Mariusz Duka

Internet rzeczy

Podstawy programowania aplikacji
i serwerów sieciowych w językach C/C++,
MicroPython i Lua na urządzeniach
IoT ESP8266, ESP32 i Arduino

Helion

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/inrzpo
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-9674-6

Printed in Poland.

SPIS TREŚCI

2. OPROGRAMOWANIE

W niniejszym rozdziale przedstawię oprogramowanie, które wykorzystasz podczas nauki tworzenia aplikacji i serwerów sieciowych w językach C/C++, MicroPython i Lua na urządzeniach IoT. Najważniejszym z przedstawionych w tym rozdziale programów jest zintegrowane środowisko programistyczne Arduino IDE, ponieważ wykorzystasz je do programowania zarówno płytek Arduino, jak i płytek rozwojowych opartych na układach ESP8266, ESP8285 czy ESP32. Oprogramowanie Arduino IDE jest łatwe w obsłudze i nie powinno sprawić Ci większego problemu podczas instalacji, a ponadto świetnie nadaje się dla początkujących programistów C/C++ piszących programy dla urządzeń IoT.

Kolejnym narzędziem programistycznym jest PlatformIO IDE, które jako dodatek do Visual Studio Code stanowi ciekawą alternatywę dla Arduino IDE. Visual Studio Code to darmowy edytor kodu udostępniony przez firmę Microsoft, w wersjach dla systemów Windows, macOS i Linux. Edytor ten za pomocą odpowiednich rozszerzeń umożliwia kodowanie w wielu językach, takich jak Python, JavaScript, C/C++, Java, PHP i Lua. Wykorzystując PlatformIO IDE, rozszerzysz Visual Studio Code o możliwość programowania układów IoT, w tym Arduino.

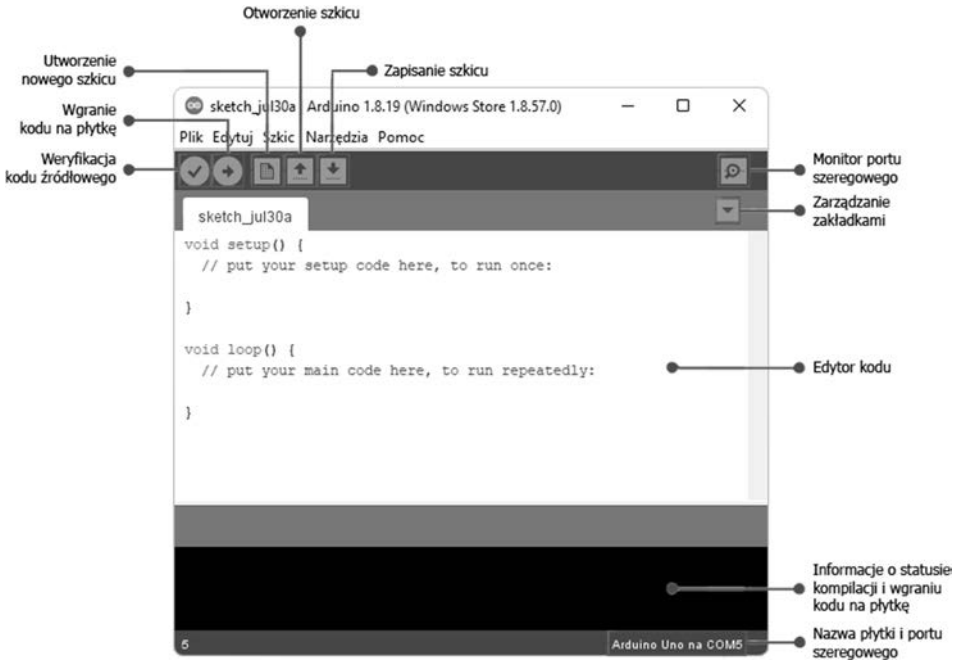
Choć Arduino IDE i PlatformIO IDE w zupełności wystarczą do napisania i wgrania programu do pamięci urządzenia IoT, warto zainstalować dodatkowe narzędzia. Umożliwią one aktualizację oprogramowania układowego (firmware'u) oraz wgranie interpretera języków MicroPython i Lua, a także ułatwią zarządzanie plikami w pamięci flash. Do takich narzędzi niewątpliwie zaliczamy ESPlorer, który pozwala tworzyć i uruchamiać kod napisany w językach Lua i MicroPython na płytkach rozwojowych NodeMCU opartych na układach ESP8266 i ESP8285 firmy Espressif Systems. Jeśli planujesz programować głównie w MicroPythonie, możesz skorzystać z narzędzi programistycznych uPyCraft i uPyLoader. Z kolei do wgrywania firmware'u wykorzystasz Flash Download Tool i NodeMCU Flasher. W przypadku korzystania z konkurencyjnych wobec ESP8266 układów W600 z pewnością przyda się EM-WiFi Configuration.

Oczywiście nie są to wszystkie programy umożliwiające zarządzanie modułami IoT, lecz w zupełności wystarczą do rozpoczęcia przygody z nauką ich programowania. Poza tym narzędzia te są łatwe w obsłudze i nie wymagają specjalistycznej wiedzy technicznej.

2.1. Arduino IDE

Zintegrowane środowisko programistyczne Arduino IDE[18] jest wykorzystywane przez wielu programistów, szczególnie początkujących, na całym świecie. Sama nazwa „Arduino" jest używana w odniesieniu zarówno do płytki Arduino, w tym najpopularniejszej Arduino UNO, jak i do całości systemu, w skład którego wchodzi oprogramowanie.

Oprogramowanie znajdziesz pod adresem *https://www.arduino.cc/en/software*. Na stronie tej są dostępne dwie wersje Arduino IDE: nowa wersja o numerze 2.0.3[19] i wcześniejsze wydanie o numerze 1.8.19. Zalecam instalację wersji starszej, ponieważ umożliwia ona aktywowanie rozszerzeń napisanych w języku Java do obsługi pamięci flash w modułach ESP i płytkach NodeMCU. Wersja 2.0.3 w chwili pisania niniejszej książki nie ma takiej możliwości. Oprogramowanie jest dostępne dla systemów Windows, macOS i Linux. Dla systemu Windows 10 i 11 instalator Arduino IDE 1.8.x znajdziesz również w sklepie Microsoft Store. Rysunek 2.1 przedstawia okno programu po instalacji w systemie Windows.

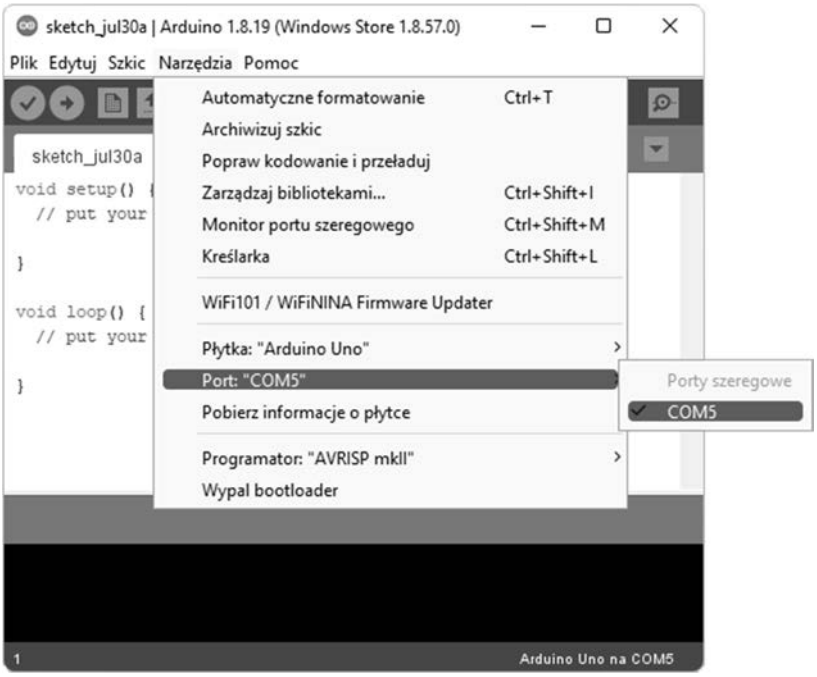


Rysunek 2.1. *Arduino IDE w systemie Windows*

[18] Arduino IDE, *https://www.arduino.cc/en/software*.

[19] Aktualne wersje IDE to 2.0.3 i 1.8.19, grudzień 2022.

Konfiguracja płytki rozwojowej w Arduino IDE polega na wybraniu w menu opcji *Narzędzia*, a następnie opcji *Płytka*. Z listy dostępnych płytek wybierz swoją, którą podłączyłeś do komputera. Kolejnym krokiem jest wskazanie za pomocą opcji *Port* portu szeregowego, pod którym płytka jest dostępna w systemie. W przypadku jednej podłączonej płytki do komputera jej konfiguracja w Arduino IDE jest bardzo łatwa. Rysunek 2.2 przedstawia informację, do którego portu szeregowego podłączona jest płytka rozwojowa Arduino UNO.

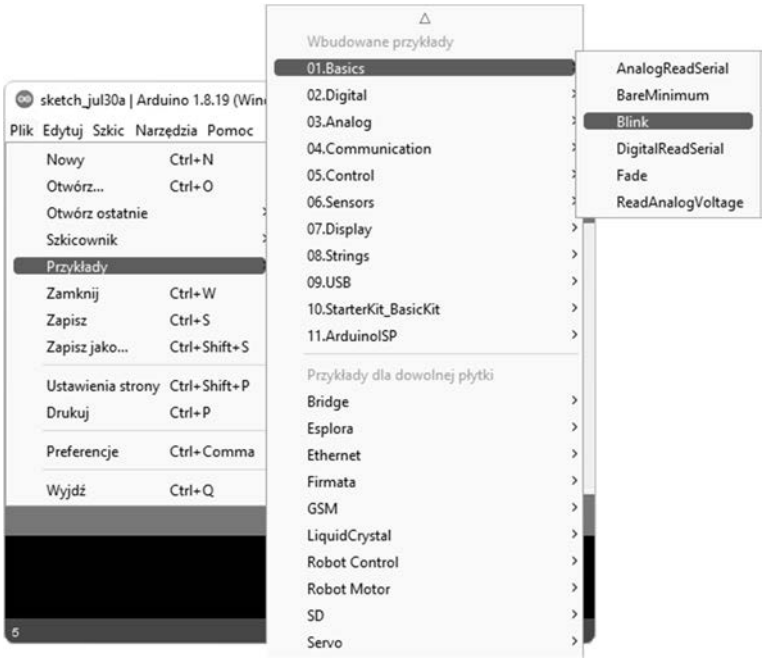


Rysunek 2.2. *Informacja o porcie szeregowym, do którego jest podłączona płytka Arduino UNO*

Arduino IDE wykorzystasz do programowania zarówno płytek Arduino, jak i modułów i płytek rozwojowych opartych na układach ESP. Projekty tworzone w środowisku Arduino IDE nazywane są **szkicami** (ang. *sketch*). Do programowania wykorzystasz język C i framework Arduino, umożliwiający stosowanie szeregu usprawnień, które pomagają pisać kod użytkownikom niezwiązanym z elektroniką. Proces kodowania polega na wyborze wersji płytki Arduino lub modułu ESP, a Arduino IDE odpowiednio skonfiguruje kompilator języka C/C++, dołączy odpowiednie biblioteki programistyczne oraz skompiluje kod i wgra do urządzenia.

Dla użytkownika dostępnych jest wiele przykładowych programów (szkiców), które można od razu wykorzystać i przetestować. W praktyce, jeśli zechcesz sprawdzić

działanie czujników, przekaźników lub diod LED, w Arduino IDE znajdziesz przykładowy kod testujący. Po zainstalowaniu i skonfigurowaniu środowiska programistycznego należy przetestować, czy kompilacja i wgranie na płytkę przykładowego kodu przebiegną bez problemów. Skorzystajmy z programu *Blink*, którego działanie polega na miganiu diody LED wbudowanej na płytce. Wybierz w menu opcję *Plik*, a następnie *Przykłady*. Pojawi się lista z przykładowymi szkicami, w której odszukaj *Blink*. Rysunek 2.3 przedstawia lokalizację programu w menu Arduino IDE.



Rysunek 2.3. *Lokalizacja przykładowego programu Blink w Arduino IDE*

Po wybraniu w menu przykładu *Blink* otworzy się nowe okno z kodem źródłowym. Teraz należy zweryfikować poprawność kodu i go skompilować. W tym celu trzeba kliknąć pierwszą ikonkę znajdującą się pod menu lub nacisnąć klawisze *Ctrl+R*. Jeśli w oknie z informacją o statusie kompilacji nie pojawią się komunikaty o błędach, wgraj skompilowany kod na płytkę, klikając drugą ikonkę pod menu lub naciskając na klawiaturze kombinację *Ctrl+U*. Zobaczysz pojawiające się informacje o statusie kompilacji i wgrywaniu kodu na płytkę. W moim przypadku, dla płytki Arduino UNO, końcowy komunikat wyglądał następująco:

```
Szkic używa 924 bajtów (2%) pamięci programu. Maksimum to 32256 bajtów.
Zmienne globalne używają 9 bajtów (0%) pamięci dynamicznej, pozostawiając
2039 bajtów dla zmiennych lokalnych. Maksimum to 2048 bajtów.
```

Komunikat informuje o maksymalnej wielkości programu po jego skompilowaniu do postaci kodu maszynowego (szesnastkowego), który możesz wgrać na płytkę do pamięci flash, oraz o wielkości dostępnej pamięci SRAM, przeznaczonej na wartości zmiennych programu.

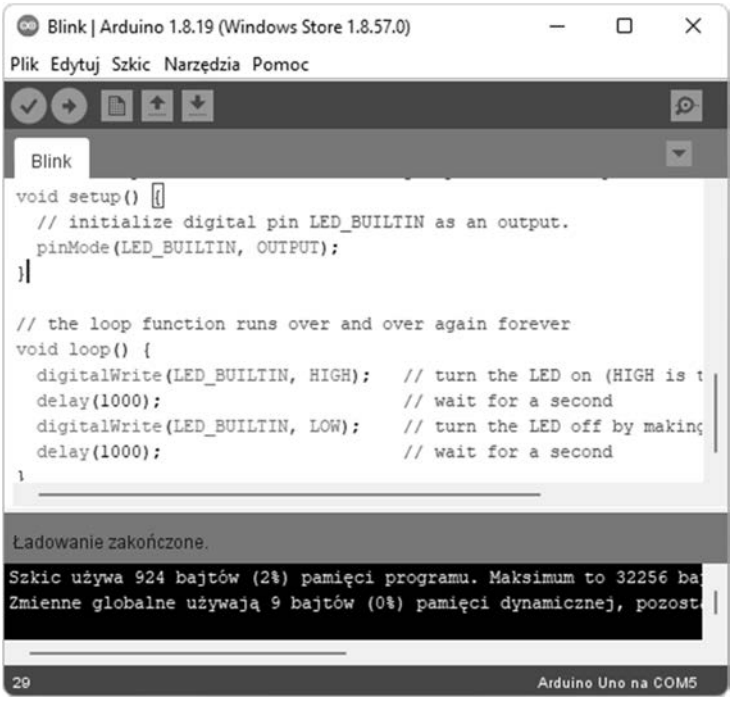
W przypadku płytki Arduino Nano komunikat wygląda następująco:

```
Szkic używa 924 bajtów (3%) pamięci programu. Maksimum to 30720 bajtów.
Zmienne globalne używają 9 bajtów (0%) pamięci dynamicznej, pozostawiając
2039 bajtów dla zmiennych lokalnych. Maksimum to 2048 bajtów.
```

Natomiast w przypadku płytki Arduino MEGA 2560 Rev3 tak:

```
Szkic używa 1536 bajtów (0%) pamięci programu. Maksimum to 253952 bajtów.
Zmienne globalne używają 9 bajtów (0%) pamięci dynamicznej, pozostawiając
8183 bajtów dla zmiennych lokalnych. Maksimum to 8192 bajtów.
```

Od razu można zauważyć różnicę w wielkość dostępnej pamięci, a także w wielkości programu skompilowanego do postaci kodu maszynowego. Rysunek 2.4 przedstawia okno narzędzia Arduino IDE po pomyślnym wgraniu kodu na płytkę Arduino UNO.



Rysunek 2.4. *Informacja o prawidłowej kompilacji i wgraniu kodu na płytkę Arduino UNO*

W tym momencie dioda LED na Twojej płytce powinna migać z częstotliwością co 1 sekundę. Możesz poeksperymentować, zmieniając wartość parametru funkcji `delay()` w kodzie programu źródłowego.

Wskazówka

Informacje na temat dostępnej pamięci na płytkach rozwojowych Arduino UNO, Nano i Mega 2560 Rev3 znajdziesz podrozdziale 1.1, „Platforma Arduino".

2.1.1. Instalacja oprogramowania dla ESP8266 i ESP32

Arduino IDE pozwala na instalację dodatkowego oprogramowania umożliwiającego kompilację kodu źródłowego dla modułów i płytek rozwojowych z układami z serii ESP8266 i ESP32, takich jak ESP-01 czy NodeMCU. W tym celu należy wybrać w menu opcję *Plik*, a następnie *Preferencje* lub nacisnąć na klawiaturze kombinację *Ctrl+,* (znak przecinka). Otworzy się okno, w którym w polu *Dodatkowe adresy URL do menedżera płytek* należy wstawić następujące adresy:

```
https://raw.githubusercontent.com/espressif/arduino-esp32/
gh-pages/package_esp32_index.json

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

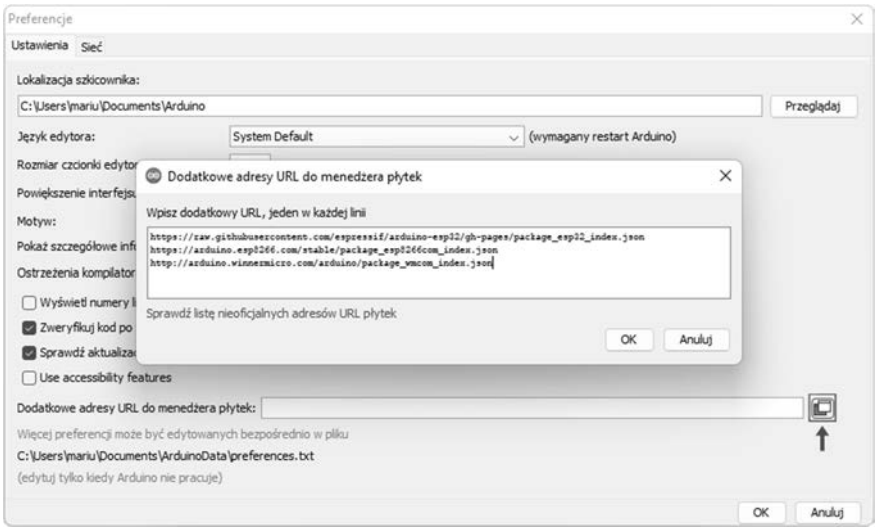
Adresy te dotyczą oprogramowania dla płytek rozwojowych z układami z serii ESP8266 i ESP32. Dodatkowe adresy URL mogą dotyczyć również innych urządzeń, takich jak choćby płytki Omega2 lub W600. Jeśli zamierzasz programować moduły ESP-01W, dodaj następujący adres URL:

```
http://arduino.winnermicro.com/arduino/package_wmcom_index.json
```

Rysunek 2.5 przedstawia okno narzędzia Arduino IDE, w którym możesz wprowadzić dodatkowe adresy URL do menedżera płytek.

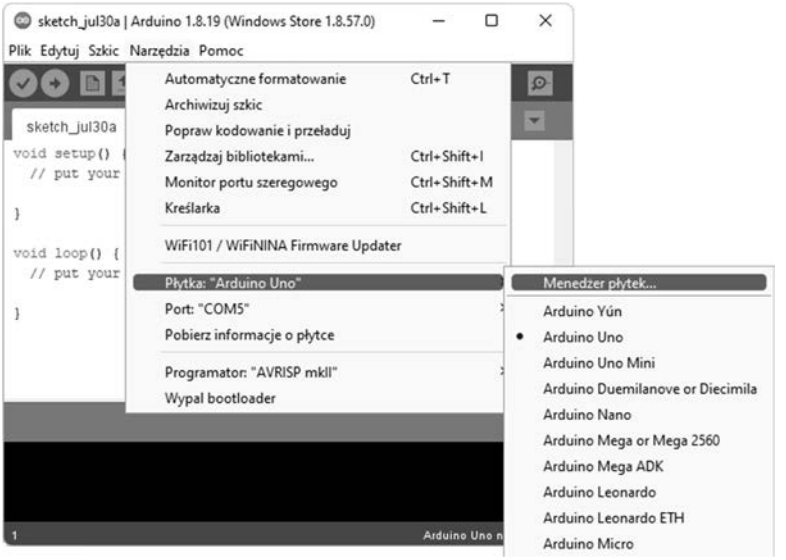
Wskazówka

Zwróć uwagę na zawartość pola *Lokalizacja szkicownika*. Wskazuje ono katalog, w którym są przechowywane szkice z kodem źródłowym. W katalogu tym możesz zainstalować wtyczki do obsługi systemu plików, wykorzystując SPIFFS (ang. *SPI Flash File System*) lub jego nowszą wersję LittleFS.



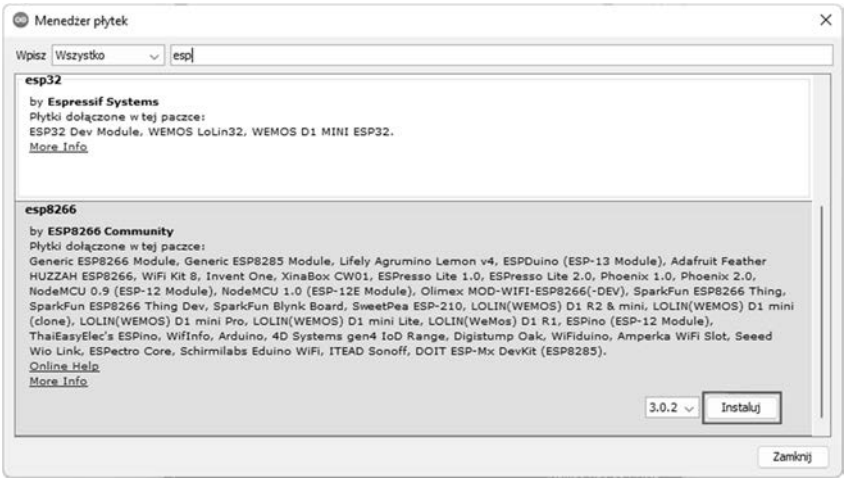
Rysunek 2.5. *Dodanie adresów URL do bibliotek programistycznych dla układów z serii ESP*

Po wpisaniu dodatkowych adresów URL do menedżera płytek wybierz w menu opcję *Narzędzia*/*Płytka*/*Menedżer płytek* (zobacz rysunek 2.6).



Rysunek 2.6. *Lokalizacja menedżera płytek w menu Arduino IDE*

Wyszukaj oprogramowanie dla płytek z układem ESP8266 lub ESP32, wpisując w polu wyszukiwania „esp”. Pojawią się dwie możliwości — *esp32* i *esp8266* (zobacz rysunek 2.7), zainstaluj tę, z której będziesz korzystać (najlepiej obie, aby instalację mieć już za sobą).

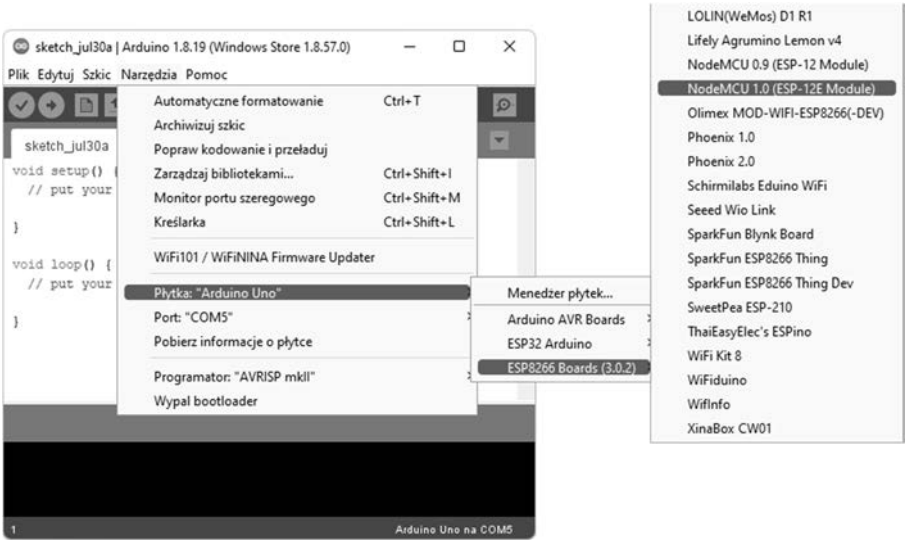


Rysunek 2.7. *Menedżer płytek w Arduino IDE*

Wskazówka

Instalacja oprogramowania dla obsługi płytek z układami ESP8266 i ESP32 (prawie 300 MB), w zależności od szybkości Twojego połączenia z siecią, może potrwać do kilku minut. W przypadku oprogramowania dla układów W600 mogą wystąpić problemy z łącznością z serwerem *arduino.winnermicro.com*.

Po instalacji oprogramowania wybierz z listy płytkę, którą masz podłączoną do komputera. W moim przypadku jest to NodeMCU 1.0 (zobacz rysunek 2.8).



Rysunek 2.8. *Wybór płytki rozwojowej NodeMCU 1.0 w Arduino IDE*

Po wybraniu płytki możesz zweryfikować poprawność jej aktywacji i konfiguracji w Arduino IDE. Najłatwiej zrobić to programem włączającym okresowo diodę LED na module ESP, czyli nieśmiertelnym *Blink*. Przepisz poniższy kod źródłowy do edytora, a następnie skompiluj i wgraj na płytkę.

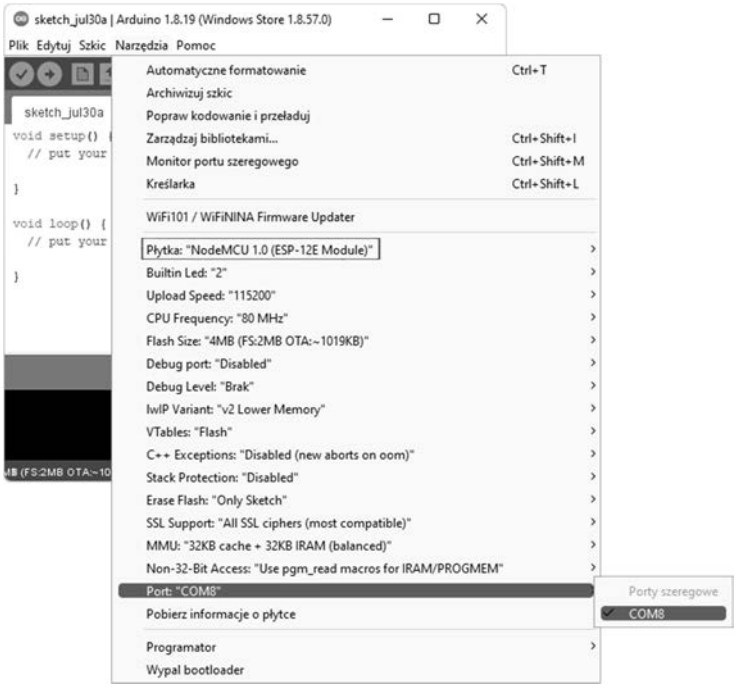
Listing 2.1. *Program Blink w języku C (Arduino) w środowisku Arduino IDE*

```
int pin = LED_BUILTIN; // numer GPIO dla diody LED

void setup() {
  pinMode(pin, OUTPUT); // ustaw GPIO jako wyjście
}

void loop() {
  digitalWrite(pin, HIGH); // włącz diodę LED stanem wysokim
  delay(1000);             // odczekaj 1 sekundę
  digitalWrite(pin, LOW);  // wyłącz diodę LED stanem niskim
  delay(1000);             // odczekaj 1 sekundę
}
```

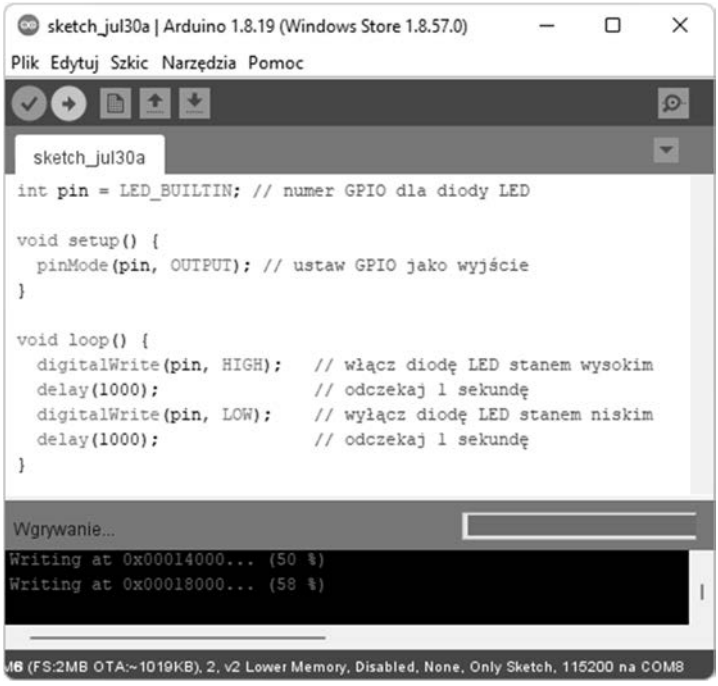
Podłączając kolejną płytkę do komputera, zwłaszcza jeśli masz dostępnych kilka portów USB, należy sprawdzić, czy w Arduino IDE prawidłowo jest wskazany port szeregowy. W moim przypadku płytka była podłączona do portu COM8, co widoczne jest na rysunku 2.9.



Rysunek 2.9. *Ustawienie portu szeregowego, do którego podłączona jest płytka rozwojowa*

Zapewne nie uszło Twojej uwadze, że dla płytek z układami ESP dostępnych jest znacznie więcej ustawień niż dla płytek Arduino. Możesz wskazać numer wyprowadzenia GPIO dla wbudowanej diody LED, zmienić prędkość transmisji UART podczas wgrywania kodu, zmienić częstotliwość taktowania procesora, jak również skonfigurować pamięć flash tak, aby możliwe było utworzenie prostego systemu plików. Dostępne są również ustawienia wykorzystywane do debugowania kodu i wsparcia dla szyfrowania SSL.

Informacje o statusie kompilacji i wgrywania programu z listingu 2.1 na płytkę będą pojawiać się w oknie Arduino IDE. W przypadku układów z serii ESP8266 i ESP32 do wgrywania kodu jest wykorzystywane narzędzie `esptool`, napisane w języku Python. Wykorzystasz je później do aktualizacji oprogramowania układowego (firmware'u). Rysunek 2.10 przedstawia moment wgrywania programu na płytkę rozwojową NodeMCU 1.0 z widocznymi informacjami o postępie.



Rysunek 2.10. *Informacja o postępie wgrywania programu na płytkę NodeMCU 1.0*

W tym momencie masz zainstalowane środowisko programistyczne Arduino IDE z dodatkowym oprogramowaniem do obsługi płytek z układami ESP firmy Espressif Systems. Od teraz możesz pisać i testować swoje pierwsze programy i uruchamiać je na urządzeniach internetu rzeczy.

W Arduino IDE, za pomocą skrótu *Ctrl+Shift+M*, jest dostępny monitor portu szeregowego (ang. *serial monitor*), który będziesz wykorzystywać do monitorowania i komunikacji z programem uruchomionym na płytce. Monitor pomaga debugować kod i możesz go traktować jako konsolę tekstową dla swoich programów. Dostępna jest również kreślarka (ang. *serial plotter*), za pomocą której możesz generować kolorowe wykresy na podstawie danych uzyskanych z programu. Z pewnością prezentacja danych na kreślarce lepiej wygląda niż pojawiające się liczby w monitorze portu szeregowego. Arduino IDE umożliwia również aktualizację kodu bootloadera odpowiedzialnego za wgrywanie kodu z pominięciem programatora.

2.1.2. Obsługa systemów plików SPIFFS i LittleFS

Projektując aplikacje dla układów z serii ESP, w których część odpowiedzialną za komunikację z użytkownikiem będziesz realizować za pomocą strony WWW, z pewnością będziesz potrzebować miejsca na zapisanie plików z kodem HTML, CSS i JavaScript. Jak już wiesz, w pamięci flash dostępnej w modułach ESP i na płytkach rozwojowych możesz utworzyć systemy plików SPIFFS lub LittleFS, które ułatwiają zarządzanie strukturą plików strony WWW lub dowolnych danych zapisywanych w plikach tekstowych.

Obsługa systemów plików SPIFFS i LittleFS poprzez Arduino IDE wymaga pobrania wtyczek napisanych w języku Java i zapisania ich w lokalizacji szkicownika (ang. *sketchbook location*), czyli w katalogu, w którym są przechowywane szkice z kodem źródłowym. Informację o lokalizacji tego katalogu uzyskasz w oknie z preferencjami Arduino IDE — w tym celu należy wybrać w menu opcję *Plik*, a następnie *Preferencje* lub nacisnąć na klawiaturze kombinację *Ctrl+,* (znak przecinka).

System plików SPIFFS

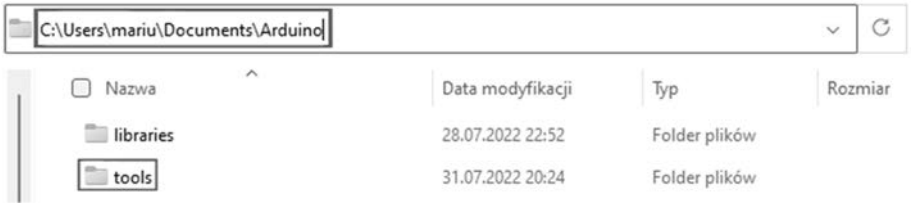
Pobierz najnowszą wersję wtyczki dla ESP8266 i ESP8285 spod adresu:

```
https://github.com/esp8266/arduino-esp8266fs-plugin/releases
```

Natomiast dla układów z serii ESP32 spod adresu:

```
https://github.com/me-no-dev/arduino-esp32fs-plugin
```

W lokalizacji szkicownika utwórz podkatalog o nazwie *tools* (zobacz rysunek 2.11) i wgraj do niego pobraną wcześniej wtyczkę (archiwum w formacie JAR).

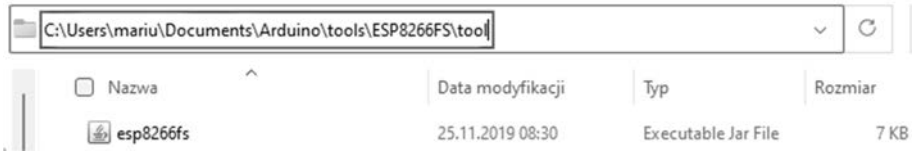


Rysunek 2.11. *Miejsce utworzenia katalogu o nazwie tools w lokalizacji szkicownika*

Przykładowa struktura katalogów dla pobranego oprogramowania ESP8266FS w systemie Windows wygląda następująco:

```
C:\Users\mariu\Documents\Arduino
C:\Users\mariu\Documents\Arduino\tools
C:\Users\mariu\Documents\Arduino\tools\ESP8266FS
C:\Users\mariu\Documents\Arduino\tools\ESP8266FS\tool
C:\Users\mariu\Documents\Arduino\tools\ESP8266FS\tool\esp8266fs.jar
```

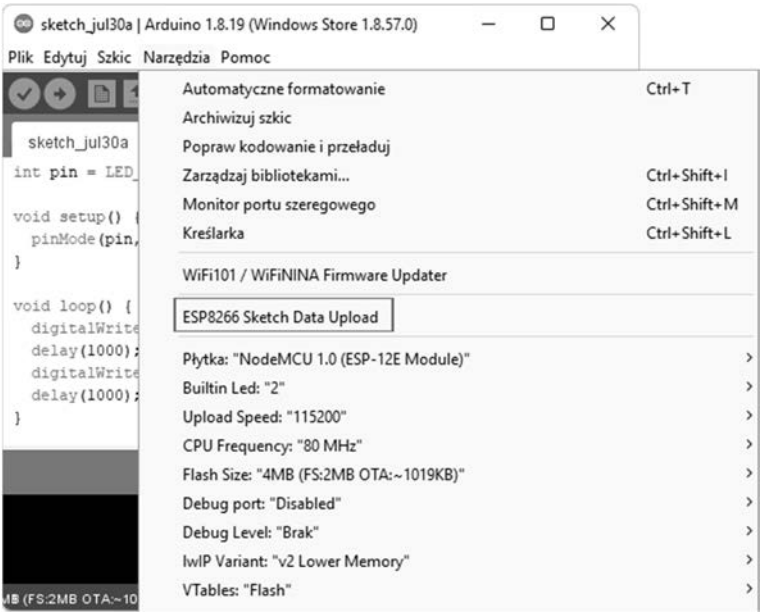
Oczywiście tak wygląda to u mnie. U Ciebie z pewnością nazwa katalogu szkicownika będzie inna. W systemie Windows lokalizację szkicownika najszybciej odnajdziesz w katalogu z dokumentami, ponieważ jest to domyślna lokalizacja w Arduino IDE. Rysunek 2.12 przedstawia lokalizację wtyczki z zaznaczoną pełną ścieżką dostępu.



Rysunek 2.12. *Lokalizacja archiwum w formacie JAR z wtyczką ESP8266FS*

Po prawidłowym utworzeniu struktury katalogów uruchom ponownie Arduino IDE. W menu *Narzędzia* powinna pojawić się nowa opcja, *ESP8266 Sketch Data Upload* (zobacz rysunek 2.13), co będzie oznaczać, że wtyczka została poprawnie zainstalowana.

Od tej chwili masz możliwość przeniesienia struktury plików do pamięci układu ESP. Struktura taka powinna znajdować się w podkatalogu o nazwie *data* umieszczonym w katalogu ze szkicem. Jej zawartość może zawierać pliki tekstowe z kodem strony WWW, pliki graficzne, a nawet pliki muzyczne — wszystko zależy od wielkości dostępnej pamięci flash i przydzielonego miejsca na dane. W komputerze struktura ta może być umieszczona w podkatalogach, natomiast w chwili przenoszenia jej do pamięci SPIFFS zostanie przekonwertowana na wersję plikową.



Rysunek 2.13. *Widok menu Narzędzia z aktywną opcją ESP8266 Sketch Data Upload*

Przykładowo, jeśli lokalizacja katalogu ze szkicem jest następująca:

```
C:\Users\mariu\Documents\Arduino\sketch_jul30a
```

to struktura plików powinna znajdować się w podkatalogu *data*:

```
C:\Users\mariu\Documents\Arduino\sketch_jul30a\data
```

Jeśli nie utworzysz tego katalogu samodzielnie dla konkretnego projektu (szkicu), uruchomienie opcji *ESP8266 Sketch Data Upload* spowoduje pojawienie się okna z komunikatem w języku angielskim *No files have been found in your data folder! Are you sure you want co create an empty SPIFFS image?*, informującym, że w katalogu ze szkicem nie odnaleziono katalogu o nazwie *data*. Wybór przycisku *Yes* spowoduje utworzenie takiego katalogu i wgranie pustej struktury do pamięci układu ESP, co będzie równoznaczne z jej sformatowaniem.

Podsumowując, kompletna struktura plików może wyglądać następująco:

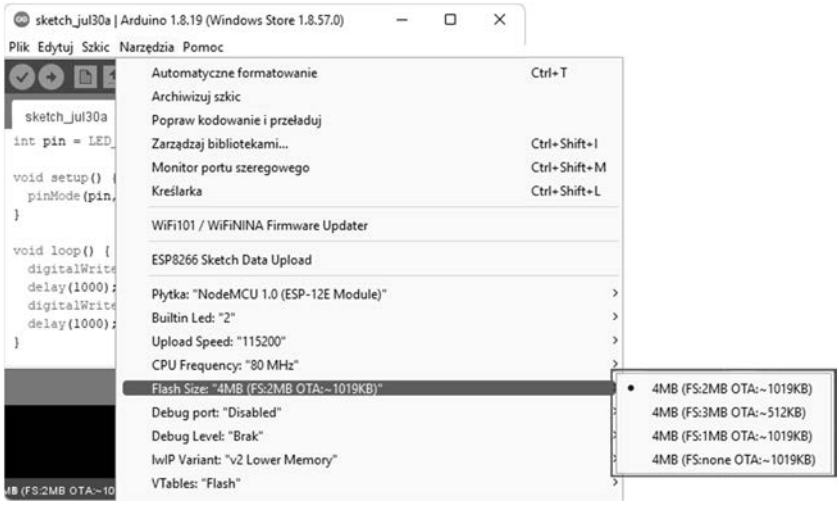
```
C:\Users\mariu\Documents\Arduino\sketch_jul30a\data
C:\Users\mariu\Documents\Arduino\sketch_jul30a\data\index.html
C:\Users\mariu\Documents\Arduino\sketch_jul30a\data\admin.html
C:\Users\mariu\Documents\Arduino\sketch_jul30a\data\css\style.css
C:\Users\mariu\Documents\Arduino\sketch_jul30a\data\js\script.js
C:\Users\mariu\Documents\Arduino\sketch_jul30a\data\img\logo.png
C:\Users\mariu\Documents\Arduino\sketch_jul30a\data\log\data.txt
```

Pierwsze dwa pliki, *index.html* i *admin.html*, znajdują się w katalogu głównym, pozostałe zaś w podkatalogach. W takiej strukturze możesz umieścić naprawdę bardzo dużo informacji dla rozbudowanych stron WWW.

Wskazówka

W wielu projektach kod HTML strony WWW jest umieszczany w kodzie aplikacji, a dokładnie w zmiennych, co znacznie utrudnia aktualizację kodu. Poza tym zmienne te zajmują i tak ograniczoną pamięć RAM, co wpływa na wydajność takiej aplikacji. Stosowanie SPIFFS lub LittleFS umożliwia swobodny dostęp do plików, ich aktualizację i usunięcie w celu zwolnienia pamięci, podobnie jak na zwykłym dysku twardym w komputerze.

Przed uruchomieniem procedury przenoszenia plików do systemów SPIFFS lub LittleFS należy zdecydować o podziale pamięci flash. Z reguły masz kilka wariantów do wyboru; na rysunku 2.14 są przedstawione możliwości dla płytki rozwojowej NodeMCU 1.0 z układem ESP-12E. Zwróć uwagę, że możesz wybrać wariant bez przydziału miejsca na system plików, pozostawiając tylko pamięć OTA.



Rysunek 2.14. *Wybór podziału pamięci flash na płytce rozwojowej NodeMCU 1.0*

System plików LittleFS

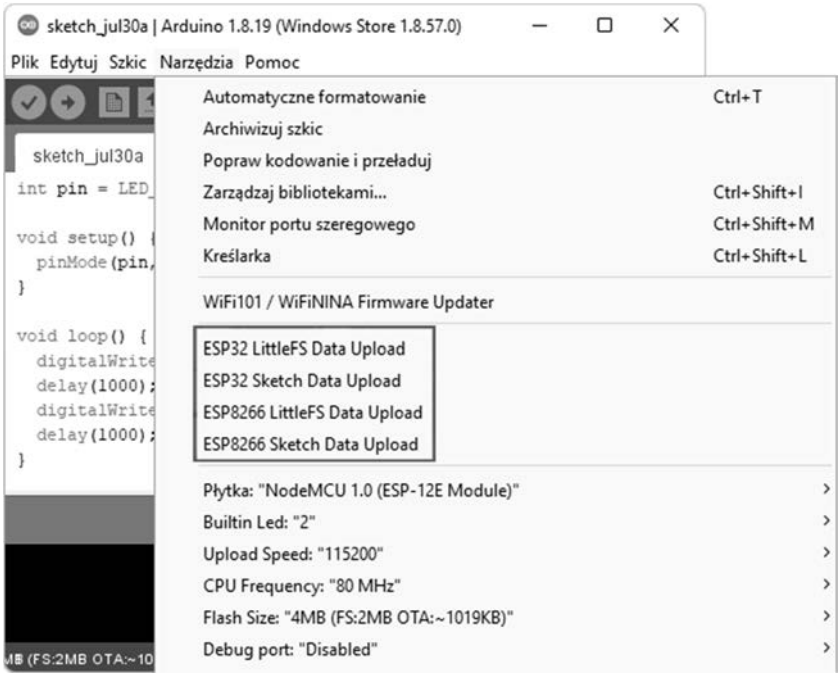
Pobierz najnowszą wersję wtyczki dla ESP8266 i ESP8285 spod adresu:

```
https://github.com/earlephilhower/arduino-esp8266littlefs-plugin/releases
```

Natomiast dla układów z serii ESP32 spod adresu:

```
https://github.com/lorol/arduino-esp32littlefs-plugin
```

Instalacja wtyczki (archiwum JAR) obsługującej system plików LittleFS przebiega podobnie jak w przypadku SPIFFS. Rysunek 2.15 przedstawia wygląd menu po aktywacji wtyczek dla wszystkich wariantów.



Rysunek 2.15. *Wygląd menu po aktywacji obsługi systemów plików SPIFFS i LittleFS*

Wskazówka

Aby ułatwić Ci zadanie z pobraniem wszystkich plików niezbędnych do aktywacji wtyczek obsługujących systemy SPIFFS i LittleFS w Arduino IDE, umieściłem je w archiwum z kodem źródłowym. Pobierz je ze strony wydawnictwa Helion i przegraj do katalogu z lokalizacją szkicownika.

Zadaniem wtyczek (zapisanych w archiwum JAR) jest uruchomienie wcześniej pobranych narzędzi `mkspiffs` i `mklittlefs`, które wchodzą w skład oprogramowania dla układów z serii ESP. Po prostu nie uruchamiamy ich ręcznie w linii poleceń, lecz są one wykonywane z poziomu Arduino IDE.

Internet rzeczy to przykład koncepcji, która — w odróżnieniu od wielu innych relatywnie młodych dziedzin IT — została więcej niż pozytywnie zweryfikowana przez czas i rynek. Już dziś IoT znajduje bardzo szerokie zastosowanie w wielu obszarach życia: od projektów stricte hobbystycznych, przez automatykę domową, handel i usługi, po systemy inteligentnych miast, przemysł i rolnictwo.

Internet rzeczy nadal ewoluuje – i staje się coraz powszechniejszy, a to może być zachętą do tego, by poznać go bliżej. *Internet rzeczy. Podstawy programowania aplikacji i serwerów sieciowych w językach C/C++, MicroPython i Lua na urządzeniach IoT ESP8266, ESP32 i Arduino* może się okazać świetną propozycją na początek przygody z IoT. Książka kompleksowo objaśnia specyfikę wybranych urządzeń i uczy, jak je programować, w sposób na tyle przystępny, że wystarczy przeciętna znajomość obsługi komputera, by zacząć tworzyć pierwsze projekty. Treść została zilustrowana przykładowymi kodami źródłowymi, co zdecydowanie ułatwia stawianie pierwszych kroków.

Dzięki książce:

- poznasz wybrane urządzenia IoT
- zaznajomisz się z narzędziami programistycznymi
- opanujesz podstawy języków programowania
- uruchomisz własny serwer dla aplikacji web
- stworzysz serwer WWW na urządzeniu IoT
- a nawet zaprojektujesz serwer Minecrafta!

Internet rzeczy ma przed sobą wielką przyszłość, bądź jej częścią!

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶

ISBN 978-83-283-9674-6

9 788328 396746

Cena: 99,00 zł