# ESP32 – moduł do IoT

## Przykładowa aplikacja w stacji pogodowej

*Oferta podzespołów do IoT poszerza się bardzo dynamicznie. Oczekujemy, że nowe komponenty będą szybsze, bardziej niezawodne, lepiej wyposażone, ale także konkurencyjne cenowo i łatwe w implementacji. Ostatnie dwie cechy niewątpliwie przyczyniły się do popularności układu ESP8266. Zobaczmy, co oferuje kolejny układ tej serii – ESP32 firmy Espressif.*


**Fotografia 1. Układ ESP32**

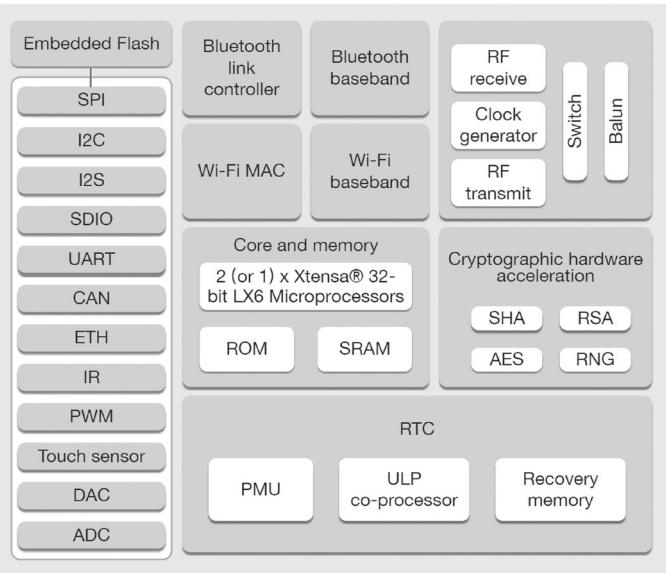
### ESP32

Podstawowe parametry układu ESP32 przedstawia **tabela 1**, dla porównania w tabeli umieszczono także parametry układu ESP8266. To oczywiste, że nowy układ jest szybszy i dysponuje większą pamięcią, ale najistotniejsze znaczenie mają dodatkowe peryferia, w które został wyposażony. Dzięki temu zmieniła się rola układu, już nie jest to tylko moduł komunikacji, teraz jest to samodzielny system. Dokładny schemat blokowy układu przedstawia **rysunek 1**.

Dwurdzeniowy procesor zastosowano z myślą o tym, aby umożliwić obsługę komunikacji Wi-Fi oraz jednoczesną realizację pozostałych funkcjonalności – programu użytkownika. Moduł Bluetooth otwiera drogę w kierunku komunikacji M2M (Machine to Machine), a w połączeniu z mocą obliczeniową procesora doskonale nadaje się do realizacji np. streamingu audio. Kontroler dotykowy Touch Sensor umożliwia bezpośrednią interakcję z użytkownikiem a jednocześnie daje dużą swobodę w zakresie designu.

Praktyczne zastosowanie technologii IoT jest możliwe tylko przy zapewnieniu bezpieczeństwa komunikacji. Problem nie jest zauważalny przy minimalnej ilości danych, ponieważ szyfrowanie można zrealizować programowo. Przy większej ilości danych takie rozwiązanie jest zbyt czasochłonne. Producent układu zdawał sobie sprawę z tego problemu i dlatego dodał sprzętowy moduł szyfrujący.

Urządzenia IoT często są zasilane bateryjnie, a wtedy istotny jest każdy mA pobierany ze źródła zasilania. Dla optymalizacji poboru energii prezentowany układ wyposażony jest w mechanizm automatycznego skalowania mocy obliczeniowej do aktualnych wymagań aplikacji oraz kilka trybów zmniejszonego poboru energii. Jednak to nie wszystko – zawiera także oddzielny blok funkcyjny przeznaczony do realizacji zadań w trybie głębokiego uśpienia i ultraniskiego poboru energii (*ULP – Ultra Low Power co-processor*).




**Rysunek 1. Schemat blokowy układu ESP32**

**Tabela 1. Porównanie parametrów ESP32 oraz ES8266.**

| | ESP32 | ESP8266 |
|---|---|---|
| Procesor CPU | Xtensa single/dual core 32-bit LX6, taktowany do 240 MHz | Tensilica L106 32-bit, taktowany do 160 MHz |
| Pamięć RAM | 520 kB SRAM + 16 kB SRAM in RTC | 160 kB w tym 80 kB user-data RAM |
| Pamięć ROM | 448 KB + dodatkowa pamięć programu w postaci zewnętrznej pamięci SPI Flash, do 16 MB | brak pamięci wbudowanej, pamięć programu w postaci zewnętrznej pamięci SPI Flash, do 16 MB |
| Porty GPIO | 34 | 16 |
| Wi-Fi | 802.11 b/g/n up to 150 Mbps | 802.11 b/g/n up to 72,2 Mbps |
| SPI/I²C/I²S/UART/PWM | Tak | Tak |
| Dodatkowe peryferia | ADC 12-bitowy, 18 kanałów<br>DAC 10-bitowy, 2 kanały,<br>Touch Sensor 10-kanałowy,<br>Wbudowany moduł Bluetooth 4.2 (wsparcie dla BLE – Bluetooth Low Energy),<br>Blok ULP – Ultra Low Power,<br>Sprzętowa akceleracja kryptograficzna. | ADC 10-bitowy, 1 kanał. |


**Fotografia 2. Moduły *WROOM* oraz *WROVER***

## WROOM czy WROVER?

Układ ESP32 dostępny jest przede wszystkim w postaci modułów – małych płytek z charakterystycznym metalowym ekranowaniem, anteną w postaci ścieżki na PCB (lub z miniaturowym złączem antenowym) oraz szeregiem wyprowadzeń-styków na krawędziach płytki. Pod ekranowaniem znajduje się procesor ESP32, pamięci FLASH i RAM oraz kilka elementów biernych niezbędnych do pracy tego mikrosystemu. Dostępne są dwie główne wersje, nazwane *WROOM* oraz *WROVER*, widoczne na **fotografii 2**.

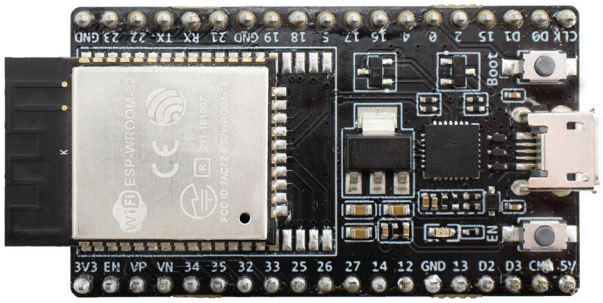
Na stronie producenta znajduje się krótka charakterystyka modułów (**tabela 2**). Oba moduły mają standardowo 4 MB pamięci Flash, ale dostępne są także wersje z pamięcią 8 lub 16 MB. Oba moduły dostępne są w wersji z anteną na PCB lub ze złączem antenowym. Natomiast tylko moduły *WROVER* są wyposażone w dodatkowe 8 MB pamięci RAM typu PSRAM (Pseudostatic RAM – to dynamiczna pamięć RAM z wbudowanym układem odświeżania i sterowania adresami, dzięki czemu zachowuje się podobnie do statycznej pamięci RAM. Łączy wysoką gęstość pamięci DRAM z łatwością użycia pamięci SRAM). Takie zestawienie kwalifikuje moduł do zastosowania np. w prostych systemach przetwarzania obrazu.

## Płytka testowa

Oferta firmy Espressif nie kończy się na modułach z procesorem, do dyspozycji mamy także gotowe płytki uruchomieniowe.

**Tabela 2. Charakterystyka modułów.**

| Moduł | Chip | Flash | PSRAM | Ant. |
|---|---|---|---|---|
| ESP32-WROOM-32 | ESP32-D0WDQ6 | 4 MB | –– | MIFA |
| ESP32-WROOM-32D | ESP32-D0WD | 4 MB | –– | MIFA |
| ESP32-WROOM-32U | ESP32-D0WD | 4 MB | –– | U.FL |
| ESP32-WROVER | ESP32-D0WDQ6 | 4 MB | 8 MB | MIFA |
| ESP32-WROVER-I | ESP32-D0WDQ6 | 4 MB | 8 MB | U.FL |
| ESP32-WROVER-B | ESP32-D0WD | 4 MB | 8 MB | MIFA |
| ESP32-WROVER-IB | ESP32-D0WD | 4 MB | 8 MB | U.FL |




**Fotografia 3. Płytka uruchomieniowa ESP32 DevKitC V4**

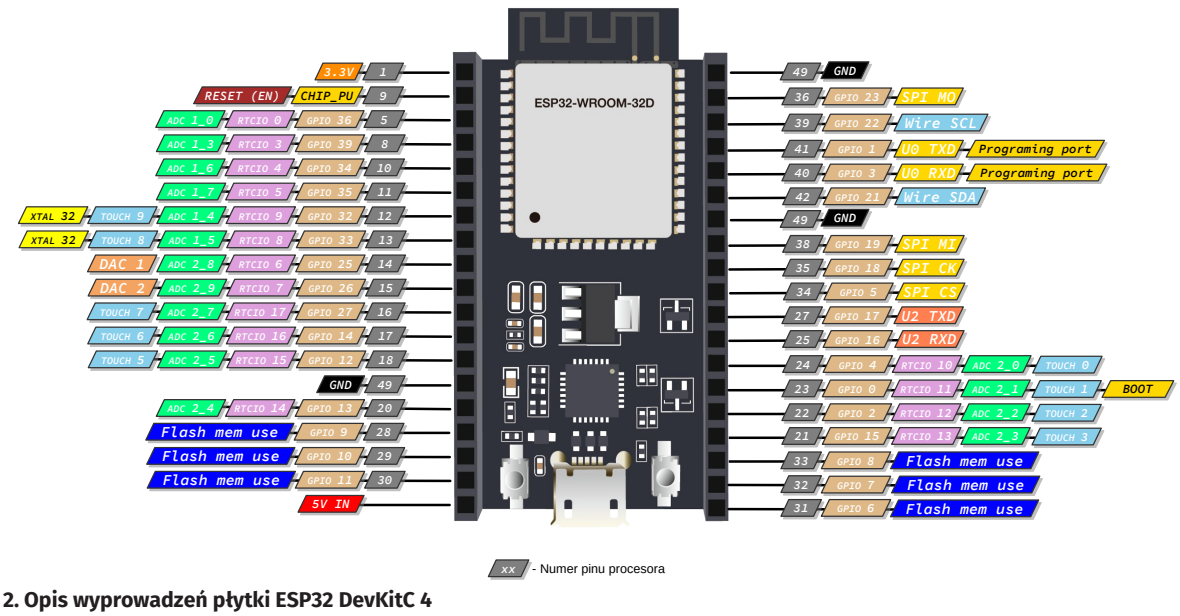
Na **fotografii 3** widoczna jest płytka *ESP32 DevKitC V4*, wyposażona w moduł typu *WROOM*.

Płytka *DevKitC V4* uzupełnia moduł o dwa istotne bloki – blok komunikacji poprzez USB oraz blok zasilania. Kontroler USB, po stronie komputera, tworzy wirtualny port szeregowy COM, a po stronie płytki połączony jest do interfejsu szeregowego UART procesora. Dzięki temu mamy możliwość programowania układu oraz monitorowania działania programu. Na płytce znajdują się przyciski *EN* oraz *Boot*, które służą do uruchomienia trybu programowania – należy trzymać przycisk *Boot* i nacisnąć przycisk *EN*. Aby jednocześnie umożliwić zasilanie modułu ze złącza USB, płytka zawiera stabilizator napięcia 3,3 V.

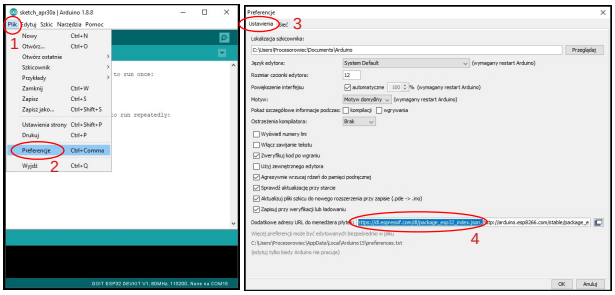
Dodatkowo płytka ma wyprowadzenia w postaci szpilek goldpin, dzięki czemu ułatwia budowę urządzeń prototypowych. Należy pamiętać, że układ ESP32 nie toleruje na wejściach napięcia wyższego od napięcia zasilającego – zatem do wejść nie wolno dołączać sygnałów o napięciu 5 V. **Rysunek 2** przedstawia opis wyprowadzeń płytki.

## Środowisko programistyczne

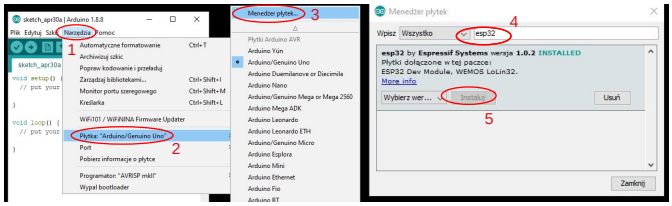
Najbardziej przystępnym narzędziem do programowania układu ESP32 jest *Arduino IDE*. Pozwala w prosty sposób zrealizować



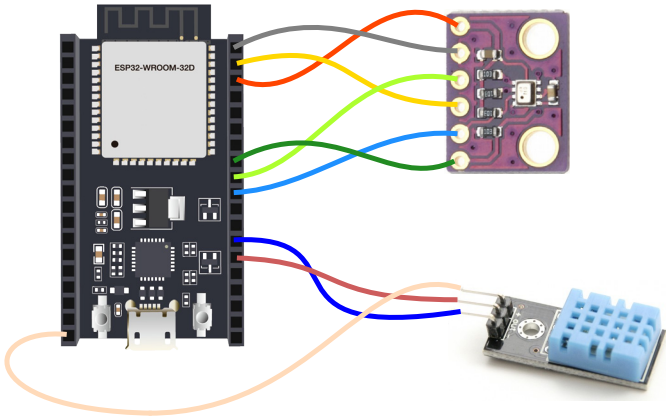

**Rysunek 2. Opis wyprowadzeń płytki ESP32 DevKitC 4**



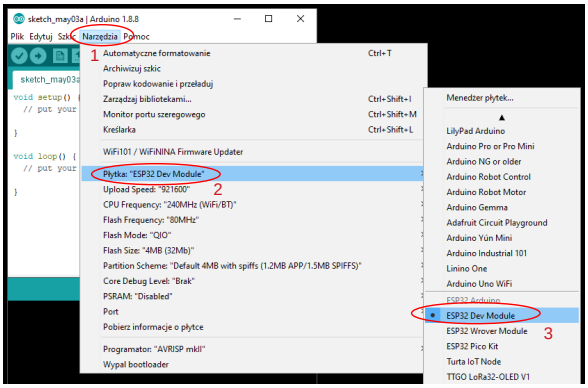
**Rysunek 3. Dodanie adresu URL do menedżera płytek**



**Rysunek 4. Zainstalowanie nowych bibliotek**




**Rysunek 5. Połączenie komponentów**



**Rysunek 6. Wybór docelowej płytki**

naprawdę złożone funkcjonalności. Alternatywą dla tego środowiska może być, dostarczany przez producenta układu, framework *ESP-IDF* (*Espressif – IoT Development Framework*), ale jest to rozwiązanie dla bardziej doświadczonych.

Środowisko Arduino wymaga krótkiego przygotowania do pracy z układem ESP32. Po pierwsze musimy dodać adres URL do menedżera płytek. W polu wskazanym na **rysunku 3** należy wstawić treść znajdującą się pod adresem: ***http://bit.ly/30tirSn***.

Jeśli środowisko *Arduino IDE* będzie używane także do programowania układów ESP8266, warto wstawić następującą, rozszerzoną treść: ***http://bit.ly/30tirSn***, ***http://bit.ly/2G476lA***.

Ostatni etap konfiguracji polega na zainstalowaniu nowych bibliotek – szczegółowe postępowanie przedstawia **rysunek 4**.

**Listing 1. Dołączenie niezbędnych bibliotek**

```
//potrzebne biblioteki sieciowe
#include <Wi-Fi.h>
#include <WebServer.h>

//potrzebne biblioteki czujników
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>

//potrzebne biblioteki interfejsów
#include <DHT.h>
#define DHTPIN 0
#define DHTTYPE    DHT11     // DHT 11
//#define DHTTYPE    DHT22     // DHT 22 (AM2302)

#include <SPI.h>
#define BMP_SCK 18
#define BMP_MISO 19
#define BMP_MOSI 23
#define BMP_CS 5
```

**Listing 2. Ustawienie parametrów sieciowych i utworzenie obiektów klas**

```
//ustawienia sieciowe
const char *apSSID = „ESP32_AP”;
const char *apPASS = „12345678”;
const int apPORT = 80;
IPAddress apSUBNET(255, 255, 255, 0);
IPAddress apGATE(10, 10, 10, 10);
IPAddress apIP(10, 10, 10, 10);
IPAddress apGetIP;

//instancje klas
WebServer webServer(apPORT);
Adafruit_BMP280 bmp(BMP_CS); // hardware SPI
DHT dht(DHTPIN, DHTTYPE);
```

**Listing 3. Uruchomienie access pointa oraz serwera**

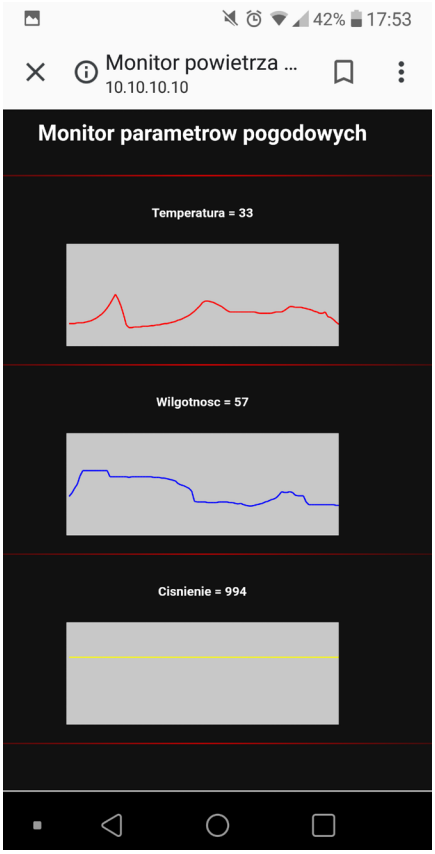
```
 //init czujników
  bmp.begin();
  dht.begin();

 //init sieci
  Wi-Fi.mode(Wi-Fi_AP);
  Wi-Fi.softAPConfig(apIP, apGATE, apSUBNET);
  Wi-Fi.softAP(apSSID, apPASS);
  apGetIP = Wi-Fi.softAPIP();

  //pokaż IP
  Serial.print(„\rAP IP address: „);
  Serial.println(apGetIP);

  //akcje serwera
  webServer.on(„/”, HandleRoot);
  webServer.onNotFound(HandleNotFound);

  //start serwera
  webServer.begin();

  Serial.println(„HTTP server started”);
```

**Listing 4. Pętla główna**

```
void loop(void) {

  //serwer
  webServer.handleClient();

  //częstość pomiarów
  now = millis();
  if ((now - prev) > dataPeriod){
    prev = now;

    //odczyt czujników
    temp = bmp.readTemperature();
    pres = bmp.readPressure();
    humm = dht.readHumidity();

    //dodanie do pamięci pomiarów
    DataFrameAdd(temp, humm, pres/1000);
    //wyświetla w terminalu
    Serial.print(„\rTemp=”);
    Serial.print(temp);
    Serial.print(„; Humm=”);
    Serial.print(humm);
    Serial.print(„; Press=”);
    Serial.println(pres);
  }
}
```

## Pierwszy projekt

Pierwszym projektem nie będzie *Helllo World*, zamiast tego uruchomimy małą stację pogodową. Użyjemy czujników: temperatury, wilgotności oraz ciśnienia i wszystkie te parametry zaprezentujemy na stronie www. Żeby umożliwić prognozowanie pogody, przedstawimy wykresy wartości tych parametrów w czasie. W pierwszej kolejności do płyki *ESP32 DevKitC V4* dołączymy czujniki *BMP280*




**Rysunek 7. Wygląd wygenerowanej strony internetowej**

**Listing 5. Kod odpowiedzialny za wygenerowanie strony internetowej**

```
//------------------------------------------------------
void HandleRoot() {

  sOut = „\
  <html>\
    <head>\
      <meta http-equiv='refresh' content='5'/>\
      <title>Monitor powietrza ESP32</title>\
      <style>\

        html{font-family:Verdana;display:inline-block;background-color:#111;}\

        .fontCl{color:white;text-align:center;text-decoration:none;color:#FFF;\
        font-weight:bold;padding:8px;margin:8px;}\

        .lineCl{border-top:3pxsolid;border-image:\
        linear-gradient(to left,#111,#E00,#111);border-image-slice:1;}\

        .graphCl{text-align:center;}\

      </style>\
    </head>\
    <body>\
      <h1 class=\"fontCl\">Monitor parametrow pogodowych</h1>";

      sOut += „<br><hr class=\"lineCl\"><br><h3 class=\"fontCl\">Temperatura = „;
      sOut += temp;
      sOut += „</h3><br><div class=\"graphCl\">";
      DataGraph(data1, 1);
      sOut += „</div>";

      sOut += „<br><hr class=\"lineCl\"><br><h3 class=\"fontCl\">Wilgotnosc = „;
      sOut += humm;
      sOut += „</h3><br><div class=\"graphCl\">";
      DataGraph(data2, 2);
      sOut += „</div>";

      sOut += „<br><hr class=\"lineCl\"><br><h3 class=\"fontCl\">Cisnienie = „;
      sOut += (pres / 100);
      sOut += „</h3><br><div class=\"graphCl\">";
      DataGraph(data3, 3);
      sOut += „</div>";

      sOut += „<br><hr class=\"lineCl\">";

    sOut += „\
        </body>\
    </html>";

  webServer.send(200, „text/html", sOut);

}
```

**Listing 6. Obrazowanie wykresów w treści strony**

```
//------------------------------------------------------
void DataGraph(int *src, int color) {

  char temp[100];

  sOut += „<svg xmlns=\"http://www.w3.org/2000/svg\" version=\"1.1\"\
    width=\"400\" height=\"150\">\n";

  sOut += „<rect width=\"400\" height=\"150\" fill=\"rgb(200, 200, 200)\
  \" stroke-width=\"1\" stroke=\"rgb(255, 255, 255)\" />\n";

  if (color == 1) sOut += „<g stroke=\"red\">\n";
  if (color == 2) sOut += „<g stroke=\"blue\">\n";
  if (color == 3) sOut += „<g stroke=\"yellow\">\n";

  int x = 0;
  int y = AbsoluteDataGet(src, x);
  for (x=1; x<dataLen ; x++) {
    int newY = AbsoluteDataGet(src, x);
    int newX = (x * 4);

    sprintf(temp, „<line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" \
    stroke-width=\"2\" />\n", newX, 150 - y, (newX + 4), 150 - newY);

    sOut += temp;
    y = newY;
  }

  sOut += „</g>\n";
  sOut += „</svg>\n";

  //webServer.send(200, „image/svg+xml", sOut);
}
```

oraz *DTH11*, zgodnie z **rysunkiem 5**. W kolejnym kroku ustawimy w *Arduino IDE* docelową płytkę. Możemy wskazać sam moduł, jak na **rysunku 6** – *ESP Dev Module*.

Kod programu omówimy tylko w istotnych fragmentach, natomiast kompletne źródła dostępne są w materiałach dodatkowych. Zaczynamy od dołączenia potrzebnych bibliotek – **listing 1**. Jeśli w czasie kompilacji programu będzie brakowało którejś biblioteki, to należy ją zainstalować, wybierając *Narzędzia → Zarządzaj Bibliotekami*. Nasza stacja pogodowa będzie działała jako access point – musimy ustawić jego nazwę, hasło dostępu oraz numer IP strony www, która będzie prezentowała wyniki – jak na **listingu 2**.

W sekcji programu określonej jako *setup* wstawiamy polecenia uruchamiające access point oraz serwer www – **listing 3**. Tych kilkanaście linii wystarczy, aby skonfigurować wszystkie elementy systemu, teraz można uzupełnić główną pętlę programu – **listing 4**. Główne zadanie, które wykonuje, to obsługa klienta – urządzenia, które łączy się z serwerem. Poza tym w głównej pętli mierzony jest czas pomiędzy pomiarami i gdy osiągnie ustaloną wartość, to następuje odczyt czujników, dodanie wyników do pamięci oraz wyświetlenie ich w terminalu.

Kod odpowiedzialny za wygenerowanie strony internetowej przedstawiony jest na **listingu 5**. Zmienna *sOut* typu *String* stanowi bufor kodu strony. Do niej dopisywane są kolejne fragmenty, a ostatecznie zostaje wysłana jako odpowiedź serwera – *webServer.send(200, „text/html", sOut);*

Obrazowanie wykresów w treści strony zostało rozwiązane poprzez wstawianie bloków *SVG* (*Scalable Vector Graphics* – uniwersalny format dwuwymiarowej grafiki wektorowej). Taki sposób można znaleźć w jednym z projektów przykładowych o nazwie *AdvencedWebServer*. Kod funkcji *void DataGraph(int *src, int color)* widać na **listingu 6**.

Aby zobaczyć efekty działania naszego urządzenia, należy, np. za pomocą smartfona, połączyć się z siecią o nazwie *ESP32_AP*, podając hasło *12345678*. Następnie należy uruchomić dowolną przeglądarkę i zamiast adresu strony wpisać numer IP o wartości 10.10.10.10. Wyświetli się strona taka jak na **rysunku 7**.

**KS**
**ep@prolab.waw.pl**