Prowadzący: dr inż. Jan Nikodem

Urządzenia Peryferyjne

Ćwiczenie 3

Zasady obsługi modemów i nawiązywania połączeń

1. Wstęp

Modem jest urządzeniem modulującym sygnał cyfrowy w elektryczny podczas nadawania danych do kabla telefonicznego oraz interpretacji sygnału otrzymywanego z tejże linii w informacje cyfrowe. Linie telefoniczne dzieli się na te działające synchronicznie i asynchronicznie. Linie synchroniczne Digital Subscriber Line (DSL) charakteryzują się tym, iż ilość możliwych bajtów wysyłanych i otrzymywanych z sieci jest taka sama. Jest to praktyczne rozwiązanie w wypadku przedsiębiorstwa, które dużo danych udostępnia w internecie (np. serwerownia). Bardziej praktycznym i częściej używanym sposobem połączenia modemu do sieci jest ADSL, co oznacza Asymmetrical Digital Subscriber Line. Jest to technologia pozwalająca na połączenie się do sieci w sposób niesymetryczny. Oznacza to, iż prędkości osiągane przy wysyłaniu danych do sieci będą znacznie mniejsze niż proces ich pobierania. Ma to zastosowanie w przypadku znacznej części komputerów stacjonarnych, gdyż operacja pobierania (np. stron internetowych, filmów czy aktualizacji) jest dużo częściej wykonywana niż wysyłanie danych do sieci.
Modem posiada zazwyczaj serię diod LED, które służą do komunikacji ze światem zewnętrznym i indykują na konkretne stany, w jakich znajduje się obecnie urządzenie.
Każda dioda informuje o innym stanie. Wygląda to następująco:

- HS (high speed) przepływ danych z maksymalną prędkością,
- AA (auto answer) stan rejestru S0, informuje czy odbiór i zaakceptowanie transmisji nastąpi automatycznie. Domyślnie SO=1. Gdy SO=0 dioda się nie świeci i nigdy nie odbierze automatycznie przychodzącej transmisji. Należy manualnie odpowiedzieć na połączenie.
- CD (carrier detected) wskazuje na to, czy modem został połączony z innym modemem za pomocą linii telefonicznej,
- OH (on hook) trwa połączenie z innym modemem przez linię telefoniczną,
- RD (read data) dane otrzymywane są z linii telefonicznej,
- TD (transfer data) dane wysyłane są na linię telefonicznej,
- MR (memory read) gotowość modemu do współpracy z komputerem.

Podczas zajęć korzystaliśmy z komend Hayes'a, dzięki którym sterowaliśmy modemami. Kilka podstawowych komend przydatnych podczas zajęć:

- ATA odbiera połączenie przychodzące,
- ATD*numer* (np. ATD3965) wybiera podany numer telefonu i próbuje nawiązać połączenie,
- ATH zerwanie połączenia,
- ATSr=n przypisanie do rejestru wybranej wartości,
- ATO powrót z trybu komend podczas połączenia (aby do niego wejść należało podać kod +++),
- ATZ powraca do ustawień początkowych modemu,
- AT&V wypisuje konfigurację,
- ATS2? Wypisuje znak wyjścia za pomocą kodu ASCII, w naszym przypadku '043', czyli '+'. W przypadku podania ATS2=45 można było podczas aktywnego połączenia przejść w tryb komend podając znaki „---„ zamiast domyślnych „+++".

2. Przebieg zajęć

Podczas zajęć laboratoryjnych mieliśmy do użytku dwa komputery stacjonarne i dwa modemy modelu Pentagram Shadow x56 połączone portem szeregowym RS-232.
W pierwszej części, aby nawiązać połączenie między komputerami, uruchomiliśmy emulator konsoli PuTTy, podając mu portu szeregowy COM1 jako sposób połączenia, modem był połączony do tego portu. Ustawiliśmy zmienne:

- Szybkość transmisji danych na 9600 bitów/sekundę,
- Przesyłana ramka miała ustawione 8 bitów,
- 1 bit stopu,
- Brak bitu parzystości.

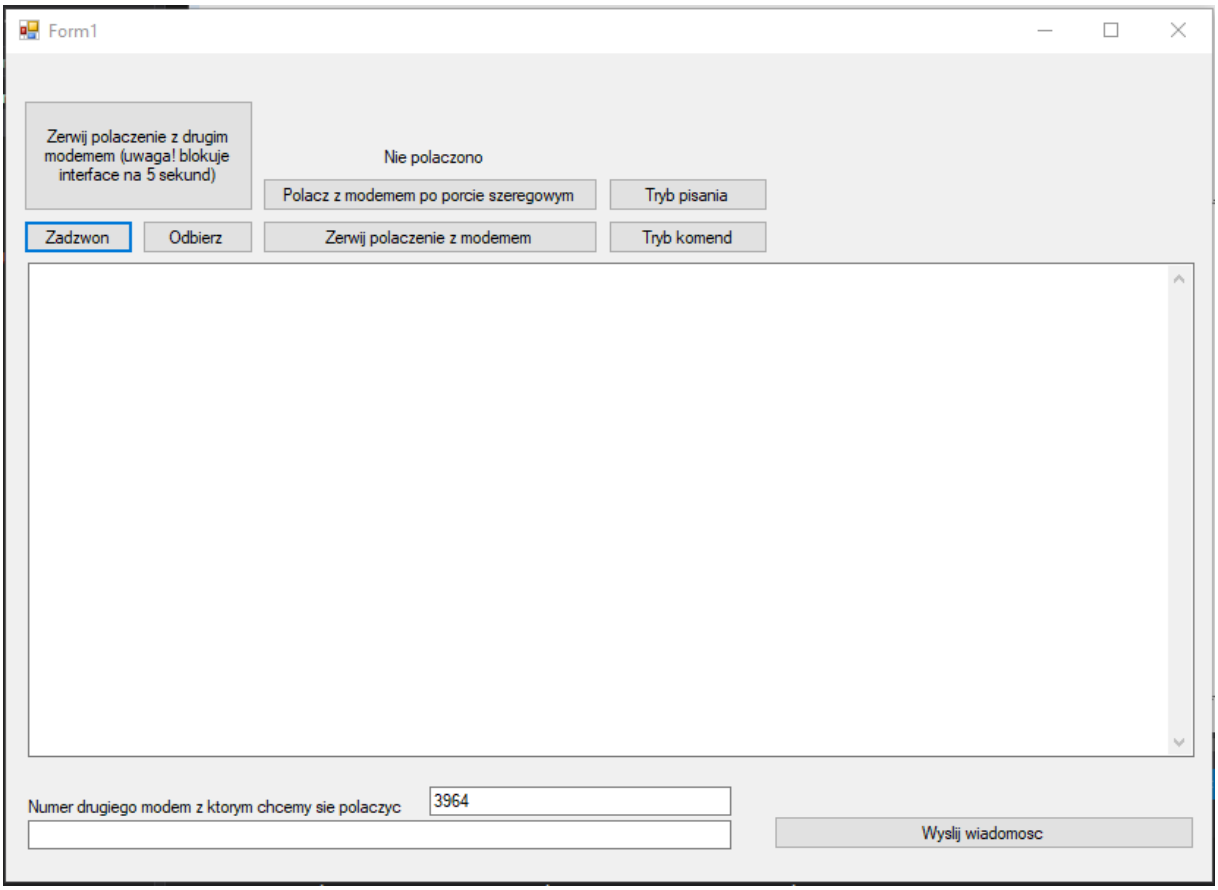
Po wprowadzeniu takiej konfiguracji przeszliśmy do połączenia się między modemami. Modemy były podłączone do wewnętrznej linii telefonicznej i miały numery następująco 3964 oraz 3965.

Aby połączyć modemy, jednym z nich zadzwoniliśmy na drugi za pomocą komendy: 'ATD3965', w tym czasie na drugim modemie pojawił się napis: 'RING', czyli przychodzące połączenie. Wpisanie komendy 'ATA' oznajmiało chęć manualnego odebrania rozmowy i jeżeli wszystko przeszło pomyślnie, rozpoczynało polaczenie między modemami. Po pojawieniu się komunikatu CONNECTED mogliśmy już zacząć wymieniać ze sobą wiadomości. Wejście do trybu komend, nie zrywając połączenia, wymagało wysłania '+++'. Aby wyjść z trybu komend i powrócić do wymiany wiadomości wystarczyło wpisać komendę 'ATO'. Do zakończenia połączenia między modemami wykorzystywaliśmy komendę 'ATH'. Po jej wykonaniu na drugim modemie pojawiała się informacja 'NO CARRIER', oznaczająca, że modem nie jest połączony z innym modemem.

Po wielokrotnych testach (których nie można nie usłyszeć z powodu dźwięków radiowych wydobywających się z modemów podczas dzwonienia), aby sprawdzić, kiedy świecą dane diody i jak modem reaguje na komendy, zabraliśmy się za ostatni etap laboratorium, czyli napisanie programu. Program miał w ostatecznej formie nawiązywać komunikację z drugim komputerem, logując się na ten komputer (albo na serwer podany przez prowadzącego, albo na drugi komputer z modem) i pobierać ze zdalnego komputera pewne dane (np. katalog systemowy) i wydrukować to na ekranie.

3. Opis programu

Program do swojej obsługi wykorzystuje przystępny i przejrzysty interfejs graficzny.



Aby połączyć się z modem, podłączonym do komputera, przez interfejs szeregowym RS-232, należy nacisnąć przycisk „Połącz z modemem po porcie szeregowym”.

```
1.  private void button2_Click(object sender, EventArgs e) {
2.      if (!port.IsOpen) {
3.          port.PortName = "COM1";
4.          port.Parity = Parity.None;
5.          port.DataBits = 8;
6.          port.BaudRate = 9600;
7.          port.StopBits = StopBits.One;
8.          port.Handshake = Handshake.RequestToSend;
9.          port.DataReceived += dataReceived;
10.         port.RtsEnable = true;
11.         port.DtrEnable = true;
12.
13.         port.WriteTimeout = 500;
14.         port.ReadTimeout = 500;
15.         try {
16.             port.Open();
17.             //port.Write("ATE0\r");   // wylaczenie ECHA
18.             WypiszNaEkran("Nawiazano polaczenie\n");
19.             label2.Text = "Polaczono";
20.         }
21.         catch (Exception ex) {
22.             WypiszNaEkran("Blad!\n" + ex.Message);
23.         }
24.     }
25. }
```

Na początku należy sprawdzić, czy przypadkiem nie jesteśmy już połączeni na wybranym porcie. Następnie następuje konfiguracja obiektu „port" klasy „SerialPort". Przypisuje się jemu wybrany numer portu „COM1", ustawia się na brak bitu parzystości, ramce przypisuje się 8 bitów, prędkość na 9600 bitów/sekundę oraz wpisuje się jeden bit stopu. Ustawienia są takie jak podczas łączenia się wcześniej przez program terminalowy Putty.
Następnie ustawienie metody Hanshakingu. Działa to tak, że podczas próby wysłania danych, urządzenie nadawcze steruje linią RTS (Ready to Send) a urządzenie odbiorcze potwierdza, że jest gotowe przyjąć dane, sygnalizując to linią CTS (Clear to Send).
W linii 8, w pole DataReceived przypisujemy metodę, która ma zostać wywołana, jeśli to na porcie szeregowym pojawią się jakieś dane. Można byłoby to również rozwiązać za pomocą utworzenia nowego wątku, który by starał się odczytać dane w tle, ale w ten sposób wykorzystujemy zdarzenie asynchroniczne, a nie używamy metody pollingu (ręcznego odpytywania).
Na koniec otwieramy port szeregowy w ten sposób nawiązując połączenie z modemem.

Aby zamknąć port, należy nacisnąć przycisk „zerwij połączenie z modemem".

```
1.  private void button6_Click(object sender, EventArgs e) {
2.      if (port.IsOpen) {
3.          port.Write("ATH\r");
4.          port.Close();
5.          label2.Text = "Nie polaczono z modemem";
6.      }
7.  }
```

Metoda ta sprawdza czy port jest na pewno otworzony, wysyła komendę „ATH" aby rozłączyć ewentualne połączenia z innym modemem oraz zamyka port.

Kolejną częścią była implementacja komunikacji z użytkownikiem. Wcześniej było wspomniane o metodzie, która działa asynchronicznie, czyli „dataReceived". Poniżej jest ona uzupełniona.

```
1.  private void dataReceived(object sender, SerialDataReceivedEventArgs e) {
2.      var data = port.ReadExisting();
3.      if (data.Contains("CON")) {
4.          polaczono = true;
5.          WypiszNaEkran("Polaczono z drugim modemem.");
6.      }
7.      WypiszNaEkran("<-" + data + "\n");
8.  }
9.
10. private void WypiszNaEkran(string data) {
11.     textBox3.Text += data;
12.
13.     textBox3.SelectionStart = textBox3.TextLength;
14.     textBox3.ScrollToCaret();
15. }
```

. Dane które zostaną odebrane, zostają wypisywane na ekranie.

Do pełni możliwości komunikowania się z modemem, należy jeszcze dodać możliwość wysłania komend (lub zwykłych ciągów znaków). Po wciśnięciu przycisku „Wyślij wiadomość” zostaną przesłane dane.

```
1.  private void button4_Click(object sender, EventArgs e) {
2.      if (port.IsOpen) {
3.          port.Write(textBox2.Text + "\r");
4.          WypiszNaEkran("->" + textBox2.Text + Environment.NewLine);
5.      }
6.  }
```

Kolejną dodaną przez nas funkcjonalnością, jest możliwość zadzwonienia pod wybrany numer lub odebrania połączenia.

```
1.  private void button3_Click(object sender, EventArgs e) {
2.      if (port.IsOpen) {
3.          polaczono = true;
4.          port.Write("ATA\r");
5.          WypiszNaEkran("Odbieranie...");
6.      }
7.  }
8.
9.  private void button1_Click(object sender, EventArgs e) {
10.     if (port.IsOpen)
11.         port.Write("atd" + textBox1.Text + "\r");
12.
13. }
```

W przypadku obierania połączenia, zostaje przesłana do modemu komenda „ATA”. Aby wykonać połączenie z wybranym modemem należało podać numer w polu tekstowym.

Dodatkową rzeczą, którą wprowadziliśmy, było umożliwienie przejścia modemu w tryb komend podczas gdy ten miał aktywne połączenie. Szybkie wysłanie znaków „+++” skutkowało tym, że odbiorca odebrał te znaki, lecz modem nie wyszedł z trybu tekstowego. Można to było rozwiązać, wpisując pojedynczy znak „+” a następnie wcisnąć przycisk „wyślij” z pewnym odstępem. My napisaliśmy metodę, która robi to za nas.

```
1.  private void button7_Click(object sender, EventArgs e) {
2.      if (port.IsOpen) {
3.          if (polaczono) {
4.              WypiszNaEkran("Tryb komend\n");
5.              port.Write("+");
6.              Thread.Sleep(100);
7.              port.Write("+");
8.              Thread.Sleep(100);
9.              port.Write("+");
10.             Thread.Sleep(100);
11.         }
12.     }
13. }
```

Wysyła ona znak „+” z pewnym krótkim odstępem czasowym.

```
1.  private void button8_Click(object sender, EventArgs e) {
2.      if (port.IsOpen) {
3.          if (polaczono) {
4.              WypiszNaEkran("Tryb pisania\n");
5.              port.Write("ato\r");
6.          }
7.      }
8.  }
```

Powyższa metoda pozwala na powrót do trybu tekstowego.

Ostatnią metodą, którą zaimplementowaliśmy, była możliwość zerwania połączenia. Można to było rozwiązać w prosty sposób, zamykając port szeregowy i otwierając go ponownie. Jednak uznaliśmy, że lepszym sposobem będzie skorzystanie z komendy „ATH". Żeby móc ją wpisać, należy tak jak wcześniej, wysłać znaki „+++". Jak zauważyliśmy w testach, aby modem na pewno zerwał połączenie, trzeba było dwa razy wysłać komendę „ATH" (nie zawsze za pierwszym razem zostawało połączenie zakończone).

```
1.  private void button5_Click(object sender, EventArgs e) {
2.      if (port.IsOpen) {
3.          if (polaczono) {
4.              WypiszNaEkran("Zrywanie...\n");
5.              port.Write("+");
6.              Thread.Sleep(100);
7.              port.Write("+");
8.              Thread.Sleep(100);
9.              port.Write("+");
10.             Thread.Sleep(100);
11.             Thread.Sleep(1000);
12.             port.Write("ATH\r");
13.             Thread.Sleep(5000);
14.             port.Write("ATH\r");
15.             polaczono = false;
16.         }
17.     }
18. }
```

4. Wnioski

Podczas zajęć proces połączenia dwóch komputerów przy pomocy modemów i linii telefonicznej z wykorzystaniem programu PuTTY wykonaliśmy szybko, więcej czasu poświeciliśmy na pełne zrozumienie komend, aby później wykorzystać je w kodzie, niż na samo połączenie. Bardzo przydała się znajomość komend Hayes'a. Wystarczającą ilość informacji potrzebnych do diagnozy stanu połączenia dają diody umieszczone na modemie. Sam modem w nietrudny sposób można też konfigurować, zmieniając wartości jego rejestrów. Napisanie programu dedykowanego wymianie informacji i połączenia się z drugim komputerem wymagało przemyślenia. Istotne jest wyznaczenie takich samych parametrów dla obu komputerów, gdyż w przeciwnym wypadku komunikacja nie powiedzie się lub będzie dawać niechciane rezultaty. Stworzyliśmy interfejs graficzny, który za pomocą przycisków zawierał wszystkie najbardziej potrzebne komendy do obsługi modemu, podczas wymiany wiadomości. Dzięki temu, pisząc, mogliśmy już skupić się głównie na wymianie informacji.