NULL, "/");

Serial.print("Token : ");

Serial.println(token);

if(strcmp(token, "0")==0){

fan\_function = 0;

flag\_arg = 1;

}

else if(strcmp(token, "1")==0){

fan\_function = 1;

flag\_arg = 1;

}

else if(strcmp(token, "2")==0){

fan\_function = 2;

flag\_arg = 1;

}

else if(strcmp(token, "3")==0){

fan\_function = 3;

flag\_arg = 1;

}

else if(strcmp(token, "4")==0){

fan\_function = 4;

flag\_arg = 1;

}

else if(strcmp(token, "5")==0){

fan\_function = 5;

flag\_arg = 1;

}

//not arg1

else{

flag = 0;

}

//arg2

if(flag\_arg == 1){

token = strtok(NULL, "/");

Serial.print("Token : ");

Serial.println(token);

if(strcmp(token, "^")==0){

//arg3

token = strtok(NULL, "/");

Serial.print("Token : ");

Serial.println(token);

if(strcmp(token, "^")==0){

//@

token = strtok(NULL, "/");

Serial.print("Token : ");

Serial.println(token);

if(strcmp(token, "@")==0){

flag = 1;

}

//not @

else{

fan\_function = -1;

flag = 0 ;

}

}

//not arg3

else{

fan\_function = -1;

flag = 0;

}

}

//not arg2

else{

fan\_function = -1;

flag = 0;

}

}

}

//not <INSTR2>

else{

flag = 0;

}

}

else if(strcmp(token, "S")==0){

//<INSTR2>

token = strtok(NULL, "/");

if(strcmp(token, "F")==0){

//arg1

token = strtok(NULL, "/");

if(strcmp(token, "T")==0){

sensor\_function = 0;

flag\_arg = 1;

}

else if(strcmp(token, "H")==0){

sensor\_function = 1;

flag\_arg = 1;

}

else if(strcmp(token, "E")==0){

sensor\_function = 2;

flag\_arg = 1;

}

//not arg1

else{

flag = 0;

}

//arg2

if(flag\_arg == 1){

token = strtok(NULL, "/");

if(strcmp(token, "^")==0){

//arg3

token = strtok(NULL, "/");

if(strcmp(token, "^")==0){

//@

token = strtok(NULL, "/");

if(strcmp(token, "@")==0){

flag = 1;

}

//not @

else{

sensor\_function = -1;

flag = 0 ;

}

}

//not arg3

else{

sensor\_function = -1;

flag = 0;

}

}

//not arg2

else{

sensor\_function = -1;

flag = 0;

}

}

}

//not <INSTR2>

else{

flag = 0;

}

}

//not <INSTR1>

else{

flag = 0;

}

}

//not mine

else{

flag = 0;

}

}

//not '@'

else{

flag = 0;

}

if(flag==1){

Serial.print("Flag = ");

Serial.println(flag);

if(fan\_function>-1){

switch(fan\_function){

case 0:

digitalWrite(RELAY1, LOW);

delay(50);

digitalWrite(RELAY2, LOW);

delay(50);

digitalWrite(RELAY3, LOW);

delay(50);

digitalWrite(RELAY4, LOW);

delay(50);

break;

case 1:

digitalWrite(RELAY2, LOW);

delay(50);

digitalWrite(RELAY3, LOW);

delay(50);

digitalWrite(RELAY1, HIGH);

delay(50);

break;

case 2:

digitalWrite(RELAY1, LOW);

delay(50);

digitalWrite(RELAY3, LOW);

delay(50);

digitalWrite(RELAY2, HIGH);

delay(50);

break;

case 3:

digitalWrite(RELAY1, LOW);

delay(50);

digitalWrite(RELAY2, LOW);

delay(50);

digitalWrite(RELAY3, HIGH);

delay(50);

break;

case 4:

digitalWrite(RELAY4, LOW);

delay(50);

break;

case 5:

digitalWrite(RELAY4, HIGH);

delay(50);

break;

default:

break;

}

return "@/" + source + "/" + macAddr + "/P/S/^/^/^/@";

}

else if(sensor\_function>-1){

DHT.read(DHT\_PIN);

switch(sensor\_function){

case 0:

m = String((int)DHT.temperature);

break;

case 1:

m = String((int)DHT.humidity);

break;

case 2:

break;

default:

break;

}

return "@/" + source + "/" + macAddr + "/P/S/^/^/^/@";

}

}

else{

return "@/" + source + "/" + macAddr + "/P/F/^/^/^/@";

}

}

5. 참고 내용

1. HEX 데이터 파싱 : <http://geekswithblogs.net/rishi/archive/2006/02/09/68893.aspx>

2. 아두이노 와이파이 쉴드 사용법 : <http://www.arduino.cc/en/Reference/WiFi>

3. 아두이노 와이파이 쉴드 구조 : <http://www.arduino.cc/en/Main/ArduinoWiFiShield>

4. strtok함수 사용법:   
<http://forum.falinux.com/zbxe/?document_srl=408126&mid=C_LIB&sort_index=readed_count&order_type=desc>

5. 아두이노 릴레이 쉴드 사용법 :   
<http://www.dfrobot.com/wiki/index.php/Relay_Shield_for_Arduino_V2.1_(SKU:DFR0144)>

개발 7~8주차 : 아두이노 에어컨 전용 클라이언트

1. 전용 클라이언트를 제작하기 위한 준비(하드웨어)

- 에어컨은 IR통신을 이용한 리모컨이 존재함, 이를 이용하여 IR신호를 통해 에어컨을 제어함

- 기본적인 구성으로 IR송신기, 와이파이 쉴드를 장착함

- IR은 적외선을 이용하여 일정 주파수 내부에 데이터 신호를 넣어 전송하는 반송파 형식을 사용하며 이를 구현하기 위해선 일반적인 pin을 사용할 수 없음(일반적인 핀은 0/1을 일정하게만 표현이 가능함)

- Arduino pin 중에서 ~표시가 되어있는 digital pin은 PWM을 의미하며 PWM은 디지털 데이터를 아날로그처럼 전송해주는 역할을 담당함

- PWM이 표시된 digital pin 3을 IR 송신기의 TX와 연결함

2. 전용 클라이언트를 제작하기 위한 준비(소프트웨어 및 소스코드)

- 소프트웨어로는 펌웨어를 올릴 수 있는 스케치와 IR 송수신을 가능하게 해주는 라이브러리가 필요함

- 스케치 및 라이브러리에 관한 설명은 개발 3주차에 되어있음

- 5주차 내용과 거의 동일함

- 추가적으로 중복된 이름을 가진 다른 라이브러리가 존재하는 경우 필요없는 라이브러리를 다른 폴더로 이동시키면 정상적으로 작동함

- 프로토콜은 5주차의 내용과 거의 일치함

3. 클라이언트 소스 코드

#include <IRremote.h>

#include <IRremoteInt.h>

#include <WiFi.h>

#include <SPI.h>

#include <dht.h>

/\* About pin number

\* if you want another pin number,

\* you could change define number

\* NOTE : pin number must equal the connection pin number

\* Used pin

\* 0, 1 : RX, TX

\* 4 : SS for SD card

\* 7 : Handshake between shield and Arduino

\* 10 : SS for WiFi

\* 11 : MOSI

\* 12 : MISO

\* 13 : SCK

\* 2, 3, 5, 6, 8, 9 : Not used pin number

\*/

#define DHT\_PIN 5

char hexval[16] = {'0', '1', '2', '3', '4', '5', '6',

'7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};

char ssid[] = "diana";

char pass[] = "flcl1300";

int cntSSID, flagStr = 0;

int status = WL\_IDLE\_STATUS;

char server[] = "210.125.31.146";

char server\_inner[] = "192.168.123.123";

WiFiClient wifiClient;

String buffer, macAddr;

byte mac[6];

char mac\_string[13];

dht DHT;

IRsend irsend;

void setup() {

Serial.begin(9600);

if (WiFi.status() == WL\_NO\_SHIELD) {

Serial.println("WiFi shield not present");

// don't continue:

while(true);

}

String fv = WiFi.firmwareVersion();

Serial.println("Firmware Version = " + fv);

cntSSID = WiFi.scanNetworks();

if(cntSSID == -1){

Serial.println("Couldn't get a wifi connection");

while(true);

}

int i=0;

/\*

while (status != WL\_CONNECTED && i<cntSSID) {

Serial.print("Attempting to connect to SSID: ");

Serial.println(WiFi.SSID(i));

switch(WiFi.encryptionType(i)){

case ENC\_TYPE\_NONE:

status = WiFi.begin(WiFi.SSID(i));

break;

case ENC\_TYPE\_CCMP:

status = WiFi.begin(WiFi.SSID(i), pass);

break;

default:

break;

}

i++;

if(i>9)

i=0;

// wait 3 seconds for connection:

delay(3000);

}

\*/

while ( status != WL\_CONNECTED) {

Serial.print("Attempting to connect to open SSID: ");

Serial.println(ssid);

status = WiFi.begin(ssid, pass);

// wait 10 seconds for connection:

delay(1000);

}

Serial.println("WiFi Connection Success!!");

WiFi.macAddress(mac);

for(int j=0;j<6;j++){

mac\_string[j\*2] = hexval[((mac[5-j]>>4) & 0xF)];

mac\_string[(j\*2)+1] = hexval[((mac[5-j]) & 0x0F)];

}

mac\_string[13] = 0;

macAddr = mac\_string;

Serial.println(macAddr);

if(wifiClient.connect(server, 80)){

Serial.println("Server Connection Success");

delay(1000);

}

else{

Serial.println("Server Connection fail");

while(true);

}

}

void loop() {

String data;

if(wifiClient.connected()){

if(wifiClient.available() > 0){

char c = wifiClient.read();

Serial.print(c);

if(c == '@' && flagStr == 0){

buffer = "@";

flagStr = 1;

}

if(flagStr == 1){

delay(500);

while(wifiClient.available()>0){

char ch = wifiClient.read();

Serial.print(ch);

buffer += ch;

if(ch == '@' && flagStr == 1){

data = protocol(buffer);

Serial.println(data);

wifiClient.print(data);

flagStr = 0;

break;

}

}

}

}

}

delay(500);

}

/\* Arduino Protocol

\* Format : @/<RX\_NAME>/<TX\_NAME>/<INSTR1>/<INSTR2>/arg1/arg2/arg3/@

\* if arg==NULL then insert ^

\* <RX\_NAME> : source node

\* <TX\_NAME> : destination node

\* <INSTR1> : C, S, P(control, Sensor, resPonse)

\* <INSTR2> : item or judgement

\* (C : F(fan), A(air conditioner) //Appliance)

\* (P : S, F, M(success, fail, mesasurement))

\* arg1 : function or item

\* (A : 0 == on/off, 1 == up, 2 == down, 3 == indensity, 4 == mode, 5 == time

\* 6 : sleep)

\* (F : 0 == on/off, 1 == indensity, 2 == nature, 3 == time, 4 == rotation,

\* 5 == lamp)

\* (S : T == temperature, H == humidity, E == electricity)

\* return : string (@/<RX\_NAME>/mac\_string/P/<INSTR2>/^/^/^/@)

\*/

String protocol(String msg){

char temp[128];

String source, m;

char \*token;

int flag = 0, flag\_arg = 0;

int fan\_function = -1;

int air\_function = -1;

int sensor\_function = -1;

int time;

strcpy(temp, msg.c\_str());

//@

token = strtok(temp, "/");

if(strcmp(token, "@")==0){

//<RX\_NAME>

token = strtok(NULL, "/");

if(strcmp(token, mac\_string)==0){

//<TX\_NAME>

token = strtok(NULL, "/");

source = token;

//<INSTR1>

token = strtok(NULL, "/");

if(strcmp(token, "C")==0){

//<INSTR2>

token = strtok(NULL, "/");

if(strcmp(token, "A")==0){

//<arg1>

token = strtok(NULL, "/");

if(strcmp(token, "0")==0){

air\_function = 0;

flag\_arg = 1;

}

else if(strcmp(token, "1")==0){

air\_function = 1;

flag\_arg = 1;

}

else if(strcmp(token, "2")==0){

air\_function = 2;

flag\_arg = 1;

}

else if(strcmp(token, "3")==0){

air\_function = 3;

flag\_arg = 1;

}

else if(strcmp(token, "4")==0){

air\_function = 4;

flag\_arg = 1;

}

else if(strcmp(token, "5")==0){

air\_function = 5;

flag\_arg = 1;

}

else if(strcmp(token, "6")==0){

air\_function = 6;

flag\_arg = 1;

}

//not arg1

else{

flag = 0;

}

//arg2

if(flag\_arg == 1){

token = strtok(NULL, "/");

if(strcmp(token, "^")==0){

//arg3

token = strtok(NULL, "/");

if(strcmp(token, "^")==0){

//@

token = strtok(NULL, "/");

if(strcmp(token, "@")==0){

flag = 1;

}

//not @

else{

air\_function = -1;

flag = 0 ;

}

}

//not arg3

else{

air\_function = -1;

flag = 0;

}

}

//not arg2

else{

air\_function = -1;

flag = 0;

}

}

}

//F

else if(strcmp(token, "F")==0){

//<arg1>

token = strtok(NULL, "/");

if(strcmp(token, "0")==0){

fan\_function = 0;

flag\_arg = 1;

}

else if(strcmp(token, "1")==0){

fan\_function = 1;

flag\_arg = 1;

}

else if(strcmp(token, "2")==0){

fan\_function = 2;

flag\_arg = 1;

}

else if(strcmp(token, "3")==0){

fan\_function = 3;

flag\_arg = 1;

}

else if(strcmp(token, "4")==0){

fan\_function = 4;

flag\_arg = 1;

}

else if(strcmp(token, "5")==0){

fan\_function = 5;

flag\_arg = 1;

}

//not arg1

else{

flag = 0;

}

//arg2

if(flag\_arg == 1){

token = strtok(NULL, "/");

if(strcmp(token, "^")==0){

//arg3

token = strtok(NULL, "/");

if(strcmp(token, "^")==0){

//@

token = strtok(NULL, "/");

if(strcmp(token, "@")==0){

flag = 1;

}

//not @

else{

fan\_function = -1;

flag = 0 ;

}

}

//not arg3

else{

fan\_function = -1;

flag = 0;

}

}

//not arg2

else{

fan\_function = -1;

flag = 0;

}

}

}

//not <INSTR2>

else{

flag = 0;

}

}

else if(strcmp(token, "S")==0){

//<INSTR2>

token = strtok(NULL, "/");

if(strcmp(token, "F")==0){

//arg1

token = strtok(NULL, "/");

if(strcmp(token, "T")==0){

sensor\_function = 0;

flag\_arg = 1;

}

else if(strcmp(token, "H")==0){

sensor\_function = 1;

flag\_arg = 1;

}

else if(strcmp(token, "E")==0){

sensor\_function = 2;

flag\_arg = 1;

}

//not arg1

else{

flag = 0;

}

//arg2

if(flag\_arg == 1){

token = strtok(NULL, "/");

if(strcmp(token, "^")==0){

//arg3

token = strtok(NULL, "/");

if(strcmp(token, "^")==0){

//@

token = strtok(NULL, "/");

if(strcmp(token, "@")==0){

flag = 1;

}

//not @

else{

sensor\_function = -1;

flag = 0 ;

}

}

//not arg3

else{

sensor\_function = -1;

flag = 0;

}

}

//not arg2

else{

sensor\_function = -1;

flag = 0;

}

}

}

//not <INSTR2>

else{

flag = 0;

}

}

//not <INSTR1>

else{

flag = 0;

}

}

//not mine

else{

flag = 0;

}

}

//not '@'

else{

flag = 0;

}

if(flag==1){

if(air\_function>-1){

switch(air\_function){

case 0:

irsend.sendNEC(0x01FE39C6, 32);

break;

case 1:

irsend.sendNEC(0x01FE29D6, 32);

break;

case 2:

irsend.sendNEC(0x01FEA956, 32);

break;

case 3:

irsend.sendNEC(0x01FE6996, 32);

break;

case 4:

irsend.sendNEC(0x01FEB946, 32);

break;

case 5:

irsend.sendNEC(0x01FEF906, 32);

break;

case 6:

irsend.sendNEC(0x01FED926, 32);

break;

default:

break;

}

return "@/" + source + "/" + macAddr + "/P/S/^/^/^/@";

}

else if(fan\_function>-1){

switch(fan\_function){

case 0:

irsend.sendMY\_FAN(0x534AAB, 23);

irsend.sendMY\_FAN(0x534AAB, 23);

break;

case 1:

irsend.sendMY\_FAN(0x534AAC, 23);

irsend.sendMY\_FAN(0x534AAC, 23);

break;

case 2:

irsend.sendMY\_FAN(0x534AB2, 23);

irsend.sendMY\_FAN(0x534AB2, 23);

break;

case 3:

irsend.sendMY\_FAN(0x534ACA, 23);

irsend.sendMY\_FAN(0x534ACA, 23);

break;

case 4:

irsend.sendMY\_FAN(0x534CAA, 23);

irsend.sendMY\_FAN(0x534CAA, 23);

break;

case 5:

irsend.sendMY\_FAN(0x534B2A, 23);

irsend.sendMY\_FAN(0x534B2A, 23);

break;

default:

break;

}

return "@/" + source + "/" + macAddr + "/P/S/^/^/^/@";

}

else if(sensor\_function>-1){

DHT.read(DHT\_PIN);

switch(sensor\_function){

case 0:

m = String((int)DHT.temperature);

break;

case 1:

m = String((int)DHT.humidity);

break;

case 2:

break;

default:

break;

}

return "@/" + source + "/" + macAddr + "/P/S/^/^/^/@";

}

}

else{

return "@/" + source + "/" + macAddr + "/P/F/^/^/^/@";

}

}

4. 참고 내용

1. IR송신기 세팅 및 사용방법 및 라이브러리 정보 : <https://learn.sparkfun.com/tutorials/ir-control-kit-hookup-guide?_ga=1.87338648.1428361935.1431942497>

개발 9주차 : 추가사항 및 결과

1. 추가사항

- 선풍기 클라이언트에 Timer를 넣어서 설정한 시간에 정지할 수 있도록 작동함(MsTimer2)

- 기존 IR 라이브러리에 IR 제어가 되는 선풍기의 IR 프로토콜 작성

2. 결과 및 앞으로 개선할 사항

- 통신은 문제 없이 잘 작동함

- 일정 시간이 지나고 작동시키려면 작동이 안되는 경우가 있음(서버가 같은 네트워크에 상주하지 않은 것과 많은 시간이 흘러 연결이 끊어진 것으로 파악)

- Arduino Uno + Wifi Shield의 비용이 많이 들어가므로 대체품인 Arduino Yun을 사용하여 비용 감면

3. 최종 소스코드

- 첨부 파일 참조(선풍기, 에어컨 제어 부분과 IR 라이브러리 참조(라이브러리에 일부 코드 추가))