System nawigacji satelitarnej GPS, część 11

Komunikacja z odbiornikiem GPS

Zajmiemy się teraz pokazaniem sposobów wydzielania i przetwarzania interesujących nas danych z wiadomości wysyłanych przez odbiornik GPS w formacie NMEA–0183. W urządzeniach elektronicznych współpracujących z odbiornikami GPS, zadanie to jest najczęściej realizowane przez odpowiednio oprogramowany mikrokontroler. Opracowanie programu mikrokontrolera, służącego do odbioru i formatowania danych nawigacyjnych, jest więc jednym z najważniejszych problemów, przed którym stają elektronicy konstruujący tego typu urządzenia.

Zasada tworzenia takiego oprogramowania jest przedmiotem dalszej części artykułu, w którym jako przykłady przedstawiono fragmenty kodu napisanego w języku C. Niektóre zamieszczone w artykule fragmenty kodu zostały napisane dla mikrokontrolerów z rodziny 8051, natomiast inne dla mikrokontrolerów AVR. Stanowią one części złożonych programów, których działanie zostało w większości przypadków sprawdzone w wykonanych prototypach urządzeń współpracujących z odbiornikami GPS.

Wykorzystanie danych w formacie NMEA we własnych aplikacjach – odbiór wiadomości

Jak już wspomniano, poszczególne dane nawigacyjne są umieszczone w postaci ciągu znaków w polach rozdzielonych znakami separującymi. Odnajdywanie i wydzielanie tych danych wymaga poszukiwania znaków separujących, poprzedzających i kończących interesujące nas pola wiadomości oraz wydzielanie zawartej pomiędzy nimi treści. Jest to podejście pewniejsze od poszukiwania i wydzielania zawartości pól na podstawie zliczania odbieranych znaków, ponieważ uniezależnia nasz program od liczby znaków w poszczególnych polach. Poszukiwanie pól wiadomości może być realizowane na bieżąco, poprzez sprawdzanie bajt po bajcie (znak po znaku) danych otrzymywanych z odbiornika GPS, lub z wykorzystaniem bufora, do którego dane te są wstępnie zapisywane, a następnie po skompletowaniu jednej lub większej ilości wiadomości są poddawane analizie i wydzielaniu informacji z poszczególnych pól. W przedstawianych tutaj przykładowych fragmentach kodu wykorzystano drugi z wymienionych sposobów.

Na **list. 1** przedstawiono przykładową funkcję służącą do odbioru wiadomości RMC z odbiornika GPS i sprawdzenia poprawności jej transmisji poprzez porównanie obliczonej i otrzymanej sumy kontrolnej. W tym przykładzie założono, że odbiornik GPS został wstępnie skonfigurowany w taki sposób, aby wysyłał wyłącznie wiadomość RMC z polem sumy kontrolnej. W związku z tym funkcja *ReceiveGPSPacket()* nie sprawdza, czy otrzymana wiadomość jest rzeczywiście oczekiwaną wiadomością RMC.

Przedstawiony fragment kodu źródłowego, pochodzi z rozbudowanego programu dla mikrokontrolera AVR ATmega128, służącego do wspólnego przetwarzania danych z odbiornika GPS i systemu nawigacji inercjalnej. Dane tych urządzeń były odbierane przez oba dostępne w ATMega128 porty szeregowe, przy czym dane z GPS odbierano przez USART0. Na początku listingu podano definicje stałych i struktury GPS, wykorzystywanych przez funkcję *ReceiveGPSPacket()*. Maksymalny rozmiar tablicy GPS.Packet[] w strukturze danych GPS, przeznaczonej do przechowywania części odebranej wiadomości RMC, zawierającej się pomiędzy znakami ‚$' i ‚*', odpowiada maksymalnej liczbie znaków, które mogą tam wystąpić MAX_RMC_SIZE. Największa liczba znaków w pełnej wiadomości RMC wynosi 75, a zatem pomniejszając ją o pomijany przy zapisie do *GPS.Packet[]* 1 znak początkowy ‚$' i 5 znaków kończących wiadomość „*CS<CR><LF>", otrzymujemy rozmiar tablicy równy 69 bajtów. Dodatkowo rozmiar tablicy powiększono o 1, aby zostawić w niej miejsce na znak ‚\0', który ułatwi w przyszłości znalezienie końca wiadomości RMC zapisanej w tablicy.

Wewnątrz funkcji *ReceiveGPSPacket()* znajdują się wywołania do własnej funkcji *ReceiveByteViaUSART0()*, która służy do odbioru danych przez port szeregowy USART0 z wykorzystaniem przerwań. Zamiast tej funkcji można jednak skorzystać ze standardowej, niewykorzystującej przerwań, funkcji bibliotecznej *getchar()*.

Na początku funkcja *ReceiveGPSPacket()* oczekuje na wiadomości przychodzące przez port szeregowy i poszukuje rozpoczynającego je znaku ‚$'. Po odnalezieniu początku wiadomości rozpoczyna się jej zapisywanie znak po znaku do tablicy GPS.Packet[] i jednocześnie obliczanie sumy kontrolnej ChSumCalc. Proces ten odbywa się w pętli *while()* do momentu

List. 1. Funkcja realizująca odbiór danych z odbiornika GPS

```
#define GPS_HEADER            ,$'
#define GPS_CHSUM_SEPARATOR   ,*'
#define MAX_RMC_SIZE    69

struct GPS_TYPE
  {
  unsigned char Packet[MAX_RMC_SIZE+1];
  unsigned char ChSumCorrect;
  } GPS;

void ReceiveGPSPacket ( void )
  {
  unsigned char   Count = 0;           // licznik odebranych znaków
  unsigned char   RxChar;              // zmienna przechowująca odebrane znaki
  unsigned char   ChSumCalc;             // suma kontrolna obliczona
  unsigned char   ChSumRcv;            // suma kontrolna odebrana z GPS

  while ( ReceiveByteViaUSART0()!=GPS_HEADER );          // oczekiwanie na początek wiadomości
  ChSumCalc = 0;
  while ( (RxChar=ReceiveByteViaUSART0()) !=                      // odbiór znaków, aż do ,*'
          GPS_CHSUM_SEPARATOR )
     {
     GPS.Packet[Count++] = RxChar;        // zapis znaków do tablicy
     ChSumCalc = ChSumCalc ^ RxChar;      // obliczanie sumy kontrolnej
     }
  GPS.Packet[Count] = ,\0';
  RxChar = ReceiveByteViaUSART0();               // odbiór 2 bajtów sumy kontrolnej
  if ( RxChar > '9' )        RxChar - = 55;
  else               RxChar - = 48;
  ChSumRcv = 16 * RxChar;
  RxChar = ReceiveByteViaUSART0();
  if ( RxChar > '9' )        RxChar - = 55;
  else               RxChar - = 48;
  ChSumRcv += RxChar;
  if ( ChSumRcv == ChSumCalc )         // porownanie odebranej i policzonej
     GPS.ChSumCorrect = 1;           // sumy kontrolnej
  else
     GPS.ChSumCorrect = 0;
     while ( ReceiveByteViaUSART0()!='\n' );                      // oczekiwanie na koniec wiadomości
  }
```

znalezienia separatora sumy kontrolnej ,*'. Następnie odbierane są 2 bajty sumy kontrolnej i obliczana jest jej wartość ChSumRcv.

Pewnego wyjaśnienia wymaga sposób obliczania sumy kontrolnej w przedstawionym na list. 1 fragmencie programu. Oba bajty sumy kontrolnej są wysyłane jako znaki ASCII odpowiadające cyfrom w kodzie szesnastkowym, tzn. np. zamiast bajtu o wartości 5 wysyłany jest bajt o wartości odpowiadającej kodowi ASCII znaku ,5', itd. Kody ASCII znaków od ,0' do ,9' są o 48 większe od liczb z przedziału 0...9, natomiast kody ASCII znaków od ,A' do ,F', odpowiadających liczbom 10...15 w kodzie szesnastkowym, są o 55 większe od liczb z przedziału 10...15. Wynika stąd, że aby policzyć sumę kontrolną należy pomniejszyć wartości odebranych bajtów odpowiednio o 48 lub 55, a następnie zsumować starszy bajt pomnożony przez 16 z młodszym bajtem. Wyjaśniono to w poniższym przykładzie:

Przykład:

```
2B - odebrane znaki pola sumy
kontrolnej
50, 66 - wartości odebranych bajtów,
odpowiadające powyższemu polu sumy
kontrolnej
50 - liczba stanowiąca kod ASCII
znaku ,2'
66 - liczba stanowiąca kod ASCII
znaku ,B'
(50-48)*16 + (66-55) = 43
- obliczona wartość sumy kontrolnej
w kodzie dziesiętnym
```

Na zakończenie działania funkcji *ReceiveGPSPacket()*, obliczona i otrzymana suma kontrolna są ze sobą porównywane, a wynik sprawdzenia jest zapisywany we wchodzącej w skład struktury danych GPS zmiennej GPS.ChSumCorrect. Informacja o zgodności sumy kontrolnej może być testowana przez inne funkcje programu np. w celu sprawdzenia, czy dane zawarte w GPS.Packet[] nadają się do dalszego wykorzystania. Po odebraniu pełnej wiadomości RMC i zakończeniu działania funkcji *ReceiveGPSPacket()*, dane nawigacyjne są umieszczone w tablicy GPS.Packet[], a flaga GPS.ChSumCorrect stanowi wygodny wskaźnik poprawności ich odbioru.

Piotr Kaniewski
pkaniewski@wat.edu.pl