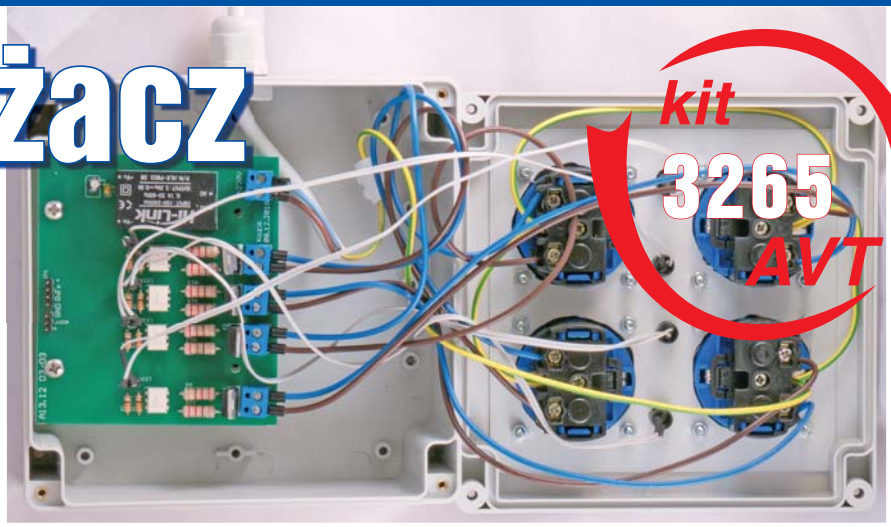
Przedłużacz Wi-Fi



Do czego służy?

Podstawowa funkcja przedłużacza jest doskonale znana. Jednak ten model pozwala dodatkowo na sterowanie podłączonymi urządzeniami poprzez stronę internetową. Dzięki wykorzystaniu obudowy i gniazd tablicowych przy starannym wykonaniu można otrzymać estetyczne i bezpieczne w użytkowaniu urządzenie. Przedłużacz Wi-Fi pozwala na zdalne sterowanie czterema urządzeniami podłączonymi do gniazd umieszczonych na jego obudowie. Dostęp do nich jest możliwy poprzez minimalistyczną, ale mam nadzieję, estetyczną stronę WWW. Pozwala ona sprawdzić aktualny stan oraz włączać lub wyłączać poszczególne urządzenia. Załączenie poszczególnych gniazd jest także sygnalizowane na diodach LED umieszczonych obok gniazd. Całość bazuje na module z bardzo popularnym układem ESP8266.

Jak to działa?

Schemat układu przedstawiony jest na **rysunku 1**. Najważniejszą częścią układu jest SJR1. Jest to moduł Wi-Fi o oznaczeniu ESP-12F. Bazuje on na popularnym układzie scalonym ESP8266. Składa się on z mało znanego 32-bitowego procesora Tensilica's L106 Diamond. W SoC-u znajduje się także pamięć SRAM oraz popularne peryferia, takie jak UART czy I2C. Jednak ten układ zawdzięcza swoją popularność zintegrowanemu z nim radiu Wi-Fi. Ma zarówno jego część cyfrową, jak i analogową. Na zewnątrz podłączona jest tylko antena. W naszym module jest ona wykonania ze ścieżki na płytce drukowanej. Niestety układ nie ma zintegrowanej pamięci nieulotnej. Dlatego w module umieszczona jest zewnętrzna pamięć Flash. Stąd wynika rozbieżność rozmiaru dostępnej pamięci pomiędzy modułami. W naszym układzie do dyspozycji mamy 4MB. W ostatnim czasie pojawiła się też nowa wersja: ESP8285, w której zintegrowana jest także pamięć nieulotna [1].

Wróćmy do schematu. Złącze JP1 pozwala na podłączenie portu szeregowego wykorzystywanego do programowania. Do aktywacji bootloadera służy zworka BOOT. Złącze RST pozwala na zresetowanie układu. Piny JP2 pozwalają na podłączenie zasilania układu na czas wgrywania oprogramowania.

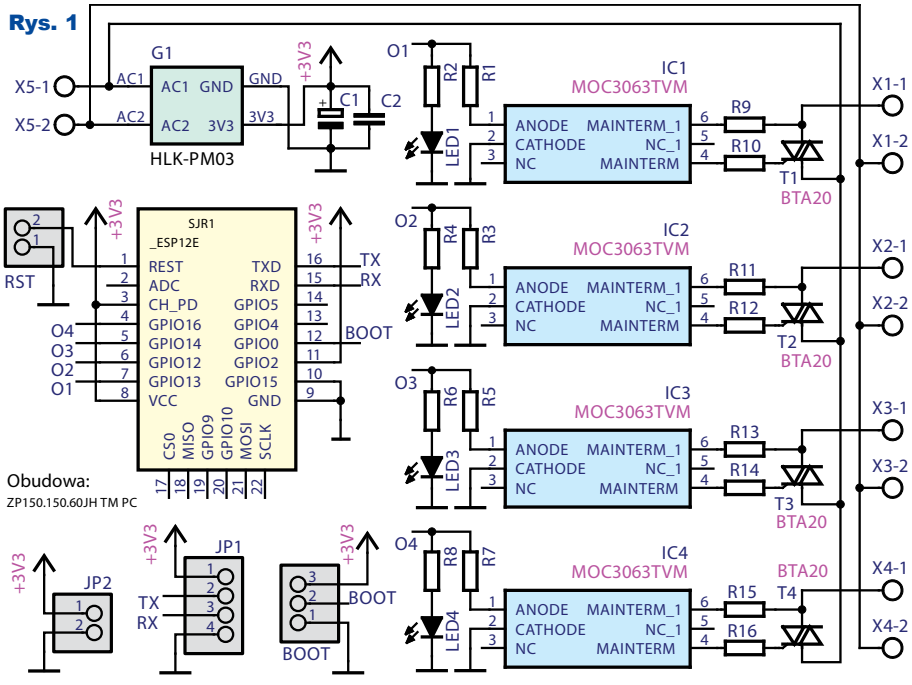
Drugi blok stanowią elementy wykonawcze. Izolację galwaniczną pomiędzy układem a sterowanymi urządzeniami zapewniają cztery optotriaki MOC3063 (IC1-IC4). Sterują one triakami BTA20 (T1-T4). Diody LED1–LED4 pokazują aktualny stan poszczególnych wyjść. Rezystory ograniczają prądy płynące przez diody LED (R2, R4, R6, R8), diody oświetlające optotriaki (R1, R3, R5, R7) oraz same optotriaki (R9-R16).

W czasie normalnej pracy stałe napięcie 3,3V jest dostarczane przez przetwornice impulsową HLK-PM03 (G1). Kondensatory C1 i C2 odpowiadają za jego filtrowanie.

Oprogramowanie dla układu ESP8266 zostało przygotowane w Arduino IDE [2]. Jego kod znajdziemy w materiałach dodatkowych oraz w repozytorium [3]. Główna część programu znajduje się w pliku *code.ino*. Na początku zdefiniowane są dwa makra. Pierwsze z nich `NETWORK_NAME` przechowuje nazwę sieci Wi-Fi, do której układ ma się połączyć, a drugie `NETWORK_PASS` jej hasło. Niżej znajduje się jeszcze zakomentowana definicja makra `DEBUG`. Jeżeli ją odkomentujemy, włączone zostanie wypisywanie na port szeregowy informacji o pracy programu.

Kod Arduino składa się zawsze z dwóch głównych części. Na początku uruchamiana jest jednorazowo funkcja `setup()`, której zadaniem jest konfiguracja środowiska. Następnie w nieskończonej pętli wywoływania jest funkcja `loop()` z główną częścią programu.

Rys. 1



W naszym przypadku funkcja `setup()` konfiguruje wyjścia sterujące gniazdami, łączy się z siecią Wi-Fi i inicjalizacje prace serwera. Jeżeli włączone jest debugowanie, to dodatkowo konfigurowany jest port szeregowy.

Następnie program przechodzi do funkcji `loop()`. Następuje tu sprawdzenie, czy nastąpiło odebranie zapytania od nowego klienta. Jeżeli tak, rozpoczyna się odczytywanie poszczególnych jego linii. Koniec zapytania sygnalizowany jest przesłaniem pustej linii. Następuje wtedy wygenerowanie strony WWW i przesłanie jej do klienta. Odpowiada za to funkcja `prepareHtmlPage`. Zwraca ona kod HTML strony WWW zawierającej informację o aktualnym stanie gniazd oraz pozwalającej na ich sterowanie.

Aby uniknąć problemów z kodowaniem, polskie znaki zostały zastąpione kodami HTML. Dla czytelności kody wszystkich „ogonków" zostały zdefiniowane jako makra w pliku `polish_chars.h`. Dzięki temu w kodzie strony internetowej możemy na przykład literę ł zastąpić makrem `l_`.

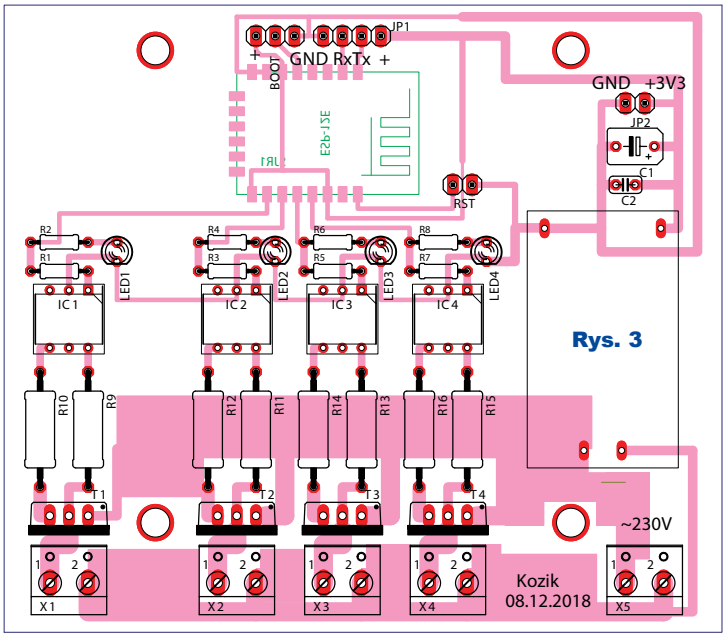
Ponieważ analizowanie kodu HTML umieszczonego wewnątrz programu napisanego w języku C++ jest mało intuicyjne, został on także umieszczony w pliku *index.html*. Można go lokalnie otworzyć w przeglądarce. Zobaczymy wtedy stronę podobną do pokazanej na **rysunku 2**. Wykorzystuje ona framework Bootstrap [4]. Ponieważ nie jest on przechowywany lokalnie, urządzenie, na którym ładujemy stronę, musi mieć dostęp do internetu. Logika strony została stworzona w języku JavaScript. Naciśnięcie przycisku powoduje wywołanie funkcji `change`, która wysyła do serwera (czyli naszego układu ESP8266) zapytanie GET z słowem kluczowym `socket`, numerem przełączanego gniazda oraz jego nowym stanem: `t` – włączony, `f` – wyłączony.

Możemy teraz znowu wrócić do kodu Arduino parsującego odebrane od klienta dane. Znajdziemy tam instrukcję warunkową, która sprawdza, czy w linii występuje słowo `socket`. Jeżeli tak, to następuje zmiana stanu odpowiedniego wyjścia.

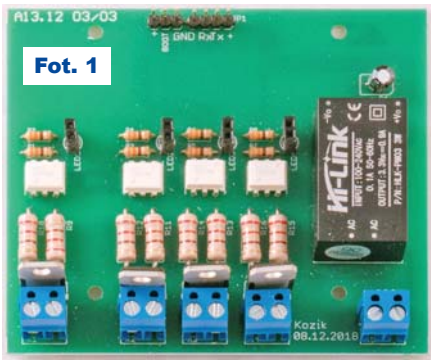
Rys. 2

Montaż i uruchomienie

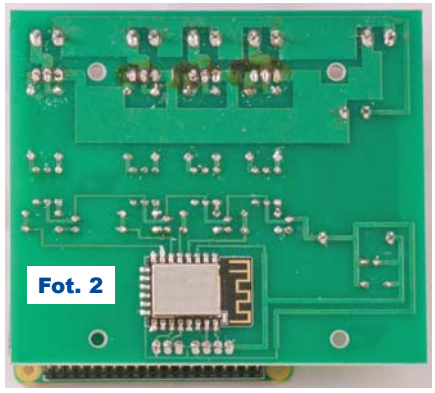
Projekt jednowarstwowej płytki PCB przedstawia **rysunek 3**. Montaż jest stosunkowo prosty. Wszystkie elementy poza modułem ESP-12F są przewlekane. Lecz nawet on nie powinien sprawić problemów podczas montażu, ponieważ odstęp pomiędzy jego wyprowadzeniami jest duży. Mimo to lutowanie warto rozpocząć właśnie od niego. Resztę elementów montujemy w kolejności od najmniejszych do największych. Ponieważ diody LED zostaną wyprowadzone na obudowę, zamiast nich lutujemy gniazdo goldpin. Do samych diod lutujemy około 30-centymetrowy przewód, który zakończamy wtykiem goldpin. Wszystkie luty warto zabezpieczyć koszulkami termokurczliwymi, aby uniknąć ewentualnych zwarć. Zmontowana płytka przedstawiona jest na **fotografiach 1** i **2**. Po zakończonym montażu, za pomocą multimetru, sprawdzamy, czy na płytce nie występują zwarcia. Jeżeli wszystko jest w porządku, możemy przejść do wgrania oprogramowania.



Rys. 3



Fot. 1

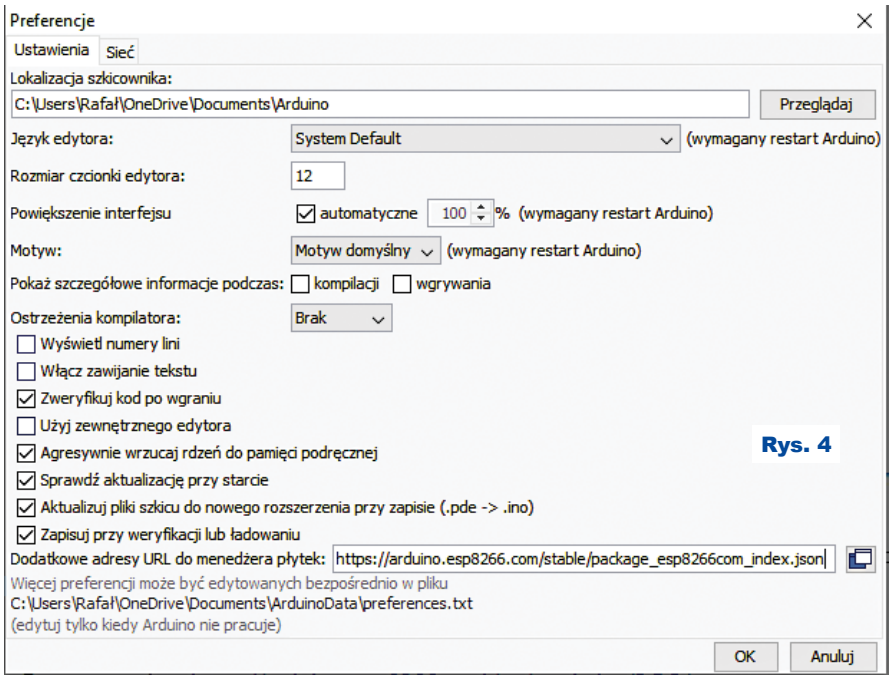

Fot. 2

Potrzebne nam będzie Arduino IDE w wersji co najmniej 1.8.7. Można je pobrać ze strony [5]. Następnie musimy doinstalować wsparcie dla ESP8266. Po włączeniu IDE, z menu *Plik* wybieramy opcję *Preferencje*. Pojawi się okno podobne do tego z **rysunku 4**. W pole *Dodatkowe adresy URL do menadżera płytek* wklejamy tekst:

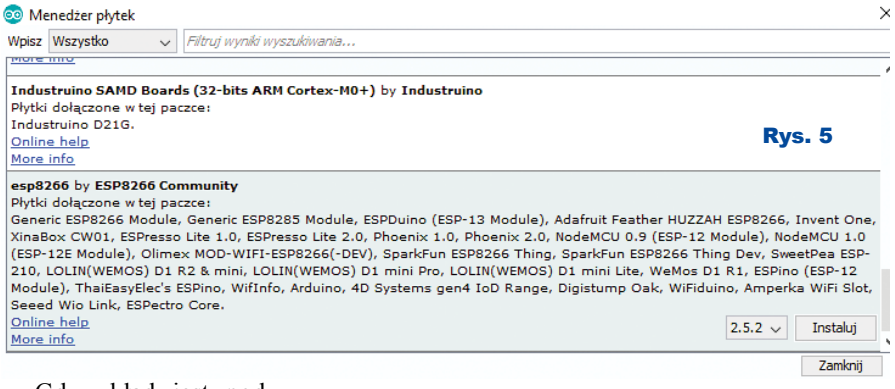
https://arduino.esp8266.com/stable/package_esp8266com_index.json

i zapisujemy, klikając OK. Następnie z menu *Narzędzia* wybieramy *Płytka:* i następnie *Menadżer płytek*.... W oknie (**rysunek 5**) zjeżdżamy na dół. Najeżdżamy na esp8266 i klikamy *zainstaluj*. Możemy także wybrać wersję modułu. Ja podczas tworzenia programu korzystałem z 2.5.2. Gdy instalacja się zakończy, zamykamy okno menadżera płytek. Teraz w menu *Narzędzia -> Płytka* pojawi się opcja *Generic ESP8266 Module*. Ustawiamy konfigurację zgodnie z przedstawioną na **rysunku 6**.

Podłączamy układ. Złącze JP1 podłączamy do komputera za pomocą konwertera USB/UART. Zasilanie (3,3V) podłączamy do pinów JP2. Ponieważ maksymalny prąd pobierany przez układ ESP8266 w czasie łączenia z siecią jest dość duży, nie możemy wykorzystać napięcia dostępnego na niektórych konwerterach USB/UART, ale musimy użyć zewnętrznego zasilacza.

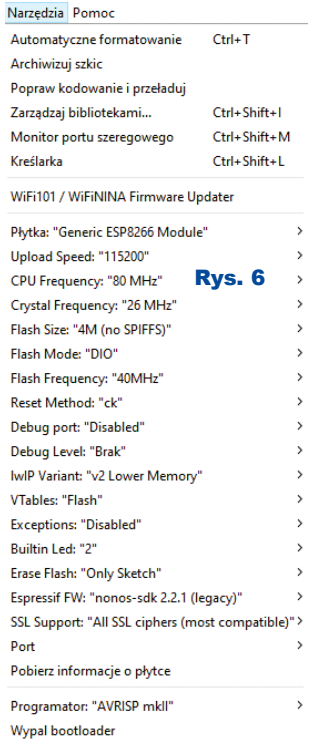


Rys. 4

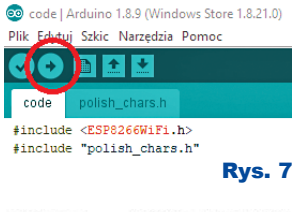


Rys. 5

Gdy układ jest podłączony, otwieramy w Arduino IDE szkic *code.ino*. Następnie musimy zmienić definicje makr `NETWORK_NAME` i `NETWORK_PASS`, aby odpowiadały parametrom naszej sieci Wi-Fi. W menu *narzędzia* → *port* wybieramy numer naszego konwertera. Następnie zwieramy złącze boot z masą i resetujemy układ poprzez krótkie zwarcie zworki rst. Rozpoczynamy programowanie, naciskając przycisk *Wgraj* (**rysunek 7**). Gdy zakończy się ono pomyślnie, zdejmujemy zworkę boot i ponownie resetujemy układ. Po włączeniu powinien od razu połączyć się z siecią Wi-Fi. Jeżeli odkomentowaliśmy makro `DEBUG`,

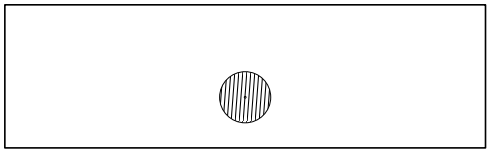


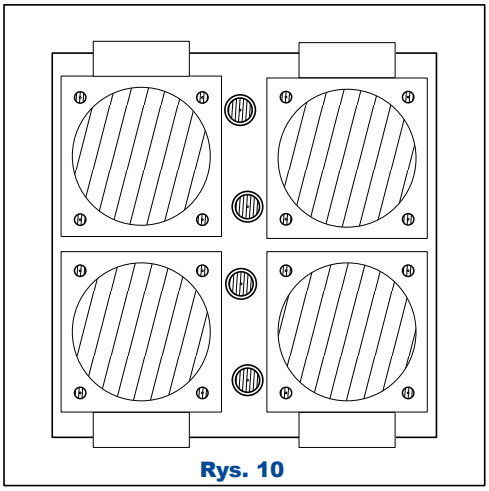
Rys. 6



Rys. 7

Rys. 9





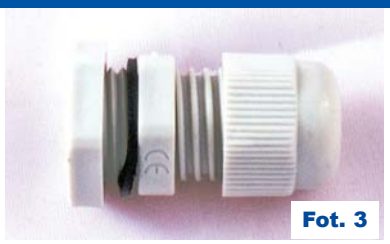
Rys. 10

informację o uzyskanym adresie IP znajdziemy na wyjściu portu szeregowego. Odpowiednie informacje znajdziemy także w konfiguracji naszego access pointu. W moim przypadku znajdują się w zakładce DHCP CLIENT DEVICES, co pokazuje **rysunek 8**. Jednak miejsce i sposób prezentacji tych informacji zależy od posiadanego sprzętu. Niektóre routery pozwalają także na przypisanie w serwerze DHCP stałego adresu IP dla zadanego adresu MAC.

Gdy wpiszemy uzyskany adres do przeglądarki, powinniśmy zobaczyć znaną nam już stronę internetową. Powinna ona także działać na telefonach komórkowych co prezentuje **rysunek 9**. Klikanie poszczególnych przełączników powinno powodować zaświecanie bądź gaszenie odpowiednich diod LED w urządzeniu.

Po wstępnych testach urządzenia możemy przystąpić do podłączenia gniazdek. Płytka drukowana została zaprojektowana do montażu wewnątrz obudowy ZP150.150.60 JH Kradex. Musimy jednak wykonać w niej odpowiednie otwory. Pomoże nam w tym szablon z **rysunku 10**. Znajdziemy go w materiałach dodatkowych do tego numeru w pliku *obudowa.svg*. Drukujemy go, pamiętając, aby wyłączyć w drukarce skalowanie. Następnie wycinamy i naklejamy na pokrywę oraz jeden z boków obudowy za pomocą kleju do papieru. Musimy wyciąć wszystkie zakreskowane elementy. Mniejsze otwory (pod śruby oraz oprawki diod LED) możemy wywiercić. Pozostałe najłatwiej wykonać za pomocą piły włoskowej po uprzednim nawierce-

Rys. 8


Fot. 3

niu. Po wykonaniu otworów odklejamy szablon. Jeżeli nie chce zejść na sucho, możemy opłukać obudowę pod ciepłą wodą.

W bocznej ściance montujemy dławik kablowy pokazany na **fotografii 3**. Służy on do wprowadzenia przewodu zasilającego. Do pokrywy za pomocą śrub o średnicy 3 mm przykręcamy gniazda tablicowe (**fotografia 4**). Następnie wkładamy diody LED w oprawkach. Aby dobrze się trzymały, można podkleić je klejem typu kropelka. Gotową obudowę przedstawia **fotografia 5**. Następnie przygotowujemy około 30cm odcinki izolowanego przewodu o przekroju co najmniej 0,5mm^2, które posłużą do połączenia gniazd z płytką. Warto także połączyć uziemienie gniazd tablicowych z bolcem uziemienia z wtyczki. Można do tego wykorzystać pojedynczą kostkę elektryczną. Na końcu za pomocą wkrętów o średnicy 3,5mm przykręcamy PCB wewnątrz obudowy. Gotowe połączenia prezentuje fotografia tytułowa. Przed podłączeniem do sieci energetycznej należy dokładnie sprawdzić, czy nie powstały żadne zwarcia. Układ jest już gotowy do pracy.

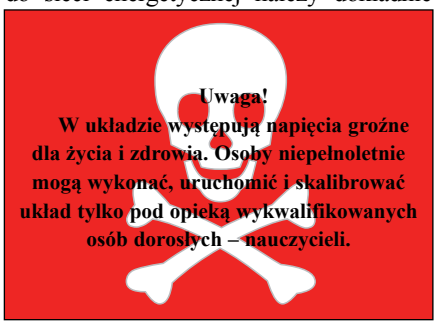

Fot. 4


Fot. 5

Wykaz elementów

R1-R8	330Ω/0,25W
R9-R16	220Ω/1W
C1	100uF
C2	100nF
T1-T4	BTA20
LED1-LED4	LED 5mm + oprawka LED
IC1-IC4	MOC3063
SJR1	ESP-12F
X1-X5	złącze ARK KF305 2pin h=12,5mm
G1	HLK-PM03
obudowa ZP150.150.60 JH Kradex	
4 ×	gniazdo tablicowe
1 ×	dławik kablowy PG9mm
20 ×	śruba M3/16mm + nakrętka M3
4 ×	wkręt 3,5/9mm
2m przewodu 3×	0,75mm2 ϕ 5,6
po 1,2 m przewodów 0,5mm^2 w kolorach brązowym, niebieskim, żółto-zielonym	
wtyczka na kabel 0	

Komplet podzespołów z płytką jest dostępny w Sklepie AVT jako zestaw AVT3265



Możliwości zmian

Kwestią ważną dla urządzeń IoT, a pominiętą w tym projekcie, jest bezpieczeństwo. Założyłem, że korzystamy z naszej prywatnej, bezpiecznej sieci Wi-Fi, do której nikt postronny nie ma dostępu. Jednak ciekawym zagadnieniem byłoby rozszerzenie oprogramowania o autoryzację i kontrolę dostępu. Zachęcam także do eksperymentowania z wyglądem i funkcjonalnościami strony WWW.

Rafał Kozik
rafkozik@gmail.com

[1] https://www.espressif.com/en/products/hardware/esp8266ex/overview
[2] https://github.com/esp8266/Arduino
[3] https://gitlab.com/kozik/wifi-extension-cord
[4] https://getbootstrap.com/
[5] https://www.arduino.cc/en/main/software