POLITECHNIKA WROCŁAWSKA
WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

Urządzenia peryferyjne

Laboratorium 2

Obsługa kamery USB

Termin zajęć: Środa TP, 17:30

Autorzy: Prowadzący zajęcia:
Daria Jeżowska, 252731 dr inż. Jarosław Mierzwa

Cel ćwiczenia

Celem ćwiczenia było:

1. Wykorzystanie dwóch modemów do przetestowania komend Hayes'a. W laboratorium znajdowały się dwa modemy połączone do linii telefonicznej Modemy były podłączone do komputerów przez port RS232.
2. Napisanie programu umożliwiającego komunikację z modemami

Wstęp

Modem jest to urządzenie elektroniczne, które moduluje sygnał w celu zakodowania informacji cyfrowych, tak by mogły być przesyłane w wybranym medium transmisyjnym, a także demoduluje tak zakodowany sygnał w celu dekodowania odbieranych danych. Najbardziej znanym przykładem jest modem akustyczny zamieniający cyfrowe dane z komputera osobistego na modulowany sygnał elektryczny w zakresie częstotliwości akustycznej kanału telefonicznego. Te sygnały mogą być przekazywane przez linie telefoniczne i demodulowane przez inny modem po stronie odbiornika, aby odzyskać dane cyfrowe.

Przebieg ćwiczenia

1. Testowanie komend Hayes'a odbyło się przy pomocy programu PuTTY.
   Zmienne w programie były ustawione następująco:
   Szybkość transmisji danych:9600 bit/s
   Przesyłana ramka: 8 bit
   1 bit stopu
   Brak bitu parzystości

   Połączenie między modemami było inicjowane komendą ATD<numer modemu>. Po wpisaniu w terminal programu PuTTY tego polecenia z właściwym numerem modemu na terminalu podłączonym do modemu drugie można było zobaczyć komunikat RING. Aby odpowiedzieć na połączenie na drugim modemie należało wpisać polecenie ATA.
   Komendy Hayes'a testowane podczas laboratorium:

➢ ATA odbiera połączenie przychodzące

➢ ATDnumer (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

2. Program został zrealizowany w języku C# z wykorzystaniem między innymi biblioteki System.IO.Ports. Biblioteka ta zawiera klasy do służące do kontrolowania portów szeregowych. Najważniejszą klasą jest SerialPost. Klasa zawiera metody synchroniczne oparte na zdarzeniach operacji wejścia/wyjścia oraz umożliwia dostęp do właściwości sterownika szeregowego. Interfejs programu jest w formie terminala.

Nawiązanie połączenia

Nawiązywanie połączenia z modemem odbywało się przy pomocy obiektu klasy SerialPort. W konstruktorze podane zostały wartości opisujące połączenie.

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

Czytanie wiadomości

Aby czytanie wiadomości wysyłanych przez modem i pisanie komend mogło odbywać się w tym samym czasie metoda odpowiedzialna za odczytywanie wiadomości z portu szeregowego uruchomiona została na innym wątku.

```
Thread readThread = new Thread(Read);
```

Metoda Read odpowiedzialna jest za sprawdzenie czy na porcie szeregowym nie pojawiły się nowe linie. Gdy takie się pojawią zostają one przekierowane na konsole programu.

```
public static void Read()
{
    while (_continue)
    {
        try
        {
            string message = _serialPort.ReadLine();
            Console.WriteLine(message);
        }
        catch (TimeoutException) { }
    }
}
```

Pisanie komend wysyłanych na port szeregowy

W monecie naciśnięcia klawisza wiadomość rozbudowywana jest o kolejne znaki. Jeśli wciśnięty klawisz to jest enter a treść wiadomości to „quit” program kończy swoje działanie w przypadku gdy nie jest to „quit” wiadomość jest kierowana na port szeregowy.

```
while (_continue)
{
    if (Console.KeyAvailable)
    {
        ConsoleKeyInfo sign = Console.ReadKey(false);
        while (sign.Key != ConsoleKey.Enter)
        {
            message += sign.KeyChar;
            sign = Console.ReadKey(false);
        }
        if (stringComparer.Equals("quit", message))
        {
            _continue = false;
        }
        else
        {
            message += "\r";
            Console.WriteLine(message);
            _serialPort.WriteLine(message);
            message = "";
        }
    }
}
```