Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej Nr 26

XIX Seminarium

ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE' 2009
Oddział Gdański PTETiS
Referat nr 2

INTEGRACJA ARCHITEKTURY J2EE Z INTERFEJSEM 1-WIRE W SYSTEMIE IDENTYFIKACJI I AKWIZYCJI DANYCH

Tomasz BORZYM[1], Lech HASSE[2]

1. GE Money Bank, Pion Informatyki, ul. Klonowa 4M, 80-264 Gdańsk
   tel: 48660713810, fax: 0583085414, e-mail: tomasz.borzym@ge.com
2. Politechnika Gdańska, ul. G. Narutowicza 11/12, 80-233 Gdańsk
   tel: 0583471884, fax: 0583416132, e-mail: lhasse@pg.gda.pl

Streszczenie: W projekcie systemu integrującego architekturę J2EE z interfejsem pomiarowym 1-wire wykorzystano: platformę TINI jako system bazowy, klucze iButton jako metodę identyfikacji, sieć internetową jako sposób integracji z innymi systemami, J2EE jako model architektury logicznej oraz technologię JSP, Sun Application Server i MySQL. System umożliwia dodanie i usunięcie klucza z bazy danych, zapisanie historii interakcji, dostarczenie mechanizmu odblokowującego dostęp do określonej lokalizacji. Oprogramowano system implementując funkcjonalność pomiaru temperatury, zarządzania dostępem do pomieszczeń, a także akwizycji czasu pracy. Większość prac programistycznych zostało przeprowadzonych w zintegrowanym środowisku Integrated Development Environment. Platforma umożliwia pisanie kodu w wielu językach programowania. Zaletą platformy jest zintegrowanie jej z serwerem aplikacyjnym Glassfish oraz bazami danych MySQL. Programowanie platformy TINI odbywało się w trybie kilkukrokowym, a program uruchamiany był w konsoli tekstowej udostępnianej przez protokół TELNET.

Słowa kluczowe: system identyfikacji i akwizycji danych, interfejs pomiarowy 1-wire, platforma TINI

1. WPROWADZENIE

Dzięki standaryzacji architektury PC, technologia informacyjna mogła pójść własnym torem. Aktualnie można wyraźnie odróżnić „programistów-elektroników" – dla których kod programu jest silnie uwarunkowany zasobami sprzętowymi, oraz „programistów-informatyków" – starających się wchodzić na coraz to wyższy poziom abstrakcji. Obie postawy mają swoje wady i zalety, jednak w niektórych przypadkach ci pierwsi mogą skorzystać z doświadczenia tych drugich. Na przykładzie zaprojektowanego systemu identyfikacji i akwizycji danych pokazano, w jaki sposób można spiąć te dwie techniki programowania i jakie korzyści można dzięki takiemu spięciu uzyskać. Przy budowie systemu wykorzystano ze świata elektroniki system pomiarowego reprezentowany przez chip komponenty iButton oraz platformę TINI w celu integracji z wysokimi warstwami programistycznymi. Stronę informatyczną reprezentuje platforma J2EE.

2. SYSTEM BAZOWY

2.1. Charakterystyka technologii platformy TINI

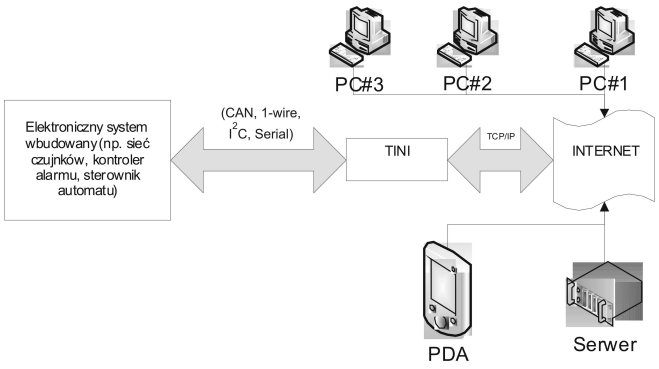
W roku 1998 w firmie Dallas Microconductors wspólnie z firmą Sun opracowano mikrosystem oprogramowany w języku Java, w celu obsługi szerokiej gamy sprzętów elektrycznych w inteligentnym budynku. Prototypowe moduły zostały zainstalowane m.in. w aplikacjach sterujących oświetleniem, systemach klimatyzacyjnych i urządzeniach gospodarstwa domowego. Aplikacje „porozumiewały" się ze sobą za pośrednictwem centralnego serwera, który dodatkowo umożliwiał centralne sterowanie każdą z nich. Ideą rozwiązania była integracja szerokiej gamy interfejsów elektronicznych w jedną sieć zapewniającą zdalne sterowanie i monitoring [1]. Programistyczne interfejsy dla aplikacji API (*ang. Application Programming Interface*) uległy później ujednoliceniu pod kątem standardów obsługi sieci – głównie przez popularyzację protokołu TCP/IP, a dzięki wyposażeniu aplikacji mikrokontrolerowych w coraz to nowocześniejsze urządzenia I/O (np. Bluetooth, Zig-Bee, WLAN, GPS) znacznie wzrósł potencjał takich aplikacji. Fundamentem platformy stały się następujące założenia:

- proces budowy aplikacji powinien być maksymalnie uproszczony,
- aplikacja docelowo powinna integrować się ze środowiskiem sieciowym,
- rozwiązanie powinno być energooszczędne i tanie,
- aplikacja kontaktuje się z otoczeniem za pomocą interfejsu ETHERNET i protokołu TCP/IP i jest wyposażona w szereg interfejsów elektronicznych umożliwiając wprowadzanie modyfikacji przez dewelopera.

Opracowany system (rys. 1) stanowi połączenie silnego Chipsetu i wirtualnej maszyny Javy. Chipset zapewnia przetwarzanie danych i komunikację na poziomie urządzenia, natomiast wszystkie funkcjonalności sprzętowe zaszyte są w doskonale udokumentowanych bibliotekach programistycznych udostępnionych deweloperowi. Celem jest zapewnienie komunikacji sieciowej dla wszystkich, dotychczas autonomicznych, urządzeń, począwszy od mikrosensorów czy siłowników, a kończąc na potężnych

Recenzent: Dr inż. Jerzy Hoja – Wydział Elektroniki Telekomunikacji i Informatyki Politechnika Gdańska

zautomatyzowanych aplikacjach fabrycznych lub urządzeniach „starej generacji”, zapewniając dla nich wsteczną kompatybilność.



Rys. 1. Schemat blokowy aplikacji na platformie TINI

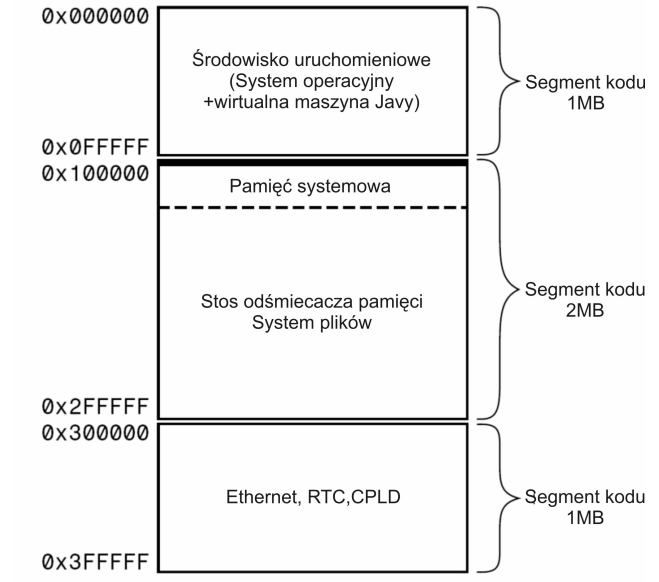
To wszystko przy dodatkowym założeniu, że interakcja będzie możliwa przez popularne standardy sieciowe, jak np. przeglądarka WWW, protokoły FTP, Telnet, etc.

2.2. Środowisko sprzętowe

Podstawową funkcjonalność platformy TINI zapewniają następujące elementy:

- mikrokontroler DS80C390/DS80C400/DS80C410,
- pamięć ROM typu FLASH,
- pamięć operacyjną SRAM,
- magistrala równoległa umożliwiająca podłączenie dodatkowych urządzeń, m.in.: kontrolera Ethernet, zegara czasu rzeczywistego, układów CPLD (*Complex Programmable Logic Device*).

Mapa pamięci (rys. 2) określa gdzie w przestrzeni adresowej mikrokontrolera znajduje się odniesienie do poszczególnych segmentów pamięci odpowiedzialnych za kod, dane i peryferia.



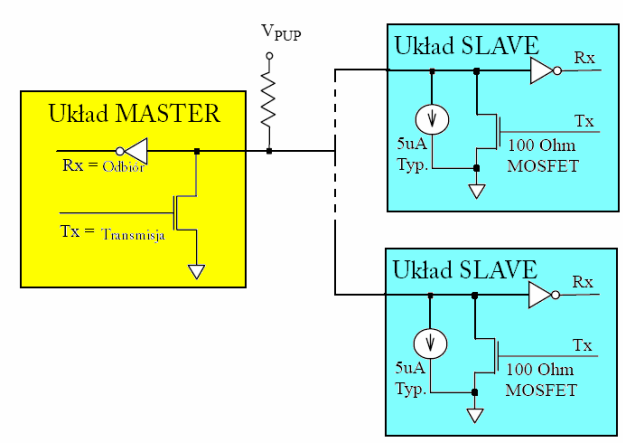
Rys. 2. Mapa pamięci środowiska TINI

Poza urządzeniami, z którymi komunikacja może się odbywać za pomocą odpowiednio adresowanej pamięci, istnieją też takie, które z punktu widzenia architektury nie nadają się do sterowania za pomocą pełnej równoległej magistrali (np. urządzenia wyposażone wyłącznie w interfejs szeregowy). Dla tych urządzeń udostępniono szereg dodatkowych interfejsów wraz z bibliotekami programistycznymi w API. Wspierane przez TINI protokoły to m.in.:

- protokół szeregowy – zbudowany na interfejsie dwuprzewodowym z komunikacją asynchroniczną; interfejs bazuje na standardzie RS232-C, dlatego doskonale nadaje się do zintegrowania z siecią urządzeń starszej generacji,
- interfejs CAN – występuje w dwóch zdefiniowanych w ISO standardach; mikrokontroler TINI dostarcza dwa zintegrowane kontrolery CAN i zestaw bibliotek do ich obsługi napisanych w języku JAVA,
- magistrala 1-Wire – interfejs stworzony przez Dallas Semiconductors, który umożliwia stworzenie sieci złożonej z pojedynczych sensorów, włączników czy też elementów pamięci dzielących ten sam przewód zarówno do komunikacji jak i zasilania,
- niezależne porty mikrokontrolera – dwukierunkowe o poziomach TTL, mogą służyć do ogólnych zastosowań (np. sterowania diodami, przełącznikami, etc.).

2.3. Magistrala 1-Wire

Technologia 1-Wire łączy w sobie zalety pełnowartościowej sieci z minimalizmem i prostotą infrastrukturalną [3]. Każde z urządzeń podłączonych do magistrali 1-Wire posiada unikalny adres, pozwalający na identyfikację urządzenia i powiązanych z nim funkcjonalności, które przekładają się na konkretne listy dostępnych rozkazów. Ponieważ komponenty sieci dzielą ten sam przewodnik, konieczne okazało się zastosowanie mechanizmu *master-slave* aby umożliwić niezakłóconą komunikację. Z założenia wszystkie podpięte do linii urządzenia posiadają status s*lave*. Dodatkowo, każdy węzeł musi zostać wyposażony w układ m*aster*, który spełnia funkcję „zarządcy” w obszarze węzła (rys. 3).



Rys. 3. Organizacja sieci 1-Wire

Linia służąca do komunikacji wewnątrz magistrali pełni jednocześnie rolę zasilacza dla podpiętych urządzeń, zatem wszystkie elementy sieci mają otwarty dren i mogą jedynie zmienić stan linii na niski. Transmisja danych po magistrali może odbywać się w dwóch prędkościach (standardowej, ok. 16 kb/s, i przyśpieszonej, 144 kb/s).

Układ *Master* posiada funkcje resetu, zapisu logicznej jedynki i zera oraz odczytu danych z magistrali. W związku z brakiem dodatkowej linii synchronizującej, komunikacji po magistrali narzucono ścisłe warunki czasowe. Odczyt

i zapis z urządzeń odbywa się w tzw. szczelinach czasowych, inicjowanych przez układ *Master*.

Proces wymiany danych w tak zdefiniowanej infrastrukturze dzieli się na następujące fazy:

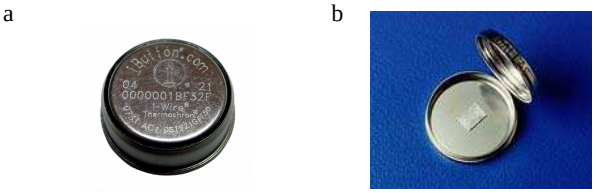
1. Faza inicjalizacji – układ master wysyła ciąg resetujący do wszystkich urządzeń znajdujących się na magistrali; po otrzymaniu tego rozkazu każde z urządzeń wysyła sygnał obecności.
2. Faza adresowania – układ master wysyła na magistralę adres urządzenia, z którym chce nawiązać komunikację; wszystkie pozostałe (niewybrane) urządzenia ustawiają się w stan wysokiej impedancji (jeśli adres urządzenia nie jest znany, master, metodą eliminacji kolejnych adresów, wyszukuje i identyfikuje elementy sieci).
3. Faza wymiany danych – na podstawie adresu (w nim tkwi informacja o rodzaju elementu) układ master wysyła odpowiednią sekwencję bitów z rozkazem do urządzenia. Po otrzymaniu instrukcji, układ dokonuje odpowiedniego przetwarzania i wysyła odpowiedź na magistralę.

2.4. Warstwa programowa

Software, na którym bazuje TINI, można podzielić na kod natywny (wykonywany bezpośrednio przez mikrokontroler) i API interpretowane jako kod bajtowy przez wirtualną maszynę Javy. Rozpoczynając pisanie aplikacji pod TINI, korzysta się z języka Java stosując instrukcje udostępnione właśnie w API, które w dalszym procesie tłumaczone są przez maszynę Javy do kodu natywnego i wykonywane przez mikrokontroler. Proces tłumaczenia obciąża w znacznym stopniu mikrokontroler, jeśli więc aplikacja narzuca pewne wymogi związane z czasem wykonywania, istnieje możliwość napisania pewnych fragmentów kodu w języku natywnym i ujęcia go w bibliotekę wywoływaną przez aplikację [1].

2.5. Chip iButton

Zastosowany w systemie iButton, podobnie jak 1-Wire został wypracowany w laboratoriach firmy Dallas.



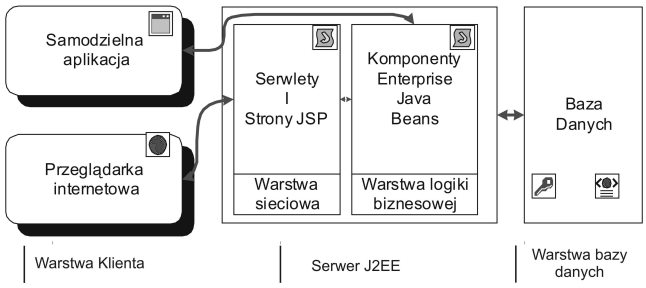
Rys. 4. iButton w wersji połączonej z czujnikiem temperatury: a - widoczny wybity numer seryjny, b –obudowa [4]

Z wyglądu element przypomina większą baterię zegarkową. Wewnątrz układu znajduje się chip elektroniczny wyposażony w unikalny 64-bitowy numer seryjny, dzięki czemu iButton może z powodzeniem spełniać funkcję identyfikatora i klucza elektronicznego. Producent postanowił zaimplementować w układziedodatkowe funkcjonalności – mamy więc iButton'y z zapisywalną pamięcią (także w wersjiszyfrowanej algorytmem SHA-1), modele wyposażone w zegar czasu rzeczywistego, czy nawet warianty zintegrowane z czujnikami temperatury (rys. 4) bądź wilgotności.

2.6. J2EE

J2EE to zbiór uporządkowanych standardów, które mogą wesprzeć proces budowy wielowarstwowych aplikacji rozproszonych, dzięki czemu można w większym stopniu skoncentrować wysiłek na wymaganiach funkcjonalnych systemu, odkładając na dalszy plan zawiłości infrastruktury - tę cześć pracy bierze na siebie Serwer Aplikacji J2EE [2]. Rosnące zapotrzebowanie na informacje, ich przetwarzanie i przechowywanie, wymusiło ewolucję standardowego wzorca aplikacji z modelu dwuwarstwowego (opartego zwykle na warstwie klienta i serwerze bazy danych) do modelu złożonego z większej liczby warstw. Powłokę klienta postanowiono „odchudzić", przenosząc logikę aplikacji wraz zadaniami wymagającymi sporej mocy obliczeniowej na serwer aplikacji. W efekcie tego procesu wzrosło bezpieczeństwo rozwiązań (aplikacje klienckie przestały odwoływać się bezpośrednio do baz danych, a do zdalnych obiektów na serwerze, które dokonywały transakcji na bazach danych), - zwiększyła się wydajność infrastruktury i zmniejszył się jej koszt dzięki skupieniu mocy obliczeniowej w jednym miejscu, - na skutek rozwoju Internetu (zwłaszcza w aspekcie nowoczesnych technologii WEB 2.0) aplikacje klienckie mogły z powodzeniem zostać przekształcone do dynamicznie tworzonych stron WWW, odwołujących się bezpośrednio do zasobów przedsiębiorstwa, gdzie aplikowany jest system.

Standard J2EE posiada zdefiniowane 4 podstawowe warstwy (rys.5):



Rys. 5. Schemat blokowy architektury J2EE

- warstwę klienta, która może występować jako aplikacja samodzielna, bądź bazować na przeglądarce internetowej,
- warstwę sieciową – złożoną z serwletów i stron wykonanych w technologii JSP (*Java Server Pages*),
- warstwę logiki biznesowej – zbudowanej z komponentów *Enterprise Java Beans*; w każdym z ziaren znajdują się konkretne usługi biznesowe, których domeną jest transakcyjność, trwałość, bezpieczeństwo i którym przyświeca idea wielodostępu, a programista zobowiązany jest jedynie do dostosowania wymogów implementacyjnych,
- warstwę zasobów informacyjnych – najczęściej w postaci serwera bazy danych.

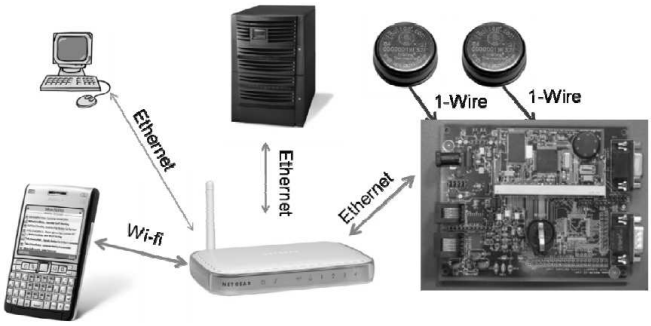
3. ARCHITEKTURA SYSTEMU

3.1. Warstwa fizyczna

Zakładając zgodność systemu ze standardem J2EE oraz przyjmując specyfikację platformy TINI opracowano model hardware'u aplikacji do akwizycji czasu pracy (rys. 6).

Do budowy prototypu zastosowano następujące elementy:

- zestaw czujników i-Button – do identyfikowania poszczególnych pracowników,
- platforma rozwojowa DS80C400-KIT do implementacji środowiska TINI,

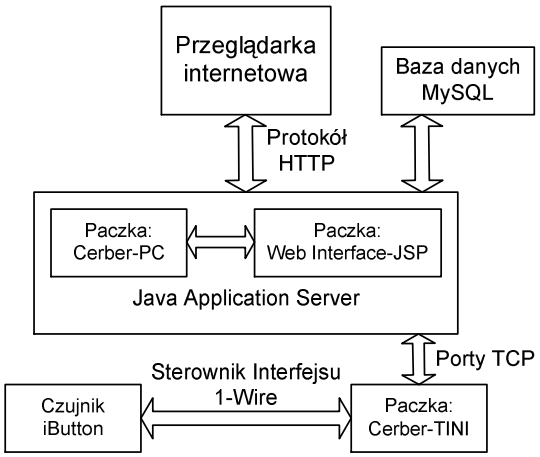


Rys. 6. Poglądowy schemat infrastruktury systemu

- fragment infrastruktury sieciowej – w systemie użyto router firmy NETGEAR wraz z okablowaniem,
- serwer - wykorzystano komputer PC z procesorem INTEL Core 2 Duo (2 GB pamięci operacyjnej, system operacyjny Windows Vista Business Edition),
- platforma klienta - komputer PC i telefon z interfejsem WIFI.

3.2. Warstwa logiczna

Architektura logiczna decyduje o sposobie komunikacji pomiędzy poszczególnymi komponentami systemu i interfejsami sieciowymi na poziomie logicznym. W zaprojektowanej aplikacji tymi elementami są (rys. 7):



Rys. 7. Schemat architektury logicznej systemu

- wirtualna maszyna JAVY – TINI,
- serwer aplikacyjny J2EE Glassfish,
- serwer bazy danych (MySQL).

Wirtualna maszyna JAVY w środowisku TINI obsługuje paczkę obiektów o nazwie Cerber-TINI, w skład której wchodzą mechanizmy komunikacji z czujnikiem iButton wspierane przez bibliotekę interfejsu 1-Wire. JVM zarządza również silnikiem aplikacji oraz algorytmami wymiany informacji przez odpowiednie porty protokołu TCP/IP. W podobny algorytm wyposażona jest jedna z klas w paczce Cerber-PC, znajdująca się na serwerze aplikacyjnym Glassfish na platformie PC. Paczka Cerber-PC umożliwia również łączność z bazą danych MySQL przy pomocy konektora JDBC. Kolejny moduł o nazwie Web-Interface implementuje warstwę użytkownika korzystając z technologii JSP.

4. PODSUMOWANIE

Efektywność programowania to w dużym stopniu wynik radzenie sobie ze złożonością, potęgowana zawsze przy rozbudowanej strukturze systemu, w którym aplikacja zostanie zaimplementowana. Nie można wyeliminować całkowicie tej zależności, natomiast można a nawet wręcz trzeba, próbować osłabić tę relację. Po zainstalowaniu w systemie platformy TINI można stwierdzić, że cel ten został w tym przypadku osiągnięty. Pisanie kodu jest łatwiejsze, program pisany jest szybciej, nabiera przejrzystości oraz nadaje się do wielokrotnej modyfikacji. Gdy połączyć ten wachlarz zalet z możliwościami architektury J2EE, otrzymuje się potężne narzędzie, które jest w stanie sprostać nawet najbardziej skomplikowanym wymaganiom aplikacji w systemie.

5. BIBLIOGRAFIA

1. Loomis D.: The TINI™ Specification and Developer's Guide, Addison-Wesley, 2001.
2. Adam Bochenek: Prosty przepis na J2EE, MIKOM, Warszawa, 2005.
3. Żurek S.: 1-wire, http://pl.wikipedia.org/wiki/1-Wire; 23.06.2006.
4. *iButton, data sheet,* Dallas Semiconductors, 2008.

INTEGRATION OF J2EE ARCHITECTURE WITH 1-WIRE INTERFACE IN THE SYSTEM FOR DATA IDENTIFICATION AND ACQUISITION

Key-words: system for data identification and acquisition, 1-wire measurement interface, TINI platform

In the design of the system integrating the J2EE architecture with 1-wire measurement interface the following features have been applied: TINI platform as a base system, iButton switches as a method of identification, internet network as a way of integration with o ther systems, J2EE as a model of logical architecture, and JSP technology, Sun Application Server and SQL. System enables adding and removal of a switch from data base, saving the interaction history and gives the mechanism of clearing an access for particular localization. The system was programmed by implementation of temperature measurement functionality, management of an access to working rooms, and also working time acquisition. The most programming works have been developed in the Integrated Development Environment. The platform enables a code writing in many programming languages. The advantage of the platform is its integration with Glassfish application server and MySQL data bases. A TINI platform programming has been realized in the several step modes, and program was executed in the text console extended by the TELNET protocol.