Politechnika Wrocławska
Wydział Elektroniki
Urządzenia Peryferyjne

GPS

Termin zajęć:
Wtorek TN 7:30

Autorzy:
Jakub Chmiel 235028
Tomasz Cieślar 235652

Prowadzący:
dr inż. Tomasz Walkowiak

5 listopada 2018

Spis treści

1 Cel ćwiczenia

- Zapoznać się z zestawem GPS oraz podłączyć via Bluetooth
- Odczytać uzyskane komendy oraz podzielić je wg typów wiadomości
- Sprawdzić ważność uzyskanych danych i przedyskutować wynik
- Obsłużyć transmisję szeregową oraz przedstawić uzyskane dane w czytelny sposób
- Zlokalizować na mapie świata (np. z Google Map) punkty, w których znajdowało się urządzenie, na podstawie samodzielnie uzyskanych danych lub od prowadzącego (plik tekstowy, format NMEA)

2 Wstęp

Global Positioning System – system nawigacji satelitarnej, stworzony przez Departament Obrony Stanów Zjednoczonych, 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.

NMEA – opublikowany przez National Marine Electronics Association protokół komunikacji między morskimi urządzeniami elektronicznymi. Ma on powszechne zastosowanie w elektronice nawigacji morskiej oraz urządzeniach GPS.

Dane są transmitowane w postaci „zdań" zapisanych kodem ASCII. Pojedyncza sekwencja zawiera do 82 znaków. Znakiem zaczynającym dane w protokole jest znak „$", dalej następuje identyfikator zdania i pola danych oddzielone przecinkami, a na końcu znajdują się symbole <CR><LF> (carriage return, line feed).

3 Założenia projektowe

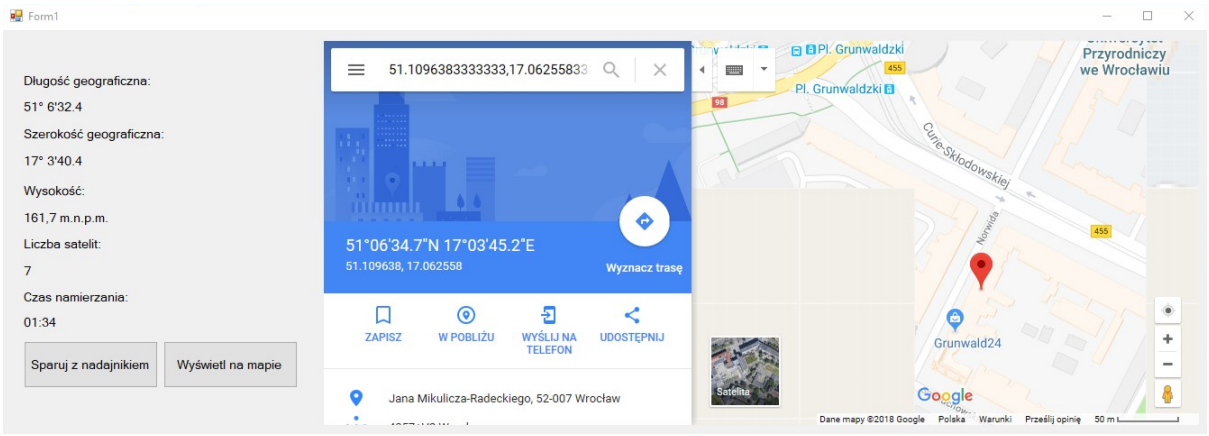
- Program był pisany w języku C#.
- Na komputerze, na którym uruchamiany był program zainstalowano system operacyjny Windows 10 w wersji 64-bitowej.

4 Wykorzystane narzędzia

- 32feet.NET - NuGet umożliwiający korzystanie z technologii Bluetooth, IrDA i OBEX w platformie .NET.
- NMEAParser GhostWare - NuGet do parsowania wiadomości NMEA w języku C#
- Windows Forms - API do implementacji interfejsu graficznego dla platformy .NET.

5 Implementacja programu

5.1 Interfejs użytkownika



Rysunek 1. Interfejs aplikacji

5.2 Kod programu

```
using [...]
namespace WindowsFormsApp2
{
    public partial class Form1 : Form
    {
        private string _latitudeFinal;
        private string _longtitudeFinal;
        private string _satellitesNumberFinal;
        private string _altitudeFinal;
        private string _longMap;
        private string _latMap;
        private Stopwatch _stopwatch;
        private Timer _timer;
        static readonly BluetoothClient BluetoothClient = new
           BluetoothClient();

        public Form1()
        {
            InitializeComponent();
            latValue.Text = "";
            longValue.Text = "";
            altValue.Text = "";
            satNumValue.Text = "";
            timerLabel.Text = "";
        }

        public void Pair()
        {
            //Wyszukanie wszystkich urzadzen BT
            var allBtDevices =
               BluetoothClient.DiscoverDevices();
```

```
        //Zabranie urzadzenia o pozadanej nazwie (nadajnik
           Pentagram)
        var wantedDevice = allBtDevices.Where(x =>
           x.DeviceName.Equals("PENTA-GPS")).
           FirstOrDefault();
        if (wantedDevice != null)
        {
            //Parowanie urzadzenia
            wantedDevice.Update();
            wantedDevice.Refresh();
            if (BluetoothSecurity.PairRequest(wantedDevice.
               DeviceAddress, "0000"))
            {
                MessageBox.Show("Urzadzenie sparowane
                   pomyslnie");
                //Asynchroniczne polaczenie z urzadzeniem.
                   Po pomyslnym wykonaniu polaczenia
                   przechodzimy do metody Connect
                BluetoothClient.BeginConnect(wantedDevice.
                   DeviceAddress,
                   BluetoothService.SerialPort, new
                   AsyncCallback(Connect), wantedDevice);
                _timer = new Timer { Interval = (1000) };
                _timer.Elapsed += new
                   ElapsedEventHandler(TimerTick);
                _stopwatch = new Stopwatch();
                _timer.Enabled = true;
                _timer.Start();
                _stopwatch.Start();
            }
            else
            {
                MessageBox.Show("Nie udalo sie sparowac
                   urzadzenia.");
            }
        }
    }

    private void Connect(IAsyncResult result)
    {


        if (result.IsCompleted)
        {
            //Pobieramy strumien danych z urzadzenia
            var nsFromDevice = BluetoothClient.GetStream();
            while (true)
            {
                if (nsFromDevice.CanRead)
```

```
                {
                    //Bufor dla odebranej wiadomosci
                    var bufferMessage = new byte[256];
                    var completeMessage = new
                       StringBuilder();

                    do
                    {
                        var numberOfBytesRead =
                           nsFromDevice.Read(bufferMessage,
                           0, bufferMessage.Length);

                        completeMessage.AppendFormat("{0}",
                           Encoding.ASCII.GetString
                           (bufferMessage, 0,
                           numberOfBytesRead));
                    }
                    while (nsFromDevice.DataAvailable);
                    try
                    {
                        //Parsowanie wiadomosci w formacie
                           NMEA
                        var nmeaParser = new NmeaParser();
                        var parsedMessage =
                           (GpggaMessage)nmeaParser.Parse
                           (completeMessage.ToString());
                        Console.WriteLine("Odebrano
                           nastepujaca wiadomosc: " +
                              completeMessage.ToString());
                        this.ShowParsedMessage
                           (parsedMessage);
                    }
                    catch (Exception exc)
                    {
                        Console.WriteLine("Wystapil blad,
                           ponowna proba pobrania danych");
                        Console.WriteLine();
                    }
                }
                else
                {
                    Console.WriteLine("Nie mozna odczytac
                       danych z tego strumienia.");
                }
            }
        }
    }

    public void ShowParsedMessage(GpggaMessage
       parsedMessage)
```

```
{
    //Szerokosc geograficzna

    //Poszczegolne skladowe wspolrzednych geograficznych
    var minutes = (parsedMessage.Latitude -
       Math.Floor(parsedMessage.Latitude)) * 60.0;
    var seconds = (minutes - Math.Floor(minutes)) *
       60.0;
    var tenths = (seconds - Math.Floor(seconds)) * 10.0;

    //Formatowanie szerokosci na potrzeby wyswietlenie
       polozenia na mapie
    _latMap = parsedMessage.Latitude.ToString();
    _latMap = _latMap.Replace(',', '.');

    //Usuniecie ulamkow
    minutes = Math.Floor(minutes);
    seconds = Math.Floor(seconds);
    tenths = Math.Floor(tenths);
    _latitudeFinal = Math.Floor(parsedMessage.Latitude)
       + " " + minutes + "'" + seconds + "." + tenths;

    //Analogicznie dla dlugosci geograficznej

    minutes = (parsedMessage.Longitude -
       Math.Floor(parsedMessage.Longitude)) * 60.0;
    seconds = (minutes - Math.Floor(minutes)) * 60.0;
    tenths = (seconds - Math.Floor(seconds)) * 10.0;

    _longMap = parsedMessage.Longitude.ToString();
    _longMap = _longMap.Replace(',', '.');

    minutes = Math.Floor(minutes);
    seconds = Math.Floor(seconds);
    tenths = Math.Floor(tenths);

    _longtitudeFinal =
       Math.Floor(parsedMessage.Longitude) + " " +
       minutes + "'" + seconds + "." + tenths;

    //Wysokosc n.p.m.
    _altitudeFinal = parsedMessage.Altitude + "
       m.n.p.m.";

    //Liczba satelit
    _satellitesNumberFinal =
       parsedMessage.NumberOfSatellites.ToString();
}

private void btShowMap_Click(object sender, EventArgs e)
```

```
{
    //Przejscie do Google Maps na znalezionym punkcie
    wbMap.Navigate("https://www.google.com/maps/?q=" +
       _latMap + "," + _longMap);
}

private void btPair_Click(object sender, EventArgs e)
{
    //BackgroundWorker odpowiada za aktualizacje
       informacji na ekranie

    var backgroundWorker = new BackgroundWorker();


    backgroundWorker.WorkerReportsProgress = true;

    backgroundWorker.DoWork += new DoWorkEventHandler(
        delegate (object o, DoWorkEventArgs args)
        {
            var secondaryBw = o as BackgroundWorker;

            for (int i = 1; i <= 1000; i++)
            {
                secondaryBw.ReportProgress(i * 10);
                Thread.Sleep(1000);
            }
        });

    backgroundWorker.ProgressChanged += new
       ProgressChangedEventHandler(
        delegate (object o, ProgressChangedEventArgs
           args)
        {
            latValue.Text = _latitudeFinal;
            longValue.Text = _longtitudeFinal;
            altValue.Text = _altitudeFinal;
            satNumValue.Text = _satellitesNumberFinal;
        });

    backgroundWorker.RunWorkerCompleted += new
       RunWorkerCompletedEventHandler(
        delegate (object o, RunWorkerCompletedEventArgs
           args)
        {
            satNumValue.Text = "Ukonczono!";
        });

    backgroundWorker.RunWorkerAsync();
    Pair();
}
```

```
        //Odmierzanie czasu od rozpoczecia namierzania
        void TimerTick(object sender, ElapsedEventArgs e)
        {
            timerLabel.BeginInvoke(new MethodInvoker(() =>
               timerLabel.Text =
               _stopwatch.Elapsed.ToString("mm\\:ss")));
        }
    }
}
```

6 Wnioski

W ćwiczeniu wykorzystaliśmy wiedzę nabytą w trakcie zajęć z Bluetooth. Bardzo przydatny okazał się NMEAParser GhostWare, dzięki niemu mogliśmy w łatwy sposób przekształcić zakodowaną wiadomość wysyłana przez modem GPS na przyjazną w odbiorze, wyświetlaną w programie.

7 Bibliografia

1. Dokumentacja 32Feet.NET:
   http : *//inthehand.github.io/html/R_Project_InTheHand.htm*
2. Parser wiadomości NMEA:
   https : *//github.com/kevingoos/NMEAParser*
3. GPS:
   https : *//pl.wikipedia.org/wiki/Global_Positioning_System*
4. NMEA:
   https : *//pl.wikipedia.org/wiki/NMEA*_0183