STUDIA INFORMATICA 2001
Volume 22 Number 3 (45)

Tadeusz PIETRASZEK, Filip ZAWADIAK
Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki

ZASTOSOWANIE STEROWNIKA Z MASZYNĄ WIRTUALNĄ JAVY™ W PROJEKCIE „INTELIGNETNY DOM"

Streszczenie. W artykule przedstawiono koncepcję zastosowania sterownika TINI z zaimplementowaną maszyną wirtualną Javy do celów sterowania urządzeniami w projekcie „Inteligentny dom". W tym celu dokonano porównania istniejących systemów sterowania i zaproponowano użycie magistrali OneWire. Wybór sterownika zapewnia możliwość wykorzystania dostępnych elementów, języka wysokiego poziomu do opisu procedur i podłączenia do Internetu.

THE APPLICATION OF MICROCONTROLLER WITH JAVA™ VIRTUAL MACHINE IN HOME AUTOMATION SYSTEM

Summary. The paper presents the idea of implementing home automation system based on TINI microcontroller with brief description of existing standards. The choice of microcontroller allows for using the wide choice of OneWire devices, possibility of writing control procedures in the high-level language. The Internet connection is also implemented.

1. Wstęp

Zagadnienie "inteligentnego domu", czyli automatycznego sterowania różnymi urządzeniami w domu jest znane od dawna i rozwiązywane przez wiele gotowych systemów. W poniższym artykule starano się przedstawić koncepcję opracowania sterownika - taniego i prostego w budowie, którego oprogramowanie oparte jest na ogólnodostępnych podzespołach z magistralą OneWire.

Mimo swojej prostoty, sterownik będzie miał szereg unikalnych cech, m.in.:

- program sterowania napisany w języku Java,

- możliwość podłączenia do sieci typu Ethernet z protokołem TCP/IP (np. dostęp przez WWW, WAP),
- możliwość dołączenia dodatkowych podzespołów - wyświetlacza LCD, układów I/O...,
- maksymalnie uproszczone elementy wykonawcze podłączane do magistrali - najprostsze zawierające tylko jeden element scalony w obudowie TO-92.

2. Istniejące standardy

2.1. X10

X10 jest obecnie najpopularniejszym i zarazem najprostszym z dostępnych standardów. Jest on bardzo rozpowszechniony w USA, a powoli staje się popularny również w Europie. Jego największą zaletą jest niski koszt urządzeń oraz użycie linii energetycznych do komunikacji. Niski koszt wynika z dostępności jednoukładowych rozwiązań, z których po dołączeniu niewielkiej liczby elementów zewnętrznych otrzymuje się gotowe urządzenie.

Użycie linii energetycznych jest bardzo wygodnym rozwiązaniem, jednakże zastosowany protokół powoduje poważne ograniczenia. Komunikacja z większością sterowników odbywa się bez potwierdzeń, z prędkością 50 bps. Wynika to z zastosowanej metody transmisji, w której bity danych nadawane są w momencie przejścia napięcia przez zero.

Sieć jest adresowana według 4-bitowego kodu domu oraz 4-bitowego numeru obwodu. Dodatkowo istnieją komendy typu "broadcast", jak np. "wyłącz/włącz wszystkie światła". Do poszczególnych 8-bitowych adresów może być podłączona dowolna liczba sterowników.

Powyższe właściwości powodują, że optymalnym zastosowaniem tego systemu jest sterowanie prostymi urządzeniami, najlepiej pod bezpośrednią kontrolą użytkownika.

2.2. P-NET

P-NET jest standardem opracowanym przez firmę Process-Data AG, który został uznany za standard europejski EN 50170 Vol.1. Jego głównym zastosowaniem jest łączenie w sieci inteligentnych sterowników.

Połączenia są realizowane wg normy RS-485 kablami połączonymi w pierścień (bez terminatorów). Prawo do nadawania jest ustalane przez "wirtualne" przekazywanie żetonu pomiędzy urządzeniami typu Master, w liczbie do 32 w pojedynczej pętli, natomiast wszystkich urządzeń może być 256. Dzięki zastosowanemu algorytmowi dostępu, sieć działa

bardzo sprawnie, pomimo stosunkowo niewysokiej prędkości transmisji wynoszącej 76800 bps.

Poszczególne pierścienie mogą być łączone ze sobą przez wieloportowe węzły typu Master. W przypadku rozbudowanych konfiguracji węzły używają "ścieżek" zawierających adresy wszystkich węzłów po drodze.

Protokół ma zdefiniowane wyższe warstwy, poszczególne właściwości sterowników są prezentowane w postaci "kanałów". Każdy kanał składa się z 16 "slotów", o z góry przypisanym znaczeniu dla każdego typu kanału.

Protokół ten jest dobrze i krótko udokumentowany, niewysokie są też opłaty licencyjne. Jedyną wadą jest podział na węzły typu Master i Slave, co uniemożliwia implementację usług powiadamiania.

2.3. EIB (European Installation Bus)

EIB jest standardem wspieranym przez większość firm europejskich, definiującym bardzo rozbudowany protokół, obsługiwany przez wiele urządzeń, głównie produkcji firmy Siemens. Z powodu komplikacji protokołu wymagana jest duża (jak np. w przypadku wyłącznika do światła) ilość pamięci RAM i ROM – typowe sterowniki mają 64 kB pamięci.

O stopniu skomplikowania może świadczyć podzielenie na warstwy OSI i to nie tylko "formalne", ale także widoczne w komendach oraz liczba nagłówków. Protokół implementuje usługi połączeniowe oraz bezpołączeniowe.

Sieć EIB może być połączona na wiele sposobów, w tym po skrętce oraz po sieci energetycznej. W tym drugim przypadku komunikacja odbywa się z prędkością 9600 bps.

Protokół w doskonały sposób definiuje wyższe warstwy. Każde urządzenie jest "widzialne" w postaci drzewa. W urządzeniach występują obiekty, które z kolei mają różne właściwości. Każda właściwość ma określony typ, może też być tablicą. Wszystkie informacje na temat urządzenia można uzyskać od samego urządzenia, co umożliwia stosowanie i konfigurację nieznanych urządzeń natychmiast po włączeniu do sieci.

Szerszemu wykorzystaniu nie sprzyjają wysokie opłaty licencyjne oraz duże wymagania sprzętowe.

2.4. LonWorks

LonWorks jest otwartm protokołem opracowanym przez firmę Echelon określonym przez standard ANSI/EIA 709.1-A-1999. Może pracować z prędkościami do 1.25 Mbps. Sieć może dzielić się na wiele podsieci, mogą też występować grupy węzłów. Co ciekawe, komunikacja w grupach także odbywa się z potwierdzeniami.

Jako węzłów sieci używa się głównie układów scalonych o nazwie "Neuron". Mają one interesującą budowę: do realizacji poszczególnych działań ma trzy "wirtualne" procesory, zrealizowane przez sprzętowe podmienianie pliku rejestrów co jeden cykl rozkazowy.

Na najwyższym poziomie komunikacja odbywa się przez użycie zmiennych "sieciowych". Istnieje też, oczywiście, możliwość wykonywania operacji typu wysyłania i odbierania pakietów.

Implementacja protokołu na "zwykłych" procesorach wymaga użycia szybkich układów lub bardzo silnego spowolnienia sieci. Dodatkowo utrudniona jest realizacja warstwy fizycznej z powodu nietypowego interfejsu.

2.5. CAN

Standard CAN i zdefiniowany przez niego protokół jest najczęściej wykorzystywany w przemyśle motoryzacyjnym, dla którego został zaprojektowany. Standard ten opiera się na szybkim przesyłaniu komunikatów, których interpretacja zależna jest od danego węzła.

Dla protokołu CAN zdefiniowano wyższe warstwy, takie jak CANopen oraz DeviceNet. Nie są one jednak publicznie dostępne.

3. Sieć komunikacyjna

W koncepcjach sterowania najogólniej możemy wyróżnić 2 typy rozwiązań:

- centralny sterownik z kartami I/O i doprowadzonymi przewodami urządzeń
- sterownik z wyprowadzoną magistralą, do której podłączane są moduły realizujące określone funkcje.

Pierwsze rozwiązanie może zostać zrealizowane przy wykorzystaniu sterowników przemysłowych z odpowiednimi kartami I/O, co jest rozwiązaniem dobrym, ponieważ sterowniki te zostały zaprojektowane z myślą o tego typu sterowaniu. Wadą tej koncepcji jest cena sterowników przemysłowych oraz konieczność prowadzenia dużej liczby przewodów, co jeszcze bardziej powiększa koszty i komplikuje instalację.

Drugie rozwiązanie polega na poprowadzeniu magistrali, do której podłączane są częściowo autonomiczne moduły komunikujące się ze sobą. W większości rozwiązań występuje moduł nadzorujący pracę pozostałych modułów, który zawiera główny program sterowania. Przy wyborze rozwiązań sprzętowych i protokołu komunikacyjnego należy zwrócić uwagę na:

1) własności magistrali, a w tym:
- możliwość podłączenia wielu urządzeń,

- odporność na zakłócenia,
- ograniczenia np. długości kabla,
- medium transmisyjne,
- złożoność urządzeń odbiorczych,

2) własności protokołu, a w tym:
- adresację urządzeń (nadawanie adresów, analiza stanu sieci),
- rozwiązanie problemów arbitrażu i kolizji,
- korekcję i detekcję błędów,
- prostotę implementacji protokołu.

Przy wyborze koncepcji sieci najbardziej atrakcyjna wydaje się możliwość wykorzystania do przesyłu danych sieci energetycznej 220 V. W rezultacie likwidujemy konieczność instalacji dodatkowych przewodów, lecz jednak pojawiają się następujące problemy:
- konieczność wbudowania zasilacza do każdego modułu włączanego do sieci,
- znacznie bardziej skomplikowana budowa transiwera - dostosowanego do pracy w sieci,
- odporność na zakłócenia występujące w sieci energetycznej (np. przy dołączeniu układów tyrystorowych czy silników komutatorowych),
- możliwość interferencji dwóch lub więcej instalacji sieci znajdujących się w pobliżu (np. 2 sąsiednie mieszkania w bloku).

Podsumowując, takie rozwiązanie jest możliwe, jednak kosztowne i trudne do zastosowania. Dążąc do maksymalnego ograniczenia kosztów i prostoty budowy, w rozwiązaniu przedstawionym poniżej autorzy zdecydowali się na wykorzystanie sieci MicroLAN (zwaną także 1WireNET), opartej na magistrali OneWire (OW) opracowanej przez firmę Dallas.

3.1. Sieć MicroLAN

Sieć MicroLAN jest siecią dwuprzewodową, w której można wyróżnić linię zerową (masy) oraz multipleksowaną linię danych i zasilania. Okazuje się, że część podłączanych urządzeń nie wymaga dodatkowego zasilania (działając przez wykorzystanie tzw. *parasite power*) i podłączana jest tylko za pomocą dwóch przewodów.

Sieć umożliwia poprawną współpracę wielu dołączonych do niej urządzeń na zasadzie *master-slave*. Każde urządzenie identyfikowane jest przez swój unikalny adres, przy czym możliwa jest dynamiczna rekonfiguracja sieci w czasie jej działania (dołączenie nowego urządzenia, odłączenie starego).

Maksymalna długość sieci to 300 m, jednak ze względu na pojemności można podzielić sieć na wybierane segmenty, z których jednocześnie tylko jeden jest aktywny. Rozwiązanie takie zostało uwzględnione na etapie założeń w przedstawionym projekcie.

Ostatecznie zdecydowano, że magistrala doprowadzona do każdego urządzenia składa się z 4 przewodów:

- masy sygnałowej (OW),
- linii danych (OW),
- zasilania 24 V,
- masy zasilania.

Dodatkowe zasilanie jest wykorzystywane do zasilania układów podłączanych do magistrali oraz elementów wykonawczych (np. przekaźniki). Przewody te mogą zostać poprowadzone 4-żyłowym kablem telefonicznym lub komputerową skrętką.

Na podstawie powyższych rozważań nie widać wyraźnej wyższości sieci MicroLAN nad np. siecią przemysłową Modbus, zbudowaną na RS-485. Rzeczywiście, pod wieloma względami sieć firmy Dallas jest gorsza od konkurencyjnych produktów, jednak posiada jedną wyróżniającą cechę - istnieje olbrzymia ilość prostych i tanich układów scalonych z zaimplementowaną obsługą magistrali OW. Układy te są tak pomyślane, że eliminują konieczność wykorzystania dodatkowych mikroprocesorów w większości modułów wykonawczych i ich oprogramowania.

3.2. Układy scalone z interfejsem OneWire

Dokonując przeglądu układów scalonych z interfejsem OW (nazywanych dalej układami), znajdujemy kilka wręcz idealnych do wykorzystania w systemie sterowania. Wszystkie układy mają następujące wspólne cechy:

- zaprogramowany przez producenta unikalny adres, zawierający rodzinę identyfikującą typ urządzenia,
- implementację protokołu umożliwiającego:
  - współdziałanie z innymi urządzeniami z magistralą OW,
  - przeszukanie sieci i wygenerowanie listy wszystkich dostępnych urządzeń, wybranie jednego urządzenia i przeczytanie jego stanu,
  - zmianę stanu wybranego urządzenia,
  - wyszukiwanie urządzeń spełniających określone kryterium (np. typ urządzenia, urządzenia, dla których wystąpiło zaprogramowane zdarzenie),
  - detekcję błędów transmisji (algorytm CRC16).

Poniżej opisano powyższe układy i ich zastosowanie w układzie sterownika.

3.2.1. DS2406

Układ jest adresowalnym układem programowalnego I/O zawierających 2 PIO z magistralę OW. Jest on produkowany w 2 wersjach: 3-pin TO-92, oraz 6-pin TSOC, przy czym

wersja 3-pin posiada tylko jedno PIO i nie ma możliwości podłączenia zewnętrznego zasilania. Układ posiada także 128 bajtów pamięci EEPROM, podzielonej na niezależnie zmieniane strony, w sposób umożliwiający zapisanie nowych danych przez przemapowanie pustej strony w miejsce już zaprogramowanej. Pamięć ta może służyć do przechowywania np. tekstowego opisu sterownika.

Wyprowadzenia PIO układu DS2406 są wyjściami typu *open drain* oraz wejściami posiadającymi dodatkowy przerzutnik *activity latch* umożliwiający zapamiętanie zmiany stanu układu wejścia pomiędzy odczytami.

Układ można skonfigurować w taki sposób, że określona zmiana na jego wejściu powoduje ustawienie flagi zdarzenia istotnej przy wyszukiwaniu warunkowym. Dzięki tej komendzie (`ConditionalSearchROM`), możliwe jest szybkie znalezienie w sieci urządzeń, dla których nastąpiła zmiana stanu wejść.

3.2.2. DS2450

Układ jest adresowalnym poczwórnym 8-bitowym przetwornikiem z magistralą OW w 8-pin obudowie SOIC, posiadającym możliwość ustawienia zakresu napięcia wejściowego oraz ustawienia progów alarmów (oddzielnie dla każdego wejścia), których przekroczenie jest uwzględniane przy wyszukiwaniu warunkowym.

W ten sposób można szybko znaleźć w sieci urządzenia, których wartości sygnałów wejściowych znajdują się poza zakresem.

Niewykorzystane wejścia analogowe można skonfigurować i wykorzystać jako wyjścia cyfrowe typu *open-drain*.

3.2.3. DS2890

Układ jest adresowalnym cyfrowym potencjometrem z magistralą OW. Jest produkowany w 2 wersjach: 3-pin TO-92 oraz 6-pin TSOC. Potencjometr cyfrowy jest typu liniowego i umożliwia ustawienie rezystancji z przedziału 0-100 kΩ na jedną z 256 wartości.

Podobnie jak w analogicznych elementach innych producentów, potencjometr zbudowany jest z drabinki rezystorów, z dołączanym środkowym wyprowadzeniem potencjometru. Potencjometr może pracować przy napięciu nie przekraczającym 11 V.

Stan potencjometru jest tracony w przypadku zaniku zasilania i ustawiana jest wartość 0. W takim jednak przypadku układ odpowiada na `ConditionalSearchROM`, dzięki czemu możliwe jest szybkie wykrycie tego zjawiska.

3.2.4. DS2423

Jest to układ z podtrzymywaną bateryjnie pamięcią RAM i czterema 32-bitowymi licznikami, z czego dwa wyzwalane są zewnętrznymi wejściami, a pozostałe dwa zliczają

zapisy do stron pamięci. Mieści się on w obudowie typu TSOC 6-pin, jest podłączony do magistrali OW i umożliwia odczyt/zapis pamięci RAM oraz wyłącznie odczyt liczników.

3.2.5. DS2480B

Układ jest driverem interfejsu OneWire podłączanym do portu szeregowego procesora. Jest zasilany napięciem 5 V, jednak umożliwia dołączenie dodatkowego napięcia programującego dla pamięci EPROM – 12 V. Dzięki zaimplementowanej maszynie stanów magistrali OW, układ umożliwia odciążenie głównego procesora sterownika od zmiany stanów linii magistrali w ściśle określonych momentach czasowych i śledzenia tych zmian.

3.2.6. DS2409

Adresowany układ umożliwiający podzielenie magistrali OW na niezależne segmenty. Działanie układu polega na tym, że można wysłać mu komendę powodującą odcięcie lub przyłączenie dalszej części magistrali znajdującej się po drugiej stronie układu. Układ może przełączać dwie magistrale (oznaczane jako główna i dodatkowa).

Zaletą układu w stosunku do zwykłego przełącznika jest to, że po odcięciu, magistrala jest zasilana, przez co nie następuje utrata danych z pamięciach RAM pozostałych układów. Dzięki wykorzystaniu DS2409 możemy zmniejszyć szkodliwe pojemności związane z długością magistrali, jednak podział sieci na segmenty komplikuje procedurę przeszukiwania sieci.

3.2.7. DS18S20

Adresowany układ termometru - termostatu z magistralą OW. Układ produkowany w obudowie typu TO-92 (3-pin) lub typu SOIC (6-pin), umożliwia odczyt temperatury układu w zakresie -10°C - +85°C z dokładnością ±0.5°C. Jeżeli taka dokładność nie jest konieczna, można skorzystać z układu DS1822, który przy tej samej funkcjonalności daje dokładność ±2°C. Układ umożliwia ustawienie w pamięci EEPROM alarmów, których przekroczenie powoduje odpowiedź układu na komendę `ConditionalSearchROM`, przez co możliwa jest szybka reakcja na zmianę temperatury.

3.2.8. DS1990A

Układ iButton z magistralą OW w obudowie MICROCAN. Najprostszy przedstawiciel rodziny iButton firmy Dallas, realizujący wyłącznie funkcję zgłaszania swojej obecności na magistrali i odpowiedzi na swój numer seryjny. Układ służy jako elektroniczny klucz w systemie sterowania.

Ze względu na to, że jest to układ dołączany w trakcie pracy systemu do działającej magistrali, interfejs powinien być tak zbudowany, żeby np. zwarcie styków nie spowodowało

zaprzestania pracy całego systemu. Najodpowiedniejsze wydaje się tu podłączenie urządzeń dołączanych do drugiej, niezależnej magistrali OW.

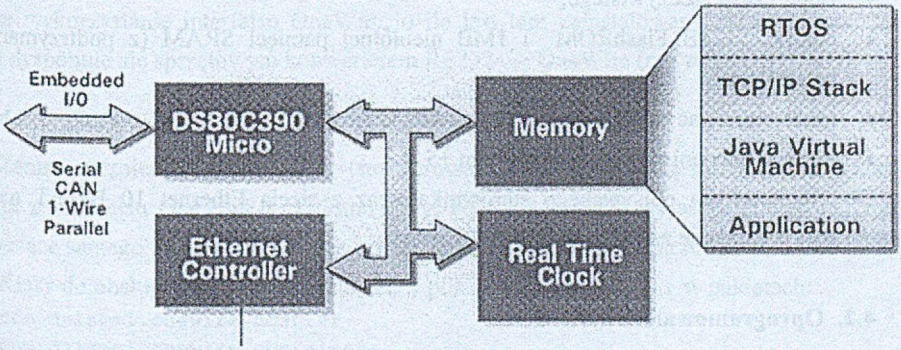
4. Jednostka sterująca

Zaproponowany system sterowania składa się z magistrali OW, do której podłączone są moduły funkcjonalne oparte na scharakteryzowanych powyżej elementach, oraz z centralnej jednostki sterującej zajmującej się kontrolą magistrali OW, dodatkowych I/O oraz realizacją programu sterowania.

Jest oczywiste, że funkcję tę powinien spełniać wydajny system procesorowy. Możliwe byłoby tutaj wykorzystanie płyty komputera IBM-PC, jednak to rozwiązanie jest niekorzystne ze względu na duże rozmiary i skomplikowaną infrastrukturę (zasilacz, karty I/O, karta sieciowa, twardy dysk lub inna nieulotna pamięć, z której uruchamia się system operacyjny). Zaproponowano wykorzystanie platformy TINI, która jest pozbawiona tych wad.

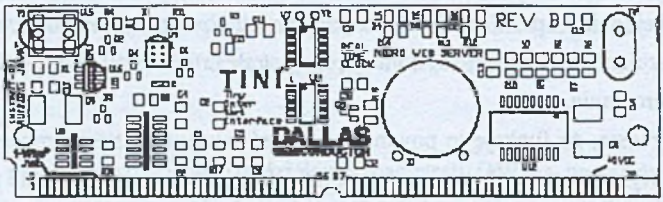
4.1. Sterownik TINI

TINI (Tiny InterNet Interface) jest platformą opracowaną przez firmę Dallas Inc. specjalnie do uruchamiania i projektowania różnorodnych systemów procesorowych podłączanych do sieci z protokołem TCP/IP. Platforma TINI oparta jest na mikrokontrolerze DS80C390 (zgodny z 8052) i systemie operacyjnym z maszyną wirtualną Javy™ (Java VM). Wersja minimalna, oprócz procesora, musi zawierać pamięć flash EEPROM i statyczną RAM. Mikrokontroler zawiera sterowniki do portów szeregowych, magistrali CAN oraz magistrali OW, obsługiwane przez system operacyjny, który może również obsługiwać kontroler sieci Ethernet.



Rys. 1. Struktura sterownika TINI
Fig. 1. The structure of TINI microcontroller

Programy uruchamiane na platformie TINI mogą być zarówno kodem wykonywalnym procesora, jak i skompilowanym programem napisanym w języku Java. Platforma TINI niekoniecznie musi być związana z konkretną konfiguracją sprzętową, jednak dla aplikacji produkowanych w mniejszej skali lub w procesie tworzenia i uruchamiania programu można skorzystać z gotowej płytki sterownika o nazwie DSTINI1, wyprodukowanej przez firmę Dallas Inc.



Rys. 2. Widok płytki ze sterownikiem (DSTINI1)
Fig. 2. DSTINI1 board overview

Jest ona w pełni funkcjonalnym komputerem o wymiarach (31.8 mm x 102.9 mm) mieszczącym się na standardowej płytce pamięci SIMM. Charakteryzuje się ona następującymi cechami:

- jest sprawdzonym środowiskiem uruchomieniowym TINI,
- ma zintegrowany kontroler Ethernetu 10 Base-T,
- umożliwia dostęp do portów I/O procesora przez API Javy,
- ma 2 porty szeregowe (jeden w standardzie RS232, drugi w standardzie TTL),
- ma wyprowadzoną podwójną magistralę CAN,
- wyprowadzone I/O procesora z możliwością dołączenia dodatkowych elementów - np. wyświetlacz, kontroler portu równoległego,
- zegar czasu rzeczywistego,
- pamięć 512kB FlashROM i 1MB nieulotnej pamięci SRAM (z podtrzymaniem bateryjnym),
- może pracować w zakresie temperatur -20°C do +70°C ,
- jest zasilany pojedynczym napięciem +5 V.

Do uruchomienia opisywanego sterownika wraz z siecią Ethernet 10 Base-T nie są praktycznie wymagane żadne dodatkowe elementy.

4.2. Oprogramowanie

Jak już wspomniano, w sterowniku TINI działa system operacyjny organizujący przełączanie zadań, system plików i zarządzający operacjami I/O. System posiada harmonogram

bezpriorytetowy, w którym zadania przełączane są co 8 ms. Wszystkie zadania systemu za wyjątkiem *garbage collector* są zadaniami Javy.

4.2.1. Uruchamianie

System operacyjny jest wgrywany do kontrolera przez port szeregowy za pomocą programu loadrea. Program loadera jest także napisany w Javie, do jego uruchomienia potrzebny jest pakiet Java Development Kit (JDK) w wersji 1.3 wraz z dodatkowym pakietem `javax.comm`.

Po wgraniu systemu operacyjnego można opcjonalnie załadować program `slush`, który jest Uniksopodobnym shellem, umożliwiającym podstawowe operacje. Oczywiście, w przypadku bardziej zaawansowanych programów, `slush` może zostać zmodyfikowany na potrzeby użytkownika, a nawet całkowicie zastąpiony własną aplikacją.

Po załadowaniu programu `slush`, można skonfigurować sieć (statyczne dane lub otrzymywane z serwera DHCP), a następnie można wszystkie pozostałe operacje wykonywać wyłącznie za pomocą sieci.

Program `slush` zapewnia serwery `telnet` i `ftp`, które umożliwiają odpowiednio zdalną pracę na TINI oraz ładowanie programów.

4.2.2. Kompilacja i uruchamianie programów

Własne programy kompiluje się na komputerze np. IBM-PC, z zainstalowanym JDK oraz klasami zaimplementowanymi na TINI (plik `tiniclasses.jar`). Po skompilowaniu plik `.class` należy przekonwertować do pliku `.tini` za pomocą specjalnie dostarczonego narzędzia. Tak przygotowany program można załadować do sterownika TINI za pomocą programu FTP, a następnie uruchomić na TINI za pomocą polecenia `java <nazwa programu> [&]`

Jeżeli nie korzysta się ze sprzętu TINI (z wyjątkiem interfejsu OneWire), programy można także uruchamiać na platformie, na której jest zainstalowany JDK (np. PC). Możliwe jest także wykorzystanie interfejsu OneWire, o ile jest tam zainstalowany pakiet `javax.comm` oraz dysponuje się sprzętowym konwerterem RS232<-> OneWire (nr części DS9097U).

4.2.3. Korzystanie z klas obsługujących OneWire

Zanim zaczniemy analizować system sterowania, przyjrzyjmy się możliwościom korzystania z interfejsu OneWire z poziomu Javy. Pozwoli to także wyjaśnić pewne szczegóły dotyczące samego interfejsu OneWire i zrozumieć sposób działania sterownika.

Klasy do obsługi OneWire znajdują się w pliku `tiniclasses.jar` w pakietach:

```
com.dalsemi.onewire.adapter
com.dalsemi.onewire.container
com.dalsemi.onewire.utils
```

Podstawowym obiektem jest adapter (`DSPortAdapter`), który odpowiada fizycznemu urządzeniu adaptera podłączonemu do komputera. Adapter (może być ich więcej niż jeden), jest obiektem zwracanym przez obiekt `OneWireAccessProvider`. Następnie należy uzyskać wyłączność na obiekt adaptera (metoda `beginExclusive()`) i można wyszukiwać urządzenia podłączone do magistrali.

Wyszukiwanie urządzeń na magistrali może dotyczyć wszystkich urządzeń lub tylko urządzeń spełniających określone warunki (np. typ urządzenia lub jego stan). Ta ostatnia operacja jest użyteczna przy wyszukiwaniu urządzeń, które wymagają określonych akcji układu sterowania (np. wzrost temperatury powyżej progu, załączenie włącznika).

Po znalezieniu urządzenia możemy albo tylko odczytać jego adres (metodą `getAddesss()`), albo stworzyć obiekt związany ze znalezionym urządzeniem. Obiekty te są zwracane jako typu `OneWireContainer` - jest to typ podstawowy posiadający metody wspólne dla wszystkich urządzeń z magistralą OneWire - jednak są skonstruowane jako obiekty pochodne dla właściwych urządzeń `OneWireContainerXX` (`XX` jest rodziną urządzenia).

W celu wykorzystania metod specyficznych dla konkretnych urządzeń, należy sprawdzić, jaki jest właściwy typ urządzenia (`instanceof`), a następnie je przerzutować na odpowiedni obiekt `OneWireContainerXX`. I tak opisanym powyżej układom odpowiadają następujące klasy:

Typ układu	Obiekt podstawowy	Implementowane interfejsy
DS2406	`OneWireContainer12`	`OneWireSensor, SwitchContainer`
DS2450	`OneWireContainer20`	`OneWireSensor, ADContainer`
DS2890	`OneWireContainer2C`	`OneWireSensor, PotentiometerContainer`
DS2423	`OneWireContainer1D`	
DS2409	`OneWireContainer1F`	`OneWireSensor, SwitchContainer`
DS18S20	`OneWireContainer10`	`OneWireSensor, TemperatureContainer`
DS1990A	`OneWireContainer01`	

Widoczne jest, że większość klas rozszerza interfejsy umożliwiające korzystanie z funkcji danego obiektu bez znajomości jego typu (np. `SwitchContainer` dla rodzin 12 i 1F).

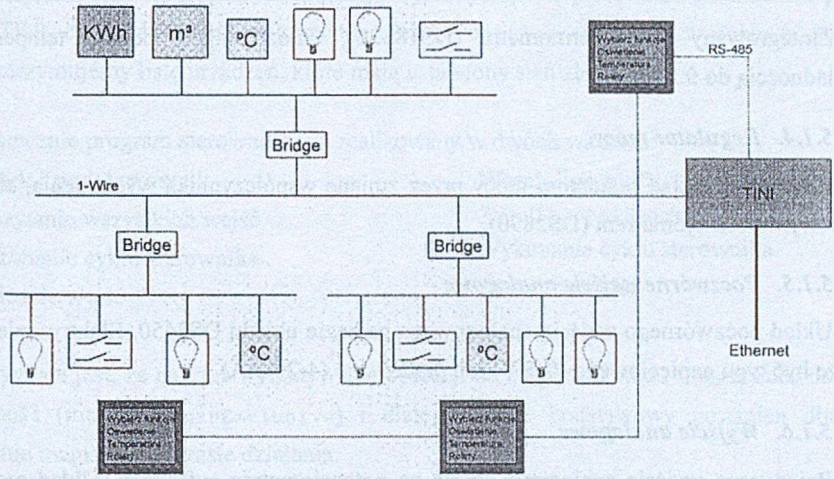
Większość funkcji wyżej wymienionych układów realizuje się przez zmianę słowa statutowego. Słowo statutowe sterownika jest odczytywane metodą `readDevice()`, a po ewentualnej zmianie metodami obiektu zapisywane metodą `writeDevice()`.

Dodatkowo, istnieją funkcje, które realizowane są przez dodatkowy odczyt/zapis sterownika za pomocą odpowiednich funkcji i w takim przypadku zmiany ustawień mają efekt natychmiastowy (np. odczyt wartości przetwornika A/D). Działanie takie mogłoby mieć

niekorzystny wpływ na system sterowania (np. kolejne odwołania do wartości wejścia w programie mogą dać różne wyniki) i z tego powodu wprowadzono dodatkowy obiekt pośredni pozbawiony tych właściwości.

5. Koncepcja systemu sterowania

Przykładowy system sterowania przedstawiono na poniższym rysunku. Składa się on ze sterownika TINI, modułów wykonawczych oraz paneli sterujących.



Rys. 3. Stuktura sieci opartej na sterowniku TINI w systemie sterowania domem
Fig. 3. The structure of TINI based home automation system

5.1. Typy modułów w układzie sterownika i odpowiadające im obiekty

Poniżej proponujemy listę podstawowych modułów możliwych do wykorzystania w systemie sterowania domem. Każdemu modułowi odpowiada oddzielna klasa w systemie.

5.1.1. Przekaźnik - wyjście cyfrowe

Podstawowy moduł wykonawczy oparty na układzie DS2406 (DS2450) umożliwiający sterowanie dwoma (czterema) przekaźnikami, sterowanymi przez klucze tranzystorowe.

Zaimplementowane metody:

- dołączenie obiektu `OneWireContainer12` (konstruktor),
- odczytanie stanu,

- zapisanie stanu i warunkowe zapisanie stanu (po zmianie),
- odczytanie stanu i-go przekaźnika,
- zmiana stanu i-go przekaźnika,
- odczytanie kodu błędu magistrali OW.

5.1.2. Przełącznik - wejścia cyfrowe

Podstawowy moduł oparty na układzie DS2406 (DS2450) umożliwiający odczyt dwóch (czterech) wejść cyfrowych. Wejścia mogą być wykonane w standardzie 5 V lub sieciowym ~220 V.

5.1.3. Termometr

Zintegrowany układ termometru (DS18S20), umożliwiający pomiar temperatury z dokładnością do 0.5°C.

5.1.4. Regulator mocy

Analogowy układ regulatora mocy przez zmianę współczynnika wypełnienia, sterowany cyfrowym potencjometrem (DS2890).

5.1.5. Poczwórne wejście analogowe

Układ poczwórnego wejścia analogowego na bazie układu DS2450. Elektrycznie wejście może być typu napięciowego (0-5 V) lub prądowego (4-20 mA).

5.1.6. Wyjście analogowe

Pojedyncze wyjście analogowe oparte na potencjometrze cyfrowym. Układ może mieć wyjścia typu napięciowego (0-5 V), prądowego (4-20 mA), oporowego (0-100 kΩ).

5.1.7. Licznik impulsów

Układ zliczający impulsy (pochodzące np. z przepływomierza czy impulsatora) oparty na układzie DS2423.

5.1.8. Bridge

Układ dzielący magistralę na segmenty. Ma znaczenie przy fizycznym podłączeniu modułów, jednak nie ma swojego odpowiednika w programie użytkownika.

5.1.9. Panel sterujący z wyświetlaczem LCD

Panel sterujący z przełącznikami i wyświetlaczem LCD do ręcznego sterowania system. Sterowany niezależnym mikrokontrolerem i podłączany do TINI przez magistralę RS-485.

5.2. Program sterownika

W klasycznych systemach sterowania, np. sterownikach przemysłowych, cykl sterownika składa się z następujących operacji:

- odczytanie stanu wejść do pamięci,
- wykonanie programu,
- uaktualnienie wyjść programu.

Projektowany sterownik działa na podobnej zasadzie jak sterowniki przemysłowe, jednak ze względu na małą szybkość działania magistrali OW pojawiłby się problem długiego czasu reakcji układu na zmianę wejść. Dotyczy to sytuacji gdy, do magistrali mamy podłączonych wiele urządzeń. Aby rozwiązać ten problem, wykorzystuje się przerwania generowane przez układy TINI, czyli odpowiedź na komendę `ConditionalSearchROM`. Po wydaniu takiej komendy otrzymujemy listę urządzeń, które mają ustawiony stan alarmu.

Ostatecznie program sterownika jest realizowany w dwóch wątkach:

Wątek "podstawowy":	**Wątek "szybki":**
Odczytanie wszystkich wejść	`ConditionalSearchROM`
Wykonanie cyklu sterownika	Wykonanie cyklu sterownika
Zapisanie wyjść	Zapisanie wyjść

Oczywiste jest, że na czas wykonywania operacji na magistrali wątki muszą mieć do niej wyłączność (metoda `beginExclusive`) i dlatego wątek podstawowy powinien dbać o zwalnianie magistrali w czasie działania.

Innym problemem wymagającym omówienia jest wykorzystanie modułów typu bridge w sieci. Moduły te zmieniają wyłącznie architekturę fizyczną układu sterownika - nie wpływają na architekturę logiczną widzianą z punktu widzenia programu sterowania, jednak ich struktura musi być uwzględniona przez procedury odczytu i zapisu stanów wszystkich urządzeń w sieci. Założyliśmy, że struktura sieci jest dwupoziomowa, tj. do urządzenia typu bridge nie może być podłączone inne urządzenie typu bridge, a jedynie moduły wykonawcze. Obiekt sterownika posiada listę wszystkich urządzeń podłączonych do magistrali, przy czym każdy *bridge* posiada listę modułów podłączonych do niego. Wówczas sekwencyjne odpytywanie modułów przebiega w następujący sposób:

```
Dla każdego gateway na magistrali
  Załącz i-ty gateway
  Odczytaj(zapis) stan wszystkich urządzeń na magistrali
  Wyłącz i-ty gateway
```

Natomiast wyszukiwanie zmienionych urządzeń:

```
Dla każdego gateway na magistrali
```

```
Załącz i-ty gatway
Dokonaj warunkowego wyszukiwania urządzeń (ConditionalSearchROM)
Wyłącz i-ty gateway
```

5.3. Projektowanie programów sterowania

Zakładamy, że napisanie programu do sterowania polega na napisaniu obiektu, rozszerzającego podstawowy obiekt sterownika. Powinien on zawierać następujące składowe:

- obiekty reprezentujące wszystkie moduły w sieci,
- obiekty typu gateway z listą modułów w ich gałęziach,
- dodatkowe zmienne wykorzystywane w pętli programu,
- procedurę programu sterownika operującą na powyższych obiektach.

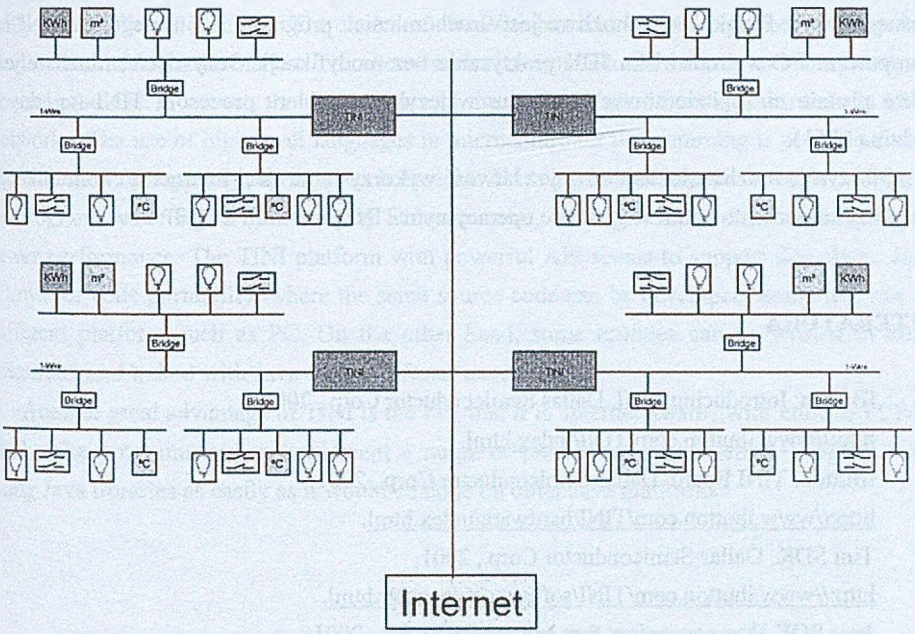
Wszystkie moduły identyfikowane są przez numer seryjny elementów OneWire, przy czym sam sterownik może mieć funkcję wyszukiwania elementów na magistrali i generowania w ten sposób listy obiektów potrzebnych powyżej. Procedura programu sterownika może zawierać dowolne konstrukcje w Javie, z czego wynika duża prostota zapisywania algorytmu sterowania. Docelowo można zaprojektować środowisko wizualne, w którym projektuje się algorytm sterownika w postaci graficznej, a następnie generujące z niego program w Javie.

Po zapisaniu programu sterownika, program powinien zostać skompilowany za pomocą kompilatora Javy postaci wykonywalnej, a następnie załadowany na sterownik i uruchomiony.

5.4. TINI w sieci Internet

Dzięki wbudowanemu protokołowi TCP/IP podłączenie sterownika do sieci komputerowej jest bardzo proste. Umożliwia to m.in.:

- podłączenie większej liczby sterowników TINI do sterowania dużymi obiektami (duża liczba modułów lub duży zasięg),
- przeglądanie i zmianę stanu systemu przez uprawnionych użytkowników z sieci Internet,
- przeglądanie i zmianę stanu przez uprawnionych użytkowników z telefonów komórkowych z wykorzystaniem technologii WAP.



Rys. 4. Podłączenie sieci sterowników TINI do sieci Internet
Fig. 4. The connection of many TINI-based networks to the Internet

6. Wnioski

W powyższym artykule przedstawiono oryginalny projekt uniwersalnego sterownika bazującego na magistrali OneWire oraz nowoczesnym języku programowania Java. Magistralę OW wybrano ze względu na dużą liczbę gotowych elementów, a co za tym idzie, maksymalną prostotę konstrukcji modułów sterownika i niską ich cenę. Do niewątpliwych wad tego rozwiązania można zaliczyć niewielką prędkość transmisji.

Dzięki szybkiemu wzrostowi mocy obliczeniowych procesorów jednoukładowych do ich programowania możliwe staje się wykorzystanie języków wyższego poziomu. Pomimo mniejszej wydajności programu, powstaje on w znacznie krótszym czasie i niższym kosztem, jest prostszy do uruchomienia i zawiera mniej błędów. Niejednokrotnie programy napisane w języku wyższego poziomu mają znacznie większą funkcjonalność niż programy napisane w języku asemblera.

Konstrukcja komputera TINI jest potwierdzeniem tej tendencji, przy czym konstruktorzy wykorzystali Javę - nowoczesny i popularny język programowania. Java posiada szereg mechanizmów wyróżniających ją wśród języków programowania - jedną z nich jest pełna prze-

nośność kodu. Dzięki temu możliwe jest uruchomienie programu napisanego na TINI na komputerze PC w środowisku JDK praktycznie bez modyfikacji. Oczywiście, możliwe jest także pisanie niskopoziomowych procedur w języku asemblera procesora TINI (zgodny z rodziną 8051).

Olbrzymie możliwości daje również łatwość wykorzystania sieci Internet do komunikacji, dzięki zaimplementowaniu w systemie operacyjnym TINI protokołu TCP/IP.

LITERATURA

1. iButton: Introducing TINI. Dallas Semiconductor Corp., 2001, http://www.ibutton.com/TINI/index.html.
2. iButton: TINI Board. Dallas Semiconductor Corp., 2001, http://www.ibutton.com/TINI/hardware/index.html.
3. Tini SDK. Dallas Semiconductor Corp., 2001, http://www.ibutton.com/TINI/software/soft_order.html.
4. Java SDK Documentation. Sun Microsystems Inc., 2001, http://java.sun.com/j2se/1.3/docs.html.
5. 1-Wire Chips Selection Table. Dallas Semiconductor Corp., 2001, http://www.dalsemi.com/products/autoinfo/1wire.html.
6. 1-Write API. Dallas Semiconductor Corp., 2001, ftp://ftp.dalsemi.com/pub/auto_id/public/owapi0_01.tgz.
7. The "Intelligent" Building at PROCES-DATA A/S, Silkeborg http://www.p-net.dk/applications/house01.html.
8. Scherg R. EIB planen und installieren, Vogel Buchverlag, Würzburg 1998.
9. The EIB Association home page. EIB Association, http://www.eiba.be.

Recenzent: Dr inż. Włodzimierz Boroń

Wpłynęło do Redakcji 2 kwietnia 2001 r.

Abstract

Some problems of home automation systems are investigated in this paper. The authors present the original idea of system with OneWire bus and recently developed microcontrolle

with Java™ Virtual Machine. The choice of OneWire bus was based on number of integrated devices available on the market. The use of such elements reduces the cost and complexity of control modules. However, the performance of the system is significantly lower than in other networks. The use of high-level languages in microcontroller programming is possible due to an increase in computing power of such devices. This approach results in faster development of programs with higher functionality at lower costs, although this is done at the price of lower performance. The TINI platform with powerful API seems to support this claim. Java allows for code portability, where the same source code can be developed, compiled, run on different platform such as PC. On the other hand, some routines can be written in 8051 assembler and linked with Java code, for faster execution.

Another great advantage of TINI is the fact that it is 'Internet aware', with built-in TCP/IP stack. The programmer can implement a range of protocols such as telnet, WWW, WAP using Java libraries as easily as it would be done on other Java platforms.