Piotr Chłystek 226100
Michał Chojnacki 225936

Data oddania sprawozdania: 4.12.2017 r.
Termin zajęć: Poniedziałek 7:30-10:15, TP

# Urządzenia peryferyjne

## Ćwiczenie 7

GPS

Prowadzący:
dr inż. Jan Nikodem

## 1. Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z zasadami działania systemu GPS, protokołem NMEA, dokumentacją odbiornika Nokia LD-1W, zasadami komunikacji urządzenia GPS z komputerem oraz obsługą map z poziomu aplikacji by w następnej kolejności napisać program, który będzie realizować obsługę systemu GPS.

## 2. Wstęp teoretyczny

GPS (z ang. Global Positioning System) to ogólnodostępne narzędzie, stworzone i administrowane przez amerykańskie wojsko. Służy ono do ustalenia pozycji geograficznej użytkownika, składającej się z parametrów takich jak długość i szerokość geograficzna oraz wysokość elipsoidalna. Długość geograficzna to kąt dwuścienny zawarty pomiędzy półpłaszczyznami południka 0° i południka, przechodzącego przez dany punkt. Przyjmuje wartość E (east) lub W (west), oznaczające kolejno wschód i zachód. Szerokość geograficzna to kąt pomiędzy płaszczyzną równika i promieniem Ziemi przechodzącym przez dany punkt. Przyjmuje wartość N (north) lub S (south), oznaczające kolejno północ i południe. Wysokość elipsoidalna to odległość od elipsoidy geocentrycznej do punktu powierzchni Ziemi, wyrażana w metrach. Wyznaczanie pozycji odbywa się dzięki 24 (wszystkich jest 31) satelitom, orbitującym na średniej orbicie okołoziemskiej (powyżej 2000 km i poniżej 36 000 km). Pozycja zostaje wyznaczona na podstawie pomiaru czasu dotarcia z satelity do odbiornika. Do wyznaczenia dokładnej pozycji potrzeba sygnału z przynajmniej 3 odbiorników. Dane przekazywane są w postaci tekstowej, zgodnej z protokołem NMEA (National Marine Electronics Association) 0183. Zdanie tego protokołu może mieć do 82 znaków, zaczyna się od znaku $, następny jest identyfikator zdania, pola danych są oddzielone przecinkami, zdanie kończy się znakami <CR><LF>, przed którym i opcjonalnie może wystąpić suma kontrolna. W pracy z technologią GPS skupimy się na danych zawartych w zdaniu $GPGGA (Global Positioning System Fix Data).

## 3. Opis programu

Zanim program zostanie uruchomiony, należy włączyć urządzenie GPS Nokia LD-1W i połączyć je przez Bluetooth z komputerem. Dane z odbiornika są przekazywane do komputera w sposób szeregowy, dlatego program zawiera pliki Serial.h i Serial.cpp, implementujące funkcje do obsługi portów szeregowych.

```
int write(const char buffer[]); //wysyła pustą tablicę znaków, jako parametr,
//zwraca ilość znaków

int write(const char *buffer, int buffLen); //wysyła wskaźnik na ciąg bajtów i
//ich liczbę na port szeregowy, zwraca liczbę wysłanych bajtów

int read(char *buffer, int buffLen, bool nullTerminate = true); //odczytuje ciąg
//bitów z portu szeregowego, zwraca liczbę przeczytanych bitów

void flush(); //funkcja czyszcząca wszystko z buforu portu szeregowego
```

W programie dodatkowo zaimplementowano funkcje

```
string doubleToString(double i); //służy do „rzutowania” liczby double na ciąg
//znaków typu string
void printChars(char* c, int len, string message); //wypisuje ciąg znaków
```

Funkcja główna, wykonująca się w programie wygląda następująco

```
int _tmain(int argc, _TCHAR* argv[])
{
	try {
		//wybieramy port urządzenia GPS, przykładowo COM7
		cout << "Enter the COM port of the GPS device [ex. COM7]: ";
		string port;
		cin >> port;
		//program pobiera z klawiatury numer/nazwę portu w formie typu string

		cout << "Opening the " + port + " port..." << endl;
		port += ":";
		TCHAR *portCom = new TCHAR[port.size() + 1];
		portCom[port.size()] = 0;
		copy(port.begin(), port.end(), portCom);
		// ustawianie portu
		tstring commPortName(portCom);
		// nawiazywanie polaczenia szeregowego na danym porcie z pomoca zew.
		//biblioteki, tworzymy zmienną typu serial i wysyłamy użytkownikowi
		//komunikat o otwarciu portu szeregowego
		Serial serial(commPortName);
		cout << "The port has been opened!" << endl;

		char tablica[RX_BUFFSIZE]; // bufor przechowujacy dane z GPSa
		//stała RX_BUFFSIZE została zdefiniowana na początku programu, wynosi
		//300, ponieważ GPS wysyła wiele zdań w protokole NMEA, nas interesują
		//jedynie te, które nastąpią po nagłówku GPGGA

		char time[6], latitude[9], longitude[9], satellites[2]; // zmienne
		przechowujace wyłuskane dane z sekwencji NMEA

		// odczyt danych z GPSa wykonywany przez iteracje
		int b = -1;
		int charsRead;
		do {
			b++;
			charsRead = NULL;
			for (int k = 0; k < 100; k++) tablica[k] = NULL;
			charsRead = serial.read(tablica, RX_BUFFSIZE);
			Sleep(1000);
		} while (b < 10);

		cout << endl;

		// szuka sekwencji $GPGGA - Global Positioning System Fix Data i pobiera
		z niej dane
		for (int i = 0; i < 100; i++) {

			if (tablica[i] == 'G' && tablica[i + 1] == 'G' && tablica[i + 2]
			== 'A')
			{
				//zgodnie z budową zdania NMEA po identyfikatorze GPGGA
				//znajduje się informacja o czasie
				i += 4;
				for (int j = 0; j < sizeof(time); j++) time[j] =
					tablica[i++];
				printChars(time, sizeof(time), "\nTime");
				i += 1;
				//informacja o szerokości geograficznej
				for (int j = 0; j < sizeof(latitude); j++) latitude[j] =
					tablica[i++];
```

```
                        latitude[sizeof(latitude) - 1] = tablica[i];
                        printChars(latitude, sizeof(latitude), "Latitude");
                        i += 3;
                        //informacja o długości geograficznej
                        for (int j = 0; j < sizeof(longitude); j++) longitude[j] =
                                tablica[i++];
                        longitude[sizeof(longitude) - 1] = tablica[i];
                        printChars(longitude, sizeof(longitude), "Longitude");
                        i += 4;
                        //informacja o ilości widocznych satelitów
                        for (int j = 0; j < sizeof(satellites); j++) satellites[j]
                                = tablica[i++];
                        printChars(satellites, sizeof(satellites), "Number of
                        satellites");
                        break;
                }
        }

        // obliczanie współrzędnych do Google Maps, rzutowanie odczytanych tablic
        //znaków na zmienne typu float
        double szer = atof(latitude);
        double dlug = atof(longitude);
        int szermin = atoi(latitude);
        int dlugmin = atoi(longitude);
        int szera = szermin;
        int dluga = dlugmin;
        szermin %= 100;  //część całkowita to stopnie, ułamkowa minuty
        szera /= 100;
        dlugmin %= 100;
        dluga /= 100;

        //string z domeną map Google, do którego „dokleimy” nasze współrzędne by
        //otrzymać link do mapy Google z zaznaczoną pozycją użytkownika
        string google = "https://www.google.pl/maps/@";

        //zamieniamy ułamki stopni na minuty
        double testszer = (((szer - (int)szer) + szermin) / 60) + szera;
        google += doubleToString(testszer);
        google += ",";
        double testdlug = (((dlug - (int)dlug) + dlugmin) / 60) + dluga;
        google += doubleToString(testdlug);
        google += ",16z";

        //program wypisuje link do mapy Google, wskazującej pozycję użytkownika
        cout << "\nGoogle Maps: " << google << endl << endl;

    } catch(const char *error) {
        cout << error << endl; //blok kodu, który się wykona jeżeli na początku
                               //programu wystąpi błąd związany z portem szeregowym
    }
    fseek(stdin, 0, SEEK_END);
    getchar();
    return 0;
}
```

## 4. Wnioski

Program napisany w trakcie zajęć działał wadliwie, został poprawiony po zajęciach. Nie posiadając właściwego sprzętu nie byliśmy w stanie sprawdzić czy ostateczna wersja programu działa w pełni prawidłowo. Testowaliśmy program (GPS-NMEA-Parser-Test na pendrive), podając mu ustalony przykładowy kod NMEA. Testy wykazały że poprawiona wersja programu powinna we współpracy z urządzeniem GPS wskazywać poprawną pozycję użytkownika.