

10.12.2024r.

# Sprawozdanie

## Internet rzeczy: KOMUNIKACJA I STEROWANIE Z WYKORZYSTANIEM SIECI BEZPRZEWODOWEJ WIFI.

**1. Cel ćwiczenia** - Celem ćwiczenia było zapoznanie z zasadą działania sterowania i komunikacji Arduino za pomocą sieci WiFi

## 2. Przebieg ćwiczenia.

### 2.1. Skanowanie dostępnych sieci oraz ich parametrów.

Postępując zgodnie z instrukcją, wykorzystaliśmy kod w niej umieszczony. Dany program skanował dostępne sieci bezprzewodowe i wyświetlał ich parametry w Serial Monitor.

```
Output  Serial Monitor x
Message (Enter to send message to 'Arduino UNO R4 WiFi' on 'COM7')

Encryption: Unknown      SSID: eduroam
18) Signal: -83 dBm      Channel: 11      BSSID: 9F:DE:57:C2:9F:02
Encryption: WPA2         SSID: HOTSPOT-Gosc
19) Signal: -84 dBm      Channel: 11      BSSID: 9F:DE:57:C2:9F:F2
Encryption: WPA2         SSID: hotspot_UniFi
20) Signal: -85 dBm      Channel: 1      BSSID: E3:E6:0C:9F:C4:6C
Encryption: WPA2         SSID: Monitor
21) Signal: -86 dBm      Channel: 1      BSSID: E1:E6:0C:9F:C4:6C
Encryption: None         SSID: AM_Student
22) Signal: -88 dBm      Channel: 1      BSSID: E0:E6:0C:9F:C4:6C
Encryption: Unknown      SSID: AM_Pracownik
23) Signal: -88 dBm      Channel: 1      BSSID: E2:E6:0C:9F:C4:6C
Encryption: Unknown      SSID: eduroam
24) Signal: -88 dBm      Channel: 1      BSSID: E5:E6:0C:9F:C4:6C
Encryption: WPA2         SSID: EuroSciPy2024
25) Signal: -88 dBm      Channel: 1      BSSID: E4:E6:0C:9F:C4:6C
Encryption: WPA2         SSID: PM_ITM
26) Signal: -90 dBm      Channel: 1      BSSID: E8:DE:57:C2:9F:F2
Encryption: WPA2         SSID: hotspot_UniFi
27) Signal: -93 dBm      Channel: 6      BSSID: 8E:01:12:C4:2C:34
Encryption: WPA2         SSID: UPC7335942

Scanning available networks...
** Scan Networks **
```

Następnie dokonaliśmy zmian w kodzie w taki sposób, aby parametry danej sieci wypisywane były w jednej linii.

```
#include <WiFiS3.h>

void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ;
  }

  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    while (true)
      ;
  }

  String fv = WiFi.firmwareVersion();
  if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
    Serial.println("Please upgrade the firmware");
  }

  Serial.println();
  Serial.println("Scanning available networks...");
  listNetworks();

  byte mac[6];
  WiFi.macAddress(mac); //print my MAC
  Serial.print("MAC: ");
  printMacAddress(mac);
}

void loop() {
  delay(10000);
  Serial.println("Scanning available networks...");
  listNetworks(); // scan networks
}

void listNetworks() {
  Serial.println("*** Scan Networks ***");
  int numSsid = WiFi.scanNetworks();
  if (numSsid == -1) {
    Serial.println("Couldn't get a WiFi connection");
    while (true)
      ;
  }
}

Serial.print("number of available networks: ");
Serial.println(numSsid); //print the list of networks

for (int thisNet = 0; thisNet < numSsid; thisNet++) {
  // Wypisanie wszystkich parametrów w jednej linii
  Serial.print(thisNet + 1);
  Serial.print(" ");
  Serial.print("Signal: ");
  Serial.print(WiFi.RSSI(thisNet));
  Serial.print(" dBm, ");
  Serial.print("Channel: ");
  Serial.print(WiFi.channel(thisNet));
  byte bssid[6];
  Serial.print(" BSSID: ");
  printMacAddress(WiFi.BSSID(thisNet, bssid));
  Serial.print(" Encryption: ");
  printEncryptionType(WiFi.encryptionType(thisNet));
  Serial.print(" SSID: ");
  Serial.println(WiFi.SSID(thisNet));
  Serial.flush();
}
Serial.println();
}

void printEncryptionType(int thisType) {
  switch (thisType) {
    case ENC_TYPE_WEP:
      Serial.print("WEP");
      break;
    case ENC_TYPE_WPA:
      Serial.print("WPA");
      break;
    case ENC_TYPE_WPA2:
      Serial.print("WPA2");
      break;
    case ENC_TYPE_WPA3:
      Serial.print("WPA3");
      break;
    case ENC_TYPE_NONE:
      Serial.print("None");
      break;
    case ENC_TYPE_AUTO:
      Serial.print("Auto");
      break;
    case ENC_TYPE_UNKNOWN:
      break;
  }
}
```

```
void printEncryptionType(int thisType) {
  switch (thisType) {
    case ENC_TYPE_WEP:
      Serial.print("WEP");
      break;
    case ENC_TYPE_WPA:
      Serial.print("WPA");
      break;
    case ENC_TYPE_WPA2:
      Serial.print("WPA2");
      break;
    case ENC_TYPE_WPA3:
      Serial.print("WPA3");
      break;
    case ENC_TYPE_NONE:
      Serial.print("None");
      break;
    case ENC_TYPE_AUTO:
      Serial.print("Auto");
      break;
    case ENC_TYPE_UNKNOWN:
      Serial.print("Unknown");
      break;
    default:
      Serial.print("Unknown");
      break;
  }
}

void print2Digits(byte thisByte) {
  if (thisByte < 0xF) {
    Serial.print("0");
  }
  Serial.print(thisByte, HEX);
}

void printMacAddress(byte mac[]) {
  for (int i = 5; i >= 0; i--) {
    if (mac[i] < 16) {
      Serial.print("0");
    }
    Serial.print(mac[i], HEX);
    if (i > 0) {
      Serial.print(":");
    }
  }
}
```

Output Serial Monitor ✕

Message (Enter to send message to 'Arduino UNO R4 WiFi' on 'COM7')

```
4) Signal: -56 dBm, Channel: 11, BSSID: 03:07:0F:9F:C4:6C, Encryption: Unknown, SSID: eduroam
5) Signal: -57 dBm, Channel: 11, BSSID: 01:07:0F:9F:C4:6C, Encryption: Unknown, SSID: AM_Pracownik
6) Signal: -57 dBm, Channel: 11, BSSID: 05:07:0F:9F:C4:6C, Encryption: WPA2, SSID: PM_ITM
7) Signal: -73 dBm, Channel: 9, BSSID: 6F:B0:63:B8:48:E8, Encryption: WPA2, SSID: TP-Link_B070
8) Signal: -75 dBm, Channel: 1, BSSID: A3:C0:0E:9F:C4:6C, Encryption: Unknown, SSID: eduroam
9) Signal: -76 dBm, Channel: 1, BSSID: A0:C0:0E:9F:C4:6C, Encryption: Unknown, SSID: AM_Pracownik
10) Signal: -76 dBm, Channel: 1, BSSID: A1:C0:0E:9F:C4:6C, Encryption: None, SSID: AM_Student
11) Signal: -76 dBm, Channel: 1, BSSID: A2:C0:0E:9F:C4:6C, Encryption: WPA2, SSID: Monitor
12) Signal: -77 dBm, Channel: 1, BSSID: A4:C0:0E:9F:C4:6C, Encryption: WPA2, SSID: EuroSciPy2024
13) Signal: -84 dBm, Channel: 6, BSSID: 48:3D:A5:DE:8A:C0, Encryption: Unknown, SSID: AM_Pracownik
14) Signal: -84 dBm, Channel: 6, BSSID: 48:3D:E5:DE:8A:C0, Encryption: None, SSID: AM_Student
15) Signal: -84 dBm, Channel: 6, BSSID: 49:3D:25:DE:8A:C0, Encryption: WPA2, SSID: Monitro
16) Signal: -85 dBm, Channel: 6, BSSID: 48:3D:25:DE:8A:C0, Encryption: WPA2, SSID: AM_ITM
17) Signal: -85 dBm, Channel: 6, BSSID: 48:3D:65:DE:8A:C0, Encryption: Unknown, SSID: eduroam
18) Signal: -86 dBm, Channel: 1, BSSID: E0:E6:0C:9F:C4:6C, Encryption: Unknown, SSID: AM_Pracownik
19) Signal: -88 dBm, Channel: 1, BSSID: E4:E6:0C:9F:C4:6C, Encryption: WPA2, SSID: PM_ITM
20) Signal: -88 dBm, Channel: 1, BSSID: E1:E6:0C:9F:C4:6C, Encryption: None, SSID: AM_Student
21) Signal: -89 dBm, Channel: 1, BSSID: E3:E6:0C:9F:C4:6C, Encryption: WPA2, SSID: Monitor
22) Signal: -90 dBm, Channel: 1, BSSID: E2:E6:0C:9F:C4:6C, Encryption: Unknown, SSID: eduroam
23) Signal: -90 dBm, Channel: 1, BSSID: E5:E6:0C:9F:C4:6C, Encryption: WPA2, SSID: EuroSciPy2024
24) Signal: -90 dBm, Channel: 1, BSSID: E8:DE:57:C2:9F:02, Encryption: WPA2, SSID: HOTSPOT-Gosc
25) Signal: -90 dBm, Channel: 1, BSSID: E8:DE:57:C2:9F:F2, Encryption: WPA2, SSID: hotspot_UniFi
```

MAC: 60:78:CC:75:54:DC

```
Output Serial Monitor X
Message (Enter to send message to 'Arduino UNO R4 WiFi on 'COM7')

SSID: linksys
BSSID: 46:E5:2E:12:07:02
signal strength (RSSI):-35
Encryption Type:2

SSID: linksys
BSSID: 46:E5:2E:12:07:02
signal strength (RSSI):-41
Encryption Type:2

SSID: linksys
BSSID: 46:E5:2E:12:07:02
signal strength (RSSI):-41
Encryption Type:2

SSID: linksys
BSSID: 46:E5:2E:12:07:02
signal strength (RSSI):-37
Encryption Type:2

SSID: linksys
BSSID: 46:E5:2E:12:07:02
signal strength (RSSI):-35
Encryption Type:2
```

## 2.3. Access point (web server) ze sterowaniem wbudowaną diodą LED.

W następnym punkcie utworzyliśmy nowy plik, do którego wpisaliśmy kod programu z instrukcji. Tworzył on z Arduino punkt dostępowy. W kodzie dokonaliśmy modyfikacji adresu IP.

```
25 String fv = WiFi.firmwareVersion();
26 if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
27   Serial.println("Please upgrade the firmware");
28 }
29 WiFi.config(IPAddress(192, 168, 4, 69)); //ip different for every accesspoint
30
31 Serial.print("Creating access point named: ");
32 Serial.println(ssid);
33
```

Następnie połączyliśmy się z Arduino poprzez przeglądarkę internetową, a za pomocą odpowiednich adresów (np. `http://adres_punktu_dostepowego/H`) mogliśmy sterować stanem diody LED.

```
Output Serial Monitor X
Message (Enter to send message to 'Arduino UNO R4 WiFi on 'COM7')

SSID: Test anteny 6G - 100% mocy
IP Address: 192.168.4.69
To see this page in action, open a browser to http://192.168.4.69
Device connected to AP
new client
GET / HTTP/1.1
Connection: close
User-Agent: Dalvik/2.1.0 (Linux; U; Android 14; SM-A546B Build/UP1A.231005.007)
Host: 192.168.4.69
Accept-Encoding: gzip

client disconnected
```



Powyżej efekt wpisywania różnych adresów w przeglądarce.

## 2.4. Dwukierunkowa komunikacja z wykorzystaniem portu UDP.

Kolejno przepisaliśmy kod programu do wymiany danych za pośrednictwem portu UDP. Poniżej test za pomocą programu Packet Sender.

Packet Sender - IPs: 10.3.10.140, fe80:a920:71b8:854e:cccd%ethernet\_32769

File Tools Multicast Panels Help

Name: Packet Name

ASCII: dane wyslane do arduino

HEX: 64 61 6e 65 20 77 79 73 6c 61 6e 65 20 64 6f 20 61 72 64 75 69 6e 6f

Address: 192.168.165.73 Port: 2390 Resend Delay: 0 UDP Send Save

Search Saved Packets... Delete Saved Packet Persistent TCP

	Send	Name	Resend	To Address	To Port	Method
1	Send	DNS dannagle.com	0	1.1.1.1	53	UDP
2	Send	DNS example.com	0	8.8.8.8	53	UDP
3	Send	FTP debian.org	0	cdimage.debian.org	21	TCP
4	Send	Google DNS over HTTPS / DoH	0	dns.google	443	HTTPS Get
5	Send	HTTP GET	0	neverssl.com	80	HTTP Get
6	Send	HTTP POST Params	0	httpbin.org	80	HTTP Post
7	Send	HTTPS GitHub API	0	api.github.com	443	HTTPS Get
8	Send	HTTPS POST ISON	0	htrhin.org	443	HTTPS Post

Clear Log (3) Log Traffic Save Log Save Traffic Packet Copy to Clipboard

	Time	From IP	From Port	To Address	To Port	Method	Error	ASCII	Hex
	2024-12-10 08:48:47.603	You	53643	192.168.165.73	2390	UDP		dane wyslane do arduino	64 61 6e 65 20 77 79 73 6c 61 6e 65 20 64 6f 20 61
	2024-12-10 08:48:46.307	You	53643	192.168.165.73	2390	UDP		dane wyslane do arduino	64 61 6e 65 20 77 79 73 6c 61 6e 65 20 64 6f 20 61
	2024-12-10 08:48:41.899	You	53643	192.168.165.73	2390	UDP		dane wyslane do arduino	64 61 6e 65 20 77 79 73 6c 61 6e 65 20 64 6f 20 61

DTLS Server Disabled UDP:53643 TCP Server Disabled SSL Server Disabled IPv4 Mode

## 2.5. Pobieranie danych z serwerów zewnętrznych na przykładzie danych pogodowych.

W kolejnym punkcie zainstalowaliśmy wymaganą bibliotekę Arduino\_JSON, a następnie przeszliśmy do wykonywania zadań. Stworzyliśmy plik nagłówkowy zawierający login oraz hasło, a potem przepisaliśmy kod podany w instrukcji.

Pierwotnie w Serial Monitor otrzymywaliśmy dane w podanej postaci.

```
Starting connection to server...
Connected to server
No JSON data received or invalid data.
No JSON data received or invalid data.
No JSON data received or invalid data.
No JSON data received or invalid data.
No JSON data received or invalid data.
No JSON data received or invalid data.

Received Data:
{"latitude":53.42,"longitude":14.559999,"generationtime_ms":0.07200241088867188,"utc_offset_seconds":3600,"timezone":"Europe/Berlin","timezone_abbreviation":"CET","elevat
0

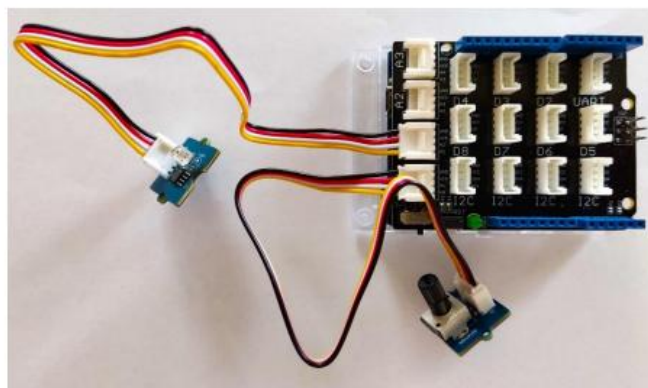
Temperature (C): 3.50
Windspeed (km/h): 10.00
```

Udało się je nam poprawić w celu lepszej widoczności. Oto rezultat.

```
Starting connection to server...
Connected to server
No JSON data received or invalid data.
No JSON data received or invalid data.
No JSON data received or invalid data.
No JSON data received or invalid data.
No JSON data received or invalid data.
No JSON data received or invalid data.
Weather Data:
-----
Temperature: 3.50 °C
Windspeed: 10.00 km/h
Humidity: 88.00 %
Pressure: 1028.40 hPa
Rainfall: 0.00 mm
Cloud Cover: 100.00 %
-----
No JSON data received or invalid data.
```

## 2.5. Web Server – odczyt wartości z wejść analogowych.

Do Arduino dołożyliśmy nakładkę Grove Shield, do której zostały podłączone potencjometr oraz czujnik światła.



Rys. 3. Podłączenie potencjometru i czujnika światła do nakładki Grove

Korzystając z kodu podanego w instrukcji wykonaliśmy program, który odbierał informacje z podłączonych komponentów. Dane były wyświetlane w przeglądarce internetowej, były odświeżane co pewną ilość czasu.

```
client disconnected
new client
GET /H HTTP/1.1
Host: 192.168.165.73
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Mobile Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Sec-GPC: 1
Accept-Language: pl-PL,pl
Referer: http://192.168.165.73/H
Accept-Encoding: gzip, deflate

client disconnected
new client
GET /favicon.ico HTTP/1.1
Host: 192.168.165.73
Connection: keep-alive
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Mobile Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-GPC: 1
Accept-Language: pl-PL,pl
Referer: http://192.168.165.73/H
Accept-Encoding: gzip, deflate

client disconnected
```

### 3. Wnioski.

- Realizacja zadań takich jak skanowanie sieci Wi-Fi, połączenie z routerem, tworzenie punktu dostępowego czy wymiana danych za pomocą protokołu UDP pokazała, jak w praktyce wykorzystać wiedzę z zakresu sieci komputerowych do sterowania i monitorowania urządzeń.
- Stworzenie Web Servera do sterowania wbudowaną diodą LED oraz do odczytu wartości z wejść analogowych ilustruje, w jaki sposób interfejsy przeglądarkowe mogą ułatwić interakcję z systemami IoT.
- Pobieranie i wyświetlanie danych pogodowych z serwera Open-Meteo pokazało, jak można wykorzystać usługi zewnętrzne do wzbogacenia funkcjonalności urządzeń IoT.