

Sprawozdanie


Internet rzeczy: PODSTAWOWE INTERFEJSY KOMUNIKACYJNE

1. Cel ćwiczenia - Celem ćwiczenia było zapoznanie z podstawowymi interfejsami komunikacyjnymi używanymi w Arduino UNO.

2. Przebieg ćwiczenia.

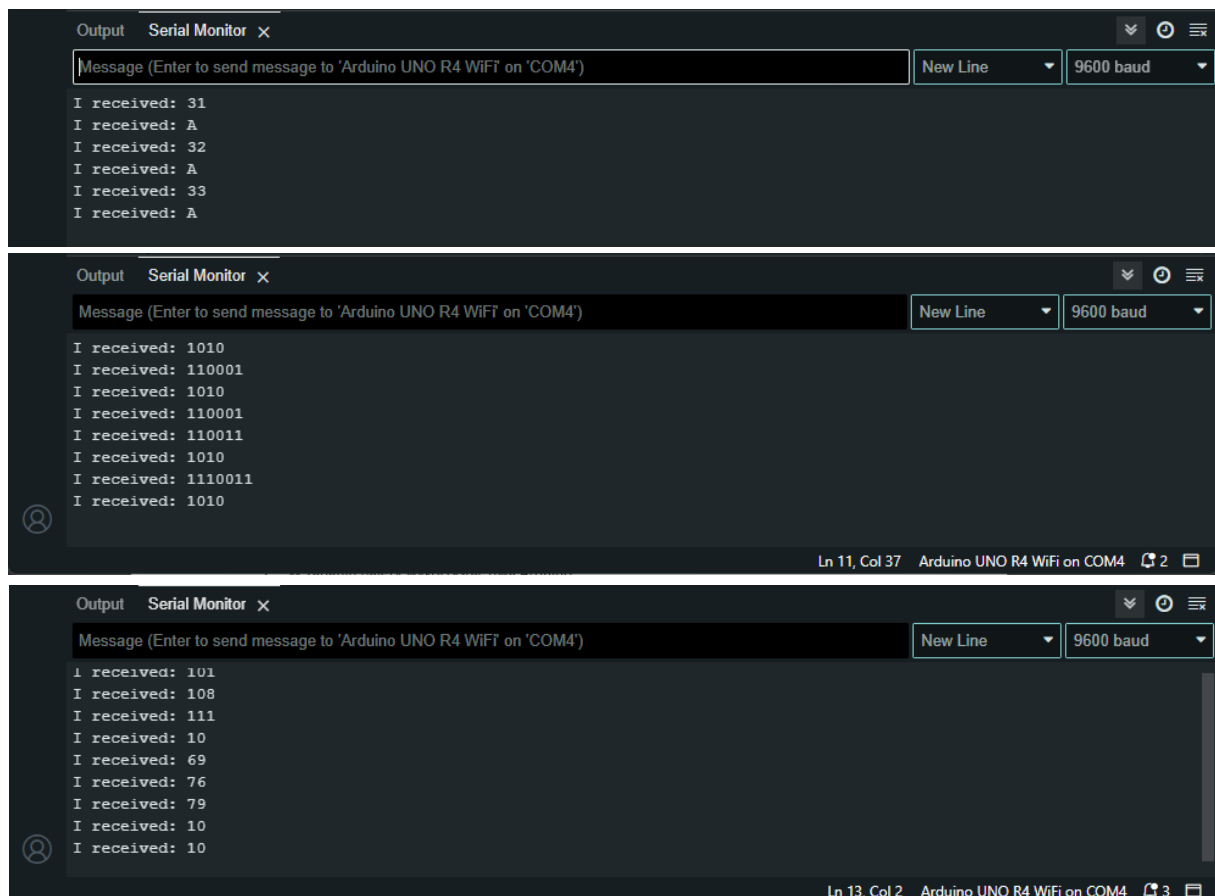
2.1. Komunikacja Arduino z komputerem z wykorzystaniem interfejsu UART i monitora portu szeregowego „Serial Monitor”.

Postępując wedle instrukcji podłączyliśmy mikrokontroler do komputera za pomocą odpowiedniego przewodu USB. Następnie w programie Arduino IDE sprawdziliśmy działanie podanego programu do komunikacji dwukierunkowej.



```
arduinko.ino
1  int incomingByte = 0;
2
3  void setup() {
4      Serial.begin(9600);
5  }
6
7  void loop() {
8      if (Serial.available() > 0) {
9          incomingByte = Serial.read();
10         Serial.print("I received: ");
11         Serial.println(incomingByte, HEX);
12     }
13 }
```

Kolejno zmienialiśmy wartość linii 11 na HEX, BIN i DEC. Oto rezultaty:

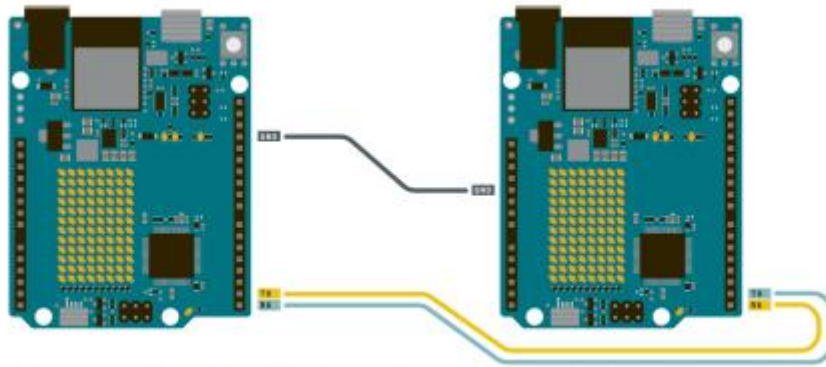


Po naciśnięciu pojedynczego znaku ENTER bez wpisywania wartości otrzymujemy:

- samo A dla HEX
- samo 1010 dla BIN
- samo 10 dla DEC

2.2. Komunikacja pomiędzy dwoma Arduino oraz komputerami z wykorzystaniem magistrali UART.

W następnym punkcie wykorzystaliśmy dwa Arduino, podłączyliśmy je według podanego schematu.



Rys. 22. Połączenie dwóch Arduino UNO do komunikacji poprzez magistralę UART

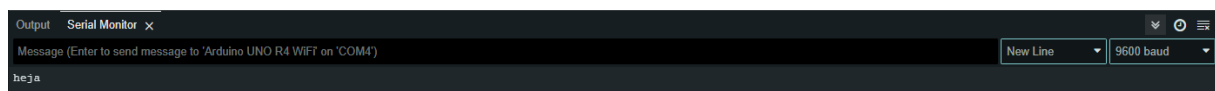
Do obu urządzeń wpisaliśmy podany w instrukcji kod.

```

arduinko.ino
1  String sendMessage;
2  String receivedMessage;
3
4  void setup() {
5      Serial.begin(9600);
6      Serial1.begin(9600);
7  }
8  void loop() {
9      // Initialize the Serial monitor for
10     // Initialize Serial1 for sending data
11     while (Serial1.available() > 0) {
12         char receivedChar = Serial1.read();
13         if (receivedChar == '\n') {
14             Serial.println(receivedMessage); // Print the received
15             receivedMessage = ""; // Reset the received message
16         } else {
17             receivedMessage += receivedChar; // Append characters to
18         }
19     }
20 }
21 if (Serial.available() > 0) {
22     char inputChar = Serial.read();
23     if (inputChar == '\n') {
24         Serial1.println(sendMessage); // Send the message through
25         sendMessage = ""; // Reset the message
26     } else {
27         sendMessage += inputChar; // Append characters to the
28     }
29 }
30 }

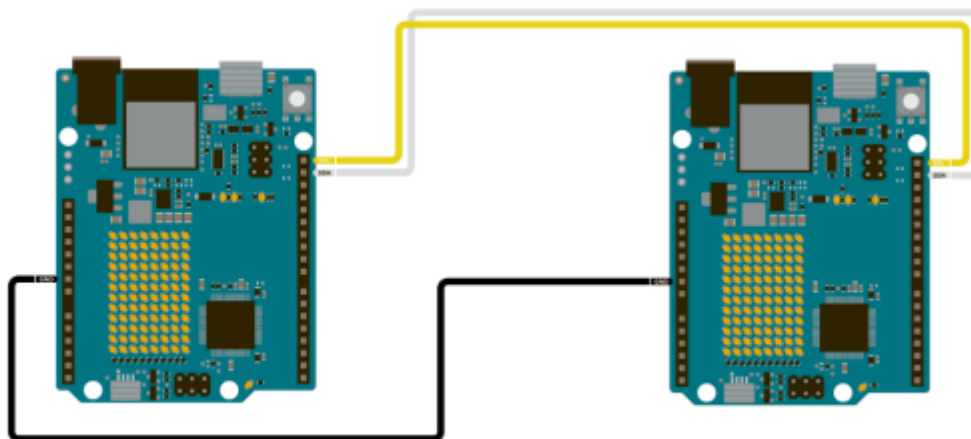
```

Rezultatem wykonanych czynności była możliwość komunikacji między mikrokontrolerami za pomocą Serial Monitor.



2.3. Komunikacja z wykorzystaniem magistrali I²C cz.1.

W zadaniu skomunikowaliśmy ze sobą dwa Arduino za pomocą synchronicznego protokołu szeregowego I²C. Nasze urządzenie pełniło rolę nadajnika.

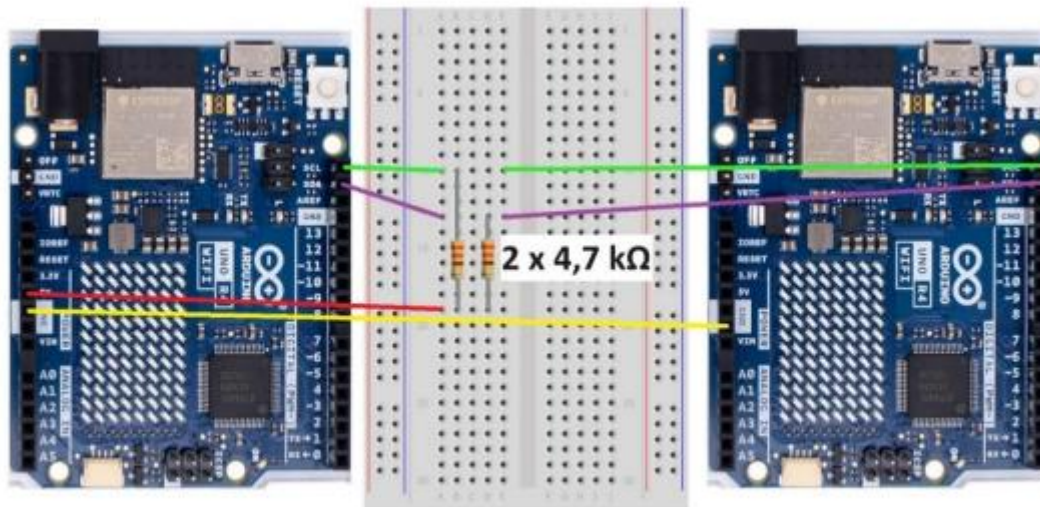


Rys. 23. Połączenie dwóch Arduino UNO do komunikacji poprzez magistralę I²C

Kod wykonawczy:

```
1  // Wire Sender
2
3  #include <Wire.h>
4
5  void setup() {
6      Wire.begin(8);
7      Wire.onRequest(requestEvent);
8  }
9
10 void loop() {
11     delay(100);
12 }
13
14 void requestEvent() {
15     Wire.write("hello ");
16 }
```

Transmisja nie była realizowana poprawnie, więc zastosowaliśmy połączenie z rezystorami pull-up.



Rys. 24. Połączenie dwóch Arduino UNO do komunikacji poprzez magistralę I2C z rezystorami pull-up

W tej sytuacji transmisja przebiegała już poprawnie. Na drugim mikrokontrolerze dało się zauważyć wiadomość „hello” wysyłaną z podanym opóźnieniem. Podłączenie rezystorów pull-up w zadaniu 3 zapewniło poprawność działania komunikacji poprzez ustabilizowanie stanów logicznych na liniach SDA i SCL. Umożliwiło to prawidłowe przesyłanie i odbieranie danych między urządzeniami, co nie byłoby możliwe bez ich zastosowania.

2.4. Komunikacja z wykorzystaniem magistrali I 2C cz.2.

W kolejnym zadaniu nasz mikrokontroler również pełnił rolę nadajnika. Do naszego Arduino wgraliśmy kod:

```
1 // Wire Master Writer
2 #include <Wire.h>
3
4 void setup() {
5     Wire.begin(); // Init i2c (dla mastera adres w nawiasie jest opcjonalny)
6 }
7
8 byte x = 0;
9
10 void loop() {
11     Wire.beginTransmission(4); //nadawanie do urządzenia o adresie #4
12     Wire.write("x is ");
13     Wire.write(x);
14     Wire.endTransmission();
15     x++;
16     delay(500);
17 }
```

Zmieniając wartości „delay” doszliśmy do wniosku, że w przypadku różnych wartości opóźnień dla nadajnika i odbiornika w magistrali I²C mogą wystąpić błędy w transmisji danych. Jeśli opóźnienie odbiornika jest zbyt krótkie w stosunku do opóźnienia nadajnika, odbiornik może wielokrotnie odczytać te same dane lub "zgubić" niektóre przesyłane bajty. Przy skróceniu opóźnienia w odbiorniku do „delay(100)” występuje częstszy odczyt danych, co może prowadzić do powtarzania wyników, gdy nadajnik nie zdąży przesłać nowych danych.

Powrót do schematu bez rezystorów pull-up skutkuje niewłaściwym działaniem układu, co potwierdza znaczenie tych rezystorów dla magistrali I²C.

2.5. Komunikacja z wykorzystaniem magistrali SPI.

W tym zadaniu zrealizowaliśmy układ oraz postępowaliśmy według podanych instrukcji, jednak nawet z pomocą prowadzących nie byliśmy w stanie zmierzyć wartości rezystancji potencjometrów. Powodem mogła być wadliwa płyta prototypowa, ponieważ po weryfikacji wszelkich podłączeń i kodu, doszliśmy do wniosku, że nie popełniliśmy żadnego błędu przy podłączaniu.

3. Wnioski.

- Ćwiczenia pokazały, jak ważne są podstawowe interfejsy komunikacyjne (UART, I²C, SPI) w integracji urządzeń elektronicznych. Różne standardy komunikacyjne mają swoje specyficzne zastosowania, jednak kluczowa dla każdego z nich jest odpowiednia konfiguracja sprzętowa i programowa.
- Rezystory pull-up są nieodzownym elementem w magistrali I²C. Ich brak prowadzi do błędów transmisji, co zostało doświadczone w praktyce. Wprowadzenie tych rezystorów stabilizuje linie SDA i SCL, zapewniając niezawodność przesyłu danych.
- Dzięki bibliotekom i prostocie implementacji Arduino pozwala na szybkie zapoznanie się z różnymi protokołami komunikacyjnymi. Ćwiczenie umożliwiło przećwiczenie zarówno transmisji jednokierunkowej, jak i dwukierunkowej z wykorzystaniem UART oraz komunikacji master-slave w I²C i SPI.