Politechnika Wrocławska Urządzenia peryferyjne – sprawozdanie z laboratorium		
Ćwiczenie 7: GPS		19.12.11 r.
Michał Kaczara (181132)	Pn / TN, 14:15 – 17:00	Prowadzący: dr inż. Jarosław Mierzwa

1. Zadania do wykonania

Zadania do wykonania na zajęciach laboratoryjnych obejmowały następujące czynności:

- zapoznanie się z zestawem GPS oraz podłączenie go do komputera poprzez Bluetooth,
- utworzenie połączenia za pomocą programu puTTY / HyperTerminal i odczytanie oraz analiza danych przesyłanych przez GPS,
- napisanie aplikacji odczytującej, z danych przesyłanych przez GPS, współrzędne geograficzne lokalizacji i wyświetlającej lokalizację w osobnym oknie z wykorzystaniem Google Maps.

2. Wstęp teoretyczny

GPS (ang. *Global Positioning System*) to najpopularniejszy z systemów nawigacji satelitarnej, obejmujący swoim zasięgiem całą kulę ziemską. System składa się z trzech segmentów: segmentu kosmicznego - 31 satelitów orbitujących wokół Ziemi na średniej orbicie okołoziemskiej; segmentu naziemnego - stacji kontrolnych i monitorujących na ziemi oraz segmentu użytkownika - odbiorników sygnału. Zadaniem systemu jest dostarczenie użytkownikowi informacji o jego położeniu oraz ułatwienie nawigacji po terenie.

Działanie polega na pomiarze czasu dotarcia sygnału radiowego z satelitów do odbiornika. Znając prędkość fali elektromagnetycznej oraz znając dokładny czas wysłania danego sygnału można obliczyć odległość odbiornika od satelitów.

Na wyposażeniu laboratorium znajduje się odbiornik LD-1W firmy NOKIA, dedykowany do użycia z telefonami komórkowymi – komunikuje się z urządzeniem poprzez bluetooth. Odbiornik przesyła dane zgodnie z protokołem NMEA. Pierwotnie był to protokół komunikacji między elektronicznymi urządzeniami morskimi, jednak szybko znalazł powszechne zastosowanie w urządzeniach GPS. W protokole tym dane są transmitowane w postaci „zdań" zapisanych w kodzie ASCII, jedno „zdanie" może mieć do 82 znaków. Znakiem zaczynającym dane jest $, następnie następuje identyfikator „zdania", pola danych oddzielone przecinkami oraz znaki CR i LF, kończące sekwencję danych.

Zestaw laboratoryjny zawiera adapter Bluetooth wykrywany przez komputer jako port szeregowy (w przypadku moich zajęć był to port COM3), dzięki temu można odbierać dane przesłane przez urządzenie LD-1W za pomocą standardowych funkcji operujących tym porcie.

3. Realizacja ćwiczenia

3.1 Odczytanie oraz analiza danych przesłanych przez GPS

Po włączeniu programu puTTY i nawiązaniu połączenia na porcie COM3 na ekranie ukazały się dane przesyłane przez odbiornik GPS. Z punktu widzenia wykonania ćwiczenia najbardziej interesowały mnie „zdania" o identyfikatorze `GPGGA`, gdyż zawierały one dane opisujące aktualną pozycję:

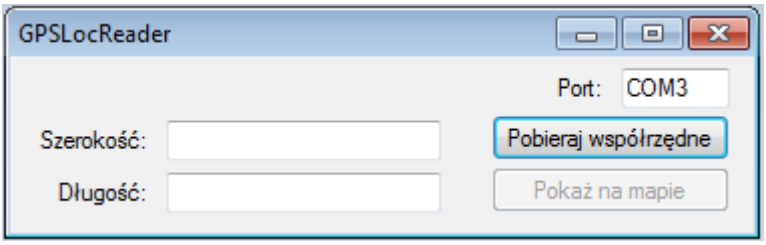
```
$GPGGA,133719.000,5106.5732,N,01703.6248,E,1,04,2.9,250.4,M,42.6,M,,0000*52
```

Odczytana pozycja to **51° 6,5732' N** i **17° 3,6248' E**.

3.2 Aplikacja współpracująca z odbiornikiem GPS

W ramach realizacji programistycznej części ćwiczenia, napisany został program w języku C#. Za komunikację z użytkownikiem odpowiedzialne jest okno dialogowe, umożliwiające:

- wprowadzenie nazwy portu, z którego mają zostać odczytane dane – np. COM3,
- rozpoczęcie / zakończenie odczytywania współrzędnych poprzez kliknięcie na przycisk *Pobieraj współrzędne* / *Zatrzymaj pobieranie*,
- wyświetlenie dodatkowego okna dialogowego z odczytaną pozycją zaznaczoną na Google Maps poprzez kliknięcie na przycisk *Pokaż na mapie*,
- obserwację odczytywanych współrzędnych w polach tekstowych.



Rysunek 1. Interfejs okna głównego programu GPSLocReader

Odpowiednie instrukcje sterujące zawarłem w metodach wywoływanych po kliknięciu na określony przycisk. Dodatkowo do projektu dodane są kontrolki *Timer* i *SerialPort* – ta pierwsza umożliwia wykonywanie funkcji co określoną ilość czasu – pozwoliło mi to na napisanie instrukcji odczytujących dane z portu szeregowego. Ta druga udostępnia obiektowy interfejs portu. Poniżej zamieszczam kod jednej z ważniejszych funkcji – przypisanej do przycisku *Pobieraj współrzędne* / *Zatrzymaj pobieranie*. Metoda ta jest odpowiedzialna za odpowiednio: otworzenie portu i włączenie zegara lub zamknięcie portu i wyłączenie zegara. Steruje też odpowiednią sekwencją wartości atrybutu *Enabled* przycisków i pól tekstowych.

Listing 1. Metoda wywoływana po kliknięciu na przycisk Pobieraj współrzędne / Zatrzymaj pobieranie

```
private void button1_Click(object sender, EventArgs e)
{
    //Jezeli zegar wlaczony lub port otrwarty
    if (zegar.Enabled == true || portSz.IsOpen)
    {
          //Zamkniecie portu i wylaczenie zegara
          portSz.Close();
          zegar.Enabled = false;
    }
    //Jezeli nie
    else
    {
          try
          {
                //Nazwa portu z pola tekstowego i jego otwarcie
                portSz.PortName = textBox1.Text;
                portSz.Open();
          }
          catch (Exception ex)
          {
                MessageBox.Show(ex.Message, textBox1.Text);
          }
    }

    //Sterowanie parametrami Text i Enabled kontrolek
    if (button1.Text == "Pobieraj współrzędne")
    {
```

```
            button1.Text = "Zatrzymaj pobieranie";
            textBox1.Enabled = false;
      }

      else
      {
            button1.Text = "Pobieraj współrzędne";
            textBox1.Enabled = true;
      }
}
```

Najważniejszą funkcją stworzonej aplikacji jest bez wątpienia metoda przypisana do zegara i wywoływana co określony odstęp czasu(okres). Ponieważ odbiornik GPS przesyła dane do Bluetooth mniej więcej raz na sekundę, okres dla zegara programu został ustawiony na 1500 ms. Sposób działania funkcji jest prosty. Na początku sprawdzany jest stan portu(czy jest otwarty), następnie do łańcucha tekstowego zapisywane są dane, które aktualnie znajdują się na porcie – metoda `ReadExisting()` obiektu `portSz`. Następnie łańcuch ten dzielony jest na tablicę łańcuchów, kryterium podziału to znak $, czyli oddzielane są kolejne „zdania" danych. Łańcuch zawierający każde „zdanie" jest z kolei dzielony przecinkiem, co pozwala wyłuskać odpowiednie fragmenty danych. Jeżeli pierwszy element tablicy z danymi „zdania" to *GPGGA* metoda przechodzi do bloku odczytu. Odczytywane są: szerokość i długość geograficzna oraz orientacja (N,S,W,E). Na potrzebę późniejszego użycia odczytane dane są konwertowane – szerokość i długość geograficzna wyrażone w stopniach i minutach na wyrażone w stopniach i ułamkach stopni. Po poprawnym odczytaniu i konwersji szerokość i długość geograficzna jest wyświetlana w polach tekstowych okna dialogowego. Kod metody wraz z komentarzami przedstawia poniższy listing.

Listing 2. Metoda wywoływana co okres zegara, odczytująca i konwertująca dane odczytane z portu szeregowego

```
private void zegar_Tick(object sender, EventArgs e)
{
      //Jezeli port jest otwarty
      if (portSz.IsOpen)
      {
            //Odczytanie tego, co jest na porcie
            string dane = portSz.ReadExisting();
            //Podzial na zdania
            string[] strTab = dane.Split('$');
            for (int i = 0; i < strTab.Length; i++)
            {
                  string tmp = strTab[i];
                  //Podzial na dane
                  string[] valTab = tmp.Split(',');
                  //Jezeli kod zdania to GPGGA - odczyt pozycji
                  if (valTab[0] == "GPGGA")
                  {
                        try
                        {
                              //Szerokosc geograficzna
                              valTab[2] = valTab[2].Replace(".", ",");
                              Console.Out.WriteLine(valTab[2]);
                              Double dSzer = Convert.ToDouble(valTab[2]);
                              dSzer = dSzer / 100;
                              string[] szer = dSzer.ToString().Split(',');

                              //Konwersja minut na ulamek stopnia i zapis, 1 st = 60 min
                              szerGeo = valTab[3].ToString() + (Convert.ToDouble(szer[0])
+ ((Convert.ToDouble("," + szer[1]) / .6))).ToString("");
                              szerGeo = szerGeo.Replace(",", ".");
```

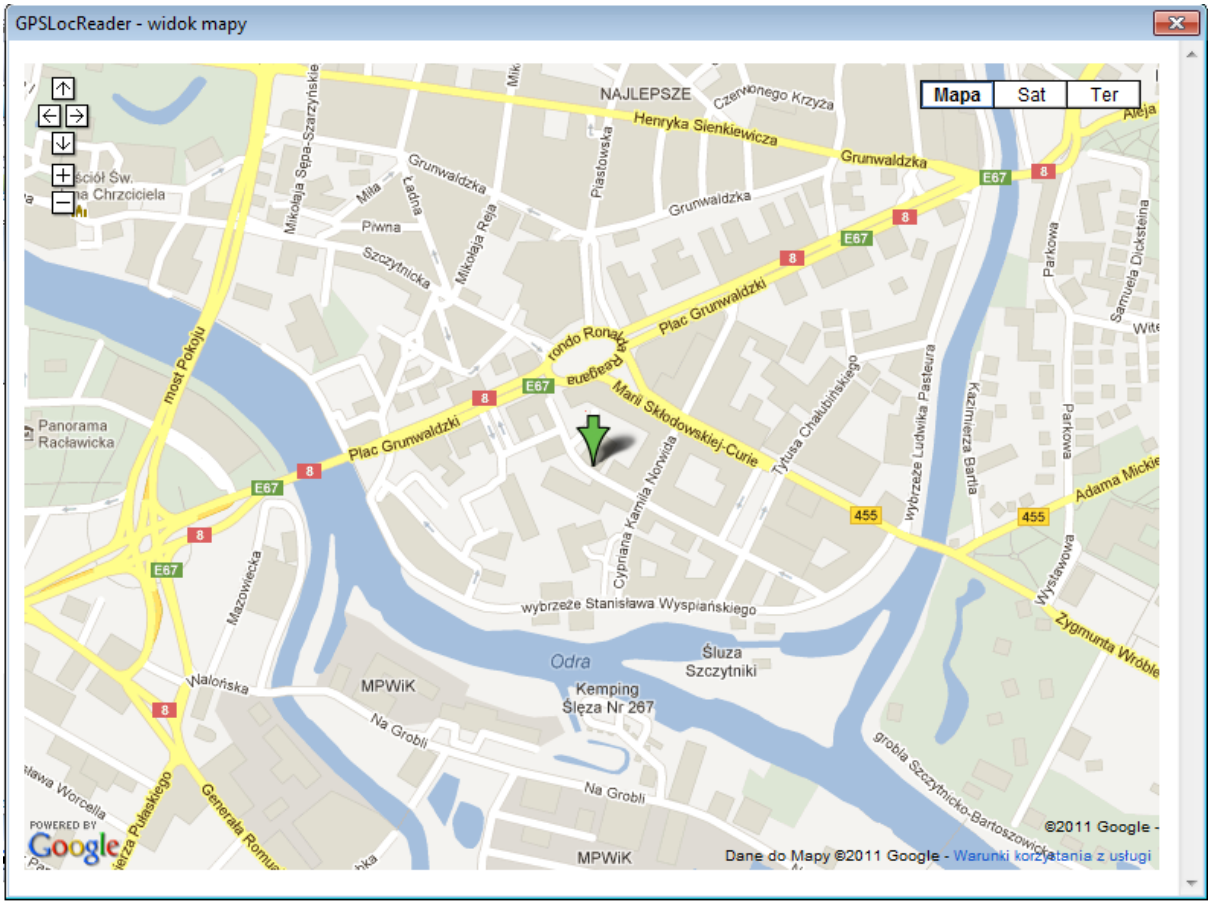
```
                        //Dlugosc geograficzna
                        valTab[4] = valTab[4].Replace(".", ",");
                        Double dDlug = Convert.ToDouble(valTab[4]);
                        dDlug = dDlug / 100;
                        string[] dlug = dDlug.ToString().Split(',');

                        //Konwersja minut na ulamek stopnia i zapis, 1 st = 60 min
                        dlugGeo = valTab[5].ToString() + (Convert.ToDouble(dlug[0])
+ ((Convert.ToDouble("," + dlug[1]) / .6))).ToString("");
                        dlugGeo = dlugGeo.Replace(",", ".");

                        //Wyswietlenie
                        txtSzer.Text = szerGeo;
                        txtDlug.Text = dlugGeo;

                        //Wlaczenie mozliwosci podgladu na mapie
                         button2.Enabled = true;
                    }
                    catch
                    {
                         //Komunikat o braku danych i blokada przycisku mapy
                        button2.Enabled = false;
                        txtSzer.Text = "Brak danych.";
                        txtDlug.Text = "Brak danych.";
                    }
                }
            }
        }
}
```

Dodatkową funkcjonalnością programu jest możliwość zaznaczenia odczytanej pozycji na mapie, wykorzystującej silnik Google Maps. W tym celu zostało stworzone dodatkowe okno dialogowe, a na nim umieszczony komponent *WebBrowser*. Potrafi on wyświetlić w oknie aplikacji stronę internetową lub zinterpretować kod HTML podany jako łańcuch tekstowy. Warto przy tym wspomnieć, że jego działanie oparte jest o silnik programu Internet Explorer. Aby nie wyświetlać bez sensu całej strony z mapą, zdecydowałem się na przekazywanie do komponentu przeglądarki kodu HTML pobranego z witryny maps.google.com, definiującego ramkę `<iframe>` zawierającą mapę. W odpowiednie miejsce kodu HTML wstawiłem zmienne odpowiedzialne za szerokość i długość geograficzną. Na potrzeby zaimplementowania opisanej funkcjonalności została stworzona funkcja pomocnicza `wyswietlHTML()`. Przyjmuje ona jako argument łańcuch tekstowy zawierający kod w języku HTML.

Listing 3. Metoda pomocnicza wyswietlHTML()

```
private void wyswietlHTML(string html)
{
      //Wyswietlenie strony about:blank
      przegladarka.Navigate("about:blank");
      //Jezeli dokument nie jest pusty
      if (przegladarka.Document != null)
      {
            //Zapisujemy "pusta" wartosc
            przegladarka.Document.Write(string.Empty);
      }
      //Zapisujemy HTML
      przegladarka.DocumentText = html;
}
```

Aby pozycja mogła zostać wyświetlona na mapie, przy tworzeniu nowego okna należy przekazać aktualne koordynaty jako parametry konstruktora. Konstruktor okna z mapą wykonuje jedynie funkcję `wyswietlHTML()`. Aby wyświetlić nowo stworzone okno potrzebne jest jeszcze wywołanie metody `Show()` obiektu formatki.



Rysunek 2. Okno 'widok mapy" aplikacji GPSLocReader; wyświetlona mapa z zaznaczoną aktualną pozycją GPS

4. Wnioski

Zajęcia laboratoryjne, na których wykonałem ćwiczenie związane z GPS pozwolił mi na zapoznanie się z możliwościami i sposobami obsługi urządzeń tego typu na komputerze. Zapoznałem się także z protokołem NMEA, powszechnie dziś stosowanym w komunikacji urządzeń elektronicznych.

Napisanie aplikacji współpracującej z odbiornikiem GPS LD-1W pozwoliło mi na poznanie podstaw języka C# oraz obsługi portu szeregowego z jego poziomu. Jak się okazało, nie jest to trudne, choć rozwiązanie z zegarem odczytującym zawartość linii danych portu co ustalony okres nie jest najlepsze – czasem nie udaje się odczytać odpowiednich danych. Zaletą takiego rozwiązania jest natomiast prostota implementacji. Rozwiązaniem w pełni poprawnym byłoby użycie funkcji, która sprawdza, czy na porcie są dane do odczytania, a dopiero później ich odczyt.

W programie duży nacisk trzeba było położyć na konwersję danych – odbiornik GPS podawał dane w stopniach i minutach, zaś API map Google wymaga, aby były one podawane w stopniach i ułamkach stopni. Samo wyświetlenie pozycji na mapie było banalnie proste. Przydatny okazał się komponent *WebBrowser* bazujący na silniku Internet Explorera. Na wyświetlanych mapach nie działa widget zmieniania stopnia przybliżenia i przesuwania mapy – to błąd charakterystyczny dla tej przeglądarki.