Urządzenia Peryferyjne

Modemy

Nikita Stepanenko, 245816

Termin zajęć Środa 11:00 TP

1.Cel ćwiczenia

Celem wykonywanego ćwiczenie było:
1) wykorzystując dwa modemy przetestować komendy Hayes`a według instrukcji umieszczonej na stronie Prowadzącego;

2) stworzenie programu, który będzie umożliwiać komunikację z modemem oraz przesyłanie plików za pomocą protokołu XModem. Program należało robić na dwa etapy:

a) w pierwszym program działa jak terminal

b) w drugim rozszerzamy program o możliwość przesyłania lub odbierania pliku za pomocą protokołu XModem.

2.Wstęp teoretyczny

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.

3.Realizacja ćwiczenia

W trakcie zajęć udało się wykonać następne zadania:
1. Przetestować komendy Hayes`a.

Modemy były podłączone do wewnętrznej linii telefonicznej i miały numery następująco 3964 oraz 3965. Dla połączenia się korzystaliśmy z emulatora konsoli „Putty”. Po konfiguracji Putty, aby połączyć modemy zadzwoniliśmy jednym modem na drugi, wprowadzając komendę „ATD3964” oraz na drugim komputerze w konsoli po kilku sekundach pojawił się napis „RING”, za pomocą

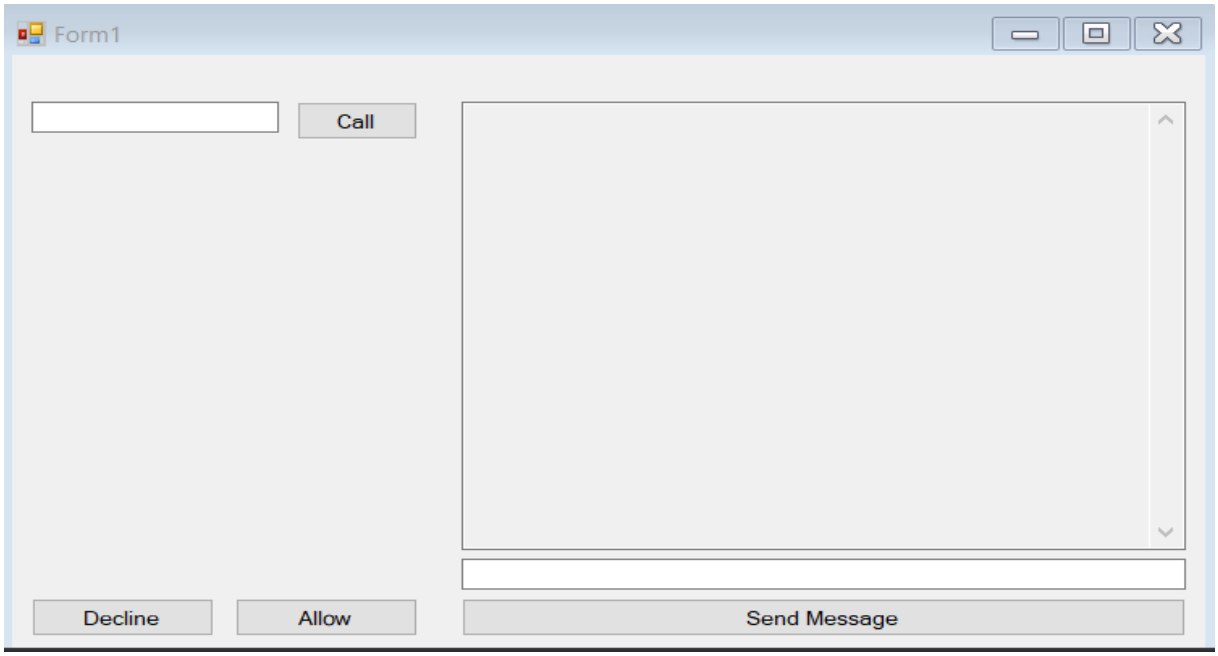
komendy „ATA” odebraliśmy rozmowę. Mogliśmy wymieniać ze sobą wiadomości. Następnie przetestowaliśmy kilka różnych komend Hayes`a oraz przystąpiliśmy do napisania programu.

2. Napisać program który umożliwia komunikację z modemem, działając jako terminal.

Opis programu

Program został napisany z wykorzystaniem języka C#. Po uruchomieniu programu mamy możliwość połączyć się z modemem.

Menu programu:



Używane biblioteki:

```
System.Collections.Generic;
System.Windows.Forms;
System.Threading;
System.IO.Ports;
```

Kod:

Dla inicjalizacji modemu w programie wykożystany następujący kod:

```
private SerialPort serialPort = new SerialPort("COM1", 9600, Parity.None, 8, StopBits.One);
```

Otwarcie portu połączenia komputera z modemem:

```
InitializeComponent();
serialPort.Open();
serialPort.RtsEnable = true;
//serialPort.Write("ATZ\r");    // resetowanie parametrów modemu
serialPort.Write("ATE1\r");
```

Polecenie ATE1 wyświetlia znaki wysłane do modemu.

Trzeba powiedzieć że ważnym atrybutem jest ‘\r’ po zakończeniu polecenia. Coś podobnego do przejścia na nową liniejkę (‘\n’).

Do dzwonienia wykożystany następujący kod:

Najpierw sprawdzamy: czy jest otwarty port połączenia między modemem a komputerem, później wykożystujemy polecenie Hayes'a ATD do połącznia z numerem wpisanym `textBoxPhoneNumber`.

```
if (serialPort.IsOpen)
{
    serialPort.Write("atd" + textBoxPhoneNumber.Text + "\r");
}
```

Do rozerwania połączenia telefonicznego wykożystany następujący kod:

Najpierw sprawdzamy: czy jest otwarty port połączenia między modemem a komputerem, później wykożystujemy polecenie +++ przechodzenia do trybu komend i wykożystujemy komendę ATH do rozerwania połączenia.

```
if (serialPort.IsOpen)
{

    serialPort.Write("+");
    Thread.Sleep(100);
    serialPort.Write("+");
    Thread.Sleep(100);
    serialPort.Write("+");
    Thread.Sleep(100);
    serialPort.Write("ath\r");
}
```

Do wysyłania wiadomości jest wykorzystany prosty kod, w którym `messageText` to wiadomość do wysyłania:

```
serialPort.Write(messageText);
```

Na następnym zdjęciu pokazany cały kod wysyłania wiadomośći, oraz przechodzenie do trybu komend i odwrotnie (oraz sprawdzenie portu połączenia):

```
messageText = textBoxSendMessage.Text;
if (serialPort.IsOpen)
{
    // przechodzenie do trybu komend
    if (textBoxSendMessage.Text[0] == '+' && textBoxSendMessage.Text[1] == '+' && textBoxSendMessage.Text[2] == '+')
    {
        commandMode = true;
        serialPort.Write("+");
        Thread.Sleep(100);
        serialPort.Write("+");
        Thread.Sleep(100);
        serialPort.Write("+");
        Thread.Sleep(100);
    }
    // jeżeli jesteśmy w trybie komend
    if (commandMode)
    {
        // wpisywanie komend
        serialPort.Write(messageText + "\r");
        // jeżeli wpiszemy ato lub ATO, to powrót do trybu danych
        if ((textBoxSendMessage.Text[0] == 'a' && textBoxSendMessage.Text[1] == 't' && textBoxSendMessage.Text[2] == 'o')
            || (textBoxSendMessage.Text[0] == 'A' && textBoxSendMessage.Text[1] == 'T' && textBoxSendMessage.Text[2] == 'O'))
        {
            commandMode = false;
        }
    }
    else
    {
        // zwykła wiadomość (przesyłanie danych)
        serialPort.Write(messageText);
    }
}

textBoxSendMessage.Text = "";
// wyświetlanie odprawionej wiadomości
textBoxMessageBox.AppendText(messageText + Environment.NewLine);
messageText = "";
```

Dla wyświetlania odebranych wiadomości korzystamy z dodatkowego wątku(żeby wiadomiści byli wyświetlane na bieżąco):

```
Thread mesThread = new Thread(messageThread);
mesThread.Start();
```

```
private void messageThread()
{
    textBoxMessageBox.Text = "";
    while (true)
    {
        try
        {
            string message = serialPort.ReadExisting();
            if (message.Length > 0)
            {
                BeginInvoke(new Action(() =>
                {
                    textBoxMessageBox.AppendText(message);
                }));
            }
        }
        catch (TimeoutException) { }
    }
}
```

4.Wnioski

W trakcie wykonywania ćwiczenia żadnych trudnych problemów nie było. Łatwo przetestowaliśmy komendy Hayes`a. Trochę czasu nam zajęło zrozumienie wszystkich komend. Niestety udało napisać tylko 1 etap programu (program działa jako terminal). Ważne przy napisaniu programu było wyznaczenie takich samych parametrów dla obu komputerów.