7주차 실습 레포트

2018125020 류호원(소프트웨어학과)

Task07-c 코드 + 코드 설명ㄴ

```
#define SWAP 0
#define EEPROM_SIZE 200
#include<WiFi.h>
#include"time.h"
#include<EEPROM.h>


# if SWAP
const char * ssid = "ESP_05";
const char * password = "123456789";
#else
const char* ssid = "hotspot";
const char* password = "asdf1234";
#endif


const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 3600 * 9; // 3600
const int daylightOffset_sec = 0; // 3600
```

```
WiFiServer server(80);

String header;

String output16State = "off";
String output17State = "off";

const int output16 = 16;
const int output17 = 17;

unsigned long currentTime = millis();
unsigned long previousTime = 0;

const long timeoutTime = 2000;

int h_time = 0;
int m_time = 0;
int s_time = 0;
int duration = 0;
int pre_time = 0;
int alram_switch = 0;
int play_switch = 0;

const int ledChannel = 0;
const int resolution = 8;
const int buzPin = 23;
```

```
const int duty = 20;



int nFrq[] = {262, 277, 294, 311, 330, 349, 370, 392, 415, 440, 466, 494, 523};
int nDur[] = { 2000, 1500, 1000, 750, 500, 375, 250 };

void playNote(int note, int dur) {

  if (note == -1) {
    ledcSetup(ledChannel, 0, resolution);
    ledcWrite(ledChannel, 0);
  }
  else {
    ledcSetup(ledChannel, nFrq[note], resolution);
    ledcWrite(ledChannel, duty);
  }
  Serial.println(String(note) + "," + String(dur));
  delay(nDur[dur]);
}




void setup() {
  Serial.begin(115200);
  EEPROM.begin(512);
```

```
    ledcAttachPin(buzPin, ledChannel);
    pinMode(output16, OUTPUT);
    pinMode(output17, OUTPUT);

    digitalWrite(output16, LOW);
    digitalWrite(output17, LOW);

#if SWAP
    WiFi.softAP(ssid, password);
    IPAddress IP = WiFi.softAPIP();
    Serial.print("AP IP address: ");
    Serial.print(IP);

#else

    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
```

```
#endif
  server.begin();
}
void printLocalTime(WiFiClient client = 0)
{
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time");
    return;
  }
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");

  Serial.println("Year: " + String(timeinfo.tm_year + 1900) + ", Month: " + String(timeinfo.tm_mon + 1));
  client.println("<script>var totalTime=" + String(timeinfo.tm_hour * 3600 + timeinfo.tm_min * 60 + timeinfo.tm_sec) +
                 ";                setInterval(function(){totalTime++; document.getElementById('timer').innerHTML='NowTime: '+Math.floor(totalTime/3600) + ':' + Math.floor(totalTime%3600/60) + ':' + totalTime%3600%60;}, 1000);</script>");
  client.println(&timeinfo, "<h2 id='timer'>NowTime: %H:%M:%S</h2>");
  client.println("Year: " + String(timeinfo.tm_year + 1900) + ", Month: " + String(timeinfo.tm_mon + 1));

}

void loop() {
```

```
  WiFiClient client = server.available(); // Listen for incoming clients
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time");
    return;
  }

  if ((timeinfo.tm_hour * 3600 + timeinfo.tm_min * 60 + timeinfo.tm_sec >= pre_time) && alram_switch == 1) {
    for (int i = 1; i < 512; i += 2) {
      playNote(EEPROM.read(i), EEPROM.read(i + 1));
    }
    ledcSetup(ledChannel, 0, resolution);
    ledcWrite(ledChannel, 0);
    alram_switch = 0;
  }
// 만약 현재 시간이 알람+현재 시간을 더한 값(pre_time) 보다 커졌을 때 그리고 알람스위치(시,분,초 값을 받았을때) eeprom에서 저장된 음과 지속시간을 읽어 들여서 음악을 재생한다
  if (client) { // If a new client connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client."); // print a message out in the serial port
    String currentLine = ""; // make a String to hold incoming data from the client
    while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while the client's connected
```

```
currentTime = millis();

if (client.available()) { // if there's bytes to read from the client,

  char c = client.read(); // read a byte, then
  Serial.write(c); // print it out the serial monitor
  header += c;
  if (c == '₩n') { // if the byte is a newline character
    // if the current line is blank, you got two newline characters in a row.
    // that's the end of the client HTTP request, so send a response:
    if (currentLine.length() == 0) {
      // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
      // and a content-type so the client knows what's coming, then a blank line:
      client.println("HTTP/1.1 200 OK");
      client.println("Content-type:text/html");
      client.println("Connection: close");
      client.println();
      // turns the GPIOs on and off
      if (header.indexOf("GET /16/on") >= 0) {
        Serial.println("GPIO 16 on");
        output16State = "on";
        digitalWrite(output16, HIGH);
      } else if (header.indexOf("GET /16/off") >= 0) {
        Serial.println("GPIO 16 off");
        output16State = "off";
        digitalWrite(output16, LOW);
```

```
} else if (header.indexOf("GET /17/on") >= 0) {
   Serial.println("GPIO 17 on");
   output17State = "on";
   digitalWrite(output17, HIGH);

} else if (header.indexOf("GET /17/off") >= 0) {
   Serial.println("GPIO 17 off");
   output17State = "off";
   digitalWrite(output17, LOW);
} else if (header.indexOf("GET /input_time") >= 0) {
   int index_end_melody = 25;
   int count = 0;
   String h = "";
   String m = "";
   String s = "";
   for (int i = header.indexOf('H') + 2; i < header.indexOf('M') - 1; i++) {
      h = h + header[i];
   }
   for (int i = header.indexOf('M') + 2; i < header.indexOf('S') - 1; i++) {
      m = m + header[i];
   }
   for (int i = header.indexOf('S') + 2; i < header.indexOf(' ', 4); i++) {
      s = s + header[i];
   }
   h_time = h.toInt();
   m_time = m.toInt();
```

```
          s_time = s.toInt();
          duration = h_time * 3600 + m_time * 60 + s_time;
          alram_switch = 1;
        }
    // 시간이 입력되면 url이 input_time으로 이동함으로 GET /input_time 이 있으면 시, 분, 초 에 해당하는 값을 모두 더하여 총 알람시간인 duration으로 만든다
        header = "";

        // Display the HTML web page
        client.println("<!DOCTYPE html><html>");
        client.println("<head><meta name=₩"viewport₩" content=₩"width=device-width, initial-scale=1₩">");
        client.println("<link rel=₩"icon₩" href=₩"data:,₩">");
        // CSS to style the on/off buttons
        // Feel free to change the background-color and font-size attributes to fit your preferences
        client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
        client.println(".button { background-color: #4CAF50;border: none; color: white; padding: 16px 40px;");
        client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
        client.println(".button2 {background-color: #555555;}");
        client.println("input{width: 40%;}</style></head>");
        // Web Page Heading
        client.println("<body><h1>ESP32 Web Server</h1>");
        // Display current state, and ON/OFF buttons for GPIO 16
        client.println("<p>GPIO 16 - State " + output16State + "</p>");
```

```
            // If the output16State is off, it displays the ON button
            if (output16State == "off") {
              client.println("<p><a href=₩"/16/on₩"><button class=₩"button₩">ON</button></a></p>");
            } else {
              client.println("<p><a href=₩"/16/off₩"><button class=₩"button button2₩">OFF</button></a></p>");
            }
            // Display current state, and ON/OFF buttons for GPIO 17
            client.println("<p>GPIO 17 - State " + output17State + "</p>");
            // If the output17State is off, it displays the ON button
            if (output17State == "off") {
              client.println("<p><a href=₩"/17/on₩"><button class=₩"button₩">ON</button></a></p>");
            } else {
              client.println("<p><a href=₩"/17/off₩"><button class=₩"button button2₩">OFF</button></a></p>");
            }



            Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");


            Serial.println("Year: " + String(timeinfo.tm_year + 1900) + ", Month: " + String(timeinfo.tm_mon + 1));
            client.println("<script>var totalTime=" + String(timeinfo.tm_hour * 3600 + timeinfo.tm_min * 60 + timeinfo.tm_sec) +
```

"; setInterval(function(){totalTime++;
document.getElementById('timer').innerHTML='NowTime: '+Math.floor(totalTime/3600) + ':' + Math.floor(totalTime%3600/60) + ':' + totalTime%3600%60;}, 1000);</script>");

client.println(&timeinfo, "<h2 id='timer'>NowTime: %H:%M:%S</h2>");

client.println("Year: " + String(timeinfo.tm_year + 1900) + ", Month: " + String(timeinfo.tm_mon + 1));

client.println("<label for =₩"fname₩"> Melody</label>");

client.println("<form action = ₩"/input_time₩">");

client.println("<input type=₩"text₩" name =₩"H₩">");

client.println("<input type=₩"text₩" name =₩"M₩">");

client.println("<input type=₩"text₩" name =₩"S₩">");

client.println("<input type = ₩"submit₩"></form></body></html>");

// input text의 형태로 시, 분, 초에 해당하는 값을 입력 받는다

// The HTTP response ends with another blank line

client.println();

// Break out of the while loop

if (alram_switch == 1) {

pre_time = timeinfo.tm_hour * 3600 + timeinfo.tm_min * 60 + timeinfo.tm_sec + duration;

}

break;

// alarm_switch가 1이라면(시,분,초에 해당하는 값을 받았다면) pre_time에 알람시간과 현재 시간을 더한 값을 넣어준다

} //** if (currentLine.length() == 0) {

else { // if you got a newline, then clear currentLine

```
                currentLine = "";
              }
            } //** if (c == '₩n') {
            else if (c != '₩r') { // if you got anything else but a carriage return character,
              currentLine += c; // add it to the end of the currentLine
            }
          } //* if (client.available()) {
        } //** while
        // Clear the header variable
        header = "";
        // Close the connection
        client.stop();
        Serial.println("Client disconnected.");
        Serial.println("");
    } //** if (client) {
} //** loop() {
```