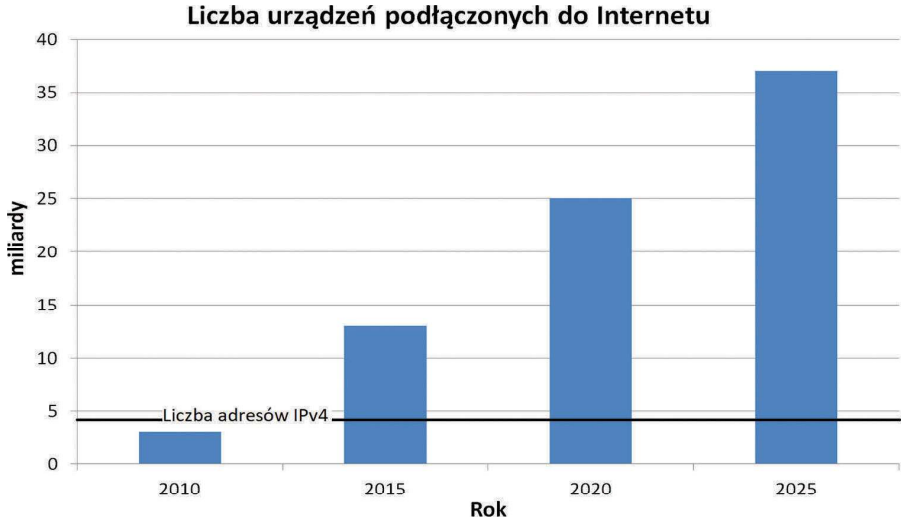
9. URZĄDZENIA INTERNETU RZECZY

Koncepcja Internetu rzeczy zakłada tworzenie sieci urządzeń (rzeczy), które mogą się komunikować przez Internet. Rzeczami internetowymi mogą być obiekty fizyczne wyposażone w interfejs komunikacyjny i jeden lub kilka następujących elementów:

- ♦ czujniki (temperatury, zanieczyszczeń, ciśnienia, wilgotności, położenia, drgań itp.),
- ♦ elementy wykonawcze (przekaźniki, sterowane zawory, silniki itp.),
- ♦ elementy sygnalizacyjne,
- ♦ elementy obliczeniowe.

Internet rzeczy jest systemem obiektów fizycznych podłączonych do globalnego Internetu, które można odkrywać, monitorować i kontrolować za pośrednictwem różnych interfejsów sieciowych. Czujniki i moduły komunikacji sieciowej stają się coraz tańsze i mogą być wbudowywane w coraz więcej obiektów fizycznych.

Według raportu Davea Evansa [7] w 2008 roku liczba rzeczy podłączonych do Internetu przewyższyła liczbę mieszkańców Ziemi. Prognoza Evansa na rok 2020 to 50 miliardów urządzeń podłączonych do Internetu. Nowsze analizy, m.in. *Strategy Analytycs*, szacują, że w roku 2020 do Internetu podłączonych jest 25 miliardów urządzeń. Wszystkie prognozy są zgodne co do tego, że liczba takich urządzeń stale rośnie i będzie rosła.



Rys. 9.1. Prognoza liczby urządzeń podłączonych do Internetu

Według *Strategy Analytycs* liczba smartfonów z dostępem do Internetu wielokrotnie przewyższa dziś liczbę komputerów PC. Największą grupą urządzeń podłączonych do Sieci są urządzenia Internetu rzeczy wykorzystywane w biznesie (Enterprise IoT), drugą w kolejności grupę stanowią inteligentne urządzenia domowe (Smart Home Devices).

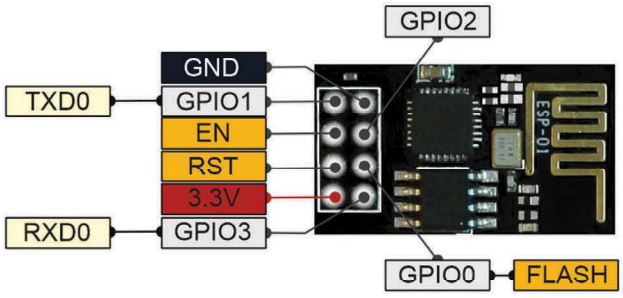
Aby rzeczy internetowe mogły się łączyć z Internetem, muszą być wyposażone w moduły zapewniające interfejs komunikacyjny. Bardzo popularne są moduły Wi-Fi, niewymagające połączenia przewodowego.

Popularne moduły Wi-Fi

Doniosły wpływ na rozwój Internetu rzeczy miało pojawienie się 30 grudnia 2013 roku taniego chińskiego układu scalonego ESP8266, wyprodukowanego przez firmę Espressif Systems [5]. Oto jego podstawowe parametry:

- napięcie zasilania: 3,3 V,
- wbudowany 32-bitowy mikroprocesor RISC Tensilica L106 o częstotliwości taktowania 80 lub 160 MHz,
- pamięć:
  - 32 KiB instruction RAM, 32 KiB instruction cache RAM,
  - 80 KiB user-data RAM,
  - 16 KiB system-data RAM,
- zewnętrzna pamięć Flash do 16 MiB (typowo od 512 KiB do 4 MiB),
- Wi-Fi IEEE 802.11 b/g/n,
- 16 pinów ogólnego przeznaczenia GPIO,
- interfejsy: SPI, I²C (realizacja softwarowa), I²S i UART,
- 10-bitowy przetwornik A/C.

W sierpniu 2014 na rynku pojawił się, wyprodukowany przez Ai-Thinker, moduł Wi-Fi ESP-01 wykorzystujący układ ESP8266. Obecnie moduł ten można nabyć w Polsce za ok. 10 zł. Ma on wbudowaną antenę i osiem wyprowadzeń, rys. 9.2.



Rys. 9.2. Schemat wyprowadzeń modułu ESP-01[16]

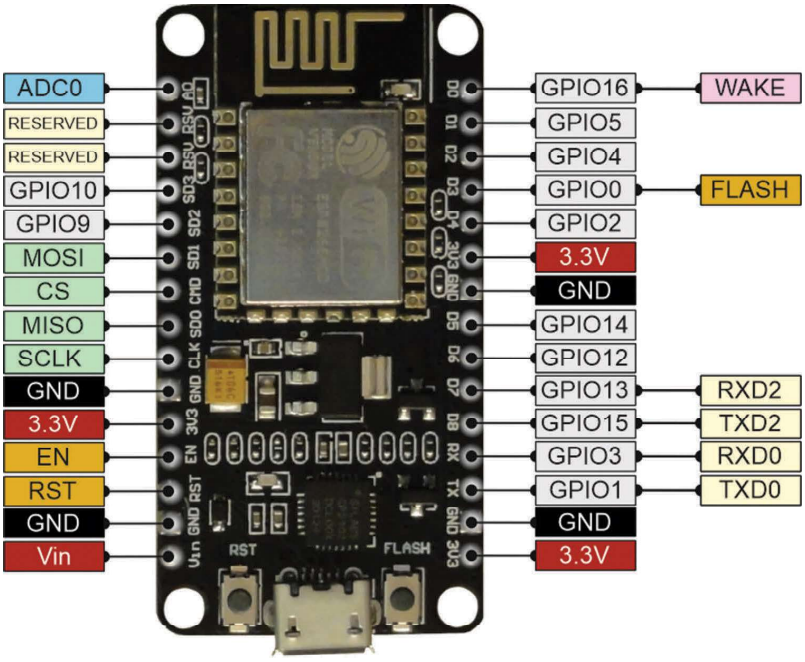
Moduł ESP-01 (niebieski) ma wbudowaną pamięć Flash o pojemności 512 KiB, kolejny moduł ESP-01 (czarny) i ESP-01S mają wbudowaną pamięć Flash o pojemności 1 MiB. Firma Ai-Thinker w kolejnych latach wypuściła na rynek kilkanaście modułów z ESP8266 różniących się wielkością pamięci Flash i liczbą wyprowadzonych pinów GPIO.

ESP-01 dał początek wielu projektom Internetu rzeczy, świetnie się nadaje do montowania w urządzeniach, ale jest mało przyjazny dla do nauki i rozwijania oprogramowania. Oprogramowanie modułu jest wgrywane do pamięci Flash za pomocą konwertera USB-UART, co jest dość kłopotliwe w warunkach laboratoryjnych. Wygodniejsze w eksperymentowaniu i nauce są płytki rozwojowe zawierające już konwerter USB-UART. Jedną z najbardziej popularnych jest moduł ESP8266 NodeMCU WiFi DevKit.

NodeMCU WiFi DevKit

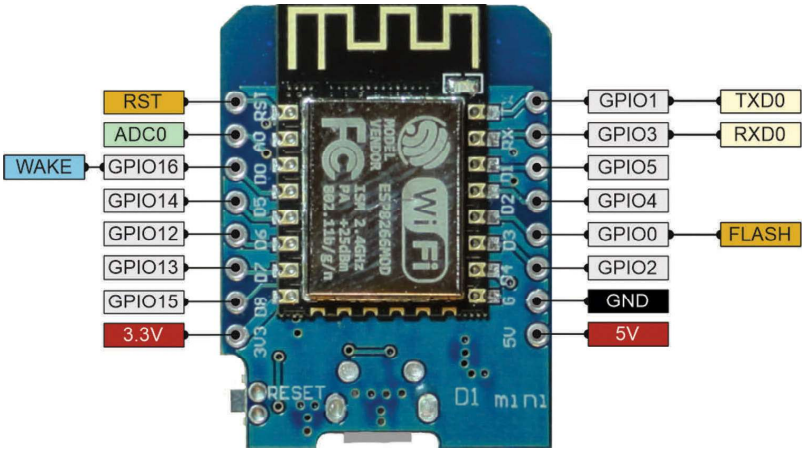
NodeMCU WiFi DevKit jest modułem przeznaczonym do rozwijania oprogramowania dla ESP8266. Ma wbudowany konwerter USB-UART, umożliwiający programowanie układu przez złącze micro USB, które służy także do zasilania układu. NodeMCU WiFi

DevKit wykorzystuje moduł ESP-12 w ekranowanej obudowie. Schemat wyprowadzeń NodeMCU WiFi DevKit przedstawiono na rys. 9.3.



Rys. 9.3. Schemat wyprowadzeń NodeMCU WiFi DevKit [16]

NodeMCU występujące w nazwie modułu to nazwa wgranego do pamięci Flash oprogramowania układowego. Pod nazwą NodeMCU zostało ono opublikowane po raz pierwszy na GitHubie w październiku 2014 roku. NodeMCU zostało stworzone w języku Lua i umożliwia programowanie ESP8266 w tym języku. Oprogramowanie było początkowo przeznaczone dla NodeMCU WiFi DevKit, obecnie może być uruchamiane na innych modułach. Najnowsza wersja jest dostępna na GitHubie pod adresem: https://github.com/nodemcu/nodemcu-firmware.



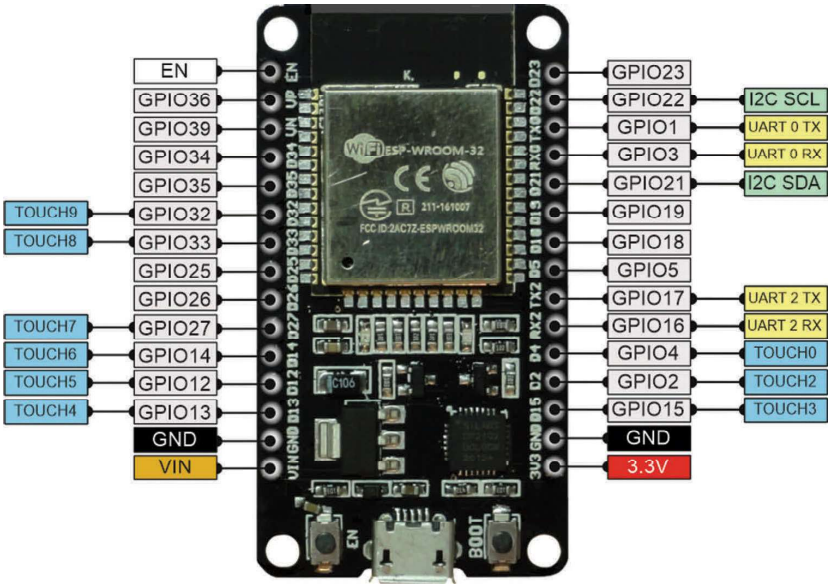
Rys. 9.4. Schemat wyprowadzeń modułu WeMos D1 Mini [16]

ESP 32

W sierpniu 2016 roku na rynku pojawił się mikrokontroler ESP32, następca ESP8266. Podstawowe parametry:

- 32-bitowy mikroprocesor LX6 o częstotliwości 160 lub 240 MHz,
- pamięć: 520 KiB SRAM,
- zewnętrzna pamięć Flash do 32 MiB (typowo: 4 MiB),
- Wi-Fi IEEE 802.11 b/g/n,
- Bluetooth v4.2,
- Interfejsy: 4 × SPI, 2 × I²C, 2 × I²S i 3 × UART,
- 12-bitowy przetwornik A/C – 18-kanałowy,
- dwa 8-bitowe przetworniki C/A,
- 10 czujników dotykowych (piny GPIO), czujnik Halla.

Wraz z pojawieniem się nowego mikrokontrolera powstały moduły Wi-Fi z tym mikrokontrolerem. Moduły mają wyprowadzenia do montażu powierzchniowego i trudno je stosować w celach laboratoryjnych. Do celów badawczo-rozwojowych powstały liczne płytki rozwojowe, podobne do NodeMCU WiFi Devkit, zwierające wlutowany moduł ESP32, wyprowadzone piny i konwerter USB-UART. Do najpopularniejszych należą: ESP32-DevKit produkcji Espressif i ESP32-CAM produkcji Ai-Thinker. Płytka rozwojowa ESP32-CAM zawiera złącze do podłączenia kamery – zwykle jest sprzedawana z kamerą OV2640. Kamera jest niskiej jakości, ale całość można kupić za 7 $, w Polsce od 40 zł. Schemat wyprowadzeń ESP32-DevKit przedstawiono na rys. 9.5., a schemat wyprowadzeń ESP32-CAM – na rys. 9.6.

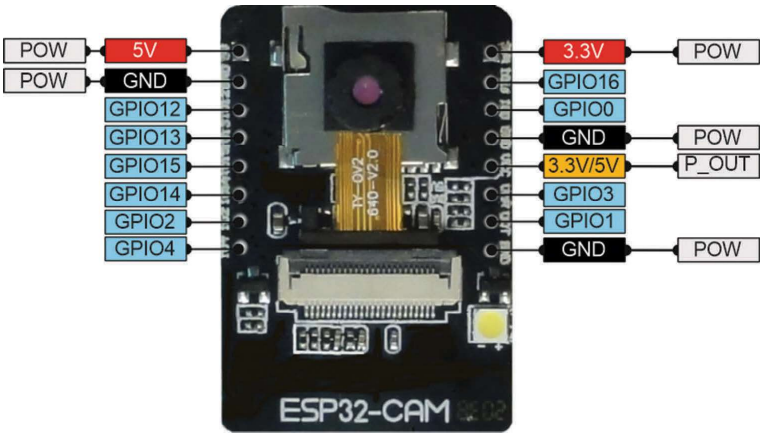


Rys. 9.5. Schemat wyprowadzeń modułu ESP32-DevKit, wersja z 30 pinami

Programowanie ESP8266 i ESP32

W październiku 2014 roku firma Espressif Systems wydała SDK (*Software Development Kit*) umożliwiający tworzenie oprogramowania dla ESP8266 [6]. Od tego czasu powsta-

ło wiele narzędzi do programowania urządzeń zawierających mikrokontrolery ESP8266 i ESP32. Bardzo popularnym środowiskiem programistycznym dla wymienionych kontrolerów jest Arduino IDE.



Rys. 9.6. Schemat wyprowadzeń modułu ESP32-CAM

Arduino IDE powstał w 2005 roku, przed pojawieniem się mikrokontrolerów ESP, i pierwotnie był przeznaczony do programowania urządzeń Arduino wykorzystujących układy z serii megaAVR. Po pojawieniu się mikrokontrolerów ESP, Arduino IDE został rozszerzony o możliwość programowania tych mikrokontrolerów. W Internecie jest bardzo wiele publikacji o programowaniu mikrokontrolerów ESP z wykorzystaniem Arduino IDE i przykładów gotowych programów. Przykładowe programy na mikrokontrolery ESP publikowane na GitHubie są głównie kodowane w Arduino IDE.

Arduino IDE wykorzystuje język programowania Arduino, przypominający język C. Program po skompilowaniu jest wgrywany do urządzenia przez konwerter USB-UART. Na części płytek rozwojowych mikrokontrolerów ESP konwerter USB-UART jest wbudowany w płytkę i wystarczy podłączyć urządzenie kablem USB. Część urządzeń trzeba programować za pomocą zewnętrznego konwertera.

Przed programowaniem modułów ESP do Arduino IDE należy doinstalować:

♦ pakiet http://arduino.esp8266.com/stable/package_esp8266com_index.json, umożliwiający programowanie modułów ESP8266, i

♦ pakiet https://dl.espressif.com/dl/package_esp32_index.json, umożliwiający programowanie modułów ESP32.

Wraz z pakietami instalowana jest bardzo dużo przykładowych programów, które obejmują podstawowe zastosowania modułów ESP. Dodatkowo można doinstalować biblioteki z wieloma kolejnymi przykładowymi programami.

W internecie znajdziemy wiele witryn publikujących opisy przykładowych programów wraz z instrukcją konfiguracji i realizacji podłączeń do czujników i elementów wykonawczych. Bardzo użyteczna może się okazać witryna https://randomnerdtutorials.com/, oferująca wiele instrukcji filmowych i tekstowych pokazujących, jak zrealizować i przetestować przykładowe programy. Autorzy witryny opublikowali też kilka płatnych podręczników.

Ćwiczenie 9.1.
Instalacja środowiska programistycznego Arduino IDE, uruchomienie i testowanie płytki ESP8266.

Polecenie dotyczy pracy zdalnej (z domu); w laboratorium środowisko Arduino IDE jest zainstalowane, a płytki są przetestowane.

1. Zainstalować Arduino IDE.

2. Podłączyć płytkę, z menu `Narzędzia,` wybrać `Narzędzia>Płytka>ESP8266 Boards(2.7.4)>NodeMCU 1.0 (ESP-12E Module) ...`

3. Z menu `Narzędzia>Port` wybrać właściwy port `COM`, następnie otworzyć `Narzędzia >Monitor portu szeregowego`.

4. W monitorze portu szeregowego ustawić szybkość transmisji 115200.

5. Uruchomić i przetestować przykładowy program skanera sieci Wi-Fi `Plik>Przykłady >ESP8266WiFi>WiFiScan`.

WiFiScan jest programem, który często uruchamia się na nowych urządzeniach w celu ich przetestowania. Program nie wymaga żadnych zmian kodu i działa od razu po uruchomieniu. Skanuje sieci Wi-Fi i wysyła komunikaty o wykryciu sieci na port szeregowy. Poniżej przykładowy efekt działania programu:

```
scan start
scan done
6 networks found
1: FunBox2-6FAF (-71)*
2: Porzycze (-71)*
3: AKA (-61)*
4: multimedia_grzegorz (-94)*
5: HomeNet (-84)*
6: HomeNet_Guest (-86)*
```

Ćwiczenie 9.2.
Program prostego serwera HTTP.

1. Wczytać przykładowy program prostego serwera HTTP: `Plik>Przykłady >ESP8266WebServer>Helloserver.`

2. W kodzie programu (szkicu) wprowadzić dane routera Wi-Fi:

```
#ifndef STASSID
#define STASSID "your-ssid"
#define STAPSK  "your-password"
#endif
```

3. Uruchomić program, następnie przetestować go z użyciem przeglądarki i programu Postman, numer IP serwera odczytać z monitora portu szeregowego, obsługiwane ścieżki i nazwę serwera można odczytać z kodu programu.

Ćwiczenie 9.3.
Program wykorzystujący RESTfull API do rejestracji wyników pomiarów

1. Pobrać kod programu *PostHttpClient_Do_WebAPI.ino*, przesyłającego do usługi WebAPI, opisanej w ćwiczeniu *Usługi internetowe RESTful WebAPI,* symulowane wyniki pomiarów.

2. Wpisać w kodzie swoje dane (AUTHOR, GAGE) i dane swojej sieci Wi-Fi, następnie uruchomić program.

3. Zainstalować bibliotekę `Arduino_JSON`.

4. Zaobserwować komunikaty wysyłane przez program na monitorze portu szeregowego.

5. Zaobserwować wysyłane przez program dane na stronie http://argo.am.gdynia.pl/www/RestWyniki/Pages/WebForm1.

Ćwiczenie 9.4.
Program zdalnego sterowania pinami ESP8266 z wykorzystaniem protokołu MQTT

1. Podłączyć diody LED (świateł drogowych) do pinów ESP8266: GND→G, R→D5, Y→D6, G→D7.

2. Zainstalować bibliotekę `PubSubClient`.

3. Pobrać kod programu mqtt_esp8266_PINY.ino zdalnego sterowania pinami ESP8266 z wykorzystaniem protokołu MQTT.

4. Wpisać w kodzie dane swojej sieci Wi-Fi.

5. Uruchomić program.

6. Za pomocą strony http://lukan.sytes.net:1880/ui/ sterować pinami zaobserwować efekt, odczytać komunikaty o otrzymanych przez Moduł ESP8266 wiadomościach.

Ćwiczenie 9.5.
Sterowanie pinami Moduł ESP8266 za pomocą programu działającego na urządzeniu przenośnym

1. Zainstalować na urządzeniu przenośnym program IoT MQTT Panel: https://play.google.com/store/apps/details?id=snr.lab.iotmqttpanel.prod&hl=pl.

2. Dodać połączenie do brokera `lukan.sytes.net` port domyślny 1883.

3. Utworzyć nowy pulpit (`Dashboard`).

4. Dodać i skonfigurować trzy kontrolki `Switch` do sterowania pinami Moduł ESP8266; nazwy tematów są widoczne w kodzie programu mqtt_esp8266_PINY.ino i na grafie http://lukan.sytes.net:1880.

5. Dodać i skonfigurować trzy kontrolki `Gauge` lub `Vertical Meter` do wyświetlania wyników pomiarów publikowanych w temacie `/TITI/pomiary/xyz1`. Wyniki są publikowane w formacie JSON, np.: `{"x":36.017,"y":90.75,"z":39.52}`. Aby z danych w formacie JSON pobrać składową `x`, należy w polu `JasonPath for subscribe` wpisać `$.x`.

6. Dodać do pulpitu kontrolkę `Line Graph` i skonfigurować ją do wykreślania wyników pomiarów `x`, `y`, `z`.

Na rys. 9.7 (s. 80) przedstawiono przykładowy wygląd skonfigurowanego pulpitu.

Projekt własnego systemu

1. Na grafie Node-RED dodać bloczek brokera MQTT aedes broker.

2. Stworzyć własny lokalny system wykorzystujący lokalnego brokera MQTT.

Rys. 9.7. Przykładowa konfiguracja pulpitu z ćwiczenia 9.5.