**Data wykonania ćwiczenia: 18.11.2014 r.**

 **Data oddania sprawozdania: 28.11.2014 r.**

Prowadzący: dr inż. Jan Nikodem Wydział Elektroniki, **Informatyka**

Grupa:

Termin: Wt TN 13:15

**Sprawozdanie z laboratorium**

**Urządzenia Peryferyjne**

*Ćwiczenie 7*

*Obsługa modułu GPS*

**Cel ćwiczenia**

Naszym zadaniem było napisanie programu komunikującego się za pomocą połączenia Bluetooth z modułem GPS firmy Nokia model LD-1W. Następnie program miał odbierać przesyłane przez Bluetooth komunikaty NMEA i odczytywać z nich czas, położenie, wysokość oraz liczbę satelitów, na podstawie której urządzenie wyznaczyło współrzędne.

**GPS, czyli Global Positioning System**

Global Positioning System (GPS) to 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. 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. Sygnał GPS zawiera w sobie informację o układzie satelitów na niebie oraz informację o ich teoretycznej drodze oraz odchyleń od niej. Odbiornik GPS w pierwszej fazie aktualizuje te informacje w swojej pamięci oraz wykorzystuje w dalszej części do ustalenia swojej odległości od poszczególnych satelitów, dla których odbiornik jest w zasięgu.

**Moduł GPS i komunikacja bluetooth**

Moduł GPS do komunikacji z hostem wykorzystuje protokół NMEA (National Marine Electronics Association). Protokół ten używany jest również w urządzeniach służących do komunikacji morskiej. Transmisja danych odbywa się w postaci wysyłania „zdań” zapisanych kodem ASCII. Pojedyncza sekwencja zawiera do 82 znaków. Znakiem zaczynającym dane w protokole jest „$”, dalej następuje identyfikator zdania i pola danych oddzielone przecinkami, a na końcu znajdują się symbole <CR><LF> (carriage return, line feed). Wewnątrz komunikatu poszczególne pojedyncze pola danych pooddzielane są znakami przecinka. W przypadku braku jakiejś danej, przecinki zostają (pomiędzy nimi jest pusty łańcuch). Po ostatnim polu danych następuje znak \* poprzedzający sumę kontrolną komunikatu (która jest ostatnią daną).

W laboratorium do dyspozycji mieliśmy moduł GPS marki Nokia model LD-W1 oraz moduł bluetooth podłączany do portu USB w komputerze. Aby nawiązać połączenie z urządzeniem, na komputerze musieliśmy wyszukać dostępne urządzenia bluetooth, następnie ze znalezionych urządzeń wybrać moduł GPS i sparować go z komputerem, używając kodu podanego w instrukcji, czyli 0000. Po sparowaniu urządzeń musieliśmy zidentyfikować port COM, na którym pracuje moduł GPS.

Urządzenia bluetooth zapewniają pewne usługi. Jedną z nich jest wirtualny port szeregowy, który znalazł zastosowanie właśnie w tym przypadku. Moduł bluetooth w komputerze jest w stanie dla takiego urządzenia utworzyć w systemie emulowany port COM, który przez system jest widziany jak normalny port szeregowy. Istnieje zatem możliwość odczytywania wysyłanych danych na porcie szeregowym, co łatwo udało się uzyskać, stosując terminal PuTTY. W ustawieniach połączenia wystarczyło podać numer portu szeregowego (COM38) oraz prędkość danych (4800b/s – zgodnie ze specyfikacją NMEA). Samo urządzenie wyposażonej jest w 3 diody, które sygnalizują stan pracy urządzenia – pierwsza sygnalizuje włączenie urządzenia, druga – nawiązanie transmisji bluetooth, natomiast trzecia - możliwość określenia pozycji (position fix) – potocznie mówi się, że urządzenie „złapało fixa”. W momencie próby nawiązania połączenia z poziomu programu PuTTY, najpierw czekaliśmy, aż zostanie ustanowiona transmisja przez port bluetooth – jej uzyskanie był sygnalizowane zapaleniem niebieskiej diody. Następnie urządzenie przesyłało puste komunikaty dotyczące pozycji (brak fixa), po kilku minutach przesyłane już były komunikaty zawierające pozycję, czas, liczbę satelit, itd. – trzecia dioda zapaliła się na zielono. Po zamknięciu konsoli port szeregowy został zwolniony.

W tym momencie mogliśmy zauważyć, jakie rodzaje komunikatów wysyła do komputera moduł GPS:

* GPGSA - dane dotyczące śledzonych satelitów oraz osłabienia precyzji pomiaru pozycji.
* GPGSV - dane dotyczące aktualnie śledzonych satelitów.
* GPRMC - paczka danych o pozycji, prędkości i czasie, tak zwane rekomendowane minimum (ang. The Recommended Minimum).
* **GPGGA - dane dotyczące aktualnego położenia – ten komunikat był kluczowy do dalszej realizacji ćwiczenia. Poniżej przedstawiona jest składnia tego komunikatu.**

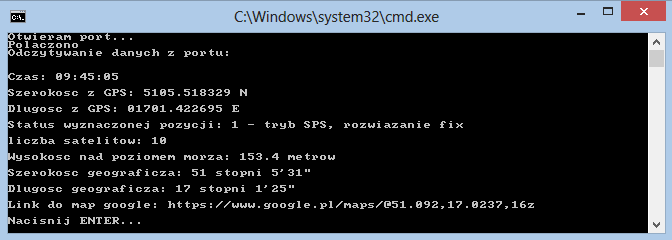
*$GPGGA,Time,Lat,LatDir,Lon,LonDir,FixQ,Sat,HDOP,Alt,AltUnit,GeoidH, GeoidHUnit,DGPSTime,DGPSid\*Checksum*

* Time - obecny czas UTC, zapisany w formacie HHMMSS, gdzie: H – godziny, M – minuty, S - sekundy
* Lat - szerokość geograficzna zapisana w formacie DDMM.MMMM, gdzie D - stopnie, M - minuty. Wartość po kropce to sekundy, określone jako ułamek minuty.
* LatDir - kierunek szerokości geograficznej, przyjmuje wartości N – północna oraz S - południowa
* Lon - długość geograficzna, format analogiczny jak dla szerokości
* LonDir - kierunek długości geograficznej, przyjmuje wartości E – wschodnia oraz W - zachodnia
* FixQ - jakość sygnału (position fix), przyjmuje wartości: 0 - brak „fixa”, 1 - fix GPS , 2 - fix DGPS.
* Sat - liczba śledzonych satelitów
* HDOP - parametr opisujący poziom dokładności. Im mniejsza wartość, tym dokładność większa, wartości powyżej 20 mają bardzo słabą jakość (błąd rzędu 100m).
* Alt - wysokość nad poziomem morza
* AltUnit - jednostka wysokości, wartość M oznacza metry
* GeoidH - wysokość geoidy nad elipsoidą Ziemi w miejscu, w którym znajduje się urządzenie GPS,
* GeoidHUnit - jednostka wysokości, wartość M oznacza metry
* DGPSTime - czas od ostatniego komunikatu DGPS
* DGPSid - identyfikator stacji bazowej DGPS2.

**Program do obsługi modułu GPS**

Analiza konstrukcji powyższego komunikatu umożliwiła nam rozkład otrzymanych danych na „czynniki pierwsze”, a następnie na dalsze ich opracowanie. Napisany przez nas program potrafi nawiązać połączenie bluetooth z modułem GPS, przechwycić dane z jego wyjścia i je wyświetlić, wyświetlić szerokość i długość geograficzną oraz wysokość, a także liczbę śledzonych satelitów. Ponadto program generuje link do serwisu Google Maps, który wystarczy skopiować i wkleić do przeglądarki, a pozycję wskazywaną przez urządzenie będziemy mogli zobaczyć na mapie. W przypadku zestawiania połączenia bluetooth kluczowym dla działania programu okazywało się otwieranie i zamykanie portu COM – bez wywołania zamknięcia, port pozostawał zablokowany, nawet po zamknięciu programu i nie dało się korzystać z niego powtórnie.

Do zestawienia komunikacji za pomocą portu szeregowego wykorzystaliśmy klasę autorstwa Hansa de Ruitera, która w prosty i przejrzysty sposób umożliwiła nam to zadanie (Serial.cpp).



Wynik działania programu

Poniżej prezentujemy napisany przez nas kod:

#include <iostream>

#include <windows.h>

#include <string>

#include <stdlib.h>

#include <sstream>

#include "Serial.h"

#include "stdafx.h"

using namespace std;

#define RX\_BUFFSIZE 512

int i;

void kopiuj(char\* tabl, char\* zmienn);

string doubl2str(double i);

int \_tmain(int argc, \_TCHAR\* argv[])

{

try

{

cout << "Otwieram port..." << endl;

Serial serial(9600);

cout << "Polaczono" << endl;

char tablica[RX\_BUFFSIZE];

cout << "Odczytywanie danych z portu: \n";

for(int a = 0; a < 5; a++)

{

int charsRead = serial.read(tablica, RX\_BUFFSIZE);

Sleep(3000);

}

cout << tablica;

char czas[20], szerokosc[20], dlugosc[20], liczbaSatelitow[4];

char fi[2], lambda[2], status[2], wspolczynnik[5], wysokosc[8];

int j = 0;

for (; i <100; i++)

if (tablica[i] == 'G' && tablica[i + 1] == 'G' && tablica[i + 2] == 'A') //naglowek protokolu

{

i += 3;

kopiuj(tablica, czas);//kopiowanie kolejnych wartości do tablic

kopiuj(tablica, szerokosc);

kopiuj(tablica, fi);

kopiuj(tablica, dlugosc);

kopiuj(tablica, lambda);

kopiuj(tablica, status);

kopiuj(tablica, liczbaSatelitow);

kopiuj(tablica, wspolczynnik);

kopiuj(tablica, wysokosc);

break;

}

cout << "\n\nCzas: " << czas[0] << czas[1] << ":" << czas[2] << czas[3] << ":" << czas[4] << czas[5] << endl;

cout << "\nSzerokosc z GPS: " << szerokosc << " " << fi << endl;

cout << "\nDlugosc z GPS: " << dlugosc << " " << lambda << endl;

cout << "\nStatus wyznaczonej pozycji: " << status;

int stan = atoi(status);

if (stan == 1)

cout << " - tryb SPS, rozwiazanie fix" << endl;

if (stan == 0)

cout << " - nieprawidlowa" << endl;

cout << "\nliczba satelitow: " << liczbaSatelitow << endl;

cout << "\nWysokosc nad poziomem morza: " << wysokosc << " metrow" << endl;

double szer = atof(szerokosc); //przeliczanie z

double dlug = atof(dlugosc); //to decimal Degrees

int szerMin = atoi(szerokosc);

int dlugMin = atoi(dlugosc);

int szerA = szerMin;

int dlugA = dlugMin;

szerMin %= 100;

szerA /= 100;

dlugMin %= 100;

dlugA /= 100;

string google = "https://www.google.pl/maps/@"; //string z adresem internetowym

double testSzer = (((szer - (int)szer) + szerMin) / 60) + szerA;

google += doubl2str(testSzer);

google += ",";

double testDlug = (((dlug - (int)dlug) + dlugMin) / 60) + dlugA;

google += doubl2str(testDlug);

google += ",16z";

cout << "\nSzerokosc geograficza: " << szerA << " stopni " << szerMin << "'" //konwersja do DMS

<< static\_cast<int>((szer - (int)szer) \* 60) << "\"" << endl;

cout << "\nDlugosc geograficza: " << dlugA << " stopni " << dlugMin << "'"

<< static\_cast<int>((dlug - (int)dlug) \* 60) << "\"" << endl;

cout << "\nLink do map google: " << google << endl << endl;

serial.close();

}

catch(const char \*msg)

{

cout << msg << endl;

}

cout << "Nacisnij ENTER..." << endl;

cin.get();

return 0;

}

void kopiuj(char\* tabl, char\* zmienn) //funkcja kopiująca z tablica do mniejszych tablic do przecinka

{

i++;

int j = 0;

for (; \*(tabl + i) != ','; i++)

zmienn[j++] = tabl[i];

zmienn[j] = 0; //znak null na końcu C-stringa

}

string doubl2str(double i) // funkcja konwertująca liczbę double do typu string

{

stringstream ss;

string temp;

ss << i;

ss >> temp;

return temp;

}

**Podsumowanie**

Powyższe ćwiczenie pozwoliło nam na zapoznanie się z zasadami działania nawigacji satelitarnej GPS, poznanie specyfiki transmisji danych w standardzie NMEA. Zyskaliśmy także umiejętność zestawiania połączenia szeregowego poprzez bluetooth, co okazało się bardzo wygodnym sposobem odbierania danych z modułu GPS.