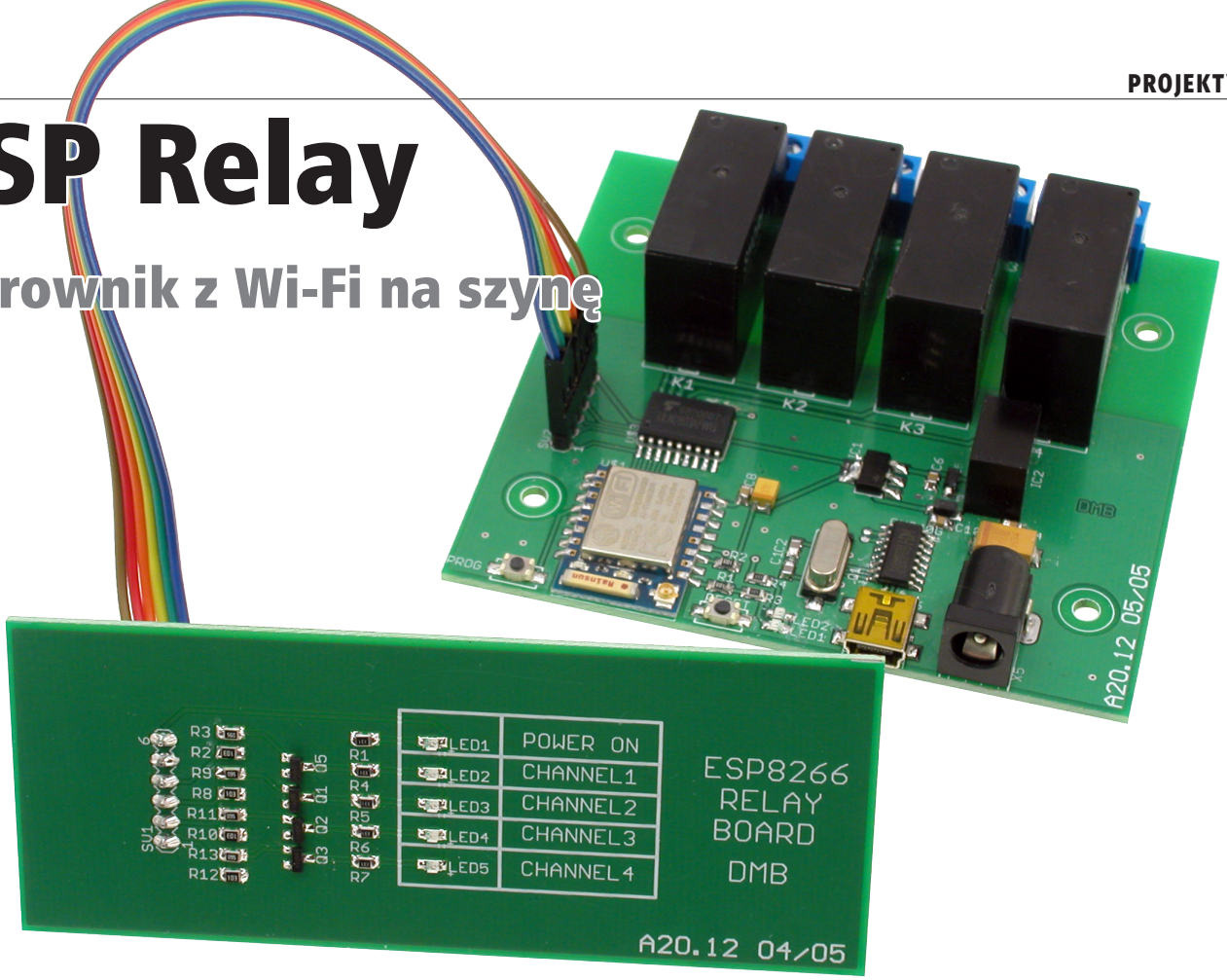
ESP Relay

Sterownik z Wi-Fi na szynę



Moduł jest przeznaczony do bezprzewodowego załączania odbiorników energii poprzez sieć Wi-Fi. Dzięki zastosowaniu interfejsu radiowego mamy do niego dostęp z dowolnego miejsca, w którym jest zasięg sieci. Płytkę przystosowano do obudowy, którą można zamontować na szynie TH35 w szafie sterowniczej.

Rekomendacje: *może służyć jako element wykonawczy do automatyki budynkowej*

Schemat ideowy sterownika zamieszczono na **rysunku 1**. Wykonano go w oparciu o moduł ESP07 z 32-bitowym mikrokontrolerem ESP8266 firmy Espressif z rdzeniem Tensilica L106 mający wbudowane peryferium obsługujące połączenie z Wi-Fi. Aby uniknąć konieczności korzystania z programatora zewnętrznego, zamontowane go na płytce sterownika, więc do zmiany oprogramowania wystarczy przewód miniUSB umożliwiający komunikację z komputerem oraz odpowiednio skonfigurowane środowisko Arduino. Rolę konwertera USB/UART odgrywa układ CH340G.

Płytka sterownika jest zasilana napięciem +12 V doprowadzonym do złącza DC. Zastosowano konwerter DC-DC AIMTEC AMSR-7805-NZ w celu obniżenia napięcia z +12 V na +5 V. Wspomniany konwerter to zamiennik popularnego układu LM7805, zgodny z nim pod względem wyprowadzeń, jednak mający większą sprawność. Napięcie +3,3 V uzyskano za pomocą stabilizatora liniowego LM1117-3.3. Dla zapewnienia odpowiedniej impulsowej wydajności prądowej zastosowano kondensatory tantalowe ze względu na ich małą rezystancję ESR. Jako zabezpieczenie obwodu zasilania zastosowano bezpiecznik polimerowy PTC 750 mA.

Przekaźniki mają cewki zasilane napięciem 12 V i są załączane za pomocą ULN2803. Jako wyprowadzenia przekaźników użyto 3-pinowych złączy śrubowych ARK500/3, dzięki czemu są dostępne styki NO i NC. Poprawne zasilanie jest sygnalizowane świeceniem LED1, natomiast LED2 sygnalizuje zerowanie i wejście w tryb programowania.

Na **rysunku 2** pokazano schemat ideowy płytki, która zawiera diody sygnalizujące załączenie przekaźników. Jej użycie jest opcjonalne – taka płytka dobrze prezentuje się, jeśli mamy obudowę Z101JFP z przezroczystym przodem. Przyda się też przy eksperymentach z programowaniem sterownika.

Montaż i uruchomienie

Schemat montażowy sterownika pokazano na **rysunku 3**, natomiast płytki z diodami sygnalizacyjnymi LED na **rysunku 4**. Moduł pasuje do obudów uniwersalnych typu Z101J oraz Z101JFP. Różnica między nimi to przezroczyste okno w wersji Z101JFP. W zależności od potrzeb można wykorzystać dodatkową płytkę z diodami

DODATKOWE MATERIAŁY NA FTP:

ftp://ep.com.pl

USER: 44747, PASS: 3qwdwa8u

W ofercie AVT*

AVT-5583

Podstawowe informacje:

- Moduł ESP07 z mikrokontrolerem ESP8266.
- Pasuje do obudów Z101J i Z101JFP montowanych na szynie TH35.
- Zasilanie 12 V DC/150 mA.
- 8 wyjść przekaźnikowych – dostępne styki NO i NC.
- Programowanie za pomocą dostępnych platform (np. Blynk – EP 4/2017) lub przez stronę www.

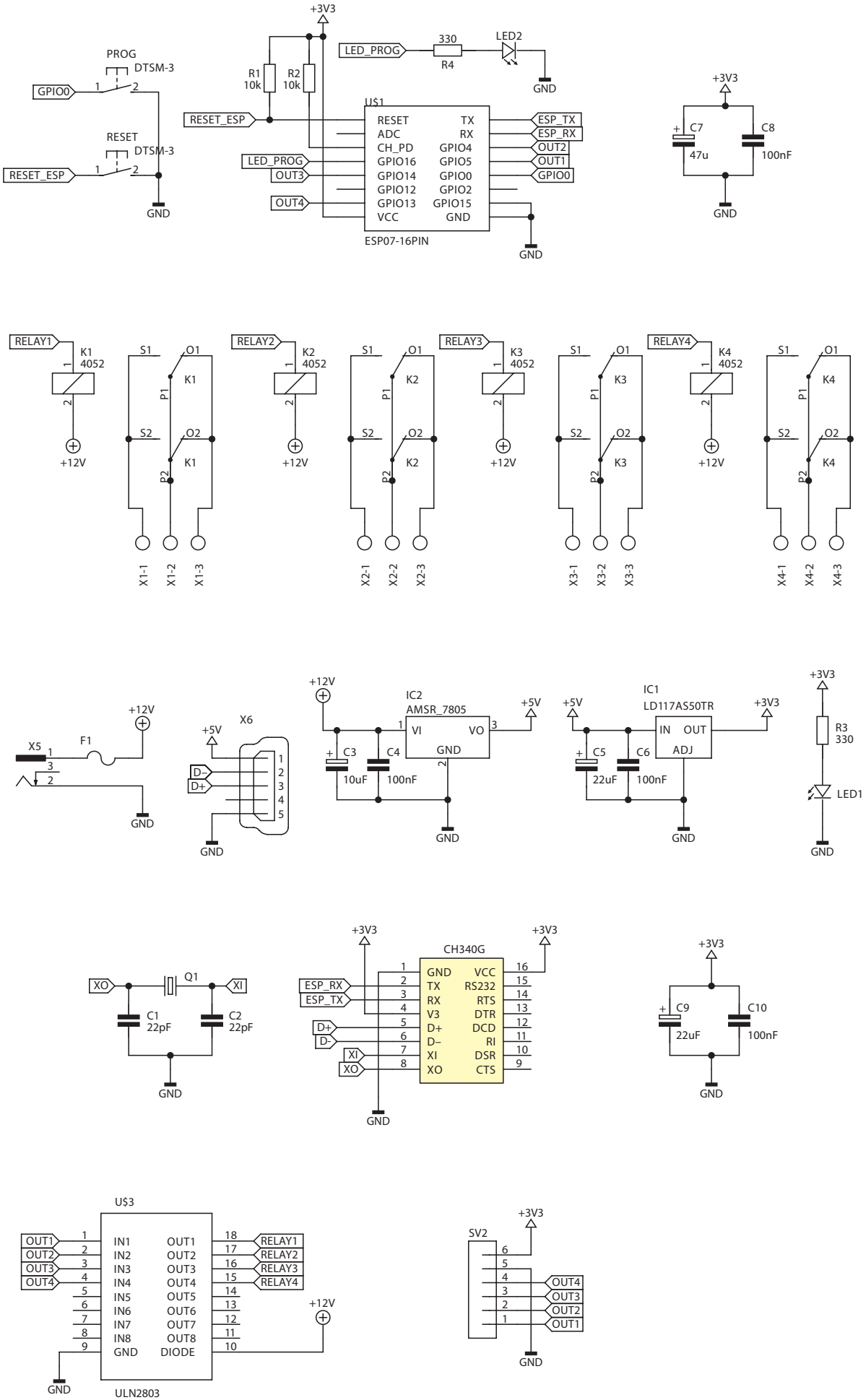
Projekty pokrewne na FTP:
(wymienione artykuły są w całości dostępne na FTP)

AVT-5530	Regulator natężenia oświetlenia z Wi-Fi (EP 1/2016)
AVT-5350	Moduł wykonawczy z interfejsem Ethernet (EP 6/2012)
AVT5250	Karta przekaźników z interfejsem Ethernet (EP 8/2010)
AVT-966	Karta przekaźników z interfejsem Ethernet (EP 2/2007)

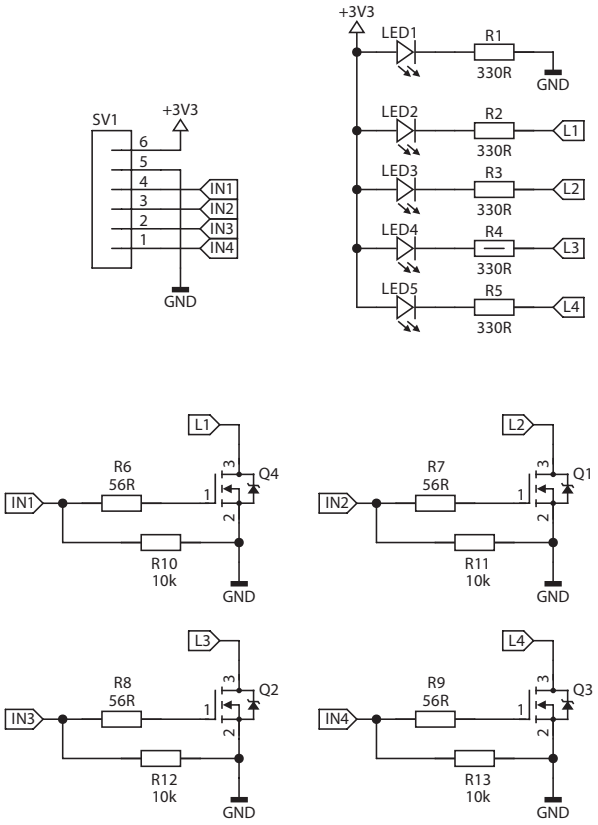
*** Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KITem (z ang. zestaw). Zestaw w wersji [**B**] zawiera elementy elektroniczne (w tym [UK] – jeśli występuje w projekcie), które należy samodzielnie 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

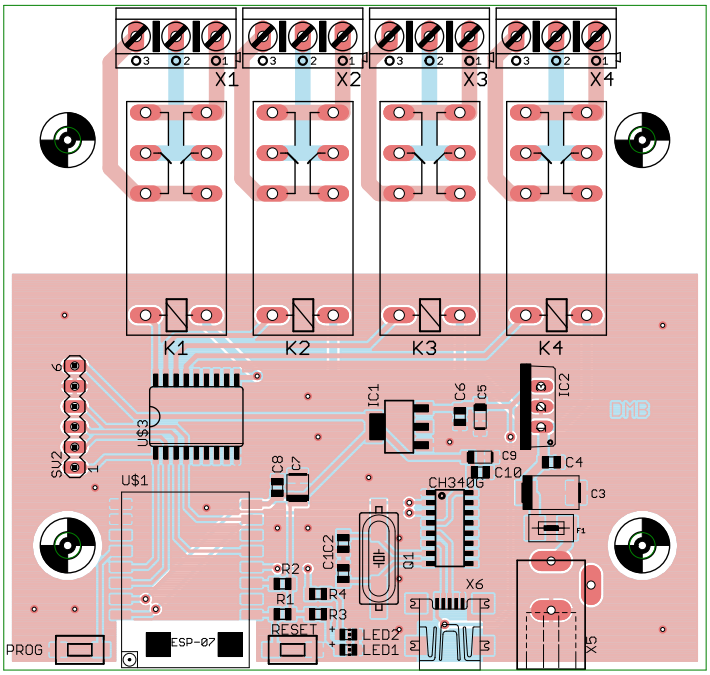
Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! ***http://sklep.avt.pl***



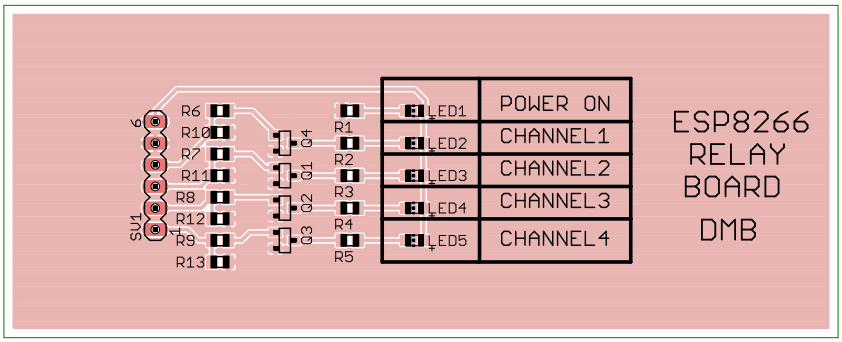
Rysunek 1. Schemat ideowy sterownika z Wi-Fi



Rysunek 2. Schemat ideowy płytki z diodami LED



Rysunek 3. Schemat montażowy sterownika z Wi-Fi



Rysunek 4. Schemat montażowy płytki z diodami LED

wyświetlającymi stan przekaźników – pasującą do wersji Z101JFP. Łączy się ona z główną płytą 6-żyłowym przewodem za pomocą złącza SV2. Niestety, w obudowie nie ma żadnych zatrzasków, więc trzeba ją zamocować samodzielnie. W obudowie trzeba wykonać dwa otwory – na przewód USB i wtyk zasilający.

Na płytce przewidziano powiększone otwory na śrubki montażowe M3, aby można było łatwiej zamontować płytkę w przypadku użycia innej obudowy.

W handlu jest dostępnych wiele modułów z mikrokontrolerem ESP8266. Różnią się one przede wszystkim wymiarami, sposobem montażu, pojemnością pamięci Flash, częstotliwością taktowania rdzenia oraz typem anteny. Moduł ESP07 ma wbudowaną antenę oraz złącze SMA dające możliwość użycia anteny zewnętrznej. Jest to o tyle ważne, że moduł można zamknąć w szafce, natomiast antenę zamocować poza nią, aby uzyskać lepszy zasięg.

Sterownik w głównej mierze składa się z komponentów SMD, z których najmniejsze mają obudowy 0805. Zamontowanie modułu ESP07 nie powinno sprawić trudności, natomiast pewien kłopot może stanowić przylutowanie gniazda microUSB, które ma niewielki raster wyprowadzeń.

Wykaz elementów:
Sterownik

Rezystory: (SMD 0805)
R1,R2: 10 kΩ
R3, R4: 330 Ω
Kondensatory:
C1, C2: 22 pF (SMD 0805)
C3: 10 µF/15 V (SMD „C”)
C4, C6, C8, C10: 100 nF (SMD 0805)
C5, C9: 22 µF/6,6 V (SMD „A”)
C7: 47 µF/6,3 V (SMD „B”)
Półprzewodniki:
U2: CH340G
U3: ULN2803
IC1: LM1117-3.3
IC2: AMSR_7805
Inne:
F1: bezpiecznik polimerowy PTC 750 mA 1812
U1: ESP-07
Q1: 12 MHz (rezonator kwarcowy)
LED1, LED2 – dioda LED zielona 0805
PROG, RESET – przycisk 3 mm
K1...K4: S4E-12B-1C (przekaźnik z cewką na 12 V DC, styki NO/NC 16 A)
X1...X4: złącza ARK 5 mm potrójne
X5: złącze DC 2,5
X6: gniazdo USB mini
SV2: goldpiny męskie 1×6 2,54

Płytka LED

R1...R5: 330 Ω
R6...R9: 56 Ω
R10...R13: 10 kΩ
LED1...LED5: dioda LED zielona 0805
Q1...Q4: IRFML2502 NMOS
SV1: goldpiny męskie 1×6

Programowanie

Do programowania układu ESP8266 zastosowanego w module możemy używać różnych kompilatorów, ale dobrym pomysłem jest użycie Arduino IDE. W programie zdefiniowano stałe służące do identyfikacji sieci Wi-Fi, z którą będziemy się łączyli, co zmusza nas do skompilowania programu za każdym razem, gdy zmieniamy sieć lub miejsce zainstalowania moduł.

Aby umożliwić programowanie modułu ESP07 z poziomu Arduino IDE, musimy zainstalować oprogramowanie umożliwiające jego obsługę. W starszych wersjach oznaczałoby to ręczne podmienianie zawartości plików w folderach konfiguracyjnych. Na szczęście od wersji 1.6 jest wspierany mechanizm automatycznego pobierania dodatkowych modułów – Additional Boards Manager.

Z menu wybieramy *Plik* → *Preferencje*. W nowo otwartym oknie odnajdujemy *Additional Boards Manager URLs:* (**rysunek 5**). Wklejamy tu link do plików z ustawieniami dla wsparcia modułów ESP: *http://arduino.esp8266.com/stable/package_esp8266com_index.json*. Po tej czynności klikamy *OK*. Teraz przechodzimy do opcji *Narzędzia* → *Płytka* → *Boards Manager*. Odszukujemy „esp8266” i instalujemy najnowszej wersję biblioteki, jak na **rysunku 6**. Po zainstalowaniu uzyskujemy dostęp do wielu przykładowych projektów i co ważniejsze – w wyborze płytek zostaną wyświetlone różne moduły z ESP8266 (**rysunek 7**).

Przetestujmy teraz, czy możemy komunikować się ze sterownikiem i zapisywać jego pamięć programu. Włączamy zasilanie (+12 V DC), a następnie przewodem miniUSB łączymy sterownik z komputerem PC. Po zainstalowaniu sterowników zostanie wyświetlony nowy, wirtualny port COM. W razie niepowodzenia instalacji sterowników – wgrywamy te z plików dołączonych do magazynu lub wyszukujemy najnowsze dla układu CH340 w wersji zależnej od posiadanego systemu operacyjnego.

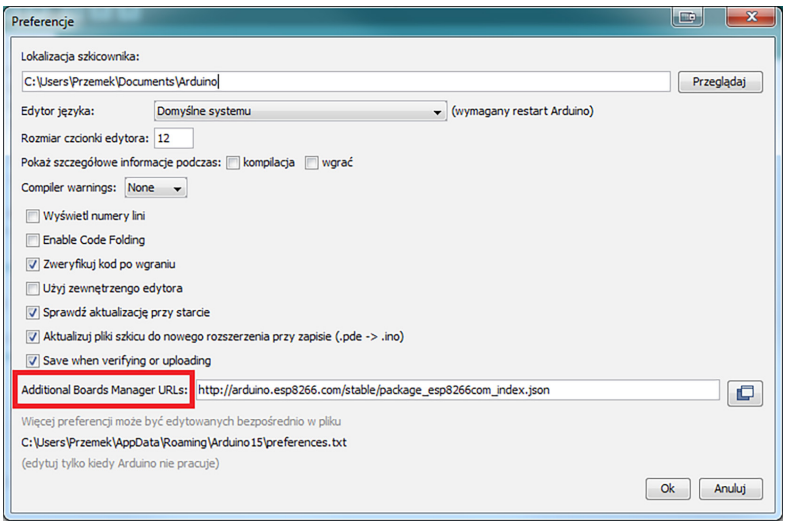
Z menu Arduino IDE wybieramy *Narzędzia* → *Płytka* → *Generic ESP8266 Module*. Zmieniamy je w taki sposób, jak pokazano na **rysunku 8**. Oczywiście, używamy takiego numeru portu COM, jaki został nadany w przez system Windows po zainstalowaniu sterowników. Następnie wybieramy *Plik* → *Przykłady* → *ESP8266* → *Blink*. Jest to program zmieniający stan wyprowadzenia z częstotliwością 1 Hz. Wprowadzamy tam jednak małą modyfikację – zamiast „LED_BUILTIN” wpiszmy wszędzie 5 – jest to pin wyjście sterujące jednym z przekaźników (program o nazwie ESP_test w materiałach).

Musimy teraz uruchomić moduł ESP07 w trybie programowania:

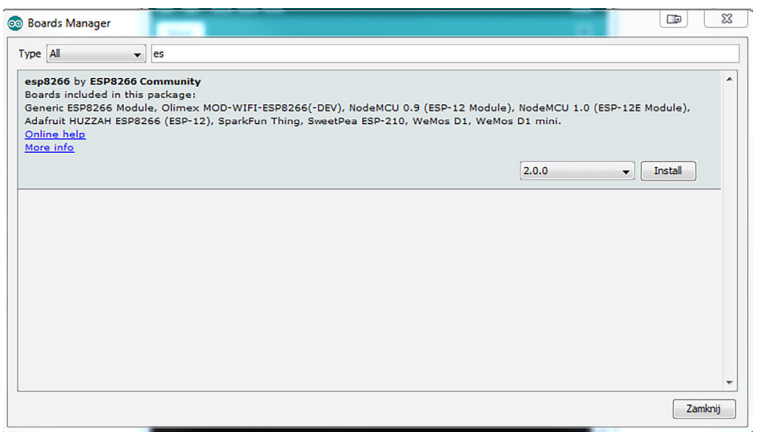
- Naciskamy przycisk *Reset*.
- Trzymając go, wciskamy przycisk *Program*.

Listing 1. Funkcja inicjująca komunikację oraz inicjowanie obsługi klientów

```
void setup ( void )
{
    // inicjalizacja pinow jako wyjscia
    pinMode ( R1, OUTPUT );
    pinMode ( R2, OUTPUT );
    pinMode ( R3, OUTPUT );
    pinMode ( R4, OUTPUT );
    // wyzerowanie pinow
    digitalWrite( R1, LOW );
    digitalWrite( R2, LOW );
    digitalWrite( R3, LOW );
    digitalWrite( R4, LOW );
    // stan domyslny przekaznikow z tablicy
    przekazniki_zapal();
    // ustawienie uartu dla debugowania
    Serial.begin ( 115200 );
    // polaczenie z siecia
    WiFi.begin ( ssid, password );
    Serial.println ( „” );
    // Oczekiwanie na polaczenie
    while ( WiFi.status() != WL_CONNECTED ) {
        delay ( 500 );
        Serial.print ( „.” );
    }
    // wyswietlenie informacji o polaczeniu
    Serial.println ( „” );
    Serial.print ( „Connected to „ );
    Serial.println ( ssid );
    Serial.print ( „IP address: „ );
    Serial.println ( WiFi.localIP() );
    if ( MDNS.begin ( „esp8266” ) )
    {
        Serial.println ( „MDNS responder started” );
    }
    // przypisanie funkcji do danych jakie moze odebrac serwer
    server.on ( „/”, handleRoot );
    server.on( „/relays”, przekazniki_funkcja );
    server.on( „/rel”, json_funkcja );
    server.onNotFound( handleNotFound );
    // uruchomienie serwera
    server.begin();
    // wyswietlenie informacji o rurchomieniu
    Serial.println ( „HTTP server started” );
}

void loop ( void )
{
    // obsluga klientow przez serwer
    server.handleClient();
}
```



Rysunek 5. Okno z parametrami menedżera płytek



Rysunek 6. Instalowanie oprogramowania dla ESP8266

- Puszczamy *Reset*, cały czas trzymając *Program*.
- Puszczamy przycisk *Program*.

Po tych czynnościach powinna zaświecić się dioda LED2 oznaczająca, że moduł ESP07 jest w trybie programowania. Teraz w Arduino IDE klikamy na *Wgraj*. Program zostanie skompilowany i rozpocznie się proces programowania. Jeśli wszystko przebiegnie prawidłowo, w polu komunikatów zostanie wyświetlona wiadomość, jak na **rysunku 9** oraz usłyszymy „stukanie" przekaźnika, a odpowiadający mu LED na panelu będzie na przemian gasł i zaświecał się.

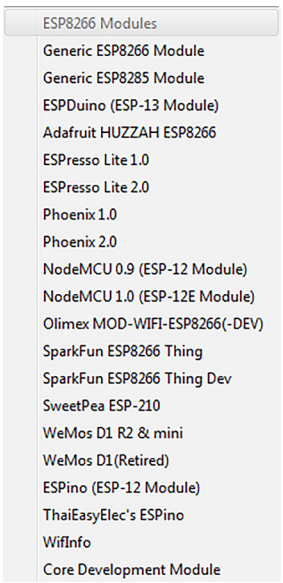
Użytkowanie sterownika

Konfiguracja środowiska za nami. Pora na opisanie dwóch sposobów kontrolowania pracy sterownika ESP Relay, są to: użycie platformy Blynk lub przez wbudowaną stronę internetową.

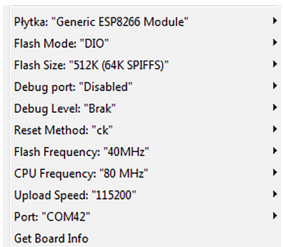
Platformę Blynk opisałem w poprzednim wydaniu „Elektroniki Praktycznej" (EP 4/2017). Korzystając z drugiej metody, możemy załączać wyjścia za pomocą dowolnego urządzenia mającego przeglądarkę internetową i dostęp do sieci, w której pracuje nasz sterownik.

Aby umożliwić sterowanie za pomocą przeglądarki, musimy uruchomić w ESP8266 oprogramowanie serwera strony www i wykonać stronę z przyciskami do sterowania. Ja do tego celu posłużyłem się przykładem „Advanced Web Serwer" z biblioteki obsługi ESP8266. Znajdziemy go w materiałach do artykułu.

Aby dostosować program do swoich potrzeb, trzeba poprawić linie:



Rysunek 7. Moduł z ESP8266 dostępny w Arduino IDE



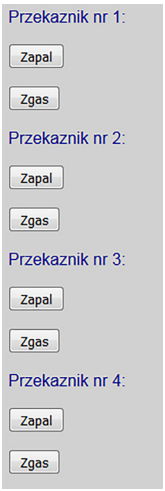
Rysunek 8. Okno parametrów środowiska Arduino

Listing 2. Sterowanie przekaźnikami oraz definicja strony w postaci stałej typu string

```
const String strona =
„<html>\
  <head>\
    <title>ESP RELAY</title>\
    <style>\
      body { background-color: #cccccc; font-family: Arial, Helvetica, Sans-Serif; Color:
#000088; }\
    </style>\
  </head>\
  <body>\
<p>Przekaznik nr 1:</p>\
<form action=\"relays\?rel1=on\" method=\"POST\">\
    <input type=\"submit\" value=\"Zapal\">\
</form>\
<form action=\"relays\?rel1=off\" method=\"POST\">\
    <input type=\"submit\" value=\"Zgas\">\
</form>\
<p>Przekaznik nr 2:</p>\
<form action=\"relays\?rel2=on\" method=\"POST\">\
    <input type=\"submit\" value=\"Zapal\">\
</form>\
<form action=\"relays\?rel2=off\" method=\"POST\">\
    <input type=\"submit\" value=\"Zgas\">\
</form>\
<p>Przekaznik nr 3:</p>\
<form action=\"relays\?rel3=on\" method=\"POST\">\
    <input type=\"submit\" value=\"Zapal\">\
</form>\
<form action=\"relays\?rel3=off\" method=\"POST\">\
    <input type=\"submit\" value=\"Zgas\">\
</form>\
<p>Przekaznik nr 4:</p>\
<form action=\"relays\?rel4=on\" method=\"POST\">\
    <input type=\"submit\" value=\"Zapal\">\
</form>\
<form action=\"relays\?rel4=off\" method=\"POST\">\
    <input type=\"submit\" value=\"Zgas\">\
</form>\
  </body>\
</html>";

void handleRoot()
{
    server.send ( 200, „text/html", strona );
}

void handleNotFound()
{
    String message = „Site Not Found\n\n";
    server.send ( 404, „text/plain", message );
}

void przekazniki_funkcja( void )
{
    String message = „Otrzymalem:\n";
    message += „URI: „;
    message += server.uri();
    message += „\nMethod: „;
    message += ( server.method() == HTTP_GET ) ? „GET" : „POST";
    message += „\nArguments: „;
    message += server.args();
    message += „\n";
    // debug na konsoli
    for ( uint8_t i = 0; i < server.args(); i++ )
    {
        message += „ „ + server.argName ( i ) + „: „ + server.arg ( i ) + „\n";
    }
    Serial.println ( message );
    // rozkodowanie komendy
    for ( uint8_t i = 0; i < server.args(); i++ )
    {
        if( server.argName( i )[ 0 ] == ‚r' && server.argName( i )[ 1 ] == ‚e' && server.
argName( i )[ 2 ] == ‚l' )
        {
        // zabezpieczenie przed przekroczeniem indexu
            if( server.argName( i )[ 3 ] - ‚1'  < 4 )
            {
                tablica_relays[ server.argName( i )[ 3 ] - ‚1' ] = (server.arg( i ) == „on"
? ‚1' : ‚0');
            }
        }
    }
    // wykonanie akcji
    przekazniki_zapal();
    // przeslanie strony glownej
    handleRoot();
}
```

```
Szkic używa 222 205 bajtów (51%) pamięci programu. Maksimum to 434 160 bajtów.
Globalne zmienne używają 31 572 bajtów (38%) dynamicznej pamięci, pozostawiając 50 348 bajtów dla lokalnych zmiennych. Maksimum to  81 920 bajtów.
Uploading 226352 bytes from C:\Users\Przemek\AppData\Local\Temp\build8b0c9f5253da3759bd7dbd37c174402e.tmp/Blink.ino.bin to flash at 0x00000000
.................................................................................. [ 36% ]
.................................................................................. [ 72% ]
.............................................................                    [ 100% ]
```

Rysunek 9. Komunikat o pomyślnym zapisie programu w pamięci sterownika



Rysunek 10. Wygląd strony głównej

```
//ustawienia parametrow sieci
const char *ssid = "b97488";
const char *password = "280740688";
```

Należy w nich podać nazwę i hasło dostępu do sieci Wi-Fi, z którą będziemy się łączyli. Następnie w przeglądarce w polu adresu wpisujemy przydzielony przez router adres IP. Możemy go poznać na kilka sposobów – po wgraniu wsadu/restarcie/uruchomieniu zasilania podłączyć się terminalem (np. Putty) do układu. Zobaczymy wtedy podgląd zdarzeń, w tym przydzielony adres IP:

Connected to b97488
IP address: 192.168.0.22
MDNS responder started
HTTP server started.

Alternatywnie możemy sprawdzić w ustawieniach routera podłączone urządzenia i ich adresy lub użyć programu typu AdvancedIPScanner. Można także, co jest szczególnie polecane, tak skonfigurować router, aby zawsze nadawał naszemu sterownikowi ten sam adres IP.

Program rozpoczyna pracę od konfiguracji wyprowadzeń jako wyjścia i ich wyzerowania. Następnie inicjalizuje pracę UART do debugowania i próbuje nawiązać połączenie z siecią zdefiniowaną w parametrach. Oczekuje przy tym na połączenie w pętli, wysyłając co 500 ms znak kropki na współpracujący terminal. Zazwyczaj połączenie następuje po 4 sekundach. Kolejnym krokiem jest przypisanie funkcji, które mają się wykonać, gdy serwer odbierze zapytanie. Zapytaniem domyślnym jest znak „/” – wtedy serwer przesyła stronę główną bez żadnej dodatkowej akcji. Gdy odbierze „relays”, serwer oczekuje na dodatkowe parametry dotyczące włączenia/wyłączenia przekaźników, wykonuje zadaną akcję i ponownie przesyła stronę startową. W przypadku błędnego adresu jest odsyłany komunikat „błąd 404” informujący o braku żądanej strony (**listing 1**).

Stronę sterującą wykonano w języku HTML. Zawiera 8 przycisków, których kliknięcie powoduje wywołanie odnośnika włączającego lub wyłączającego odpowiedni przekaźnik. Jego przesłaniem zajmuje się pokazana na **listingu 2** funkcja *handleRoot()*. Źródło strony zapisano w programie w postaci stałej typu string, stąd obecność znaków „/” na końcach linii i przed znakami specjalnymi. Wygląd strony głównej sterownika pokazano na **rysunku 10**.

Przemek Michalak
projektydmb.blogspot.com