

Jakub Gubar nr albumu 29205

03.12.2024r.

Sprawozdanie

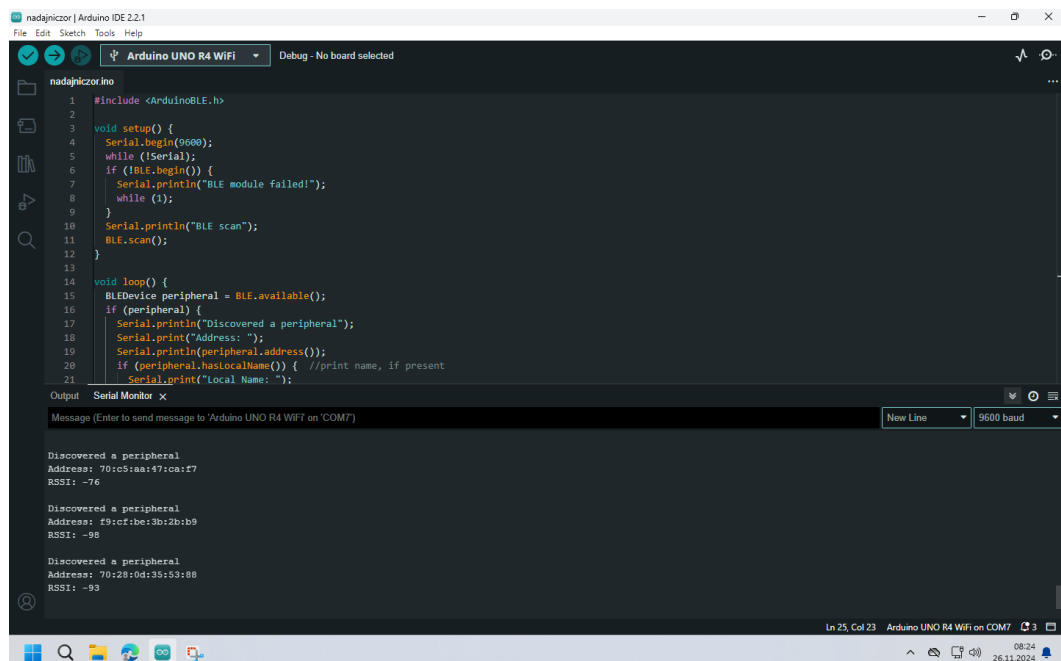
Internet rzeczy: BLUETOOTH - KOMUNIKACJA I STEROWANIE BEZPRZEWODOWE

1. Cel ćwiczenia - Celem ćwiczenia było zapoznanie z zasadą działania i sposobem wykorzystania modułu Bluetooth

2. Przebieg ćwiczenia.

2.1. Skanowanie dostępnych sieci oraz ich parametrów.

Postępując zgodnie z instrukcją napisaliśmy program skanujący dostępne sieci bezprzewodowe i wyświetlający parametry dostępnych sieci w „Serial Monitor”.



The screenshot shows the Arduino IDE interface. The top panel displays the code for 'nadajniczor.ino'. The code includes the 'ArduinoBLE.h' library and implements a setup and loop function. The setup function initializes the serial port at 9600 baud and starts the BLE module. The loop function scans for available BLE devices and prints their addresses and RSSI values to the serial monitor.

```
1 #include <ArduinoBLE.h>
2
3 void setup() {
4   Serial.begin(9600);
5   while (!Serial);
6   if (!BLE.begin()) {
7     Serial.println("BLE module failed!");
8     while (1);
9   }
10  Serial.println("BLE scan");
11  BLE.scan();
12}
13
14 void loop() {
15  BLEDevice peripheral = BLE.available();
16  if (peripheral) {
17    Serial.println("Discovered a peripheral");
18    Serial.print("Address: ");
19    Serial.println(peripheral.address());
20    if (peripheral.hasLocalName()) { //print name, if present
21      Serial.print("Local Name: ");
```

The bottom panel shows the 'Serial Monitor' output, which displays the results of the BLE scan:

```
Discovered a peripheral
Address: 70:c8:aa:47:ca:f7
RSSI: -76

Discovered a peripheral
Address: f9:c7:be:3b:2b:b9
RSSI: -90

Discovered a peripheral
Address: 70:28:0d:35:53:88
RSSI: -93
```

Program skanuje dostępne sieci oraz wyświetla ich MAC adresy oraz wskaźnik sygnału radiowego RSSI.

2.2. Sterowanie diodą LED wbudowaną w Arduino za pomocą bluetooth LE.

Następnie wykorzystaliśmy kod podany w instrukcji i napisaliśmy program służący do sterowania wbudowaną diodą LED.

```
#include <ArduinoBLE.h>
BLEService ledService("19B10000-E8F2-537E-4F6C-D104768A1214"); // Bluetooth
LE LED Service
BLEByteCharacteristic switchCharacteristic("19B10001-E8F2-537E-4F6C-D104768A1214", BLERead | BLEWrite);
const int ledPin = LED_BUILTIN;
void setup() {
  Serial.begin(9600);
  while (!Serial);
  pinMode(ledPin, OUTPUT);
  if (!BLE.begin()) {
    Serial.println("starting Bluetooth® Low Energy module failed!");
    while (1);
  }
  BLE.setLocalName("UNO R4 LED xx");
  BLE.setAdvertisedService(ledService);
  ledService.addCharacteristic(switchCharacteristic);
  BLE.addService(ledService);
  switchCharacteristic.writeValue(0);
  BLE.advertise();
  Serial.println("BLE LED Peripheral");
}

void loop() {
  BLEDevice central = BLE.central();
  if (central) {
    Serial.print("Connected to central: ");
    Serial.println(central.address()); // print MAC address:
    while (central.connected()) {
      if (switchCharacteristic.written()) {
        if (switchCharacteristic.value()) { // any value other than 0
          Serial.println("LED on");
          digitalWrite(ledPin, HIGH); // will turn the LED on
        } else { // a 0 value
          Serial.println("LED off");
          digitalWrite(ledPin, LOW); // will turn the LED off
        }
      }
    }
  }
  Serial.print("Disconnected from central: ");
  Serial.println(central.address());
}
```

Po zainstalowaniu i konfiguracji aplikacji LightBlue byliśmy w stanie sterować wbudowaną diodą.

08:39 65%

< Characteristic Hex

Device
UNO R4 LED 69

Service UUID
19b10000-e8f2-537e-4f6c-d104768a1214

Write

Write New Value

Tue Nov 26 08:39:11 GMT+01:00 2024
0x00

Tue Nov 26 08:38:59 GMT+01:00 2024
0x01

Tue Nov 26 08:38:52 GMT+01:00 2024
0x00

Tue Nov 26 08:38:43 GMT+01:00 2024
0x01

Read/Notified Values

08:36 66%

< Peripheral : Connected

UNO R4 LED 69
DC:54:75:CC:78:61

Advertisement Data

Services

Generic Access

Device Name
Arduino

Appearance
Properties: Read

Generic Attribute

Service Changed
Properties: Indicate

19b10000-e8f2-537e-4f6c-d104768a1214

19b10001-e8f2-537e-4f6c-d104768a1214
4
Properties: Read, Write

08:38 66%

< Hex

Property
Hex Value

1

Write

| | | |
|---|---|------|
| D | E | F |
| A | B | C |
| 7 | 8 | 9 |
| 4 | 5 | 6 |
| 1 | 2 | 3 |
| x | 0 | Done |

08:38 66%

< Characteristic Hex

Read/Notified Values

Read

No value read recently
Tap on one of the buttons above – if available – to begin

Descriptors

No Data

Properties

✓ Readable
Able to be read from

✓ Writable
Able to be written to

✗ Does not support notifications/indications
Unable to be subscribed to for notifications/indications on changes to the characteristic

08:39 66%

< Hex

Property
Hex Value

0

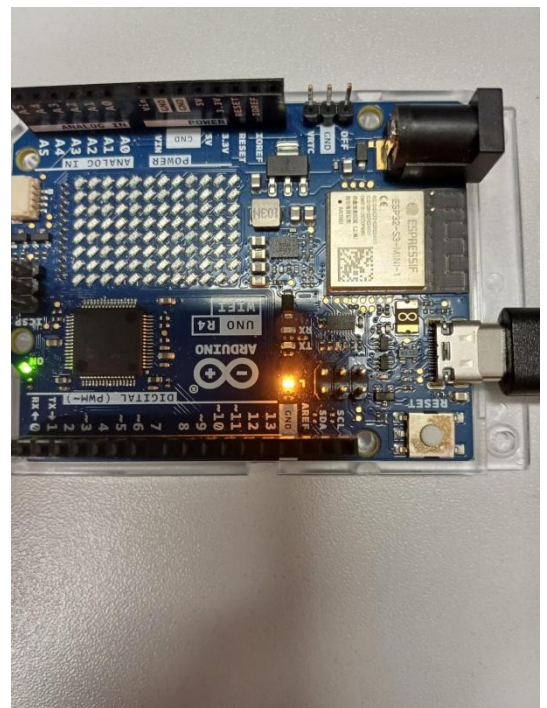
Write

| | | |
|---|---|------|
| D | E | F |
| A | B | C |
| 7 | 8 | 9 |
| 4 | 5 | 6 |
| 1 | 2 | 3 |
| x | 0 | Done |

```
21
Output Serial Monitor x
Message (Enter to send message to 'Arduino UNO R4 WiFi' on 'COM7')

RSSI: -81

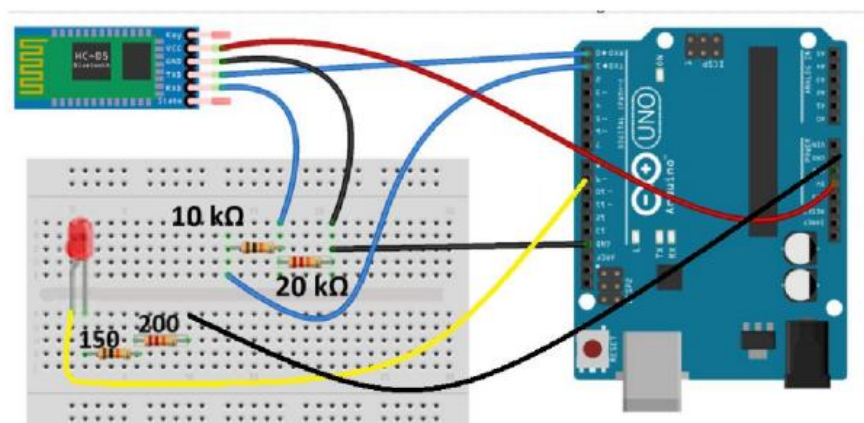
starting Bluetooth® Low Energy module failed!
starting Bluetooth® Low Energy module failed!
starting Bluetooth® Low Energy module failed!
💎💎💎💎💎
Connected to central: 72:a3:e5:bf:3b:5f
Disconnected from central: 72:a3:e5:bf:3b:5f
Connected to central: 44:3b:60:95:82:37
LED on
LED off
LED on
LED off
```



W Serial Monitor widzimy komunikaty zawarte w kodzie świadczące o połączeniu się z urządzeniem oraz z aktualnym stanem diody.

2.3. Sterowanie diodą LED wbudowaną w Arduino za pomocą bluetooth HC-05.

Postępując zgodnie ze schematem zamieszczonym w instrukcji, podłączyliśmy układ potrzebny do realizacji tego zadania.



Następnie zrealizowaliśmy program do włączania i wyłączania diody LED za pomocą aplikacji Bluetooth Electronics.

```
#include <SoftwareSerial.h>

char BluetoothData;
char OutputInfo;

int P1_value=0;
int P1_value_c1=0;
int P1_value_c2=0;
int P1_value_c3=0;

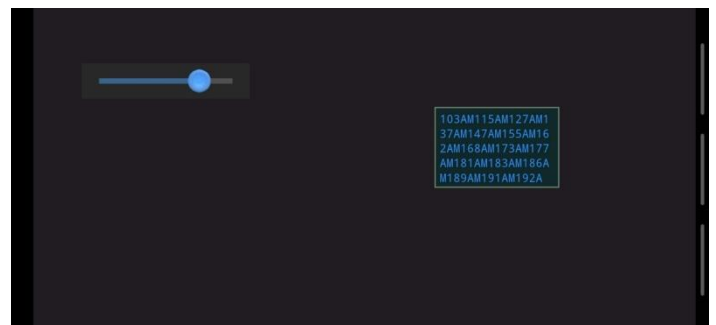
SoftwareSerial mySerial(2,3);
void setup() {
  // put your setup code here, to run once:
  pinMode(9, OUTPUT);
  Serial.begin(9600);
  mySerial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (mySerial.available())
  {
    BluetoothData = char(mySerial.read());

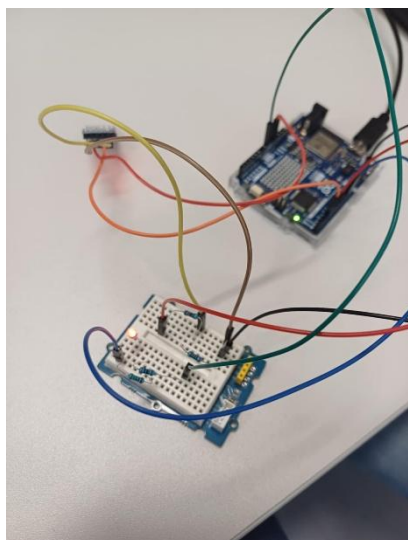
    if (BluetoothData == 'M')
    {
      P1_value = mySerial.parseInt();
      P1_value_c1 = P1_value/100+48;
      P1_value_c2 = abs((P1_value/10-(P1_value/100)*10))+48;
      P1_value_c3 = abs(P1_value-(P1_value/100)*100-((P1_value/10-
(P1_value/100)*10))*10)+48;
      Serial.write(P1_value_c1);
      Serial.write(P1_value_c2);
      Serial.write(P1_value_c3);
      Serial.write('\n');
    }
  }

  analogWrite(9, P1_value);
}
```

W aplikacji zaprojektowaliśmy panel do sterowania diodą według podanych wytycznych. Poniżej rezultat przesuwania slidera w Serial Monitor.



Celem zastosowania równań dla P1_value, P1_value_c1, P1_value_c2 i P1_value_c3 jest podzielenie liczby na cyfry i przesłanie ich jako znaki ASCII przez interfejs szeregowy. Jest to przydatne, gdy komunikujemy się z urządzeniem, które interpretuje dane w formacie tekstowym.



2.4. Sterowanie diodami LED za pomocą bluetooth HC-05 i prezentacja relacji pomiędzy intensywnościami świecenia dwóch LED.

Postępując wedle instrukcji, dokonaliśmy modyfikacji kodu dodając obsługę drugiej diody LED. Poniżej rezultaty.

```
#include <SoftwareSerial.h>
#include "Arduino_LED_Matrix.h"

// Deklaracje zmiennych do obsługi Bluetooth
char BluetoothData;

// Zmienne przechowujące jasności dwóch diod LED
int P1_value = 0;
int P2_value = 0;

// Inicjalizacja SoftwareSerial na pinach 2 (RX) i 3 (TX)
SoftwareSerial mySerial(2, 3);

// Inicjalizacja matrycy LED
ArduinoLEDMatrix matrix;

// Bufor ramki dla matrycy LED
uint8_t frame[8][12];

void setup() {
    // Ustawienie pinów jako wyjścia dla dwóch diod LED
    pinMode(9, OUTPUT); // LED 1 podłączona do pinu 9
    pinMode(10, OUTPUT); // LED 2 podłączona do pinu 10

    // Inicjalizacja komunikacji szeregowej
    Serial.begin(9600);
    mySerial.begin(9600);

    // Inicjalizacja matrycy LED
    matrix.begin();
}

void loop() {
    // Sprawdzenie, czy są dostępne dane z Bluetooth
    if (mySerial.available()) {
        BluetoothData = char(mySerial.read());

        // Jeśli odebrano 'M', czytamy jasność dla LED 1
        if (BluetoothData == 'M') {
            P1_value = mySerial.parseInt();
            Serial.print("P1_value: ");
            Serial.println(P1_value);
        }
    }

    // Ustawienie jasności dwóch diod LED
    analogWrite(9, P1_value);
    analogWrite(10, P2_value);

    // Obliczenie różnicy w jasności
    int difference = abs(P1_value - P2_value);

    // Mapowanie różnicy na liczbę kolumn do wyświetlenia (0 do 12)
    int numColumns = map(difference, 0, 255, 0, 12);

    // Wyświetlenie różnicy na wbudowanej matrycy LED
    displayDifference(numColumns);
}

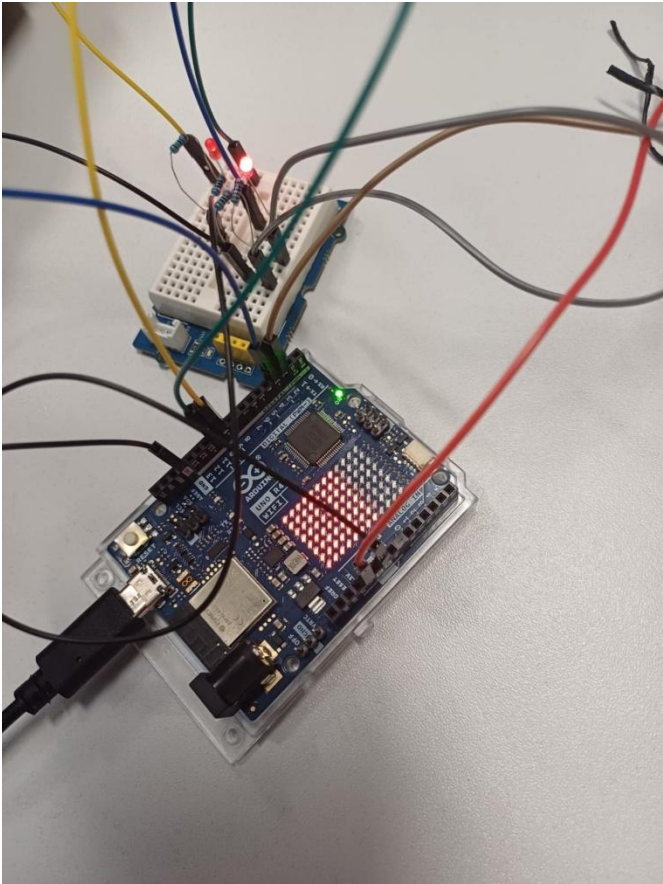
void clear_frame() {
    for (int row = 0; row < 8; row++) {
        for (int col = 0; col < 12; col++) {
            frame[row][col] = 0;
        }
    }
}

void display_frame() {
    matrix.renderBitmap(frame, 8, 12);
}

void displayDifference(int numColumns) {
    // Wyczyść bufor ramki
    clear_frame();

    // Ustaw piksele w ramce zgodnie z wartością numColumns
    for (int col = 0; col < numColumns; col++) {
        for (int row = 0; row < 8; row++) {
            frame[row][col] = 1; // Włącz piksel
        }
    }

    // Wyświetl ramkę na matrycy LED
    display_frame();
}
```

3. Wnioski.

- Udało się nawiązać komunikację między modulem HC-05 a aplikacją mobilną. Dzięki temu można było sterować jasnością diody LED w Arduino.
- Dane odbierane przez Bluetooth (zmienna P1_value) zostały prawidłowo rozbite na cyfry (setki, dziesiątki, jedności) i przesłane w formie tekstowej, co ułatwia ich interpretację przez aplikację.
- Użytkownik mógł sterować działaniem diody LED za pomocą prostego interfejsu w aplikacji, co pokazuje praktyczne zastosowanie technologii IoT.