



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Instrukcja współfinansowana przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
w projekcie

*„Innowacyjna dydaktyka bez ograniczeń
– zintegrowany rozwój Politechniki Łódzkiej – zarządzanie Uczelnią,
nowoczesna oferta edukacyjna i wzmacniania zdolności
do zatrudniania osób niepełnosprawnych”*

Instrukcja jest dystrybuowana bezpłatnie.

Instrukcja do laboratorium

Piotr Korbel

Projektowanie i programowanie systemów bezprzewodowych

Korzystanie z urządzeń peryferyjnych

Zadanie nr 14 – Studia podyplomowe „Bezprzewodowe systemy nadzoru i monitorowania”



Politechnika Łódzka
Instytut Elektroniki

90-924 Łódź, ul. Żeromskiego 116,
tel. 042 631 28 83
www.kapitalludzki.p.lodz.pl

Korzystanie z urządzeń peryferyjnych

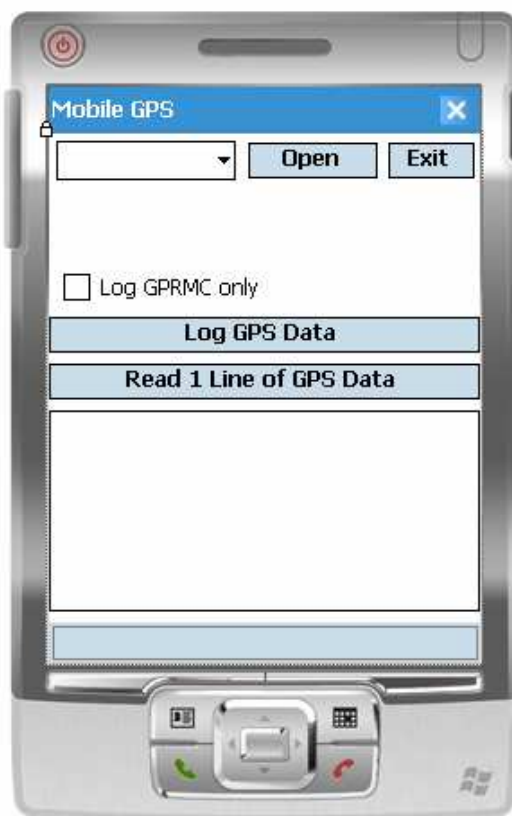
1. Wprowadzenie

Celem zajęć laboratoryjnych jest zapoznanie słuchaczy ze sposobami obsługi urządzeń peryferyjnych urządzeń mobilnych (wbudowanych lub zewnętrznych podłączanych za pomocą wybranych interfejsów komunikacyjnych). W ramach realizacji ćwiczeń przeanalizowane oraz uruchomione zostaną programy pobierające dane z odbiornika satelitarnego systemu pozycjonowania GPS oraz obsługujące wbudowany aparat fotograficzny.

2. Odczyt danych z odbiornika systemu GPS

Wymagania – do realizacji ćwiczenia wymagane jest urządzenie przenośne z wbudowanym odbiornikiem systemu GPS (np. ASUS MyPAL 696 lub HP iPAQ 614c) lub zewnętrzny moduł odbiornika systemu GPS wyposażony w interfejs (np. Bluetooth) umożliwiający połączenie do urządzenia przenośnego.

Szablon oprogramowania znajduje się w folderze **MobileGPS**. Po uruchomieniu pakietu Visual Studio należy otworzyć plik **MobileGPS.sln**, a następnie w zakładce Solution Explorer wskazać plik **Form1.cs** i wybrać opcje *View Code* oraz *View Designer* w celu wyświetlenia kodu źródłowego (Dodatek A) oraz projektu formularza głównego aplikacji (Rys. 1).



Rys. 1. Widok formularza głównego programu odczytującego dane z odbiornika GPS



Za pomocą dostępnych kabli USB podłączyć urządzenie przenośne do stacji deweloperskiej oraz skompilować i uruchomić program (przycisk *F5*) wskazując jako docelową platformę urządzenie z systemem Windows Mobile 6 Professional (*Windows Mobile 6 Professional Device*).

Po uruchomieniu programu wybrać z listy nazwę portu, pod którym widoczny jest odbiornik GPS oraz nawiązać komunikację z portem poprzez naciśnięcie przycisku **Open**. Uruchomić odczyt danych z odbiornika poprzez naciśnięcie przycisku **Read 1 Line of GPS Data** (odczyt pojedynczej linii komunikatów GPS) lub przycisku **Log GPS Data** (odczyt ciągły połączony z zapisem odczytanych informacji do pliku dziennika *test.txt*). Po odczycie komunikatu zawierającego informację o pozycji odbiornika, współrzędne położenia geograficznego wyświetlone zostaną w pasku statusu zlokalizowanym w dolnej części formularza głównego aplikacji.

Po wybraniu opcji **Log GPRMC Only**, program przetwarza jedynie komunikaty NMEA rozpoczynające się od sekwencji \$GPRMC (tylko te komunikaty zostają wyświetlone i zapisane do pliku dziennika).

Po zakończeniu pracy wyłączyć tryb zapisu danych (**Stop Logging GPS Data**) oraz zamknąć komunikację z portem (**Close**).

W ramach ćwiczeń można wprowadzić modyfikacje warunków filtrowania komunikatów odczytywanych z odbiornika GPS (przetwarzać różne komunikaty, dokonać ekstrakcji dodatkowych, np. o prędkości poruszania się odbiornika, liczbie widocznych satelitów, itd.).

3. Obsługa wbudowanego aparatu fotograficznego

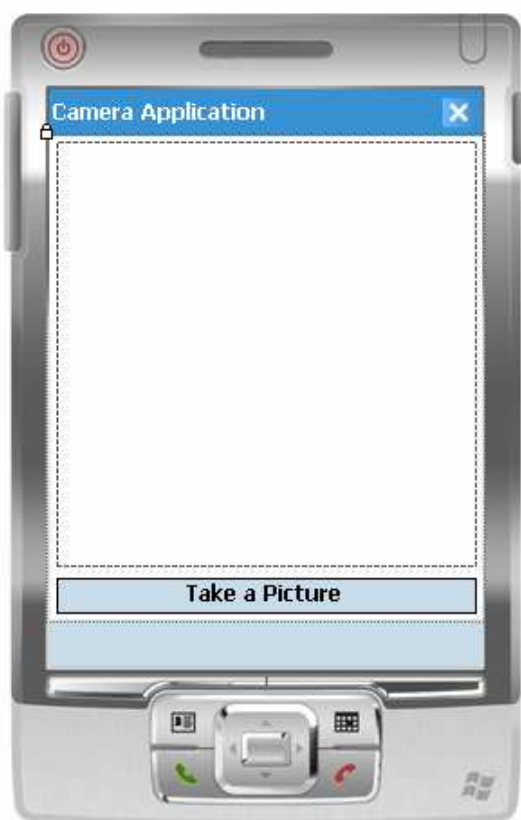
Wymagania – do realizacji ćwiczenia wymagane jest urządzenie przenośne z wbudowanym aparatem fotograficznym (np. HP iPAQ 614c).

Program demonstruje możliwość sterowania pracą wbudowanego w urządzenie aparatu za pomocą funkcji oferowanych przez biblioteki Microsoft.WindowsMobile.dll oraz Microsoft.WindowsMobile.Forms.dll.

Szablon oprogramowania znajduje się w folderze **Camera_App**. Po uruchomieniu pakietu Visual Studio należy otworzyć plik **Camera_App.sln**, a następnie w zakładce Solution Explorer wskazać plik **Form1.cs** i wybrać opcje *View Code* oraz *View Designer* w celu wyświetlenia kodu źródłowego (Dodatek B) oraz projektu formularza głównego aplikacji (Rys. 2).

Za pomocą dostępnych kabli USB podłączyć urządzenie przenośne do stacji deweloperskiej oraz skompilować i uruchomić program (przycisk *F5*) wskazując jako docelową platformę urządzenie z systemem Windows Mobile 6 Professional (*Windows Mobile 6 Professional Device*).

Po naciśnięciu przycisku **Take Picture** wywołane zostaje okno obsługi wbudowanego aparatu fotograficznego z programowo ustawionymi parametrami. Należy przetestować działanie programu a następnie przeprowadzić próby modyfikacji ustawień aparatu poprzez odpowiednie zmiany w kodzie źródłowym programu.



Rys. 2. Widok formularza głównego programu odczytującego dane z odbiornika GPS

Literatura – źródła internetowe

- [1] <http://msdn.microsoft.com/>
- [2] <http://www.devx.com/>
- [3] <http://www.codeplex.com/>



Dodatek A

Kod programu odczytującego dane z odbiornika GPS

Form1.cs

```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.IO.Ports;
using System.Text;
using System.Windows.Forms;

namespace MobileGPS
{
    public partial class FormGPS : Form
    {
        //Łańcuch przechowujący odczytaną linię tekstu z
        //odbiornika GPS
        string gpsLine;
        private StreamWriter sw = null;

        //Inicjalizacja głównego formularza aplikacji
        public FormGPS()
        {
            InitializeComponent();
            this.MinimizeBox = false;

            textBox1.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
        }

        //Czynności wykonywane podczas ładowania formularza
        //głównego
        private void FormGPS_Load(object sender, EventArgs e)
        {
            //Zapełnienie kontrolki combobox nazwami
            //dostępnych portów (COM1, COM2, ...)

            comboBoxPort.Items.Clear();
            string[] ports = SerialPort.GetPortNames();
            foreach (string prt in ports)
```





```
        {
            comboBoxPort.Items.Add(prt);
        }
    }

    //Połączenie/rozłączenie ze wskazanym portem
private void buttonPort_Click(object sender, EventArgs
e)
{
    if (buttonPort.Text == "Open")
    {
        if (serialPort1.IsOpen)
        {
            serialPort1.Close();
        }
        int id = comboBoxPort.SelectedIndex;
        if (id > 0)
        {
            serialPort1.PortName =
comboBoxPort.Items[id].ToString();

            if (!serialPort1.IsOpen)
            {
                serialPort1.Open();

                //Ustawienie szybkości transmisji
danych portu
                string br = "4800";
                serialPort1.BaudRate =
Convert.ToInt32(br);
                if (serialPort1.IsOpen)
                {
                    buttonPort.Text = "Close";
                    //Wyświetlenie odczytanej
szybkości transmisji (można usunąć)
                    statusBar1.Text = "Port OK, Baud
Rate: " + serialPort1.BaudRate.ToString();
                }
            }
        }
    }
    else
    {
        if (serialPort1.IsOpen)
        {
            serialPort1.Close();
        }
        if (!serialPort1.IsOpen)
```



```
        {
            statusBar1.Text = "Port Successfully
Closed";
            buttonPort.Text = "Open";
        }
    }

    //Czynności wykonywane podczas zamykania aplikacji -
zamknięcie portu
    private void FormGPS_Closing(object sender,
CancelEventArgs e)
    {
        if (serialPort1.IsOpen)
        {
            serialPort1.Close();
        }
    }

    //Odczyt danych z GPS (pojedynczej linii)
    private void buttonReadGps_Click(object sender,
EventArgs e)
    {
        if (serialPort1.IsOpen)
        {
            gpsLine = serialPort1.ReadLine() + "\n";
            textBox1.Text += gpsLine;
            extractCoordinates(gpsLine);
        }
    }

    private void extractCoordinates(string gpsMessage)
    {
        //Odczytanie współrzędnych z linii komunikatu GPS
($GPGGA lub $GPRMC) i wyświetlenie współrzędnych w pasku
statusu

        if (gpsMessage.StartsWith("$GPGGA"))
        {
            string[] result;
            result = gpsMessage.Split(',');

            try
            {
                if (result[2] != "")
                {
```

```
        statusBar1.Text = "Lat: " + result[3]
+ result[2] + "; Lon: " + result[5] + result[4];
    }
}
catch (Exception Ex)
{
}
}
else if (gpsMessage.StartsWith("$GPRMC"))
{
    string[] result;
    result = gpsMessage.Split(',');
    try
    {
        if (result[3] != "")
        {
            statusBar1.Text = "Lat: " + result[4]
+ result[3] + "; Lon: " + result[6] + result[5];
        }
    }
    catch (Exception Ex)
    {
    }
}

e)
private void buttonExit_Click(object sender, EventArgs
{
    timerLogGps.Enabled = false;
    Close();
}

private void buttonLogGps_Click(object sender,
EventArgs e)
{
    if (buttonLogGps.Text == "Log GPS Data")
    {
        sw = new StreamWriter("test.txt", true);
        timerLogGps.Enabled = true;
        buttonLogGps.Text = "Stop Logging GPS Data";
    }
    else
    {
        timerLogGps.Enabled = false;
        sw.Close();
        buttonLogGps.Text = "Log GPS Data";
    }
}
```




```
    }  
  
e) private void timerLogGps_Tick(object sender, EventArgs  
    {  
        if (textBox1.TextLength > 300)  
        {  
            textBox1.Text= " ";  
        }  
        if (serialPort1.IsOpen)  
        {  
            gpsLine = serialPort1.ReadLine() + "\n";  
            if (checkBoxGprmc.Checked)  
            {  
                if (gpsLine.StartsWith("$GPRMC"))  
                {  
                    textBox1.Text += gpsLine;  
                    sw.Write(gpsLine);  
                    extractCoordinates(gpsLine);  
                }  
            }  
            else  
            {  
                textBox1.Text += gpsLine;  
                sw.Write(gpsLine);  
                extractCoordinates(gpsLine);  
            }  
        }  
    }  
}
```



Dodatek B

Kod programu obsługującego wbudowany aparat fotograficzny urządzenia przenośnego

Form1.cs

```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.WindowsMobile.Forms;

namespace Camera_App
{
    public partial class Form1 : Form
    {
        string fName = string.Empty;
        string fNameWithPath = string.Empty;
        private Bitmap Img;

        public Form1()
        {
            InitializeComponent();
            this.MinimizeBox = false;
        }

        private void buttonStillPic_Click(object sender,
EventArgs e)
        {
            CameraCaptureDialog ccd = new
CameraCaptureDialog()
            {
                Resolution = new Size(100, 200),
                Mode = CameraCaptureMode.Still,
                InitialDirectory = @"My Documents\",
                DefaultFileName = @"test.jpg"
            };

            ccd.ShowDialog();
            if (ccd.FileName != string.Empty)
            {
```



```
fName =  
ccd.FileName.Replace(ccd.InitialDirectory, " ");  
fNameWithPath = ccd.FileName;  
  
if (Img != null)  
{  
    Img.Dispose();  
}  
pictureBoxStill.SizeMode =  
PictureBoxSizeMode.StretchImage;  
Img = new Bitmap(ccd.FileName);  
pictureBoxStill.Image = (Image)Img;  
}  
}  
}
```

