Jakub Gubar nr albumu 29205

10.12.2024r.

Sprawozdanie

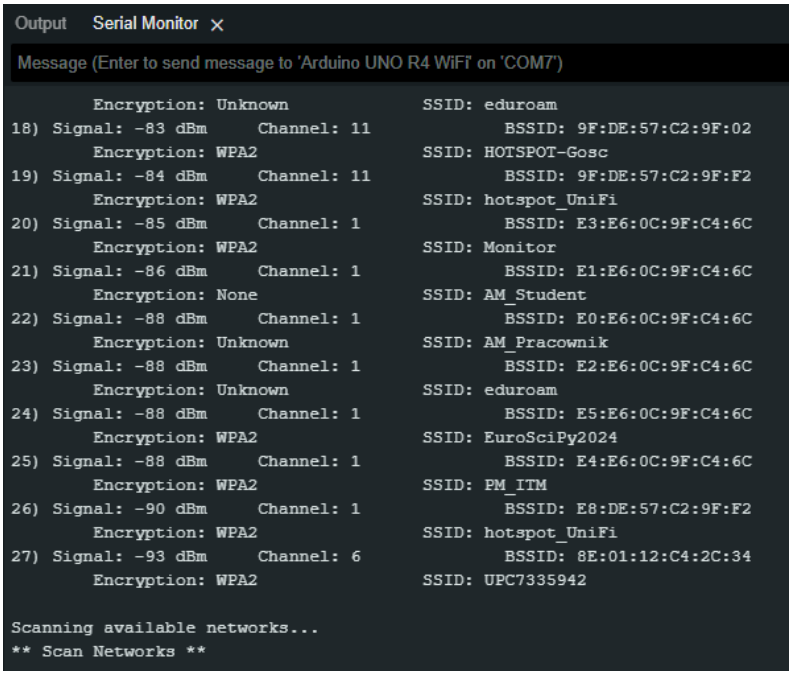
Internet rzeczy: KOMUNIKACJA I STEROWANIE Z WYKORZYSTANIEM SIECI BEZPRZEWODOWEJ WIFI.

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.



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:
```

```
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:
    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
```

2.2. Połączenie z routerem sieci bezprzewodowej oraz sczytanie parametrów z routera.

Kolejno wykorzystaliśmy kod z drugiego ćwiczenia, który odczytywał parametry routera. Jako routera użyliśmy telefonu o podanych w instrukcji parametrach, czyli SSID „linksys” i o pustym haśle.

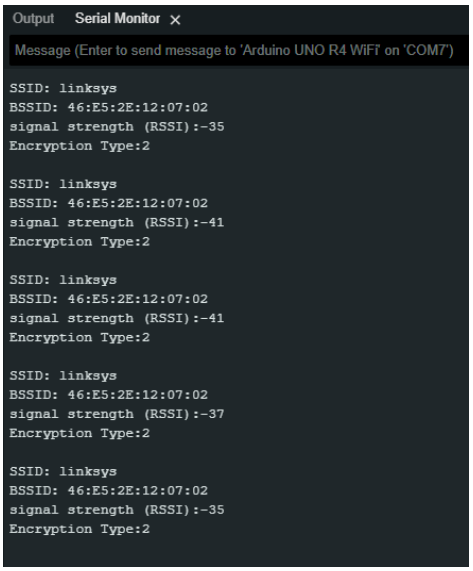
arduino_login_and_password.h

```
1  //arduino_login_and_password.h header file
2  #define MYSSID "linksys"
3  #define MYPASS ""
```

W rezultacie otrzymaliśmy to co poniżej.

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

Attempting to connect to WPA SSID: linksys
Attempting to connect to WPA SSID: linksys
Attempting to connect to WPA SSID: linksys
Attempting to connect to WPA SSID: linksys
Attempting to connect to WPA SSID: linksys
Attempting to connect to WPA SSID: linksys
Attempting to connect to WPA SSID: linksys
Attempting to connect to WPA SSID: linksys
Attempting to connect to WPA SSID: linksys
Attempting to connect to WPA SSID: linksys
Attempting to connect to WPA SSID: linksys
```

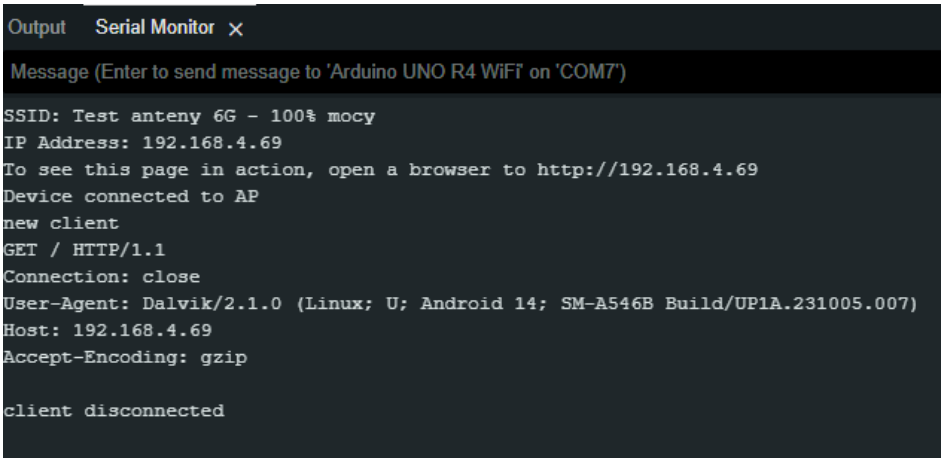


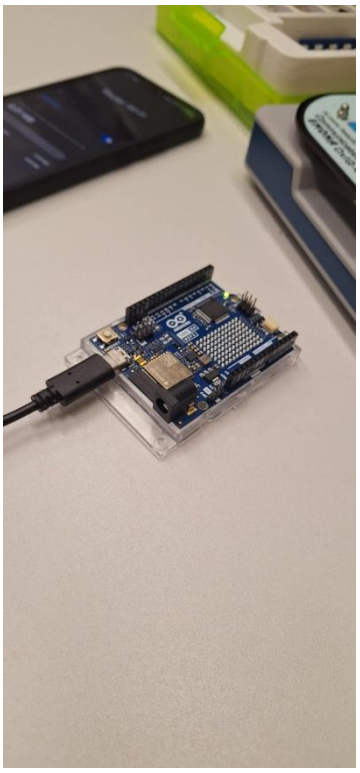
2.3. Access point (web server) ze sterowaniem wbudowaną diodą LED.

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_dostępowego/H) mogliśmy sterować stanem diody LED.

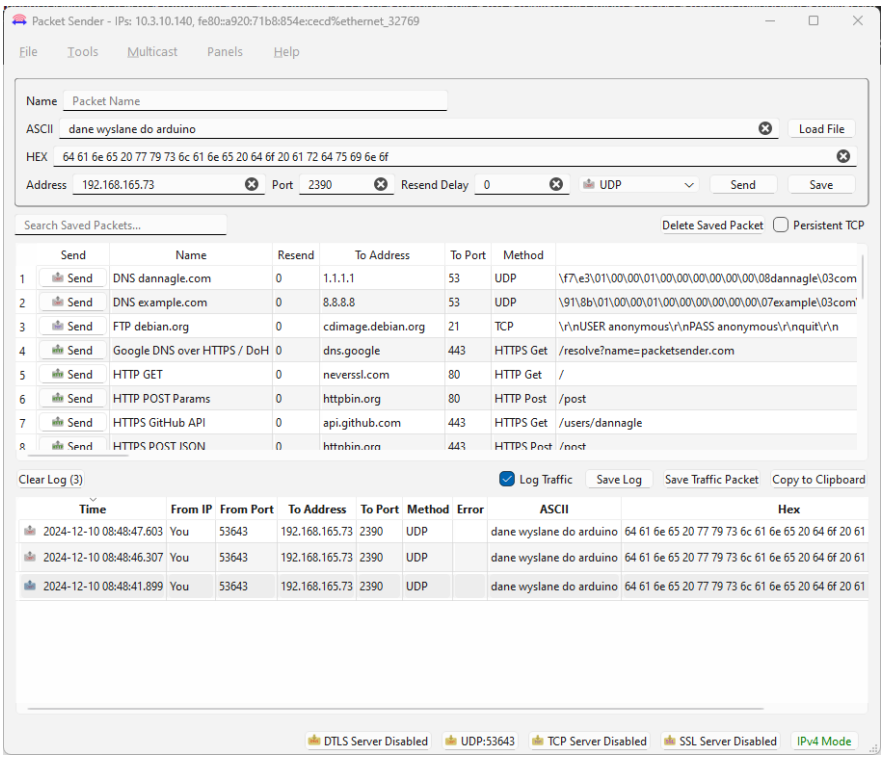




Powyżej efekt wpisywania różnych adresów w przeglądarce.

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.



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

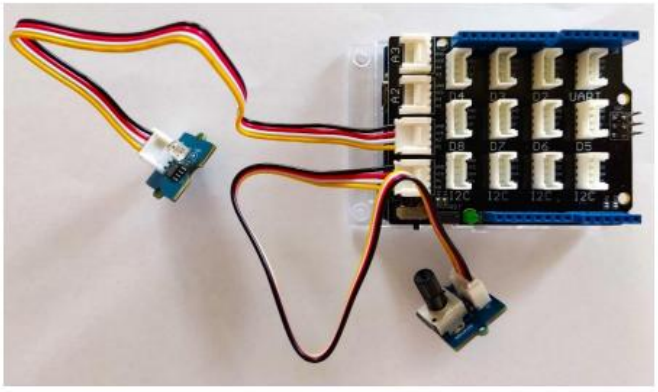
W kolejnym punkcie zainstalowaliśmy wymaganą bibliotekę Arduino_JSON, a następnie przeszliśmy do wykonywania zadań. Stworzyliśmy plik nagłówkowy zawierający login oraz hasło, a potem przepisaliśmy kod podany w instrukcji.

Pierwotnie w Serial Monitor otrzymywaliśmy dane w podanej postaci.

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

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.