

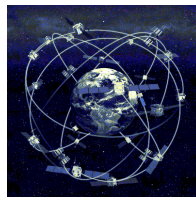


Katedra Systemów Geoinformatycznych

System GPS i jego zastosowania

Laboratorium

Ćwiczenie 3 – Komunikacja szeregowa z odbiornikiem GPS w technologii .NET Compact Framework



Wstęp

W ramach ćwiczenia Student zapozna się z podstawami odczytu danych z portu szeregowego w urządzeniach przenośnych wyposażonych w system Windows Mobile w wersji 5.0 lub wyższej. Zadanie zostanie zrealizowane na przykładzie aplikacji odczytującej dane pochodzące z symulatora odbiornika GPS przekazującego dane w formacie NMEA 0183.

Krótko o systemie GPS

System GPS powstał w roku 1973 na zlecenie Departamentu Obrony USA. Miał on służyć wojsku do bardzo precyzyjnego ustalania pozycji w dowolnym miejscu na świecie niezależnie od warunków klimatycznych, geograficznych, oraz pory dnia i nocy. Początkowo nazywał się on DNSS (Defense Navigation Satellite System) - Obronny Nawigacyjny System Satelitarny i przeznaczony był dla lotnictwa i marynarki wojennej USA. Jego koncepcja została zatwierdzona w roku 1979 pod nazwą Navstar GPS (Navigational Satellite Time and Ranging, Global Positioning System) - Układ Nawigacji Satelitarnej Określania Czasu i Odległości Globalnego Systemu Pozycjonowania. System GPS opiera się na zespole 24 satelitów rozmieszczonych na orbicie okołoziemskiej, na wysokości ok. 20 162 km nad ziemią. Satelity te okrążają kulę ziemską dwa razy w ciągu doby (okres obiegu Ziemi przez satelitę wynosi 11h 57m 27s) transmitując w sposób ciągły, w kierunku globu, sygnały radiowe o częstotliwościach 1227,60 MHz (częstotliwość wojskowa) oraz 1575,42 MHz (częstotliwość cywilna). W styczniu 1978 r. został umieszczony na orbicie pierwszy satelita systemu, a w lipcu 1995r. system uzyskał pełną sprawność operacyjną. Decyzja Kongresu USA, GPS został dopuszczony do zastosowań cywilnych, lecz przy ograniczonej dokładności do 40 metrów. Dokładność taka była wynikiem celowego wprowadzenia programu zakłócającego SA (Selective Availability) - system ograniczonego dostępu. Dnia 01-05-2000r. decyzją rządu USA program ten został wyłączony. Obecnie system jest zarządzany przez połączone biuro Navstar (GPS JPO - Navstar GPS Joint Program Office), złożone z przedstawicieli sił powietrznych, marynarki, sił lądowych, piechoty morskiej, straży przybrzeżnej USA, kwatery głównej NATO i Australii. W odpowiedzi na amerykański GPS w Rosji powstał system GLONASS. Jego znikoma obecność na rynku tłumaczy brak odbiorników cywilnych. Występują one tylko jako wojskowe i okrętowe.

System GPS a urządzenia mobilne

Obecna popularność systemu GPS wynika w dużej mierze z postępu technologicznego w dziedzinie tworzenia odbiorników GPS skutkującego między innymi w spadających kosztach produkcji. Obecnie, miniaturowe odbiorniki GPS montowane są w wielu rodzajach urządzeń przenośnych umożliwiając stały rozwój systemów informacyjnych korzystających z usług lokalizacji. Na rynku dostępne są także miniaturowe wersje odbiorników GPS, które można zamontować na dowolnym urządzeniu elektronicznym, które w połączeniu np. z modemem GSM umożliwiają zdalne monitorowanie pozycji geograficznej.

Protokół NMEA 0183

Protokół NMEA został opracowany przez National Marine Electronics Association w celu standaryzacji przesyłania danych pomiędzy urządzeniami elektronicznymi stosowanymi do monitoringu środowiska w warunkach morskich. Celem opracowanego standardu było umożliwienie jednolitej komunikacji pomiędzy szeregiem urządzeń elektronicznych pracujących na statkach. Obecnie NMEA 0183 jest najczęściej stosowaną formą przesyłania danych także w odbiornikach GPS. W praktyce oprogramowanie odbiornika PGS polega, więc na odczytywaniu odpowiednich sentencji protokołu NMEA oraz na ich prawidłowej interpretacji. Co ważne, dane przekazywane przez protokół NMEA zawierają nie tylko informację o pozycji odbiornika GPS, ale także dane dotyczące ilości śledzonych satelitów, ich położeniu, jakości wyznaczanej pozycji oraz wiele innych przydatnych informacji nawigacyjnych. W niniejszej instrukcji ograniczono się jedynie do analizy podstawowych informacji protokołu NMEA zawartych w sentencji GGA, której przykładową zawartość pokazano poniżej:

\$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

- GGA - Nazwa znacznika: Global Positioning System Fix Data,
- 123519 - czas wygenerowania pozycji: 12:35:19 UTC,
- 4807.038,N - Szerokość geograficzna 48 stopni, 07.038' N,
- 01131.000,E - Długość geograficzna 11 stopni, 31.000' E,
- 1 - jakość wyznaczenia pozycji:
 - 0 = nieprawidłowa,
 - 1 = Standardowa jakość pozycji (SPS),

- 2 = Z wykorzystaniem DGPS,
- 3 = Z wykorzystaniem PPS,
- 4 = Tryb RTK (Real Time Kinematic),
- 5 = Tryb Float RTK,
- 6 = pozycja przybliżona w trybie zliczania pozycji (dead reckoning),
- 7 = tryb manualny,
- 8 = tryb manualny symulacyjny,
- 08 - ilość satelit śledzonych przez odbiornik GPS,
- 09 - poziome rozmycie pozycji (ang. horizontal dilution of position),
- 545.4,M - wysokość w m.n.p.m,
- 46.9,M - wysokość geoidy względem elipsoidy WGS-84w zadanej pozycji,
- puste pole – czas w sekundach jaki upłynął od ostatniego wyznaczenia pozycji w trybie DGPS,
- puste pole (2) – numer stacji DGPS,
- *47 - suma kontrolna zawsze zaczynająca się od znaku *.

Zadanie 1

Zadanie pierwsze polega na prawidłowej konfiguracji komputera PC, która umożliwi napisanie aplikacji odczytującej dane z portu szeregowego i wyświetlenie ich na ekranie urządzenia mobilnego. Ze względu na skomplikowaną konfigurację sprzętową oraz różnorodność modeli urządzeń mobilnych zadanie zostanie zrealizowane jedynie na bazie emulatora urządzenia mobilnego z systemem Windows Mobile.

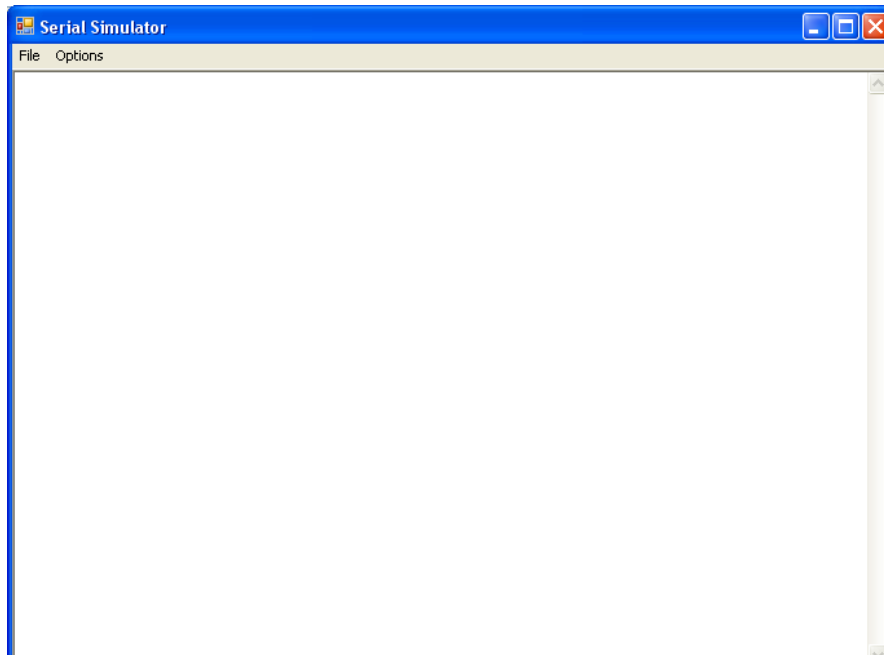
W zadaniu wykorzystane zostaną programy umożliwiające odpowiednią konfigurację komputera, a mianowicie:

- **com0com** – program stanowiący wirtualny most pomiędzy dwoma portami szeregowymi na lokalnym komputerze PC. W obecnej konfiguracji program umożliwia przekazywanie danych między portem szeregowym COM0 a COM2. Konfigurację wirtualnego mostu można sprawdzić uruchamiając narzędzie konfiguracyjne programu w *Menu Start->com0com->Setup*
- **mtty** – program umożliwiający prowadzenie nasłuchu na zadanym porcie szeregowym komputera PC

- **SerialSimulator** – projekt programu VisualStudio umożliwiającego wysyłanie zawartości pliku tekstowego na wybrany port szeregowy, w naszym przypadku COM 0. Do projektu dołączony jest plik tekstowy zawierający przykładowe sentencje protokołu NMEA 0183.

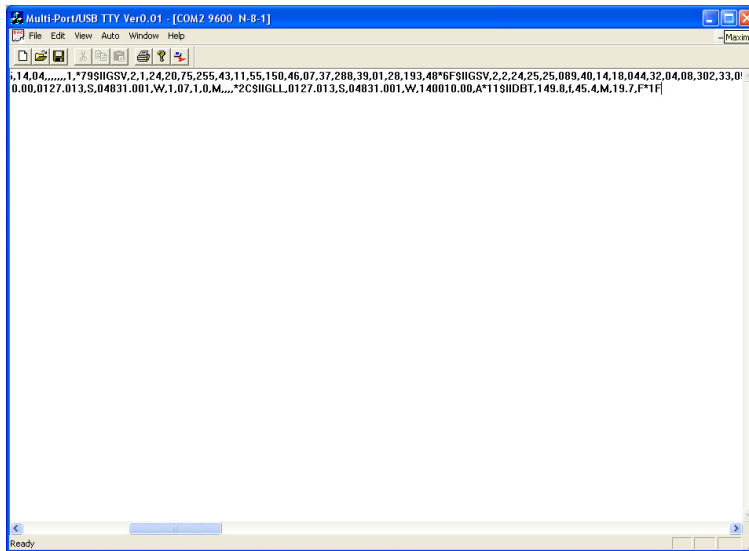
Realizację zadania 1 zacznij od uruchomienia programu *mtty.exe*. Ustaw nasłuch na port COM2 (wyjście wirtualnego mostu com0com), prędkość przesyłania danych (ang. baud rate) to 9600, brak kontroli parzystości, ilość bitów danych 8, 1 bit stopu.

Następnie, uruchom program Visual Studio i otwórz projekt *SerialSimulator*. Przeanalizuj kod projektu oraz skompiluj i uruchom program. Po uruchomieniu programu pojawi się okno dialogowe tak jak pokazano na Rys. 1.



Rys. 1. Okno dialogowe programu Serial Simulator.

Program w kodzie źródłowym ma zapisaną ścieżkę (plik *MainForm.cs*) do pliku *nmea.log* (w katalogu projektu), w którym zapisane są przykładowe dane nawigacyjne w standardzie NMEA. Po naciśnięciu *Menu->File->Start* program wywoła metodę *simulation* zdefiniowaną w pliku *MainForm.cs* która będzie odczytywała kolejne linie pliku *nmea.log* i wysyłała je na port szeregowy COM0, reprezentowany przez obiekt *serialPort* klasy **SerialPort**. W wyniku działania programu wirtualny most com0com będzie przekazywał dane z portu szeregowego COM0 na port szeregowy COM2, a w oknie aplikacji *mtty* powinny pojawić się dane nawigacyjne w standardzie NMEA, tak jak pokazano na Rys. 2.




Rys. 2. Zawartość okna aplikacji mtyy odczytana z wyjścia portu COM2.

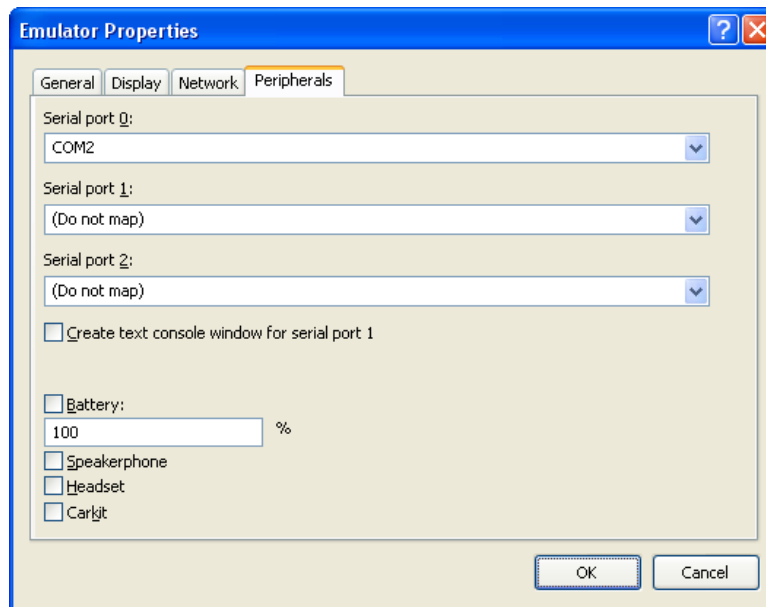
Prawidłowo przeprowadzona konfiguracja komputera PC stanowi koniec zadania 1. Przed przejściem do zadania 2 wyłącz program MTTY, tak aby nie blokować portu COM2.

Zadanie 2

W zadaniu 1 skonfigurowaliśmy wirtualny most pomiędzy portem szeregowym COM0 i COM2 na komputerze PC. Uruchomiliśmy także aplikację, która wysyła dane nawigacyjne w formie NMEA symulując tym samym działanie odbiornika GPS po stronie komputera PC. Aby dane odczytywać na emulatorze urządzenia mobilnego trzeba jeszcze prawidłowo skonfigurować sam emulator – zadanie to zostanie opisane w tym punkcie.

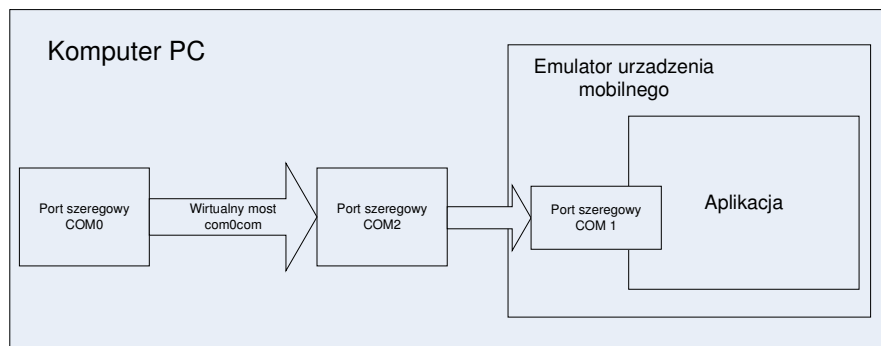
Aby przystąpić do stworzenia aplikacji na urządzeniu mobilnym, która będzie odczytywała dane z portu szeregowego należy uruchomić drugą instancję Visual Studio i stworzyć projekt typu SmartDevice o nazwie **GPS_Zad1**.

Następnie, za pomocą przycisku  uruchom emulator docelowego urządzenia mobilnego. W emulatorze należy odpowiednio skonfigurować przekierowanie portu szeregowego COM2 (wyjścia wirtualnego mostu com0com) na lokalnym komputerze PC na port szeregowy COM 1 emulatora. Realizuje się to uruchamiając okno konfiguracji emulatora (*Menu->File->Configure->Zakładka Peripherals*) i wpisując w polu *Serial Port 0*: wartość COM2 tak jak pokazano na Rys. 3.



Rys. 3. Okno właściwości emulatora.

W wyniku przeprowadzonej konfiguracji, dane, które zostaną przesłane na port COM0 komputera PC, zostaną przekierowane poprzez port COM2 komputera PC na port COM1 emulatora urządzenia mobilnego, co zostało schematycznie przedstawione na Rys. 4.



Rys. 4. Schemat przepływu danych w bieżącej konfiguracji.

Następnie możemy przystąpić do tworzenia aplikacji odczytującej dane z portu szeregowego przeznaczonej do wykonywania na urządzeniach mobilnych. Programowanie zaczynamy od deklaracji odpowiednich zmiennych w pliku **Form1.cs**, który zawiera kod źródłowy odpowiedzialny za uruchomienie głównego okna aplikacji. W klasie Form1 deklarujemy zmienne:

```
public bool czySprawdzac = false;

public delegate void UpdateTextCallback(string text);

public SerialPort serialPort;
```


Pierwsza z nich będzie stanowiła warunek kontynuacji pętli odczytu z portu szeregowego. Z uwagi na fakt, iż czytanie z portu szeregowego odbywa się w sposób asynchroniczny, aby nie blokować GUI użytkownika, aplikację zrealizujemy w trybie wielowątkowym. Potrzebny będzie więc delegat (`UpdateTextCallback`) przekazujący w zmiennej `text` zawartość odczytanej wartości z portu szeregowego reprezentowanego przez obiekt klasy `SerialPort`. Więcej na temat wykorzystania delegatów oraz programowania wielowątkowego w technologii .NET można znaleźć pod adresem:

<http://msdn.microsoft.com/en-us/library/aa446540.aspx>

Następnie, w konstruktorze klasy `Form1` po wywołaniu metody `InitializeComponent` należy utworzyć referencję na obiekt `serialPort` z odpowiednimi parametrami:

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

oraz otworzyć sam port:

```
serialPort.Open();
```

Następnie definiujemy metodę uaktualniania otrzymanego z portu tekstu (znajdującego się obiekcie `text` klasy **string**):

```
private void UpdateText(string text)
{
    // Set the textbox text.
    textBox.Text = text;
}
```

, gdzie `textBox` to nazwa obiektu klasy **TextBox**, który w GUI użytkownika pozwala nam wyświetlić dowolny łańcuch znaków. Obiekt `textBox` należy dodać z wykorzystaniem okna **Designer** tak jak to pokazano we wcześniejszych ćwiczeniach.

Należy także zdefiniować metodę odczytu z portu szeregowego, która będzie uruchamiana w osobnym wątku tak aby nie blokować GUI użytkownika:

```
private void checkPort_Thread()
{
    while (czySprawdzac)
    {
        String text = serialPort.ReadExisting();
        textBox.Invoke(new UpdateTextCallback(this.UpdateText),
        new object[] { text});
        Thread.Sleep(1000);
    }
}
```

Metoda ta po uruchomieniu będzie przekazywała zawartość portu do zmiennej *text* oraz wywołała, za pomocą metody *Invoke*, delegat *UpdateTextCallback* powodujący uaktualnienie tekstu w obiekcie *textBox* GUI użytkownika (patrz metoda *UpdateText*). Odczytywanie z portu w sposób ciągły jest zbędne, a ponadto zajmowałoby znaczną część czasu pracy procesora, dlatego jeden obrót pętli wykonuje się co sekundę poprzez usypianie wątku (*Thread.Sleep(1000)*).

Ostatnim elementem zadania 2 jest implementacja metody tworzącej nowy wątek, w którym będzie odbywało się czytanie z portu oraz przekazywanie odczytanej informacji do GUI użytkownika. Jej zawartość pokazano poniżej:

```
czySprawdzac = true;  
Thread newThread = new Thread(new ThreadStart(checkPort_Thread));  
newThread.Start();
```

Metoda powinna być uruchamiana po naciśnięciu klawisza menu o nazwie „*GetCOM*” tak jak to pokazano na Rys. 5.



Rys. 5. Wynik zadania 2

Zadanie 3

W celu weryfikacji umiejętności w zadaniu 3 należy zmodyfikować powstałą aplikację tak, aby pokazywała pozycję geograficzną oraz czas w systemie UTC. Należy się w tym celu posłużyć informacjami zawartymi we wstępie do niniejszej instrukcji.