IoT 프로젝트 최종 보고서

스마트 홈

3+4 조

이름 및 학번 : 변승훈
이의종
전세환
한규현

교과목 : 임베디드시스템

날짜 : 2023.06.01

목차

Ⅰ. 기능 (Function)

디바이스	핀	기능 설명	로컬 인터페이스 방법		웹 인터페이스 방법	
			Display	Control	Display	Control
ESP-8266	D0 D1 D2	보드와 연결된 I/O 데이터를 웹에 전송하거나 받는 기능을 수행한다.			보드에 연결되어 있는 센서들의 값을 표시한다. 온도, 습도, led (On/Off)	On: LED_status=1 Off: LED_status=0
FRDM-K64F		사용자 시간을 제공한다. 사용자는 시간을 조정할 수 있고, 정각마다 스피커에 알림을 요청한다. Thread를 시작하고 scheduling 한다. Mutex 기능을 사용하여 동기화 작업을 한다.				
조이스틱	A2 A3 A4 A5 D4	시스템의 시/분에 해당하는 시간을 조정하는 Interrupt를 제공한다.		Up: H+1 Down: H-1 Left: M-1 Right: M+1 Center: Reset		
LCD	D7 D10 D11	사용자가 시간을 확인할 수 있도록, 현재 시간을 출력한다.	현재 시간을 출력한다. (시:분:초)			

	D13 D12					
Buzzer	D6	정각에 시간이 되면 스피커를 통해 간단한 알람 소리를 울린다.		정각 시간에 알람 소리를 출력한다.		
DHT	A0	스마트 홈에 온도 및 습도를 감지하여 결과 값을 제시한다.		온도 및 습도를 읽어서 esp보드로 통신을 통해 값을 전달한다.	현재 온도 및 습도 값 제시한다.	
LED(Red)	D4	스마트 홈에 LED를 제어하여, 조명을 제어한다.	LED가 켜지고 꺼진다.		LED On/Off 결과를 표시한다.	사용자가 웹을 통해서 LED를 On/Off한다.

Ⅱ. 하드웨어 (Hardware)

1. ESP 8266

	설명
디바이스 설명	HUZZAH ESP8266으로 Adafruit 사에서 판매하는 아래와 같은 기능을 가지는 WIFI MCU 웹 서버 동작을 위해 사용하며, FRDM-K64F에서 명령에 따라 FRDM-K64F와 115,200 baud UART 통신을 한다. FRDM-K64F와 통신 하여 HTTP 프로토콜을 이용해 받은 HTML 데이터를 클라이언트에게 전송한다. 세부사항은 아래와 같다. - Espressif 사의 32-bit 80MHz MPU - NodeMCU (open source Lua based firmware) - 802.11 b/h/n protocol - TCP/IP Protocol stack with DNS support - UART(TX/RX) connection - Reset button, GPIO0 button, Red LED - 9 GPIO pins(#0, #2, #4, #5, #12, #13, #14, #15, #16) can be used for SPI and I2C - 2 x 3-6V inputs(VBat, V+), 3.3V output(3V)
Pin	D0, D1, D2
소프트웨어	ESP 8266은 ESP-CALL 라이브러리의 도움을 받아 KRDM-K64F가 ESP 8266에 명령어를 전송한다. 시리얼 통신을 통해 PC와 ESP 모듈 간의 통신을 설정한다.

2. FRDM K64F – 타이머

	설명
디바이스 설명	FRDM-K64F는 NXP사의 개발 보드로, ARM Cortex-M4 기반의 마이크로컨트롤로를 탑재한 보드이다. FRDM-K64F에서 지원되는 타이머를 활용해 스마트 홈의 시계 기능을 구현한다. Mutex 기능을 지원하여 공유자원에 lock을 걸어 동기화를 시켜준다. thread를 scheduling하여 멀티 쓰래딩 기능을 지원 한다.
소프트웨어	TIMER에서 사용되며, timerInit으로 타이머를 동작시킵니다. _threadTimer 쓰레드에서 timer.read()로 1초 단위의 시간 값을 받습니다. Mutex에서 사용하며 mutex.lock()으로 lock을 걸어주고 mutex.unlock()으로 lock을 해제한다. th1.start()를 사용하여 thread를 시작하고 시작된 thread를 scheduling 해준다.

3. Mbed Application Shield - 조이스틱

		설명
디바이스 설명	설명	Mbed Application Sheild에 연결된 5-Way joystick(5-Way 조이스틱)으로 시간을 조절한다.
	A2(상)	시간에 시(Hour)를 + 1 한다.
	A3(하)	시간에 시(Hour)를 - 1 한다.
	A4(좌)	시간에 분(Min)을 + 1 한다.
	A5(우)	시간에 분(Min)을 - 1 한다.
	D4(중앙)	시간에 초(Sec)를 Reset 한다.
Pin		A2, A3, A4, A5, D4
소프트웨어		1. 5-way joy stick 핀 설정 up, down, left, right, center에 대해 각각 InterruptIn 객체가 생성하고 각각의 핀은 디지털 입력으로 설정되어 있다. 2. 인터럽 서비스 루틴 함수 인터럽트 객체에서 rise가 발생할 때 호출되어 각각의 역할을 수행한다.

4. Mbed Application Shield - LCD

	설명
디바이스 설명	Mbed Application Sheild에 연결된 128x32 LCD로, 스마트 홈의 시간을 Display 해주는 화면이다.
Pin	D7, D10, D11, D13, D12
소프트웨어	C12832 라이브러리를 사용하여 시간을 디스플레이 한다. (0,6)에 Digital Clock!을 출력하고 (0,16)에 Current Time:시:분:초를 출력한다.

5. Buzzer

	설명
디바이스 설명	소리를 발생시키는 부품으로, 짧은 사운드 신호를 생성한다. 정각이 되면 스마트 홈의 사용자가 정각임을 알 수 있도록 알람 사운드를 출력한다.
Pin	D6
소프트웨어	Mbed Application Sheild에 연결된 Buzzer로, 스마트 홈의 시각이 정각을 나타낼 때 Buzzer를 440Hz로 1초간 울려 정각임을 알리는데 사용된다.

6. DHT

	설명
디바이스 설명	Digital Humidity and Temperature의 약자로, 주변 환경의 온도와 습도를 측정하는데 사용되는 센서이다. 전원(VCC), 접지(GND), 데이터(DAT), 3핀으로 구성되어 있으며. VCC와 GND 핀을 통해 센서를 전원에 연결하고, DAT 핀을 통해 데이터를 전송받는다. 센서를 통해 측정된 온도와 습도 데이터를 ESP-8266을 활용해 웹 페이지에 전송한다.
Pin	A0
소프트웨어	Mbed 플랫폼에서 작동하는 온습도 센서인 DHT를 DHT 라이브러리를 사용하여 제어합니다. 스마트홈의 온, 습도를 값을 hum_sensor.ReadTemperature(CELCIUS), hum_sensor.ReadHumidity()를 사용하여 값을 받는다.

7. LED

	설명
디바이스 설명	LED를 On/Off하여 스마트 홈에 조명을 제어한다. 웹페이지에서 led on off 버튼으로 값을 submit한다.
Pin	D4
소프트웨어	DigitalOut 클래스를 사용하여 LED를 제어하는데 필요한 핀을 설정하고 LED의 상태는 thread_light() 함수를 사용하여 업데이트한다.

Ⅲ. 소프트웨어 (Software)

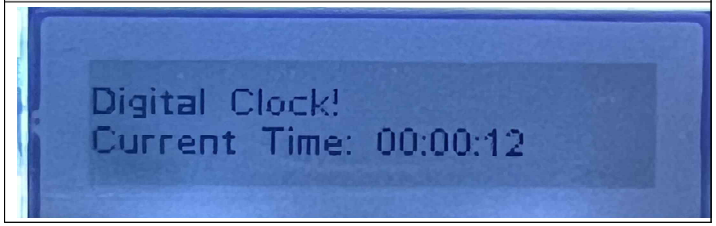
함수명	설명
NodeMCU_PGM http_server[]	NodeMCU(ESP8266)보드를 사용하여 웹 서버를 구축하는 코드로, http_server 배열은 웹 서버를 구성하고 동작 시키기 위한 NodeMCU Lua 스크립트이다. Wi-Fi 연결을 설정하고 웹 서버 설정 및 동작시킨다.
callback_send_uptime	‘cel’과 ‘hum’ 값을 포맷하여 시리얼 통신을 통해 출력한다. cel은 온도 값, hum은 습도 값을 나타낸다. 이 함수는 미리 정의된 온도(cel)와 습도(hum)값을 시리얼 포트를 통해 출력하는 역할을 수행한다. 이러한 함수를 main에서 1초마다 호출하여 온습도 값을 업데이트한다.
ISR_up()	“up” 인터럽트 서비스 루틴 ‘offset’ 변수에서 1분을 증가하는 일을 수행한다. (offset = offset + (60*1000))
ISR_down()	“down” 인터럽트 서비스 루틴 ‘offset’ 변수에서 1분을 감소하는 일을 수행한다. (offset = offset - (60*1000))
ISR_left()	“left” 인터럽트 서비스 루틴 ‘offset’ 변수에서 1시간을 감소하는 일을 수행한다. (offset = offset - (60*60*1000))
ISR_rite()	“rite” 인터럽트 서비스 루틴 ‘offset’ 변수에서 1시간을 증가하는 일을 수행한다. (offset = offset + (60*60*1000))
ISR_center()	현재 시간과 offset 값을 사용하여 현재시간을 조정한다. offset의 합을 1분으로 나눈 나머지만큼 감소시켜 초만 0초로 조정한다.
thread_light()	LED의 상태를 감지하여 제어하는 thread이다, ‘LED_status’값을 웹페이지에서 받아와서 ‘led_r’에 저장하여 상태를 업데이트 한다. mutex를 사용하여 공유 변수에 대한 접근을 보호한다.
thread_clock()	디지털 시계를 표시하는 thread로 현재 시각을 lcd에 뜨우기 위해 사용될 변수 time, h, m, s, ms을 선언한다. up, down, left, rite, center 객체의 rise() 메서드를 호출하여 특정 입력신호의 rise 엣지를 감지하여 해당 인터럽트 서비스 루틴을 실행하는 기능을 한다. LCD화면에 시간을 띄우기 위해 현재 시간을 offset을 통해 계산하여 time 변수에 저장시킨다. 시간에 음수 값이 표시되는 것을 방지 하기 위해 offset 값을 현재 시간과 더한 값이 0보다 작으면 offset을 현재 시간의 음수로 설정하게 했

	다. 시간값을 읽은 값은 ms초 단위로 읽어오기 때문에 이를 계산하여 현재 시간, 초, 분을 각각 변수 h, m, s에 저장합니다. 이를 LCD 화면에 출력하여 현재 시간을 표시한다. 정각마다 부저가 울리게 하기 위해서 분이 0이고 초가 3보다 작을 때 부저를 440Hz 주파수로 설정하고 1초 동안 울리게 하여 정각임을 나타냈다. 3초 보다 작게 한 이유는 thread간의 scheduling 과정에서 최적화 되지 못하여 안울리는 경우를 대비해 설정했다.
thread_read()	온습도 센서에서 값을 읽고 저장하는 thread로 온습도 센서에서 온도와 습도 값을 반복적으로 읽어와서 cel, hum 변수에 저장한다. error가 1인경우 데이터 읽기에 실패하였으므로 잠시 후 다시 시도하고, error가 0인 성공한 경우에는 일정 시간 간격으로 데이터를 갱신한다. 이를 위해서 쓰레드를 반복적으로 실행하게 했다.
main()	main 함수의 동작은 다음 순서와 같이 동작한다. 1. PC와 ESP간의 통신을 위해 통신을 설정. 2. ESP를 재설정.(ESP 초기화) 3. ESP의 noecho 모드 설정. 4. NodeMCU 프로그램을 실행.(http_server 배열에 저장된 NodeMCU 프로그램의 명령어를 차례대로 실행) 5. ESP에서 PC로의 데이터 수신 설정.(ESP에서 PC로의 데이터 수시을 처리하는 인터럽트 콜백 함수를 설정) 6. 1초마다 호출되는 콜백함수 설정.(PC에 현재시간을 전송하는 역학을 수행) 7. 타이머 시작.(프로그램 실행 시간을 측정하는데 사용 됨.) 8. thread 시작.(thread_clock, thread_light, thread_read 3개의 thread를 시작.) 9. 무한루프를 통해 프로그램이 종료되지 않고 계속 실행하도록 설정.

IV. 인터페이스 (Interface)

1. LCD

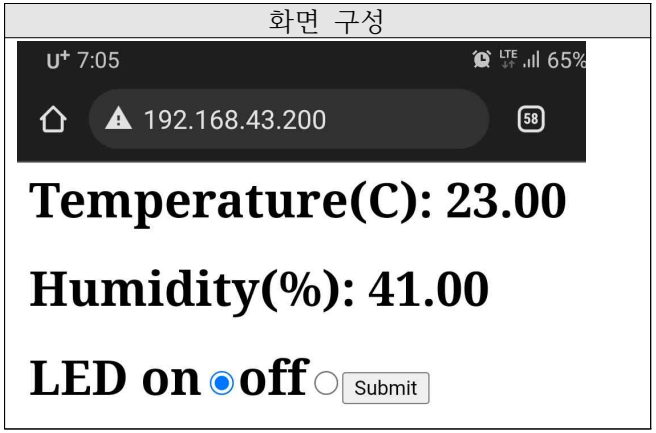
화면 구성



현재 시간을 표시합니다. 현재 시각을 설정하는 데 있어서 app-shield에 5-way joystick을 사용하여 시간을 설정할 수 있다. current time이 정각을 나타낼 때마다 부저를 울려 정각임을 나타낸다.

2. 웹 페이지

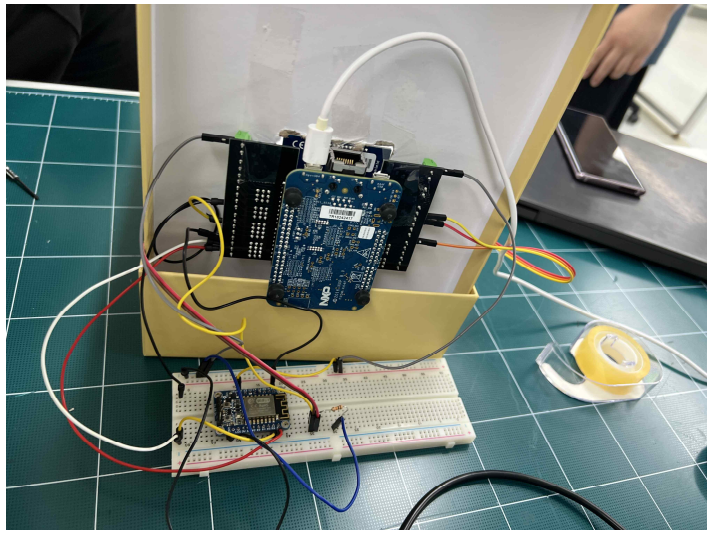
화면 구성



브라우저를 통해 실시간으로 온도, 습도, LED를 제어하는 버튼을 출력한다.
DHT 센서를 통해 입력받은 온, 습도 값을 받아와 웹 브라우저에 출력한다.
LED는 집에 조명을 의미하며 브라우저에서 on, off를 summit하면 LED를 제어할 수 있다.

V. 데모 (Demo)

1) 사진

프로젝트 결과물	회로 구성 사진
	

2) 영상

클락
https://youtu.be/bKISI4MPAGA

온습도, LED
https://youtu.be/VzfGpjw99C8

VI. 소스 코드

1. main.cpp

main.cpp

```
#include "mbed.h"
#include "ESP-call.h"
#include "LM75B.h"
#include "DHT.h"
#include "C12832.h"
#include <string>

//thread
Thread th1, th2, th3, th4, th5;

//mutex
Mutex mutex;

Ticker ticker; // A 1.0 sec ticker

//sensor info
float cel = 0.0;
float hum = 0.0;

//timer offset
long offset = 0;
```

```
Timer timer;

//temp&hum
DHT hum_sensor(A0, SEN11301P);

//lcd
C12832 lcd(D11, D13, D12, D7, D10);

//joystick
InterruptIn sw2(SW2); // sw2 = SW2
InterruptIn sw3(SW3);
InterruptIn up(A2); // up = A1
InterruptIn down(A3);
InterruptIn left(A4);
InterruptIn rite(A5);
InterruptIn center(D4);

//buzzer
PwmOut buzzer(D6);
DigitalOut fan(D8);

//led
DigitalOut led_r(D5);

NodeMCU_PGM http_server[] = {
```

```
"", 1,
"wifi.setmode(wifi.STATION)",1,
"", 1,
"station_cfg={}", 1,
"station_cfg.ssid=₩"freeW"", 1,
"station_cfg.pwd=₩"12345678₩"", 1,
"station_cfg.save=false", 1,
"wifi.sta.config(station_cfg)", 1,
"", 1,
"wifi.sta.connect()", 1,
"", 1,
"print(wifi.sta.status())", 1,
"print(wifi.sta.getip())", 1,
"", 1,
"Red=₩"checked₩"", 1,
"_Red=₩"₩"", 1,
"hum=0.0", 1,
"cel=0.0", 1,
"http_resp = ₩"HTTP/1.0 200 OKWWrWWnContent-Type: text/htmlWWrWWnWWrWWnW"", 1,
"html_main1 = ₩"<h1>Hello</h1>₩"", 1,
"html_main2 = ₩"<h1>Temperature(C): %5.2f</h1>₩"", 1,
"html_main3 = ₩"<h1>Humidity(%%): %5.2f</h1>₩"", 1,
"html_main4 = ₩"<form id=WWW"newWWW" method=WWW"POSTWWW">₩"", 1,
"html_main5 = ₩"<h1>LED ₩"",1,
"html_main6 = ₩"on<input type=WWW"radioWWW" name=WWW"LED_RWWW" value=WWW"onWWW" %s>₩"",1,
```

```
"html_main7 = ₩"off<input type=₩₩₩"radio₩₩₩" name=₩₩₩"LED_R₩₩₩" value=₩₩₩"off₩₩₩" %s>₩"",1,
"html_main8 = ₩"<input type=₩₩₩"submit₩₩₩" value=₩₩₩"Submit₩₩₩"></h1>₩"",1,
"html_main9 = ₩"</form>₩"", 1,
"", 1,
"srv = net.createServer(net.TCP)", 1,
"srv:listen(80, function(conn)", 1,
"    conn:on(₩"receive₩", function(sck, payload)", 1,
"        if string.match(payload, ₩"LED_R=off₩")", 1,
"        then print(₩"^0₩") Red=₩"₩" _Red=₩"checked₩" end", 1,
"        if string.match(payload, ₩"LED_R=on₩")", 1,
"        then print(₩"^1₩") Red=₩"checked₩" _Red=₩"₩" end", 1,
"        print(payload)", 1,
"        sck:send(http_resp)", 1,
"        sck:send(html_main1)", 1,
"        sck:send(string.format(html_main2, cel))", 1,
"        sck:send(string.format(html_main3, hum))", 1,
"        sck:send(html_main4)", 1,
"        sck:send(html_main5)", 1,
"        sck:send(string.format(html_main6, Red))", 1,
"        sck:send(string.format(html_main7, _Red))", 1,
"        sck:send(html_main8)", 1,
"        sck:send(html_main9)", 1,
"        end)", 1,
"    conn:on(₩"sent₩", function(sck) sck:close() end)", 1,
"end)", 1,
```

```
    NULL, 0,
};

// Callback function that sends uptime to ESP
void callback_send_uptime(void)
{
    ESP.printf("cel=%5.2f₩r₩n", cel);
    ESP.printf("hum=%5.2f₩r₩n", hum);
}

void ISR_up(){
    offset = offset + (60*1000);
}

void ISR_down(){
    offset = offset - (60*1000);
}

void ISR_left(){
    offset = offset - (60*60*1000);
}

void ISR_rite(){
    offset = offset + (60*60*1000);
}
```

```
void ISR_center(){
    offset = offset - ((timer.read_ms()+offset) %(1000*60));
}
//light thread
void thread_light(){
    while(true){
        mutex.lock();
        led_r = LED_status;
        mutex.unlock();
    }

}

//clock thread
void thread_clock(){

    long time;
    unsigned char h,m,s,ms;

    up.rise(&ISR_up);
    down.rise(&ISR_down);
    left.rise(&ISR_left);
    rite.rise(&ISR_rite);
    center.rise(&ISR_center);
```

```
lcd.cls();
lcd.locate(0,6);
lcd.printf("Digital Clock!");
while(true){
    mutex.lock();
    time = timer.read_ms();
    if((time + offset) < 0) offset = (-1)*time;
    time = time + offset;

    s = (time / 1000) % 60;
    m = ((time / 1000) / 60) % 60;
    h = ((time / 1000) / (60*60)) % 24;
    lcd.locate(0, 16);
    lcd.printf("Current Time: %02d:%02d:%02d", h, m, s);

    thread_sleep_for(10);
    if(m == 0 && s<3){
        buzzer.period(1.0/440);
        buzzer = 0.5;
        thread_sleep_for(1000);
        buzzer = 0.0;
    }
    mutex.unlock();
}
```

```
}


void thread_read(){
    int error;
    while(true){
        mutex.lock();
        cel = hum_sensor.ReadTemperature(CELCIUS);
        error = hum_sensor.readData();
        if(error == 0){
            hum = hum_sensor.ReadHumidity();
            thread_sleep_for(5000);
        }
        else thread_sleep_for(1000);
        mutex.unlock();
    }
}

int main()
{
    // Config baudrate of PC and ESP
    PC.baud(115200);
    ESP.baud(115200);

    // Reset ESP
```

```
PC.printf("₩r₩nReset ESP...₩r₩n");
ESP_reset();

// Setup ESP noecho mode
PC.printf("Setup ESP noecho...₩r₩n");
ESP_noecho();

// Execute a NodeMCU program
PC.printf("Execute a NodeMCU program...₩r₩n");
ESP_call_multi(http_server);

// Config ESP to PC receive mode
PC.printf("₩r₩nESP receive mode...₩r₩n");
ESP.attach(&ISR_ESP_to_PC, Serial::RxIrq);

// Attach 1.0 sec ticker callback function
ticker.attach(&callback_send_uptime, 1.0);

timer.start();

th1.start(thread_clock);
th2.start(thread_light);
th3.start(thread_read);
```

```
        // Main thread sleeps
        while(1) {
        }
}
```

2. Esp-call.h

Esp-call.h

```
// ================================================================

typedef struct NodeMCU_PGM_STRUCT{
    const char *code;// NodeMCU code
    intdelay;// delay time in 1/10 sec unit
} NodeMCU_PGM;

// ================================================================

externRawSerial PC;// PC = (USBTX, USBRX)
externRawSerial ESP;
extern charbuf;
extern intLED_status;// ESP = (D1=TX, D0=RX)

// ================================================================

extern charESP_recv_buffer[];// ESP receive buffer
extern intESP_recv_buffer_index;// ESP receive buffer index

// ================================================================

void ISR_PC_to_ESP();
void ISR_ESP_to_PC();

void ISR_ESP_to_recv_buffer();

// ================================================================

void ESP_reset(void);

void ESP_echo(void);
```

```
void  ESP_noecho(void);

char  *ESP_call_single(NodeMCU_PGM  pgm);
void  ESP_call_multi(NodeMCU_PGM  pgms[]);

// ========================================================================
```

3.Esp-call.cpp

Esp-call.cpp

```
// ================================================================

#include "mbed.h"
#include "ESP-call.h"

// ================================================================

RawSerial PC(USBTX, USBRX);// PC = (USBTX=TX, USBRX=RX)
RawSerial ESP(D1, D0);// ESP = (D1=TX, D0=RX)
DigitalOut ESP_reset_pin(D2);// ESP_reset_pin = D2

// ================================================================

char ESP_recv_buffer[1028];// ESP receive buffer
intESP_recv_buffer_index;// ESP receive buffer index
charbuf;
intLED_status = 0;
// ================================================================
// ISR for redirecting PC RX to ESP TX

void ISR_PC_to_ESP()
{
    while(PC.readable()) {
        ESP.putc(PC.getc());
}
}

// ================================================================
// ISR for redirecting ESP RX to PC TX

void ISR_ESP_to_PC()
```

```
{
    while(ESP.readable()) {
        if((buf=ESP.getc())!='^')      PC.putc(buf);
        elseLED_status = ESP.getc() - '0';
}
}

// ====================================================================
// ISR for redirecting ESP RX to ESP_recv_buffer[]

void ISR_ESP_to_recv_buffer()
{
    while(ESP.readable()) {
        ESP_recv_buffer[ESP_recv_buffer_index++] = ESP.getc();
}
}

// ====================================================================
// Reset ESP and wait for 1 second

void ESP_reset(void)
{
ESP_reset_pin = 0;
ESP_reset_pin = 1;
    thread_sleep_for(1000);
}

// ====================================================================
// Setup ESP echo/noecho mode
// ====================================================================

void ESP_echo(void)
{
```

```
NodeMCU_PGM uart_setup_echo ={
        "uart.setup(0, 115200, 8, uart.PARITY_NONE, uart.STOPBITS_1, 1)₩r₩n", 1,

    ESP_call_single(uart_setup_echo);
}

void ESP_noecho(void)
{
NodeMCU_PGM uart_setup_noecho ={
        "uart.setup(0, 115200, 8, uart.PARITY_NONE, uart.STOPBITS_1, 0)₩r₩n", 1,

    ESP_call_single(uart_setup_noecho);
}

// ================================================================
// Send NodeMCU_PGM to ESP and execute

char *ESP_call_single(NodeMCU_PGM pgm)
{
                        x = 0;
    ESP.attach(&ISR_ESP_to_recv_buffer, Serial::RxIrq);
    ESP.printf("%s₩r₩n", pgm.code);
    thread_sleep_for((pgm.delay) * 100);
    ESP.attach(NULL, Serial::RxIrq);
    ESP_recv_buffer[ESP_recv_buffer_index] = '₩0';
    returnESP_recv_buffer;
}

void ESP_call_multi(NodeMCU_PGM pgms[])
{
    inti;
    char *p;
```

```
    for(i = 0; pgms[i].code != NULL; i++) {
        PC.printf("%s₩r₩n", pgms[i].code);
p = ESP_call_single(pgms[i]);
        PC.printf("%s", p);
}
}

// ========================================================================
```

Ⅷ. 참고 문헌

1. 신동하, "1 Embedded systems(임베디드 시스템)", 상명대학교, 2023.
2. 신동하, "3 Using ARM Mbed(ARM Mbed 사용)", 상명대학교, 2023.
3. 신동하, "4 Digital input and output(디지털 입출력)", 상명대학교, 2023.
4. 신동하, "5 Analog input and output(아날로그 입출력)", 상명대학교, 2023.
5. 신동하, "6 Pulse-width modulation(펄스-폭 변조)", 상명대학교, 2023.
6. 신동하, "8 Mbed application shield(Mbed 응용 쉴드)", 상명대학교, 2023.
7. 신동하, "9 Real-time programming(실시간 프로그래밍)", 상명대학교, 2023.
8. 신동하, "10 Internet communication(인터넷 통신)", 상명대학교, 2023.