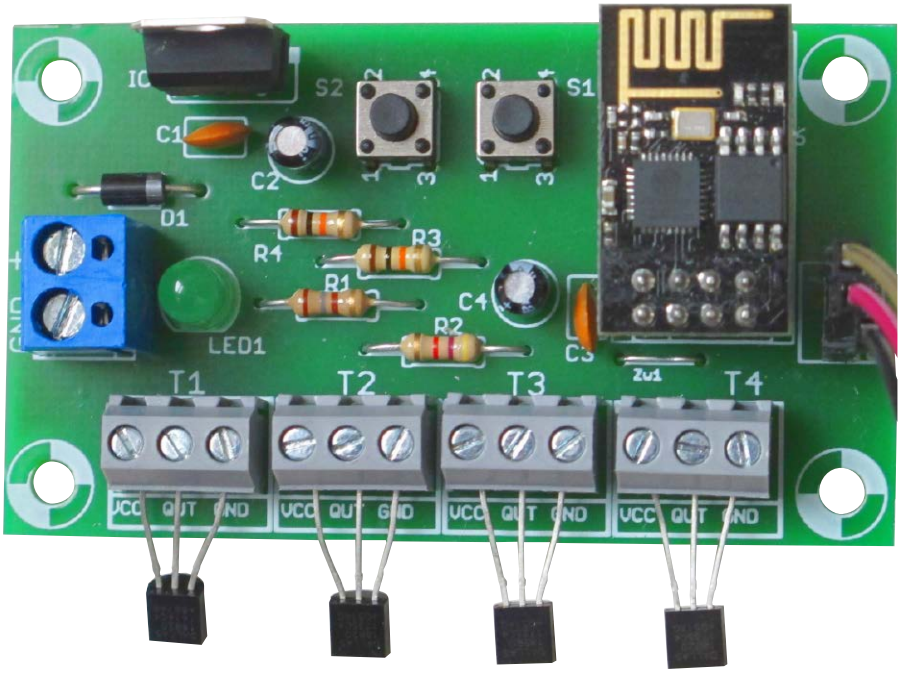
4-kanałowy termometr z interfejsem Wi-Fi

Termometr elektroniczny z 4-punktowym odczytem temperatury przez lokalną sieć Wi-Fi. Wyniki pomiarów są prezentowane z pomocą aplikacji Termik4Temp na wyświetlaczu telefonu komórkowego lub tabletu z systemem operacyjnym Android. Jako czujniki temperatury zastosowano popularne układy cyfrowe DS18B20.

***Rekomendacje:** dzięki możliwości wprowadzania ustawień widoków w aplikacji termometr może znaleźć liczne zastosowania w wielu dziedzinach życia.*



DODATKOWE MATERIAŁY DO POBRANIA ZE STRONY:

www.media.avt.pl

W ofercie AVT*

AVT-5623

Podstawowe informacje:

- Popularny moduł Wi-Fi z ESP8266.
- Napięcie zasilające 5...9 V DC/0,5 A.
- Cztery czujniki DS18B20 z interfejsem 1-Wire.
- Obsługa przez aplikację Termik4Temp pracującą pod kontrolą systemu Android.

Projekty pokrewne na www.media.avt.pl:

AVT-5566	THPStation – rozbudowany termometr z Wi-Fi (EP 1/2017)
AVT-1941	8-kanałowy termometr z I²C (EP 1/2017)
AVT-5573	Nieskomplikowany termometr-rejestrator (EP 11/2016)
AVT-5535	Termometr 2-kanałowy z interfejsem Bluetooth (EP 4/2016)
AVT-5518	Termometr bezprzewodowy (EP 11/2015)
AVT-1863	Termometr z interfejsem Bluetooth (EP 8/2015)
AVT-1790	Termometr XXL (EP 2/2014)
AVT-5489	8-kanałowy termometr z alarmem i wyświetlaczem LCD (EP 11/2013)

*** Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KITem (z ang. zestaw). Zestaw w wersji [**B**] zawiera elementy elektroniczne (w tym [UK] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.
Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [**C**] zmontowany, uruchomiony i przetestowany zestaw [**B**] (elementy wlutowane w płytkę PCB)
- wersja [**A**] płytka drukowana bez elementów i dokumentacja

Kity w których występuje układ scalony wymagający zaprogramowania, posiadają następujące dodatkowe wersje:
- wersja [**A+**] płytka drukowana [**A**] + zaprogramowany układ [**UK**] i dokumentacja
- wersja [**UK**] zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! ***http://sklep.avt.pl***

Od lat starałem się zbudować elektroniczny termometr do pomiaru temperatury wewnętrznej, zewnętrznej i... nawet kilka zbudowałem. Niestety, zawsze brakowało mi postawienia przysłowiowej kropki nad i, czyli wykonania estetycznej obudowy, która przypadłaby do gustu pozostałym domownikom. Ostatecznie z braku determinacji i akceptacji rodziny (z uwagi na walory estetyczne obudowy) większość projektów lądowała na strychu w „skrzyni skarbów" i tak z sześć razy...

Po latach usilnych starań udało mi się jednak zbudować termometr bez konieczności wykonywania obudowy, który przypadł do gustu wszystkim domownikom. Może on pracować z czterema czujnikami temperatury, a odpowiednio do ich liczby, w specjalnej aplikacji możemy wybrać odpowiadający im widok oraz nadać unikatową nazwę.

Budowa

Schemat ideowy termometru pokazano na **rysunku 1**. Wykonano go w oparciu o moduł ESP8266-01 realizujący aplikację użytkownika. To dzięki temu „maleństwu" tak wiele skomplikowanych działań jest możliwych do realizacji w tak prosty i wręcz banalny sposób, i to bez stosowania dodatkowych układów elektronicznych. Moduł ten szeroko opisywany był na łamach Elektroniki Praktycznej, zatem odsyłam do wcześniejszych artykułów i publikacji na jego temat.

Jako czujniki temperatury zastosowano termometry z interfejsem 1-Wire typu DS18B20. Są one dołączane do złączy oznaczonych T1... T4. Wyjścia czujników są polaryzowane za pomocą rezystora R2 i dołączone do wyprowadzenia GPIO2 modułu ESP8266. Kondensatory C3 i C4 pełnią funkcję filtra przeciwzakłóceniowego. Dioda prostownicza D1 stanowi zabezpieczenie przed odwrotną polaryzacją napięcia zasilającego. Układ LF33CV stabilizuje napięcie zasilania na poziomie 3,3 V. Kondensatory C1 i C2 to kolejne filtry przeciwzakłóceniowe.

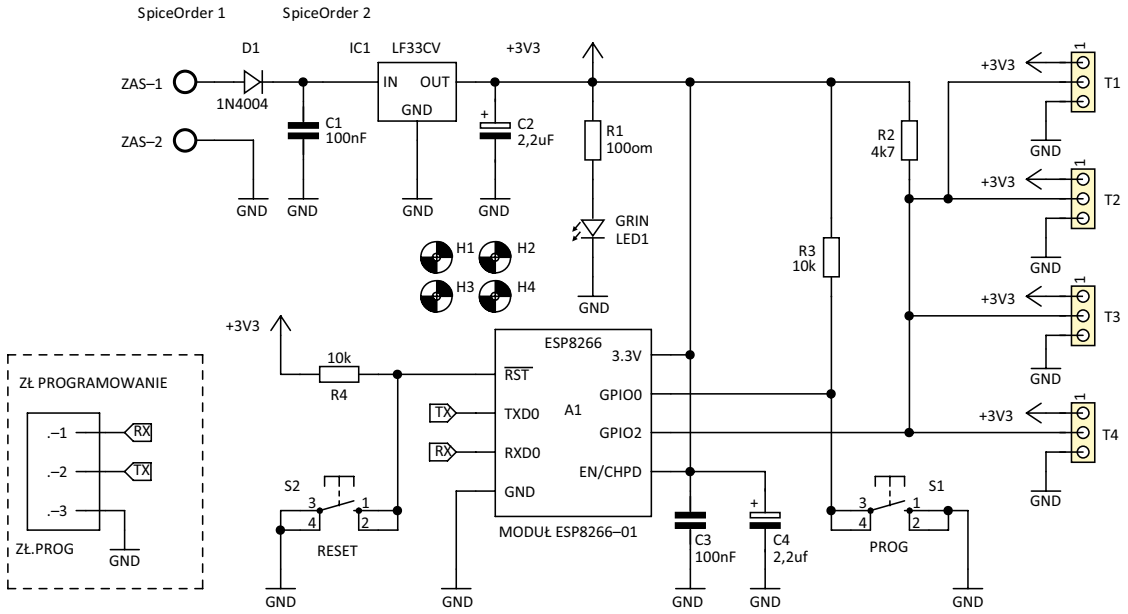
Moduł ESP8266 jest dość „prądożerny" i w czasie pracy pobiera prawie 300 mA, więc zaleca się zastosowanie zasilacza impulsowego o odpowiedniej wydajności prądowej. Dioda Led1 to wskaźnik obecności napięcia zasilania, który można wyprowadzić na zewnątrz obudowy.

Ostatnia grupa elementów przyciski S1 i S2, rezystory R3 i R4 oraz złącze ZŁ.PROG. Są one używane jedynie podczas programowania modułu ESP, które to zostanie szerzej omówione w dalszej części artykułu.

Zasada działania

Urządzenie współpracuje z aplikacją **Termik-4Temp** (całkowicie darmowa do pobrania ze sklepu Google Play). Dane są przesyłane za pomocą protokołu UDP.

Termometr nasłuchuje w sieci Wi-Fi na określonym porcie. Jeżeli nadejdzie pakiet



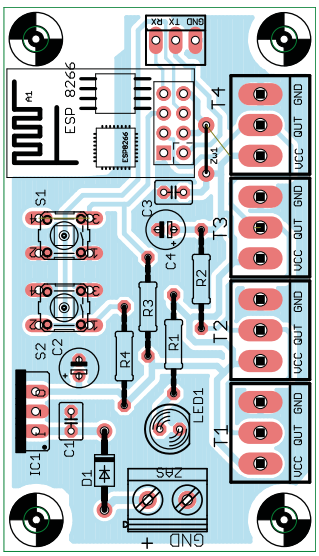
Rysunek 1. Schemat ideowy 4-kanałowego termometru z Wi-Fi

od aplikacji z żądaniem odczytu temperatur, urządzenie dokonuje pomiaru i odsyła pakiet danych z pomiarem na adres IP, z którego nadeszło zgłoszenie. Dla odświeżania wyników pomiarów, żądanie pomiaru z aplikacji wysyłane jest co 2 sekundy. Kiedy nadejdzie pakiet zwrotny z pomiarem, aplikacja przetwarza dane i wyświetla je na ekranie. Gdy „milczy" aplikacja, to „milczy" również termometr. Odczyt temperatur możliwy jest za pomocą wielu telefonów czy tabletów w tym samym czasie.

Montaż i uruchomienie

Schemat montażowy termometru pokazano na **rysunku 2**. Montaż rozpoczynamy typowo – od najniższych elementów (zwora, rezystory itp.). Po zmontowaniu całości, sprawdzamy poprawność montażu, mierzymy wartość napięcia zasilania (3,3 V) itp.

Do komunikacji termometru z komputerem potrzebować będziemy środowiska



Rysunek 2. Schemat montażowy 4-kanałowego termometru z Wi-Fi

programistycznego Arduino oraz konwertera USB-UART, który to dołączamy z przeplotem (Rx-Tx, Tx-Rx) do gniazda ZŁ.PROG termometru. Nie zapomnijmy też o połączeniu masy.

Na komputerze uruchamiamy środowisko Arduino pamiętając, aby ustawić odpowiedni numer portu, do którego jest dołączony konwerter. Jeśli wszystko jest przyłączone i skonfigurowane poprawnie, możemy przystąpić do pierwszego uruchomienia termometru.

Po włączeniu napięcia zasilania powinna zaświecić się czerwona dioda LED na module ESP, a niebieska może „błysnąć" kilka razy. Teraz z menu „Narzędzia" środowiska Arduino wybieramy zakładkę „Monitor portu szeregowego" i wysyłamy komendę AT (wpisujemy *AT+enter*). Gdy otrzymamy odpowiedź OK możemy uznać, że nasz układ jest sprawny. Za pomocą komend AT sprawdzamy jeszcze wersję firmware modułu ESP, którą w razie potrzeby należy uaktualnić do najnowszej dostępnej wersji.

Na **listingu 1** pokazano szkic ARDUINO, który należy wgrać do naszego modułu ESP8266 (wsad o nazwie **termometr4xds-18b20AVT**). Zanim jednak to nastąpi musimy odpowiednio przygotować szkic, tak, aby mógł pracować w sieci Wi-Fi. W pierwszej kolejności nadajemy adres IP termometrowi. Dzięki temu adres będzie stały i DHCP po restarcie rutera nie zmieni go. **Uwaga!** Wprowadzony adres nie może kolidować z innym adresem w naszej sieci. W kolejnym kroku wprowadzamy pozostałe dane, które możemy uzyskać w łatwy sposób wpisując polecenie w wyszukiwarce systemu Windows ***cmd+enter***, a następnie w otwartym okienku „wiersz polecenia" ***ipconfig+enter***. Dzięki tym czynnościom uzyskamy widok, z którego odczytamy konfigurację naszej sieci Wi-Fi. Określamy jeszcze port,

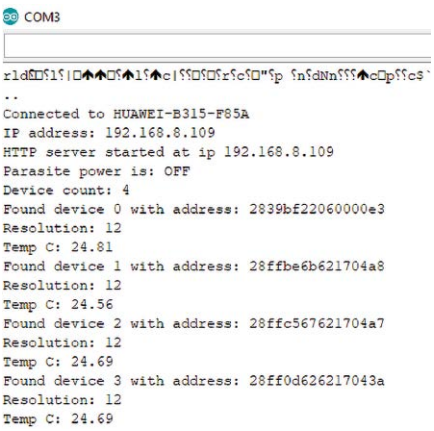
Wykaz elementów:

Rezystory:
R1: 100 Ω
R2: 4,7 kΩ
R3, R4: 10 kΩ

Kondensatory:
C1, C3: 100 nF
C2, C4: 2,2 µF

Półprzewodniki:
D1: 1N4004
IC1: LF33CV
LED1: zielona dioda LED
T1...T4: DS18B20

Inne:
Moduł ESP8266-01
S1, S2: przycisk miniaturowy



Rysunek 3. Adresy czujników odczytane za pomocą portu szeregowego

na którym będzie odbywać się komunikacja (powyżej wartości 1024). W przykładzie jest to port 12346. W szkicu znajdują się liczne komentarze, które ułatwią samodzielną konfigurację oprogramowania.

Teraz rzecz najtrudniejsza. Musimy odczytać adresy czujników DS18B20 i odpowiednio przypisać je do t1, t2, t3, t4 w kodzie naszego szkicu. W tym celu

Listing 1 Oprogramowanie modułu ESP8266

```
// Termometr 4-kanałowy, współpracuje z aplikacją Termik4Temp
#include <DallasTemperature.h>
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <OneWire.h>

/* tutaj musimy ustawić adres ip esp8266 oraz wprowadzić inne dane naszej sieci; po tej czynności będziemy mieli na stałe przypisany adres
ip w module ESP8266 */

IPAddress ip (192, 168, 8, 151);       // tu przypisujemy ip modułowi
IPAddress gateway(192, 168, 8, 1);     // Default Gateway
IPAddress subnet(255, 255, 255, 0);    // Subnet Mask
const char* ssid = "**************";    // nazwa naszej sieci WiFi
const char* password = "************"; // Hasło

void wifi_init(){
    WiFi.config(ip, gateway, subnet);
    delay(100);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    Serial.println();
    Serial.print("Connecting");
    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(200);
    }
    Serial.println();
    while (WiFi.waitForConnectResult() != WL_CONNECTED)
    {
        Serial.println("Fail connecting");
        delay(5000);
        ESP.restart();
    }
    Serial.print("   OK  ");
    Serial.print("Module IP: ");
    Serial.println(WiFi.localIP());
}

#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
// W ten sposób przypisujemy adresy czujników wcześniej odczytane
DeviceAddress t1 = { 0x28, 0x39, 0xbf, 0x22, 0x06, 0x00, 0x00, 0xe3 }; // - adres DS18B20 - 1
DeviceAddress t2 = { 0x28, 0xff, 0xbe, 0x6b, 0x62, 0x17, 0x04, 0xa8 }; // - adres DS18B20 - 2
DeviceAddress t3 = { 0x28, 0xff, 0xc5, 0x67, 0x62, 0x17, 0x04, 0xa7 }; // - adres DS18B20 - 3
DeviceAddress t4 = { 0x28, 0xff, 0x0d, 0x62, 0x62, 0x17, 0x04, 0x3a }; // - adres DS18B20 - 4

WiFiUDP Udp;
unsigned int localUdpPort = 12346; // Ustalamy port powyżej 1024
char incomingPacket[255]; //
char replyPacekt[] = "Dokonano pomiaru temperatur :-)";

void setup()
{
    Serial.begin(115200);
    Serial.println();
    wifi_init();
    delay(500);
    Serial.print(".");
    Serial.println(" connected");
    Udp.begin(localUdpPort);
    Serial.printf("Now listening at IP %s, UDP port %d\n", WiFi.localIP().toString().c_str(), localUdpPort);
}

void loop()
{
    int packetSize = Udp.parsePacket();
    if (packetSize)
    {
        //odbiór pakietów UDP
        sensors.requestTemperatures();// Polecenie odczytu temperatur
        Serial.printf("Received %d bytes from %s, port %d\n", packetSize, Udp.remoteIP().toString().c_str(), Udp.remotePort());
        int len = Udp.read(incomingPacket, 255);
        if (len > 0)
        {
            incomingPacket[len] = 0;
            sensors.requestTemperatures();
            delay(750);
        }
        Serial.printf("UDP packet contents: %s\n", incomingPacket);
        float temp1;
        temp1 = sensors.getTempC(t1);
        String Pomiar1;
        Pomiar1 = String(temp1, 1);
        Serial.println(Pomiar1);
        temp1 = sensors.getTempC(t2);
        String Pomiar2;
        Pomiar2 = String(temp1, 1);
        Serial.println(Pomiar2);
        temp1 = sensors.getTempC(t3);
        String Pomiar3;
        Pomiar3 = String(temp1, 1);
        Serial.println(Pomiar3);
        temp1 = sensors.getTempC(t4);
        String Pomiar4;
        Pomiar4=String(temp1, 1);
        Serial.println(Pomiar4);
        Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
        String tempp1 ;
        tempp1 += "n1=";
        tempp1 += Pomiar1;
        tempp1 += "x";
        tempp1 += "n2=";
        tempp1 += Pomiar2;
        tempp1 += "x";
        tempp1 += "n3=";
        tempp1 += Pomiar3;
        tempp1 += "x";
        tempp1 += "n4=";
        tempp1 += Pomiar4;
        tempp1 += "x";
        Udp.println(tempp1);
        Udp.endPacket();
    }
}
```

Rysunek 4. Adresy czujników odczytane za pomocą przeglądarki

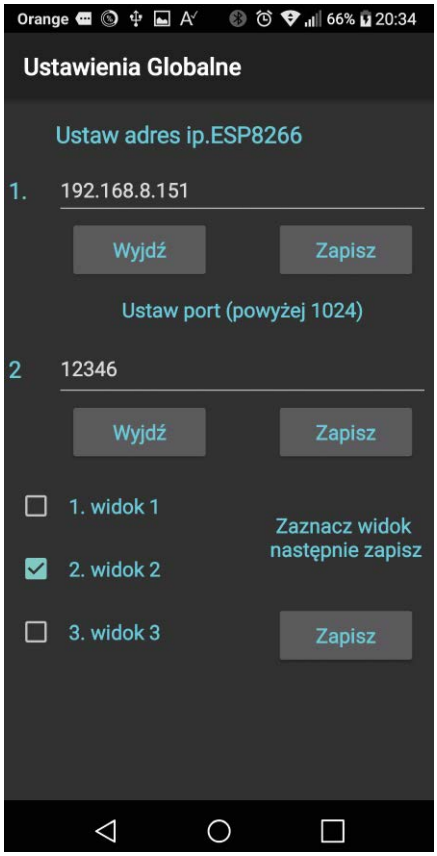
możemy wykorzystać inny szkic Arduino zaczerpnięty ze strony internetowej ***http://bit.ly/2H1Go9L*** oznaczony „wsad_odczytserialds18b20". Tutaj wprowadzamy tylko nazwę sieci i hasło dostępu.

Procedura wgrywania szkicu (dotyczy obu wersji)

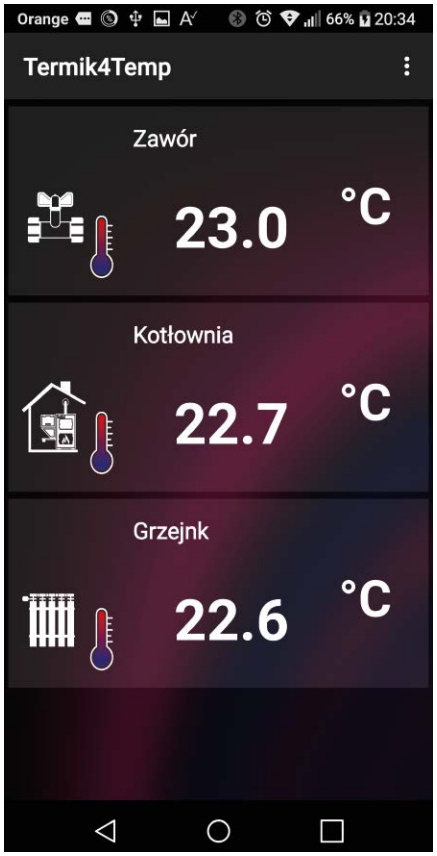
Przy wyłączonym napięciu zasilania dołączamy czujniki **DS18B20** do złącz T1…T4. Następnie włączamy napięcie zasilające i po chwili naciskamy przycisk S2 (RESET). Trzymając wciśnięty S2 wciskamy S1 (PROG), puszczamy S2 (RESET) i puszczamy S1 (PROG). Moduł ESP8266 wprowadzony został w tryb programowania. Teraz możemy wgrać nasz szkic.

Dzięki tej procedurze możemy również dokonać zmiany firmware modułu ESP8266. W niektórych jednak przypadkach przycisk S2 (RESET) może nie działać. Należy wówczas wyłączyć napięcie zasilania, wcisnąć przycisk S1 (PROG) i trzymając wciśnięty podać napięcie zasilania. Po chwili zwolnić przycisk i przystąpić do procedury wgrywania nowego firmware.

Po udanym wgraniu naszego pomocniczego szkicu przechodzimy do monitora portu szeregowego (Arduino) i naciskamy S2 (RESET). Naszym oczom powinien ukazać się widok pokazany na **fotografii 3**. Teraz możemy odczytać adresy czujników i przypisać je we właściwym szkicu. Odczytany adres



Rysunek 5. Ustawienia globalne aplikacji Termik4Temp



Rysunek 6. Ekrany z wynikami odczytu temperatury

IP naszego ESP, który tymczasowo został przydzielony przez DHCP (w tym przykładzie 192.168.8.109) możemy wpisać na pasku adresu przeglądarki internetowej. Po nawiązaniu połączenia uzyskamy widok pokazany na **fotografii 4** – z tabeli możemy odczytać numery seryjne czujników temperatury. Ten adres możemy wykorzystać i przypisać go na stałe naszemu termometrowi. Dzięki temu uzyskamy pewność, że nie wejdziemy w kolizję z innym adresem IP.

Tak przygotowany szkic możemy wgrać teraz do modułu ESP8266 we wcześniej opisany sposób. Po restarcie na monitorze szeregowym powinien wyświetlić się nadany wcześniej adres IP, nazwa sieci oraz status połączenia. Pozostało już tylko zainstalowanie i skonfigurowanie aplikacji Termik4Temp na telefonie lub tablecie z systemem Android. W aplikacji należy ustawić port oraz nadany wcześniej termometrowi adres IP oraz zrestartować aplikację (łącznie z rzutem w tle).

Po ponownym uruchomieniu aplikacji nawiązuje ona połączenie z modułem ESP termometru i całość rozpoczyna pracę. Niebieska dioda LED na module ESP zaczyna cyklicznie „błyskać", a po chwili na ekranie telefonu pojawiają się wyniki pomiarów temperatury (**rysunek 6**).

Przewidywane są dalsze modyfikacje i rozbudowa aplikacji o pomiar z wielu różnych modułów ESP8266 oraz umożliwienie zdalnego dostępu przez Internet.

Krzysztof Gruca
kisoftgk@gmail.com