Grzegorz CHMAJ, Tomasz BOJKO
AKADEMIA GÓRNICZO-HUTNICZA, WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI, KATEDRA ROBOTYKI I DYNAMIKI MASZYN

Aplikacja radiowej akwizycji danych z rozproszonych obiektów pomiarowych

Mgr inż. Grzegorz CHMAJ

Doktorant w Katedrze Robotyki i Dynamiki Maszyn AGH. Jego zainteresowania dotyczą mechatroniki, akwizycji sygnałów pomiarowych oraz zastosowań systemów informatycznych w dziedzinie pomiarów i mechatroniki. Jest autorem prac dotyczących systemów informatycznych stosowanych w zagadnieniach akwizycji sygnałów.

e-mail: chmaj@agh.edu.pl

Dr inż. Tomasz BOJKO

Adiunkt w Katedrze Robotyki i Dynamiki Maszyn AGH. Jego zainteresowania dotyczą mechatroniki, robotyki i automatyki. Jest autorem prac dotyczących robotyki, układów napędowych robotów, szybkiego prototypowania układów sterowania, układów MEMS oraz rozwiązań komunikacji bezprzewodowej.

e-mail: bojko@uci.agh.edu.pl

Streszczenie

Współcześnie dużego znaczenia nabierają prace związane z bezprzewodowymi systemami pomiarowymi realizowanymi drogą radiową. Rozważając możliwości zastosowań tych systemów szybko znajdujemy szereg dziedzin, gdzie radiowa akwizycja danych jest techniką bardzo wygodną i niejednokrotnie jedyną z możliwych do zastosowania. Systemy radiowe, ze względu na swój charakter, mogą pracować w środowiskach, gdzie wymagana jest akwizycja sygnałów z różnych – oddalonych od siebie punktów pomiarowych. Cecha ta sprawiła, że systemy te są często stosowane w celach aktywnego monitoringu obiektów czy konstrukcji mechanicznych, komunikacji pomiędzy robotami mobilnymi, czy w takich dziedzinach jak geodezja czy tektonika. W niniejszym artykule zostanie przedstawiona aplikacja służąca do akwizycji danych pomiarowych z bezprzewodowych układów pomiarowych firmy Crossbow pracujących z systemem operacyjnym o nazwie TinyOS.

PC based application of data collecting from distributed measured objects

Abstract

Nowadays, applied research has been developing very rapidly in wireless sensor networks. Considering possibilities of applying those systems, we quickly find a couple of disciplines, where wireless sensor networks are very comfortable in use and many a time the only possible ones. Those systems, for the sake of their character, can run in environments, where data collecting from different places of measured object is required. This feature makes that wireless sensor networks are often used to perform active monitoring of buildings or mechanical constructions, mobile robots' navigation and also in different disciplines like geodesy. The article is going to present an application for collecting data which works with wireless sensors made by Crossbow company. The sensors exploit the TinyOS operational system.

1. Wprowadzenie

Szukając rozwiązań dla bezprzewodowy systemów pomiarowych trafiamy na szereg firm oferujących takie rozwiązania. Jedną z nich jest firma Crossbow, od której w ramach prowadzonego projektu badawczego w Katedrze Robotyki i Dynamiki Maszyn AGH został zakupiony pakiet bezprzewodowych układów pomiarowych pracujących z systemem operacyjnym o nazwie TinyOS - systemem typu open-source. TinyOS zawiera szereg bibliotek dostarczających między innymi: protokół sieciowy, sterowniki do różnorakich czujników oraz narzędzia do akwizycji danych, użytkownik natomiast pracuje na wyższej warstwie dostępu korzystając z implementacji tychże bibliotek. W celu zbudowania kompletnego systemu pomiarowego współpracującego z zakupionymi układami pomiarowymi, umożliwiającego akwizycję danych z tych układów, zaistniała potrzeba opracowania aplikacji kontrolującej sieć bezprzewodowych układów pomiarowych. Opracowana aplikacja o nazwie Mote Viewer została napisana w języku C# na platformie .NET.

W obecnej wersji aplikacja obsługuje trzy różne programy akwizycji danych z sieci bezprzewodowych układów pomiarowych. Wszystkie programy jak i sposób komunikacji pomiędzy węzłami sieci zostaną omówione w niniejszym artykule.

2. Opis programów pomiarowych, sposób działania

Obsługiwane przez aplikację Mote Viewer programy pomiarowe różnią się między sobą przede wszystkim stopniem zaawansowania. Dwa pierwsze z nich są to programy testowe mające na celu testowanie komunikacji pomiędzy siecią czujników a stacją bazową oraz aplikacją nadzorującą sieć. W tym celu pierwszy z obsługiwanych programów działa na zasadzie nasłuchu portu szeregowego oczekując na pakiety dostarczane przez jeden węzeł sieci. Pakiety te są wysyłane przez węzeł z ustalonym interwałem czasowym i zawierają informację o stanie czterech różnych czujników: magnetonometru, akcelerometru, fotodiody, mikrofonu oraz informację o stanie akumulatorów zasilających węzeł. Drugi program testowy ma zaimplementowaną pełną kontrolę nad węzłem sieci. Umożliwia to użytkownikowi między innymi wybranie czasu startu pomiaru, interwału czasowego, liczby próbek czy sterowanie mocą radia.

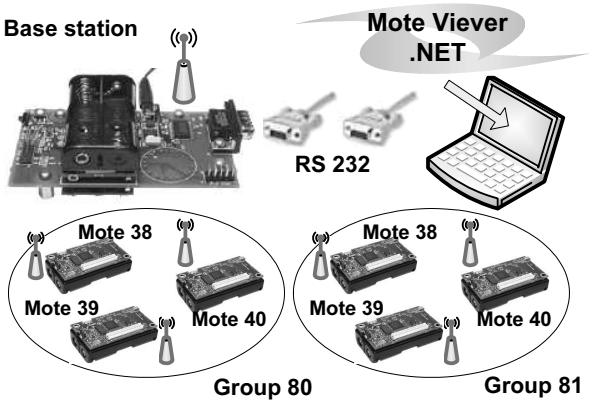
Główny założeniem aplikacji Mote Viewer było umożliwienie przeprowadzenia pomiaru, w którym może brać udział dowolna dostępna liczba węzłów sieci. W tym celu została zaimplementowana obsługa trzeciego programu przeznaczonego do pomiaru przyśpieszenia drgań mechanicznych. Zasada działania programu polega na jednoczesnym przesłaniu do wszystkich węzłów sieci rozkazu startu pomiaru zawierającego dodatkowo informację o częstotliwości próbkowania oraz wymaganej liczbie próbek pomiarowych. Podczas trwania pomiaru każdy węzeł gromadzi rejestrowane próbki w swojej pamięci. Akwizycja danych przez aplikację wygląda następująco: po skończonym pomiarze każdy węzeł sieci zostaje odpytany z osobna w celu przesłania do aplikacji zgromadzonych danych pomiarowych. W taki sposób wszystkie zgromadzone dane pomiarowe są dostępne dla użytkownika w aplikacji Mote Viewer, gdzie mogą zostać poddane obróbce i analizie.

W celu lepszego zrozumienia komunikacji pomiędzy siecią czujników, stacja bazową oraz aplikacją nadzorującą sieć zostanie omówiony przepływ danych w systemie.

3. Przepływ danych, komunikacja z siecią

Na rysunku 1 został przedstawiony schemat opisujący komunikację aplikacji z siecią. Jak wynika z rysunku, aplikacja Mote Viewer poprzez port szeregowy komputera komunikuje się ze stacją bazową sieci. Stacja bazowa nie posiada żadnych informacji odnośnie sieci.

Jedynym jej zadaniem jest wysyłanie oraz odbieranie pakietów sieciowych. Dla przykładu rozważmy sytuację, w której aplikacja Mote Viewer wysyła rozkaz pomiaru do wszystkich węzłów w sieci, następnie każdy węzeł zostaje odpytany z osobna, w celu przesłania do aplikacji zarejestrowanych danych. Aby rozpocząć pomiar we wszystkich węzłach sieci, zostaje zbudowany odpowiedni pakiet sieciowy zawierający między innymi informację do kogo adresowany jest pakiet.



Rys. 1. Komunikacja aplikacji Mote Viewer z siecią czujników

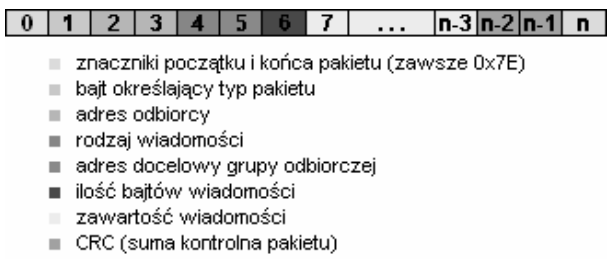
Każdy węzeł posiada swój adres i nasłuchuje eter w celu wyłapania pakietów zaadresowanych do niego, lecz istnieje adres o wartości 65535, którym można zaadresować pakiet w celu odebrania go przez wszystkie węzły w sieci. W ten sposób aplikacja wysyła rozkaz startu pomiaru do wszystkich węzłów. Po odebraniu takiego pakietu, węzeł rozpoczyna pomiar. Częstotliwość próbkowania oraz liczba próbek do zarejestrowania również umieszczana jest w pakiecie.

Po zarejestrowaniu danych zostaje zbudowany pakiet zawierający rozkaz odczytu danych. Każdy pakiet zostaje zaadresowany do pojedynczego węzła i w momencie odebrania go przez adresata rozpoczyna się transfer zarejestrowanych danych. Wszystkie dane z każdego węzła zostają zgromadzone w aplikacji. Dalsza ich obróbka zależy już od użytkownika.

Istotną cechą systemu TinyOS jest fakt, że węzły w sieci mogą zostać podzielone na grupy, których może być do 256. Dla każdej grupy można wyodrębnić do 65535 unikalnych adresów sieciowych, po których są identyfikowane węzły w sieci. Maksymalna ilość grup, jak i ilość węzłów w sieci zdeterminowane są ilością bajtów przewidzianych w nagłówku pakietu - dla adresów grup 1 bajt , dla adresów węzłów 2 bajty.

4. Opis pakietu sieciowego

Opis pakietu sieciowego obsługiwanego przez system TinyOS został przedstawiony na rysunku 2. Długość pakietu sieciowego może być różna i zależy od dwóch czynników. Pierwszy z nich określa długość właściwej wiadomości (bajty od 7 do n-3).



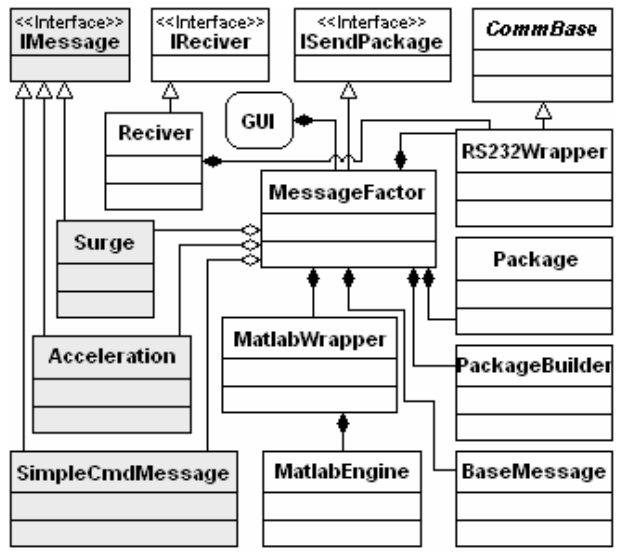
Rys. 2. Budowa pakietu sieciowego systemu TinyOS

W zależności od zainstalowanego programu w układzie pomiarowym długość ta może ulegać zmianie. Drugi czynnik mówi o ilości wystąpień wartości 0x7E oraz 0x7D w pakiecie. Ponieważ wartość 0x7E jest zarezerwowana dla bajtów początku i końca pakietu nie może się ona pojawić wewnątrz pakietu. Jeśli jednak przetwornik A/C zwróci taką wartość zostanie ona zamaskowana poprzez dwa specjalne bajty: 0x7D oraz wynik operacji bitowej 0x7E XOR 0x20. Tak więc miejsca, gdzie pojawią się bajty o wartości 0x7E zostaną zamaskowane dwoma bajtami o wartościach 0x7D oraz 0x5E. Skoro bajt 0x7D jest używany do operacji maskowania, w przypadku pojawienia się na wyjściu przetwornika wartości 0x7D zostaje ona również poddana operacji bitowej różnicy symetrycznej by do pakietu wstawić bajty 0x7D oraz 0x5D. Każdy pakiet odebrany czy to przez aplikację Mote Viewer czy to przez układ pomiarowy zostaje sprawdzony pod kątem wystąpienia zamaskowanych bitów, by wrócić do ich pierwotnej wartości.

Po omówieniu pakietu sieciowego systemu TinyOS zostanie przedstawiona struktura samej aplikacji akwizycji danych.

5. Struktura aplikacji Mote Viewer. Diagram klas

Strukturę aplikacji akwizycji danych pomiarowych wraz z dodatkowymi modułami przedstawia poniższy diagram klas zbudowany w języku UML.



Rys. 3. Diagram klas aplikacji Mote Viewer

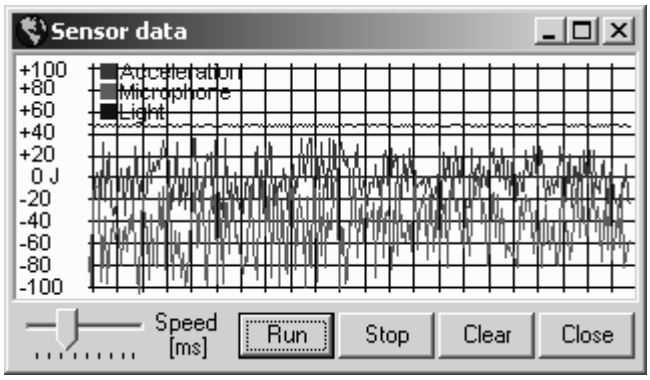
Całość aplikacji składa się z szeregu klas pełniących odpowiednie funkcje oraz panelu użytkownika - Graphics User Interface (GUI)- skąd cała aplikacja jest sterowana. Główną klasą aplikacji jest klasa *MessageFactory*, która jest klasą nadrzędną dla pozostałych. Dziedziczy ona z interfejsu *ISendPackage* implementując jego metody w celu zapewnienia sobie możliwości wysłania informacji do panelu użytkownika o każdym pakiecie, który został odebrany z sieci. Odbiorem i wysyłaniem pakietów zajmuje się klasa *RS232Wrapper*, która jest odpowiedzialna za komunikację z portem szeregowym komputera. Zanim jednak jakikolwiek pakiet zostanie wysłany do węzłów w sieci, musi on zostać w odpowiedni sposób zbudowany i sprawdzony. Wszystko to dzieje się w klasie *Package*, w której zostały zaimplementowane metody budowy pakietów sieciowych zgodnych z specyfikacją systemu TinyOS.

Odbiorem pakietów przesyłanych przez węzły z sieci zajmuje się klasa *Reciver*. Klasa ta ma za zadanie budowę pakietów z pojedynczych bajtów odczytanych z portu szeregowego. Na podstawie otrzymanych bajtów następuje rozpoznanie początku

i końca pakietu. Jeżeli pakiet został pomyślnie zbudowany zostaje on przekazany do klasy *PackageBuilder*, gdzie gromadzone są wszystkie otrzymane pakiety.

Każdy program zaimplementowany w aplikacji (szare bloczki diagramu z rysunku 3) jest osobną klasą, dziedzicząca z interfejsu *IMessage* i implementującą w ten sposób niezbędne metody budowy komunikatów sterujących węzłami w sieci. Rozwiązanie takie, w łatwy sposób pozwala na rozszerzanie aplikacji o nowe programy, dla których rozkazy sterujące mogą się różnić. Dla przykładu, opisywany wcześniej program służący do równoczesnego pomiaru we wszystkich węzłach sieci został zaimplementowany w klasie *Acceleration*.

Po zakończonym pomiarze i akwizycji danych, wszystkie próbki pomiarowe z poszczególnych węzłów są przechowywane, jak już wspomniano wcześniej, w klasie *PackageBuilder*. Użytkownik ma do dyspozycji panel, na którym może oglądnąć zarejestrowane sygnały, lub może je wyeksportować do przestrzeni roboczej środowiska Matlab. Zadaniem tym zajmuje się klasa *MatlabWrapper*, która jest odpowiedzialna za nawiązanie sesji z środowiskiem Matlab oraz transfer danych. Wspomniany wcześniej panel służący do odtwarzania zarejestrowanych sygnałów może również służyć jako panel do monitoringu próbek odbieranych od węzła sieci. Aby taki monitoring był możliwy, wymagany jest odpowiedni program zainstalowany na poszczególnych węzłach sieci, który na bieżąco będzie przesyłał zarejestrowane dane do aplikacji. W aplikacji została zaimplementowana obsługa takiego programu w klasie Surge, a przykładowe przebiegi wyświetlane przez panel zostały pokazane na rysunku 4.

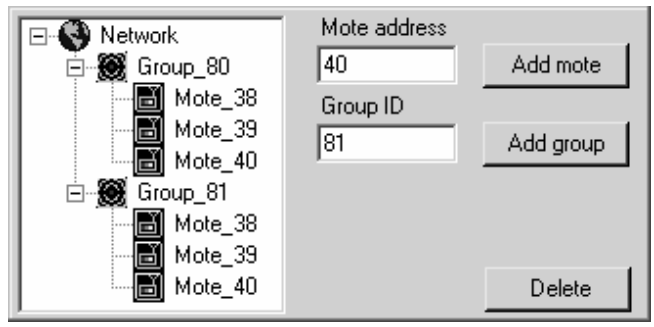


Rys. 4. Przebiegi czasowe z monitoringu obiektu pomiarowego

Po przedstawieniu struktury aplikacji, struktury przepływu danych oraz budowy pakietu sieciowego zostanie omówiona przykładowa sesja pomiarowa.

6. Opis sesji pomiarowej, eksport danych

Opis przykładowej sesji pomiarowej zostanie przedstawiony dla aplikacji Acceleration. Zadaniem aplikacji jest pomiar przyśpieszenia drgań mechanicznych równocześnie we wszystkich węzłach sieci, a po skończonym pomiarze akwizycja danych. W pomiarze będzie brało udział sześć węzłów podzielonych na dwie grupy. Jednej z grup nadano identyfikator o numerze 80 drugiej o numerze 81. Każdemu węzłowi w danej grupie przydzielone zostały odpowiednio adresy 38, 39 oraz 40. Pomiar zostanie przeprowadzony przy udziale wszystkich węzłów należących do grupy 80, a następnie do grupy 81. Pierwszym krokiem będzie zbudowanie sieci odpowiadającej opisanej sytuacji. Do tego celu służy obiekt z panelu użytkownika pokazany na rysunku 5. Budowa sieci następuje poprzez dodanie do drzewa odpowiedniej liczby grup, a następnie do każdej grupy węzłów do niej należących. Drugim krokiem jest ustalenie interwału czasowego oraz wymaganej liczby próbek, wybranie grupy, w której ma być przeprowadzony pomiar, a następnie przesłanie tak zbudowanego rozkazu do sieci. Jak pamiętamy warunkiem jednoczesnego rozpoczęcia pomiaru we wszystkich węzłach danej grupy jest zaadresowanie pakietu sieciowego adresem 255.



Rys. 5. Budowa sieci pomiarowej w aplikacji Mote Viewer

Po przeprowadzonym pomiarze w grupie 80 zmieniamy adres grupy na 81 i rozpoczynamy pomiar w tej grupie. Po zakończonej sesji pomiarowej w dwóch grupach jesteśmy gotowi do przeprowadzenia akwizycji próbek pomiarowych.

Czytanie danych z węzłów polega na zaznaczeniu odpowiedniego węzła z rysunku 5, a następnie wysłaniu do sieci rozkazu czytania danych (pakiet zostanie automatycznie zaadresowany wybranym węzłem z drzewa). Jeśli dane z wszystkich węzłów sieci zostaną zgromadzone w aplikacji, można je wyeksportować do przestrzeni roboczej środowiska Matlab, gdzie mogą zostać poddane procesowi przetwarzania.

7. Podsumowanie

Aplikacja Mote Viewer została zaprojektowana do celów akwizycji sygnałów pomiarowych z rozproszonych sieci czujników bezprzewodowych. Jest przeznaczona do współpracy z radiowymi układami pomiarowymi pracującymi z systemem operacyjnym TinyOS. Duża liczba układów pomiarowych mogących brać udział w pomiarze sprawia, że system może być stosowany do monitoringu dużych obiektów pomiarowych.

W trakcie dalszy prac aplikacja Mote Viewer zostanie rozszerzona o kolejne programy monitoringu i akwizycji danych oraz moduł zdalnego przeprogramowywania układów pomiarowych. Zadaniem takiego modułu będzie zdalna instalacja nowego programu w układzie pomiarowym bez konieczności demontażu takiego układu z obiektu pomiarowego.

Dodatkowo aplikacja Mote Viever zostanie rozbudowana o moduł dostępu do bazy danych w celu archiwizacji rejestrowanych sygnałów pomiarowych.

Prace finansowane są z projektu badawczego T1825130728.

8. Literatura

[1] MTS/MDA Sensor and Data Acquisition Boards User's Manual, Document 7430-0020-03 Revision A, Crossbow, April 2004
[2] MPR/MIB User's Manual, Document 7430-0021-06 Revision A, Crossbow, August 2004
[3] Getting Started Guide, 7430-0022-05, Crossbow, August 2004
[4] Steven John Metsker, Design Patterns in C#, Helion 2005
[5] Wrycza S., Marcinkowski B., Wyrzykowski K., Język UML 2.0 w modelowaniu systemów informatycznych, Helion, 2006
[6] Mielczarek W., Szeregowe interfejsy cyfrowe, Helion, 1994

Artykuł recenzowany