

# SPRAWOZDANIE Z ZAJĘĆ PBL1

*Platform Processing*

*Kinga Konieczna*

*Jan Czechowski*

*28.11.2024*

# 1. Postawienie serwera

## 1.1. Cel

Celem projektu było skonfigurowanie i uruchomienie serwera HTTP na mikrokontrolerze ESP8266, połączenie urządzenia z siecią Wi-Fi oraz umożliwienie wymiany danych pomiędzy serwerem a przeglądarką internetową.

## 1.2. Wykorzystane narzędzia i komponenty

- Mikrokontroler ESP8266
- Oprogramowanie: Arduino IDE
- Dostęp do sieci bezprzewodowej

## 1.3. Przebieg realizacji

### 1.3.1. Konfiguracja ESP8266

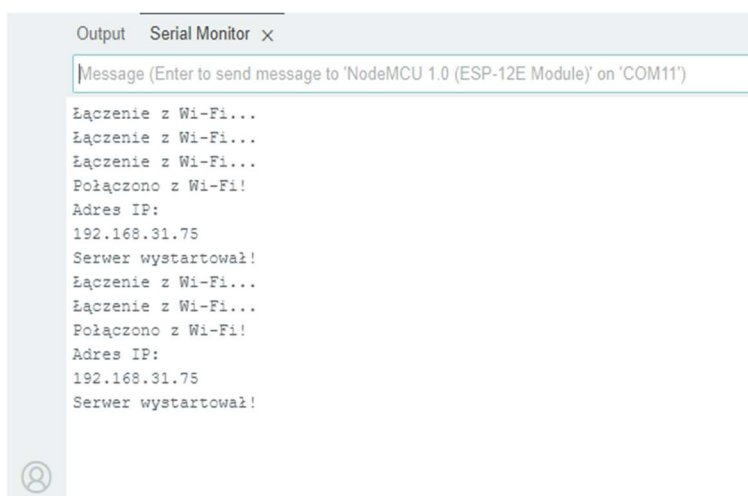
- W kodzie zdefiniowano nazwę i hasło sieci Wi-Fi.
- Po połączeniu z siecią ESP8266 wyświetlało w monitorze portu szeregowego przydzielony adres IP.

### 1.3.2. Uruchomienie serwera

- Utworzono serwer HTTP na porcie 80 z użyciem biblioteki ESP8266WebServer.
- Zaimplementowano funkcję obsługującą główną stronę serwera, która zwracała tekst lub dane w odpowiedzi na żądanie HTTP.

### 1.3.3. Testowanie

- Po wgraniu kodu do ESP i podłączeniu do Wi-Fi, adres IP urządzenia został otwarty w przeglądarce.
- Serwer na ESP8266 został poprawnie uruchomiony i obsługiwał żądania HTTP.
- Mikrokontroler wyświetlał dane w przeglądarce na prostym interfejsie tekstowym



The screenshot shows the Serial Monitor window in the Arduino IDE. The title bar reads 'Output Serial Monitor x'. Below the title bar is a text input field with the placeholder text 'Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM11')'. The main area of the window displays the following text output from the microcontroller:

```
Złączenie z Wi-Fi...
Złączenie z Wi-Fi...
Złączenie z Wi-Fi...
Połączono z Wi-Fi!
Adres IP:
192.168.31.75
Serwer wystartował!
Złączenie z Wi-Fi...
Złączenie z Wi-Fi...
Połączono z Wi-Fi!
Adres IP:
192.168.31.75
Serwer wystartował!
```

Rys. 1. Start serwera. [opracowanie własne]

## 2. Podłączenie czujnika ultradźwiękowego do ESP8266

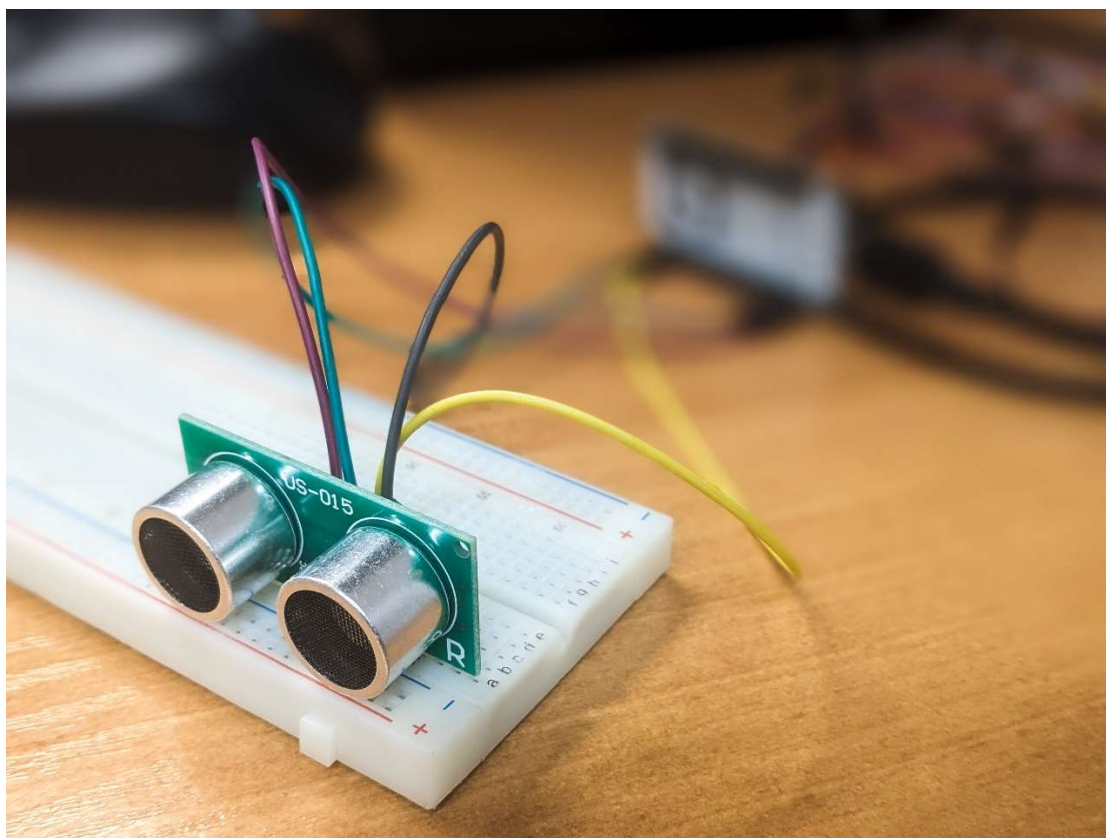
### 2.1. Cel

Celem było podłączenie czujnika ultradźwiękowego US-015 do modułu ESP8266 w celu pomiaru odległości oraz przesyłanie danych na lokalny serwer HTTP.

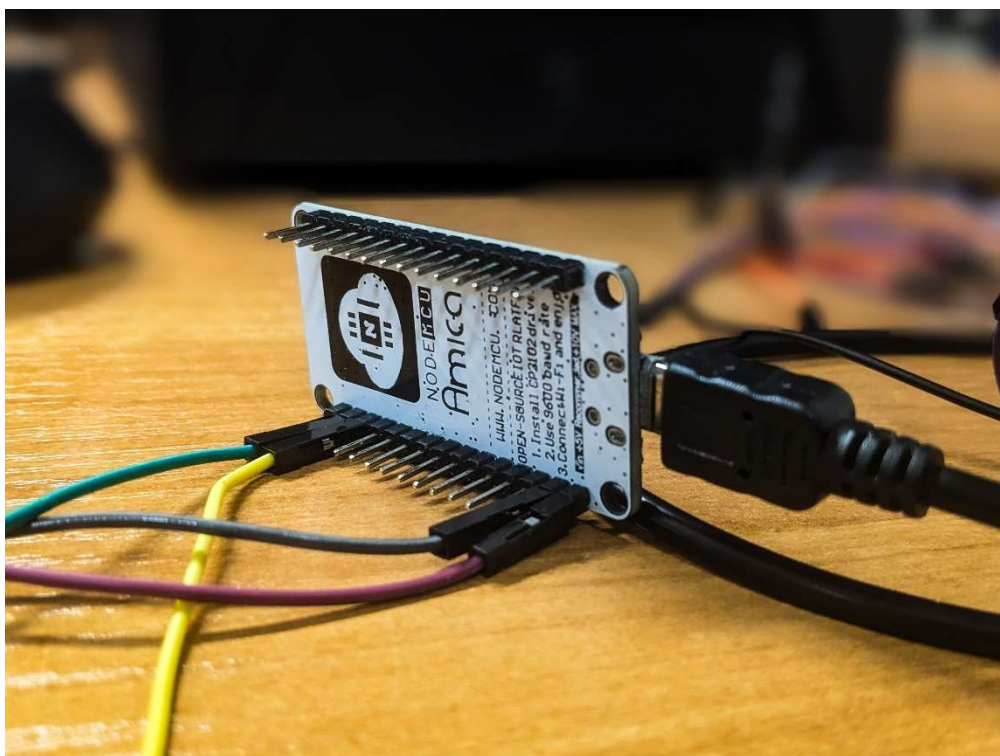
### 2.2. Podłączenie sprzętu:

Czujnik ultradźwiękowy **US-015** ma cztery piny:

- **VCC** – zasilanie, podłączone do **3.3V** na ESP8266
- **GND** – masa, podłączona do **GND** na ESP8266
- **Trig** – pin do wysyłania impulsu, podłączony do jednego z pinów GPIO (w naszym przypadku D1)
- **Echo** – pin do odbioru sygnału odbitego, podłączony do innego GPIO (w naszym przypadku D2)



Rys. 2. Podłączenie czujnika US-105. [opracowanie własne]



Rys. 3. Podłączenie do PINów ESP8266 czujnika US-105. [opracowanie własne]

## 2.3. Zaprogramowanie czujnika

W funkcji **handleDistance()** obliczamy odległość na podstawie czasu trwania impulsu odbitego przez czujnik. Używamy funkcji **pulseIn()** do zmierzenia czasu trwania impulsu na pinie **Echo**. Następnie dane zostają przesłane na lokalny serwer HTTP.

## 3. Processing

### 3.1. Cel

Celem było stworzenie programu w **Processing**, który łączy się z modulem **ESP8266** i wyświetla zmierzoną odległość z czujnika ultradźwiękowego **US-015** na ekranie w formie tekstu oraz graficznej wizualizacji (pasek).

### 3.2. Opis działania

#### 3.2.1. Połączenie z ESP8266

Program używa klasy **URLConnection** do wysyłania zapytania HTTP GET na adres IP ESP8266 (w tym przypadku 192.168.31.75) w celu pobrania zmierzonej odległości z endpointu `/distance`.

### 3.2.2. Odczyt danych

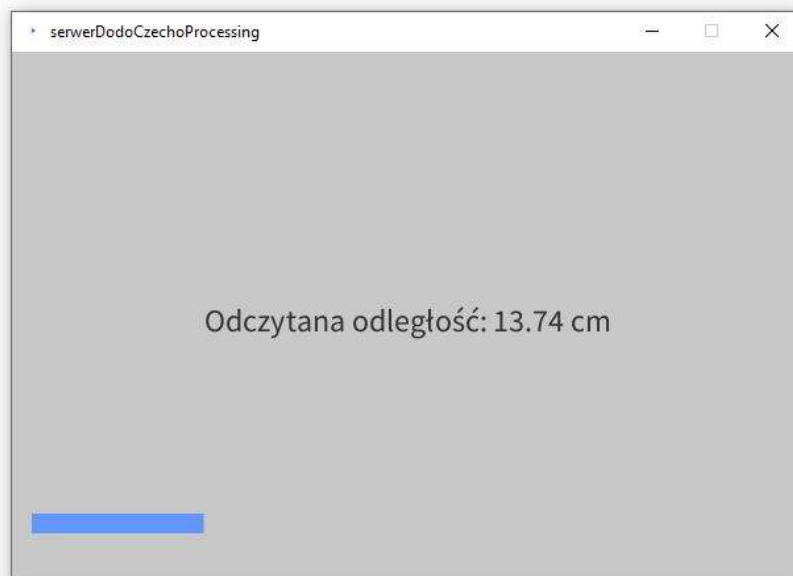
Po udanym połączeniu, odpowiedź z serwera ESP8266 (zwracającą odległość w centymetrach) jest odczytywana z **BufferedReader**. Wartość jest parsowana na typ float, aby umożliwić jej dalsze użycie.

### 3.2.3. Wizualizacja

W głównym oknie programu (o wymiarach 600x400 pikseli) odczytana odległość jest wyświetlana na środku ekranu w postaci tekstu. Dodatkowo, na dole ekranu rysowany jest pasek, którego długość jest proporcjonalna do odległości. Pasek jest skalowany tak, by mieścił się w szerokości okna, przy założeniu maksymalnej odległości 100 cm.

### 3.2.4. Aktualizacja danych

Program aktualizuje dane co sekundę dzięki ustawieniu `frameRate(1)`, co sprawia, że odczyt i wizualizacja są odświeżane na bieżąco.



---

Rys. 4. Design w Processingu. [opracowanie własne]