강의명: 임베디드 시스템
숙제 번호: 6
숙제 제목: Internet communication(인터넷 통신)
학생 이름: 한규현(팀장), 전세환

1. 프로그램 http-server-temperature

1.1 프로그램 코드 쓰기

```
#include "mbed.h"
#include "ESP-call.h"
#include "LM75B.h"
Ticker ticker; // A 1.0 sec ticker
uint64_t uptime =0; // Current uptime
float cel =0.0;
LM75B temp(D14, D15);
NodeMCU_PGM http_server[] = {
    "", 1,
    "wifi.setmode(wifi.STATION)",1,
    "", 1,
    "station_cfg={}", 1,
    "station_cfg.ssid=\"free\"", 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,
    "uptime=0", 1,
    "cel=0.0", 1,
          "http_resp     =     \"HTTP/1.0     200     OK\\r\\nContent-Type:
text/html\\r\\n\\r\\n\"", 1,
    "html_main1 = \"<h1>Hello, NodeMCU!</h1>\"", 1,
    "html_main2 = \"<h1>Temperature(C): %5.2f</h1>\"", 1,
    "html_main3 = \"<h1>Uptime(secs): %d</h1>\"", 1,
    "html_main4 = \"<h1>Bye!</h1>\"", 1,
    "", 1,
```

```
    "srv = net.createServer(net.TCP)", 1,
    "srv:listen(80, function(conn)", 1,
    "   conn:on(\"receive\", function(sck, payload)", 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, uptime))", 1,
    "       sck:send(html_main4)", 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)
{
    uptime = uptime +1;
    ESP.printf("cel=%5.2f\r\n", cel);
    ESP.printf("uptime=%llu\r\n", uptime);
}


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);

    // Main thread sleeps
    while(1) {
        cel =temp.read();
    }
}
```

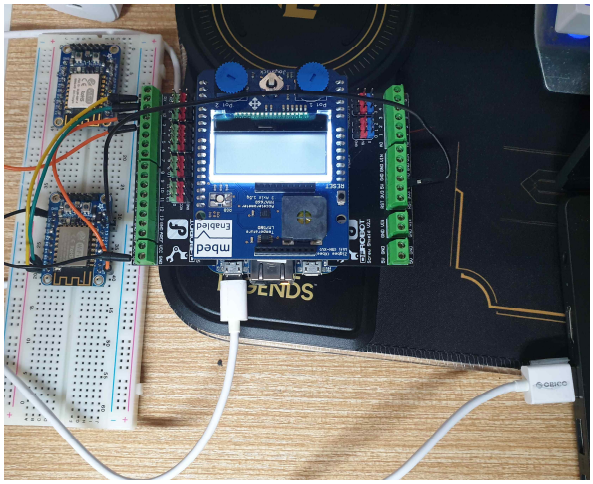
1.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

Ticker ticker는 1초 주기로 콜백 함수를 호출하기 위한 Ticker 객체이다. uint64_t uptime은 현재 디바이스의 uptime을 저장하는 변수로 callback_send_uptime에서 매 초마다 1씩 증가한다. float cel은 LM75B 온도 센서에서 측정한 썹씨 온도 값을 저장하는 변수로 main에서 LM75B 객체를 생성하고, while 루프 안에서 온도 값을 읽어와서 cel 변수에 저장한다. ticker.attach(&callback_send_uptime, 1.0)으로 1초마다 callback_send_uptime 함수를 호출하도록 Ticker 객체를 설정한다.

위 코드는 시리얼 통신을 설정하고 PC와 ESP를 115200 baudrate로 설정하고 reset과 noecho모드 설정을 완료한다. 이후 서버를 실행하고 데이터를 PC로 전송하는 ISR_ESP_to_PC를 데이터를 보낼때마다 실행한다. 이후 ticker.attach(&callback_send_uptime, 1.0)을 통해 1초마다 callback 함수가 호출되게 한다. while문에서는 LM75B를 읽어 cel 변수에 저장하고 HTML 코드에 온도를 출력하도록 한다.

callback_send_uptime 함수는 Ticker 객체에서 1초마다 호출되는 콜백함수로, 전역 변수 uptime을 1씩 증가시키고, 현재 온도 값을 ESP 모듈로 시리얼 통신을 보내는 역할을 한다. ESP 모듈의 printf() 함수를 사용하여 시리얼 통신을 하며, 현재 온도와 uptime을 cel과 uptime 문자열과 함께 출력한다. 이렇게 출력된 문자열은 ESP 모듈로 전송되어 NodeMCU 프로그램에서 처리된다.

1.3 하드웨어 구성 사진 첨부하기



1.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기
https://youtu.be/raRqdm_Rxq4

2. 프로그램 http-server-led

2.1 프로그램 코드 쓰기

main.cpp

```
#include "mbed.h"
#include "ESP-call.h"
#include "LM75B.h"
PwmOut led_r(PTA2);
NodeMCU_PGM http_server_l[] = {
    "", 1,
    "wifi.setmode(wifi.STATION)", 1,
    "", 1,
    "station_cfg={}", 1,
    "station_cfg.ssid=\"free\"", 1,
    "station_cfg.pwd=\"12345678\"", 1,
    "station_cfg.save=false", 1,
    "wifi.sta.config(station_cfg)", 1,
    "", 1,
    "wifi.sta.connect()", 80,
    "", 1,
    "print(wifi.sta.status())", 1,
    "print(wifi.sta.getip())", 1,
    "", 1,
```

```
    "Red=\"checked\"", 1,
    "_Red=\"\"", 1,
    "uptime=0", 1,
          "http_resp    =    \"HTTP/1.0    200    OK\\r\\nContent-Type:
text/html\\r\\n\\r\\n\"",1,
    "html_main1 = \"<h1>Hello, NodeMCU!</h1>\"",1,
    "html_main2 = \"<form id=\\\"new\\\" method=\\\"POST\\\">\"", 1,
    "html_main3 = \"<h1>LED \"",1,
      "html_main4  =  \"on<input  type=\\\"radio\\\"  name=\\\"LED_R\\\"
value=\\\"on\\\" %s>\"",1,
     "html_main5  =  \"off<input  type=\\\"radio\\\"  name=\\\"LED_R\\\"
value=\\\"off\\\" %s>\"",1,
               "html_main6      =      \"<input      type=\\\"submit\\\"
value=\\\"Submit\\\"></h1>\"",1,
    "html_main7 = \"<h1>Bye!</h1>\"",1,
    "html_main8 = \"</form>\"", 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(\"^1\") Red=\"\" _Red=\"checked\" end", 1,
    "if string.match(payload, \"LED_R=on\")", 1,
    "then print(\"^0\") Red=\"checked\" _Red=\"\" end", 1,
    "print(payload)",1,
    "sck:send(http_resp)",1,
    "sck:send(html_main1)",1,
    "sck:send(html_main2)",1,
    "sck:send(html_main3)",1,
    "sck:send(string.format(html_main4, Red))",1,
    "sck:send(string.format(html_main5, _Red))",1,
    "sck:send(html_main6)",1,
    "sck:send(html_main7)",1,
    "sck:send(html_main8)",1,
    "end)",1,
    "conn:on(\"sent\", function(sck) sck:close() end)",1,
    "end)",1,
    NULL, 0,
};
int main()
```

```
{
    PC.baud(115200);
    ESP.baud(115200);
    led_r =0;
    PC.printf("\r\nReset ESP...\r\n");
    ESP_reset();
    PC.printf("Setup ESP noecho...\r\n");
    ESP_noecho();
    PC.printf("Execute a NodeMCU program...\r\n");
    ESP_call_multi(http_server_1);
    PC.printf("\r\nESP receive mode...\r\n");
    ESP.attach(&ISR_ESP_to_PC, Serial::RxIrq);
    while(1) {
        led_r = LED_status;
    }
}
```

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
int ESP_recv_buffer_index;          // ESP receive buffer index
char buf;
int LED_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);
        else                      LED_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)
{
    ESP_recv_buffer_index =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';
    return ESP_recv_buffer;
}
void ESP_call_multi(NodeMCU_PGM pgms[])
{
    int i;
    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);
    }
}
// ================================================================================
```

ESP-call.h

```
// ================================================================================
typedef struct NodeMCU_PGM_STRUCT {
    const char *code;               // NodeMCU code
    int delay;                      // delay time in 1/10 sec unit
} NodeMCU_PGM;
// ================================================================================
extern RawSerial PC;                // PC = (USBTX, USBRX)
extern RawSerial ESP;               // ESP = (D1=TX, D0=RX)
// ================================================================================
```

```
extern char ESP_recv_buffer[];      // ESP receive buffer
extern int ESP_recv_buffer_index;   // ESP receive buffer index
extern char buf;
extern int LED_status;
// ========================================================================
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[]);
// ========================================================================
```

2.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

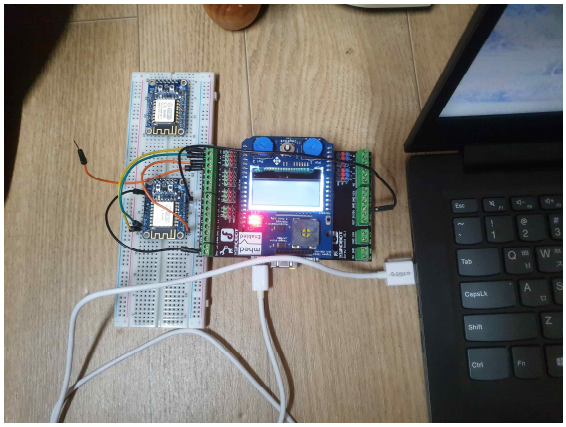
PwmOut led_r(PTA2)를 작성해 PTA2핀에 연결된 LED를 제어하는 PwmOut 객체를 선언했다.

NodeMCU_PGM http_server[]에서는 lua 스크립트이며 클라이언트가 요청을 보내면 LED를 제어할 수 있도록 한다. radio 버튼과 Submit 버튼을 만들었다. 또한 collectgarbage() 함수를 호출하여 메모리 누수를 방지하였다.

main 문 안에는 led_r을 0으로 초기화 하고, 무한 루프를 돌면서 led_r에 LED_status에 따라 led On/Off가 반영되도록했다.

ESP-call.cpp와 ESP-call.h에 char buf와 LED_status를 선언하여 ESP에서 읽어오는 값이 on일때 0을 전송하고 off일때 1을 전송한다. frdm보드에서는 읽어오는 값을 LED_status에 저장하여 led를 제어했다.

2.3 하드웨어 구성 사진 첨부하기



2.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기
https://youtu.be/OGwJiMGoQ3w

끝.