```
/*

  温湿度表示器
    有機液晶表示器(128x64dot)を使用
    Wi-Fi対応(CPUはESP-WROOM-02)
  ――――――――――――――――――――
      2017.10.11 Susumu Shikata

*/
//  ESP-WROOM-02に関する定義
#include <ESP8266WiFi.h>
extern "C" {
#include "user_interface.h"
}
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

//  I2C関連デバイス定義
#include "SparkFun_Si7021_Breakout_Library.h"   //  SI7021温湿度センサ
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>                       //
OLED表示用グラフィックライブラリ
#include <Fonts/FreeSerifBold18pt7b.h>
//  フォント指定(サンセリフ／ボールド)
#include <ESP_Adafruit_SSD1306.h>               //  OLED表示ドライバ
#define OLED_RESET 4
//  SI7021からの読み取り値格納用グローバル変数
float humidity = 0;                             //  湿度
float tempf = 0;                                //  温度(華氏℉)
float tempc = 0;                                //  温度(摂氏℃)

//  OLEDの初期化
Adafruit_SSD1306 display(OLED_RESET);

//Create Instance of HTU21D or SI7021 temp and humidity sensor and MPL3115A2
barrometric sensor
Weather sensor;

const char* ssid = "ssid";                 //  Wi-Fi A/P
const char* password = "password";        //  同上用パスワード

ESP8266WebServer server(80);                    //  サーバの接続ポート番号



void handleRoot() {
  char tmp[400];
```

```
  snprintf ( tmp, 400,

             "<html>\
  <head>\
    <meta http-equiv='refresh' content='5'/>\
    <meta charset='UTF-8'>\
    <title>Get HTU21 status</title>\
    <style>\
      body { background-color: #cccccc; font-family: Arial, Helvetica,
Sans-Serif; Color: #000088; }\
    </style>\
  </head>\
  <body>\
    <h2>Weather condition now</h2>\
    <h1>気温 %02d.%02d℃ 湿度 %02d.%02d%</h1>\
  </body>\
</html>",
             int(tempc), int( (tempc - floor( tempc)) * 100 ), int(humidity ),
int( (humidity - floor( humidity )) * 100 )
           );
  server.send ( 200, "text/html", tmp );
}

void handleNotFound() {
  String message = "File Not found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) {
    message += " NAME:" + server.argName(i) + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
}

void setup() {
  int i;

  Wire.begin( 5, 4 );                                 //
(SDA,SCL):ESP8266(IO5)-OLED(SDA),(IO4)-OLED(SCL)
  display.begin(SSD1306_SWITCHCAPVCC, 0x78 >> 1); // OLED ADDRESS
  display.clearDisplay();                             // Clear the buffer.

  //  温湿度センサを初期化
```

```
  sensor.begin();

  WiFi.begin ( ssid, password );

  // サーバへの接続待ち
  for ( i = 0; i <= 20; i++ ) {
    if ( WiFi.status() == WL_CONNECTED ) break;   //  接続タイムアウトなら脱出
    delay ( 500 );
  }
  //  Wi-Fiへの接続が成功したらサーバーを起動する。
  if ( i < 20 ) {
    server.on ( "/", handleRoot );
    server.on ( "/inline", []() {
      server.send ( 200, "text/plain", "this works as well" );
    } );
    server.onNotFound ( handleNotFound );

    server.begin();
  }
}

void loop() {
  int i;

  getWeather();
  for ( i = 0; i <= 200; i++ ) {
    server.handleClient();
    delay( 10 );
  }

  display.clearDisplay();       //  全画面消去

  //  ローカルIPアドレスの表示
  display.setCursor( 20, 57 );
  if ( WiFi.status() == WL_CONNECTED )  display.println ( WiFi.localIP()
);       //  DHCPで得られたローカルIP
  else display.println ( "WiFi not found." );           //
接続できなかった時の表示
  //  温度表示
  display.setFont(&FreeSerifBold18pt7b);
  display.setTextColor(WHITE);
  display.setCursor(20, 23);
  display.print( tempc );
  display.setFont();
  display.print(" C");
  //  湿度表示
  display.setFont(&FreeSerifBold18pt7b);
  display.setCursor(20, 52);
```

```
  display.print( humidity );
  display.setFont();
  display.print(" %");
  display.display();            //  表示を実行
}

//---------------------------------------------------------------
//  温湿度センサからデータを読み込む
//---------------------------------------------------------------
void getWeather()
{
  // Measure Relative Humidity from the HTU21D or Si7021
  humidity = sensor.getRH();
  tempc = sensor.getTemp();
  // Temperature is measured every time RH is requested.
  // It is faster, therefore, to read it from previous RH
  // measurement with getTemp() instead with readTemp()
}
```