Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki Nr 9, Rok 7, 2013, s. 53-62

Automatyczna aktualizacja oprogramowania w urządzeniach embedded

Marek Sawicki*, Krzysztof Różanowski**, Robert Czapkowski***

Streszczenie

W artykule zaprezentowano implementację systemu automatycznej aktualizacji danych w urządzeniach embedded z mikrokontrolerem jednoukładowym. Przeanalizowano mechanizmy i protokoły stosowane w istniejących rozwiązaniach pracujących pod kontrolą systemów operacyjnych. Zaproponowano niezbędne optymalizacje, pozwalające na implementacje mechanizmu aktualizacji w systemach z ograniczonymi zasobami pamięci. Opisano implementacje wybranych wariantów zaproponowanych rozwiązań w mikrokontrolerze z rdzeniem Cortex. W pracy zwrócono również uwagę na aspekt zapewnienia bezpieczeństwa mechanizmów aktualizacji automatycznej zarówno pod kątem nieautoryzowanego dostępu, jak też odporności na awarię sieci komunikacyjnej.

***Słowa kluczowe**: aktualizacja automatyczna, systemy embedded, mikrokontrolery jednoukładowe*

1 Wprowadzenie

Automatyczna aktualizacja oprogramowania głównie kojarzona jest z systemami operacyjnymi instalowanymi na sprzęcie o różnej architekturze. Jej mechanizmy umożliwiają łatwą dystrybucję poprawek oraz uaktualnień dla komputerów PC, urządzeń mobilnych, a także sprzętu sieciowego (routery, przełączniki). Tymczasem w prostych systemach z mikrokontrolerem jednoukładowym do komunikacji coraz częściej stosuje się Ethernet. Podyktowane jest to przede

* Wojskowa Akademia Techniczna.
** Warszawska Wyższa Szkoła Informatyki.
*** Wyższa Szkoła Policji w Szczytnie.

wszystkim możliwością uzyskiwania dużych przepływności, galwaniczną separacją współpracujących systemów oraz istniejącą w miejscu instalacji infrastrukturą.

Tego rodzaju medium transmisyjne, oprócz wymienionych zalet, daje również możliwość implementacji mechanizmów automatycznej aktualizacji oprogramowania. Może ona być przeprowadzana przy użyciu dedykowanego sprzętu – programatora (np. komputer PC z odpowiednim oprogramowaniem) lub całkowicie autonomicznie z użyciem wskazanej lokalizacji sieciowej. W przypadku zapewnienia połączenia sieci lokalnej z siecią Internet, pobieranie oprogramowania może odbywać się bezpośrednio z serwerów producenta urządzenia.

2 Aktualizacja oprogramowania w urządzeniach z systemami operacyjnymi

Przykładem zdalnej aktualizacji oprogramowania jest sposób stosowany w systemach z rodziny Windows. W zależności od ustawień użytkownika, system okresowo łączy się ze zdalnym serwerem i sprawdza, a następnie automatycznie pobiera i instaluje dostępne poprawki. Wymiana komunikatów w sieci Ethernet podczas przykładowego sprawdzenia aktualizacji w systemie Windows 7 x 64 przedstawiona została w tabeli 1.

Tabela 1. Fragment komunikacji w sieci Ethernet podczas sprawdzania aktualizacji przez system Windows

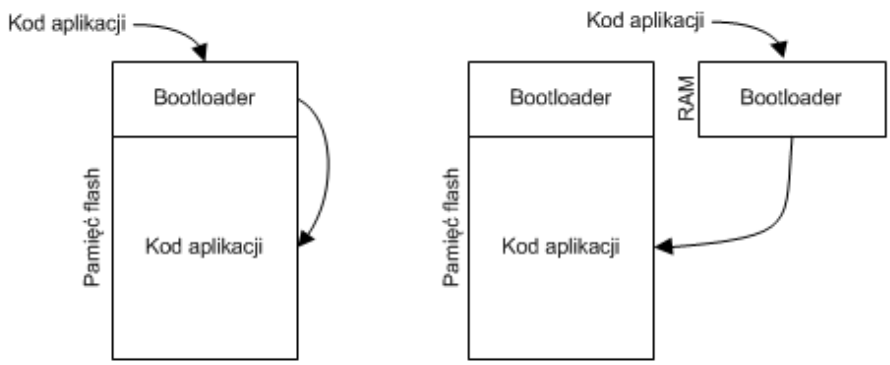
Adres źródłowy	Adres docelowy	Protokół	Informacja o pakiecie
192.168.0.201	157.56.96.59	TCP	62671 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
157.56.96.59	192.168.0.201	TCP	http > 62671 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
192.168.0.201	157.56.96.59	TCP	62671 > http [ACK] Seq=1 Ack=1 Win=65700 Len=0
192.168.0.201	157.56.96.59	HTTP	HEAD /v10/1/windowsupdate/selfupdate/wuident.cab?1304151514 HTTP/1.1
157.56.96.59	192.168.0.201	TCP	[TCP segment of a reassembled PDU]
192.168.0.201	157.56.96.59	TCP	62671 > http [ACK] Seq=172 Ack=265 Win=65436 Len=0
192.168.0.201	157.56.96.59	TCP	62675 > https [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
157.56.96.59	192.168.0.201	TCP	https > 62675 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
192.168.0.201	157.56.96.59	TCP	62675 > https [ACK] Seq=1 Ack=1 Win=65700 Len=0
192.168.0.201	157.56.96.59	TLSv1	Client Hello

Sprawdzenie aktualizacji rozpoczyna się od zestawienia połączenia TCP z serwerem aktualizacji. Następnie poprzez zapytanie HTTP metodą HEAD sprawdzana jest osiągalność serwera aktualizacji. Dalsza transmisja odbywa się już w sposób niejawny za pomocą protokołu TLS. W podobny sposób pobierane są aktualizacje. Szyfrowanie kluczem asymetrycznym, a ściślej faza jego generowania, dostarcza serwerowi informacji o licencji systemu operacyjnego, zezwalając na pobranie oprogramowania.

3 Aktualizacja oprogramowania dla systemów z mikrokontrolerem jednoukładowym

Metody dystrybucji oprogramowania wraz z protokołami uwierzytelniania oraz szyfrowania stosowane w urządzeniach z systemami operacyjnymi nie mogą być wprost zaimplementowane w urządzeniach z mikrokontrolerami jednoukładowymi. W systemach operacyjnych pobierane są aplikacje instalujące właściwe oprogramowanie, które po odszyfrowaniu uruchamiane są jak instalatory aplikacji użytkownika. Mechanizmy te nie są jednak przewidziane do aktualizacji oprogramowania stacji roboczej (przeinstalowania systemu operacyjnego). Typowe oprogramowanie na mikrokontrolera jednoukładowego bez zaimplementowanego systemu operacyjnego stanowi integralną całość. Instrukcje kodu mogą odwoływać się bezpośrednio do układów peryferyjnych. Specjalne bloki sprzętowe mogą również wymusić wykonywanie określonych fragmentów kodu w dowolnej chwili (system przerwań). Oprogramowanie tego rodzaju systemów musi więc być aktualizowane zawsze w całości.

Implementacja wymiany oprogramowania w systemach z mikrokontrolerem sprowadza się do napisania własnych funkcji tzw. bootloadera, pobierającego dane z określonego źródła (pamięć zewnętrzna, komputer PC, serwer FTP) i za pomocą specjalnych instrukcji procesora wykonują kasowanie i zapisywanie w pamięci kodu programu. W przypadku mikrokontrolerów z rdzeniem Cortex [1], możliwe są dwa scenariusze programowania wewnętrznej pamięci (rysunek 1).



Rysunek 1. Tryby pracy bootloadera użytkownika w mikrokontrolerach z rdzeniem Cortex

Bootloader może być wykonywany z pamięci kodu programu i zapisywać inny obszar tej samej pamięci. Podczas normalnej pracy systemu, po zresetowaniu program ten modyfikuje licznik kodu programu i rozpoczyna się wykonywanie programu użytkownika. Bootloader może też skopiować się do pamięci RAM i stamtąd rozpocząć operacje aktualizacji kodu programu. W tym przypadku możliwe jest zaktualizowanie całej pamięci Flash łącznie z kodem bootloadera.

Protokół pobierania oprogramowania z serwera przygotowano tak, aby możliwe było jego uruchomienie na serwerze hostingowym przeznaczonym dla stron www [2]. Urządzenie, które zamierza pobrać nowe oprogramowanie z serwera, zestawia połączenie TCP na porcie 80. Następnie wysyłane jest standardowe dla protokołu HTTP żądanie GET. Na serwerze uruchamiany jest skrypt napisany w języku PHP z takimi parametrami jak nazwa urządzenia (*name*) oraz jego numer seryjny (*id*). Dodatkowo, w polu *sys* przesyłana jest informacja, czy urządzenie żąda pliku z oprogramowaniem (*sys=update*), czy tylko oznaczenia jego wersji (*sys=ver*) (List. 1).

List. 1. Żądanie aktualizacji oprogramowania z użyciem metody GET

```
GET /index.php?id=1234567890&name=ArtNetDmx&sys=update HTTP/1.1
Host: www.firmware.msawiko.pl
User-Agent: EccDevice
Accept: */*
```

Skrypt dystrybucji oprogramowania identyfikuje urządzenia, odpowiada na przesłane żądania oraz zapisuje do pliku informacje o czasie i parametrach otrzymywanych zapytań. Identyfikacja odbywa się na podstawie wysłanego przez urządzenie numeru seryjnego oraz nazwy, które są weryfikowane w bazie SQL serwera. Mechanizm ten ogranicza dostęp do aktualizacji dla niezarejestrowanych przez producenta urządzeń. Dodatkowo, każda próba połączenia się z serwerem aktualizacji jest rejestrowana w bazie SQL. Na podstawie zebranych informacji producent lub dystrybutor sprzętu elektronicznego może sprawdzić, z jaką wersją oprogramowania pracują poszczególne egzemplarze urządzeń. Ma to znaczenie przede wszystkim w procesie sprawnego usuwania zgłaszanych usterek, szczególnie w fazie wdrożenia produktu. Algorytm obsługi żądań przez skrypt PHP przedstawiono na rysunek 2 na następnej stronie.

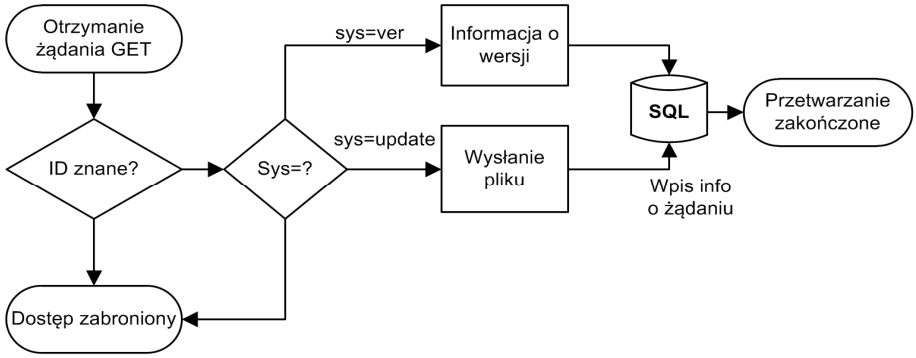
Odpowiedz serwera aktualizacji na poprawnie sformułowane żądanie GET rozpoczyna się nagłówkiem serwera, po którym w zależności od parametru *sys* zapytania przesyłana jest informacja o wersji oprogramowania lub zawartość wskazanego pliku (List. 2).

List. 2. Odpowiedź serwera aktualizacji na żądanie z parametrem sys=wer

```
HTTP/1.1 200 OK
Date: Sat, 20 Apr 2013 08:30:20 GMT
```

```
Server: Apache/2
X-Powered-By: PHP/5.2.17
Vary: Accept-Encoding,User-Agent
Transfer-Encoding: chunked
Content-Type: text/html

3.1.23
```



Rysunek 2. Algorytm obsługi żądań przez serwer aktualizacji

Plik z oprogramowaniem jest transmitowany bajt po bajcie w pakietach TCP. Zgodnie ze specyfikacją [3][4], w polu ładunkowym TCP mogą być wysyłane bajty o dowolnej wartości i nie ma potrzeby kodowania przesyłanego pliku. Parametry poprawnych żądań, na które serwer odpowiedział, są rejestrowane w bazie danych i dostępne w poprzez stronę internetową w postaci tabeli (tabela 2).

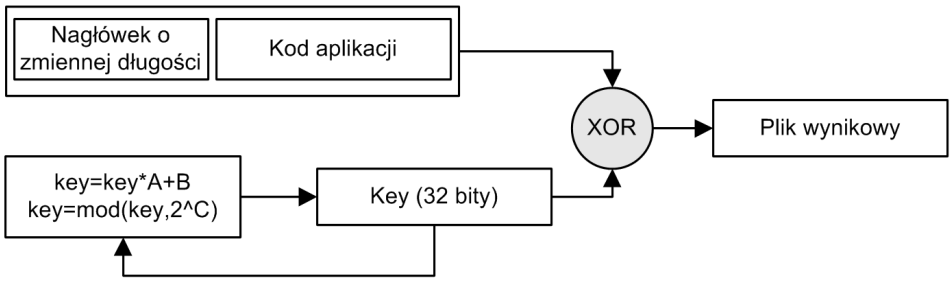
Tabela 2. Informacje o żądaniach dostępne po zalogowaniu dla administratora systemu dystrybucji aktualizacji

L.P.	IP	Data	Godzina	Time stamp	URL	User Agent	ID	NAME	SYS
22	164.127.43.30	2013-04-20	10:35:39	1366446939	www.firmware.msawko.pl/index.php?id=1234567890&name=ArtNetDmx&sys=update	EccDevice	1234567890	ArtNetDmx	update
21	164.127.43.30	2013-04-20	10:30:20	1366446620	www.firmware.msawko.pl/index.php?id=1234567890&name=ArtNetDmx&sys=ver	EccDevice	1234567890	ArtNetDmx	ver
20	164.127.43.30	2013-04-20	10:28:20	1366446500	www.firmware.msawko.pl/index.php?id=1234567890&name=ArtNetDmx&sys=ver	EccDevice	1234567890	ArtNetDmx	ver
19	88.198.65.178	2013-04-19	18:58:45	1366390725	www.firmware.msawko.pl/index.php?id=1234567890&name=ArtNetDmx&sys=update	EccDevice	1234567890	ArtNetDmx	update
18	88.198.65.178	2013-04-19	18:20:38	1366388438	www.firmware.msawko.pl/index.php?id=1234567890&name=ArtNetDmx&sys=update	EccDevice	1234567890	ArtNetDmx	update
17	88.198.65.178	2013-04-19	18:19:42	1366388382	www.firmware.msawko.pl/index.php?id=1234567890&name=ArtNetDmx&sys=update	EccDevice	1234567890	ArtNetDmx	update
16	193.105.35.187	2013-04-19	10:55:48	1366361748	www.firmware.msawko.pl/index.php?id=1234567890&name=ArtNetDmx&sys=update	EccDevice	1234567890	ArtNetDmx	update

4 Bezpieczeństwo dystrybucji oprogramowania

Zdalny system dystrybucji oprogramowania może nieść zagrożenia zarówno dla aktualizowanych urządzeń, jak również interesów firm, które wytwarzają to oprogramowanie i je serwisują. Zapewnienie odpowiedniego poziomu bezpieczeństwa w systemach dystrybucji oprogramowania ma na celu głównie ochronę przed nieautoryzowaną aktualizacja kodu urządzenia. Rozważając kwestie bezpieczeństwa zdalnej aktualizacji oprogramowania, należy uwzględnić możliwość zaprogramowania urządzeń niewłaściwym kodem. Sytuacja taka może być spowodowana błędami transmisji lub świadomym działaniem mającym na celu unieruchomienie określonej grupy urządzeń lub systemów.

W systemie dystrybucji oprogramowania przewidziano mechanizmy zabezpieczeń zarówno po stronie serwera aktualizacji, jak również urządzeń. Serwer weryfikuje zapytania o numer identyfikacyjnych oraz nazwę żądanego zasobu. Obsługiwane są tylko te żądania, dla których w bazie istnieje wpis ze zgodną parą pól *id* oraz *name*. Dodatkowo wymagane jest, aby w polu zwykle zawierającym informację o przeglądarce www (*User-Agent*) była wartość *EccDevice*. Powyższe mechanizmy zabezpieczają system przez przypadkowymi zapytaniami. Nie stanowią jednak zabezpieczenia przed podszyciem się nielegalnej kopi urządzenia i pobraniem pliku. Główną ochroną przesyłanego kodu oprogramowania jest jego zaszyfrowanie za pomocą klucza o takiej samej długości jak szyfrowana informacja. Szyfrowanie strumieniowe jest wykonywane za pomocą operacji XOR. Klucz szyfrujący jest natomiast ciągiem pseudolosowym, generowanym według schematu z rysunku 3.

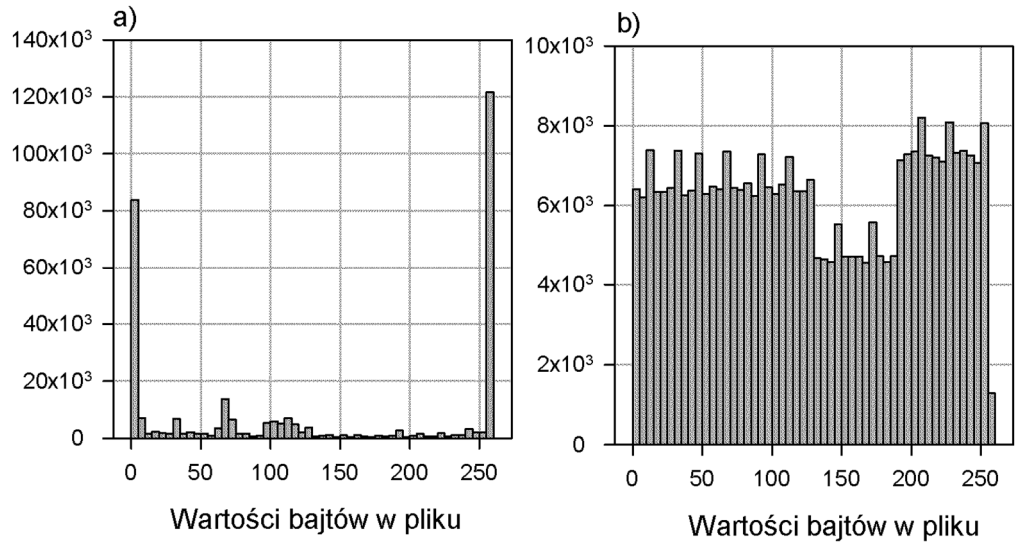


Rysunek 3. Proces szyfrowania kodu programu umieszczanego na serwerze aktualizacji

Ciąg szyfrujący zależy od współczynników (A, B, C) oraz wpisanej wartości początkowej tzw. ziarna. Aby utrudnić znalezienie ziarna ciągu, do pliku z oprogramowaniem dodawany jest nagłówek z następującymi informacjami: nazwa wersji, wskaźniki na fragmenty pamięci z danymi użytkowników oraz wskaźnik na właściwy kod w zaszyfrowanym pliku. W ten sposób, nawet znając początkową (na ogół stałą) zawartość kodu aplikacji, jak też metodę generacji

klucza, trudno jest ustalić ziarno generatora. Jednocześnie, zarówno generowanie klucza, jak i deszyfrowanie, są realizowane przy użyciu elementarnych operacji procesora.

Wartości współczynników generowania klucza (A, B, C) oraz jego wartość początkowa dobrane zostały tak, aby wygenerowany ciąg bajtów miał rozkład w przybliżeniu równomierny, co spowoduje, że równomierny będzie również rozkład wartości w pliku wynikowym. Przykładowy histogram kodu aplikacji przed oraz po zaszyfrowaniu przedstawiono na rysunku 4. Powyższe mechanizmy zapewniają wystarczający poziom zabezpieczenia przed skopiowaniem urządzenia i nieautoryzowanym jego zaprogramowaniem. Odszyfrowanie pliku przed zaprogramowaniem pamięci odbywa się w mikrokontrolerze w programie bootloadera. Są tam też zapisane klucze (współczynniki A, B, C oraz wartość początkowa) potrzebne do generowania ciągu deszyfrującego.



Rysunek 4. Histogram kodu programu przed (a) oraz po zaszyfrowaniu (b)

Pamięć z kodem bootloadera jest zabezpieczona przed odczytem, a jej podgląd jest możliwy jedynie przez aplikację użytkownika. Mechanizmy zabezpieczeń stosowane przez producentów mikrokontrolerów nie pozwalają jednak na częściowe skasowanie i zaprogramowanie zabezpieczonej przed odczytem pamięci Flash. Próba wgrania za pomocą systemu aktualizacji (poprzez skopiowanie skryptu serwera) odpowiednio spreparowanego kodu aplikacji również jest utrudniona, gdyż odszyfrowywany w mikrokontrolerze plik musi być spójny (poprawne sumy kontrolne) zanim zostanie wgrany do pamięci kodu programu.

5 Implementacja automatycznej aktualizacji w mikrokontrolerze z rdzeniem Cortex

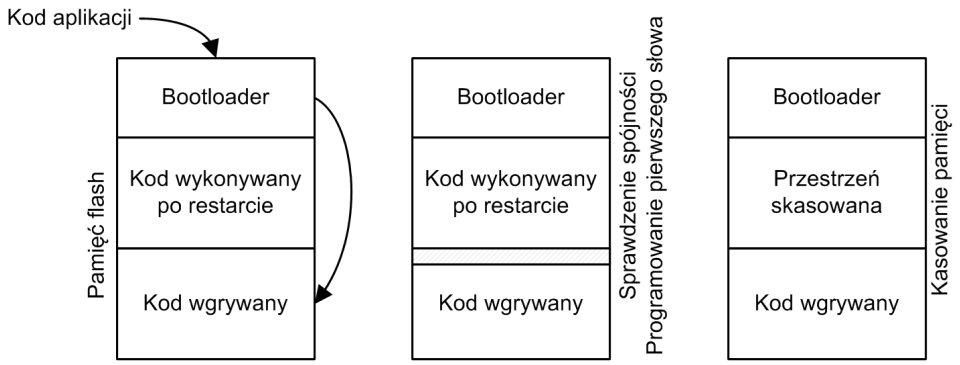
Przygotowując system mikroprocesorowy do pobierania kodu programu z sieci Internet, należy wydzielić na kod programu ładującego wystarczającą ilość miejsca do zaimplementowania niezbędnych protokołów sieciowych oraz mechanizmów szyfrowania i uwierzytelniania. Z funkcji sieciowych można korzystać z poziomu programu użytkownika, jednak w praktyce wygodniej jest całkowicie rozdzielić funkcje bootloadera od programu użytkownika, tworząc każde z nich niezależnie. Program ładujący oprogramowanie powinien być umieszczony od domyślnego adresu resetu mikrokontrolera.

W przypadku układów z rdzeniem Cortex jest to adres zerowy. Na kod bootloadera przeznaczono 32 kB kodu programu, co w przypadku układów rodziny STM32F107 stanowi 16 sektorów pamięci Flash, a dla układów STM32F2xx oraz STM32F4xx są to dwa pierwsze sektory [1]. W przeznaczonej przestrzeni zaimplementowano minimalną wersję stosu TCP/IP. Zawarto w niej między innymi: obsługę ramek ARP, ICMP oraz protokół TCP, co pozwala na otwarcie i utrzymywanie jednego połączenia [5].

Zaimplementowano również funkcję klienta DHCP. Program bootloadera uruchamia się po resecie mikrokontrolera i sprawdza jedną z flag zegara RTC systemu. Jeżeli jest ona ustawiona, następuje sprawdzenie wersji dostępnego oprogramowania i ewentualnie jego podmiana. Następnie licznik kodu programu ustawiany jest na adres 0x0800 8000, od którego rozpoczyna się inicjalizacja mikrokontrolera (*startup.s*) zgodnie z potrzebami programu użytkownika. Sprawdzenie ustawienia flagi zegara RTC powoduje, że aktualizacja nie jest dokonywana po każdym uruchomieniu systemu (reset sprzętowy). Aktualizacja może być wywoływana na życzenie (np. przyciskiem) lub okresowo przez aplikację użytkownika, która ustawia odpowiednią flagę zegara RTC, a następnie wykonuje programowy reset mikrokontrolera.

Zaimplementowano również mechanizmy zabezpieczenia systemu przed uszkodzeniem wskutek błędów w transmisji, nieoczekiwanego przerwania połączenia lub też awarią zasilania w chwili pobierania aktualizacji. Ich działanie polega na zdublowaniu przestrzeni na kod aplikacji. Aktualizacja pobierana i odszyfrowywana jest do nieużywanego fragmentu pamięci Flash, a po sprawdzeniu poprawności procesor przełączany jest na nowy kod programu (rysunek 5 na następnej stronie).

Przełączenie kodu następuje przez odpowiednie zaprogramowanie pierwszych 4 bajtów nowego kodu (pominięte przy ściąganiu aktualizacji) oraz skasowaniu dotychczas używanej przestrzeni. Poprawność wykonania pierwszej operacji determinuje odporność systemu na uszkodzenie wskutek przerwy w zasilaniu urządzenia podczas programowania. W przypadku zaniku zasilania po pierwszej fazie przełączenia bootloader dokończy operację na podstawie informacji o wersji zawartej w każdym bloku programu. Omówione mechanizmy bezpieczeństwa powodują, że ryzyko uszkodzenia oprogramowania podczas aktualizacji zostało znacznie zminimalizowane.



Rysunek 5. Etapy aktualizowania oprogramowania w mikrokontrolerze

6 Podsumowanie

Przedstawiony mechanizm synchronizacji oprogramowania systemów mikroprocesorowych pozwala wyposażyć projektowane urządzenia w bardzo użyteczną dla serwisantów oraz producenta funkcję. Umożliwia ona nie tylko zmniejszenie kosztów serwisowania sprzętu, ale też przyczynia się do poprawy komfortu ich użytkowania. Ograniczony czas tworzenia wbudowanego oprogramowania powoduje, że bardzo trudno jest uniknąć różnego rodzaju błędów. Dysponując systemem dystrybucji oprogramowania dla wdrożonych systemów, możliwe jest poprawianie błędów w sposób niezauważalny dla użytkowników na podstawie zgłoszeń od użytkowników oraz własnych spostrzeżeń. Aktualizacja nie powoduje wyłączenia urządzeń z pracy na czas serwisowania. Przedstawiony system dystrybucji kodu programu został tak zaprojektowany, aby mógł być używany również przez niewielkie firmy produkujące sprzęt elektroniczny. Do jego uruchomienia wystarczy typowy serwer http, na którym zwykle uruchomiona jest strona internetowa producenta. Implementacja automatycznej aktualizacji oprogramowania, oprócz podstawowej funkcjonalności dostarcza również producentowi informacji o tym, jakie urządzenia pracują z poprawionym kodem, a do którego odbiorcy należy skierować prośbę o podłączenie sprzętu do Internetu lub dostarczenie do serwisu.

Bibliografia

[1] Yiu J., *The definitive guide to the ARM Cortex-M3*, Newnes/Elsevier, Amsterdam −Boston 2010

[2] Gerner J., [i in.], *Linux, Apache, MySQL i PHP: zaawansowane programowanie*, Helion cop., Gliwice 2006

[3] *Transmission Control Protocol*, http://tools.ietf.org/pdf/rfc793 [Data uzyskania dostępu: 19 kwietnia 2013]
[4] *TCP Extensions for High Performance*, http://tools.ietf.org/html/rfc1323 [Data uzyskania dostępu: 19 kwietnia 2013]
[5] Peczarski M., *Mikrokontrolery STM32 w sieci Ethernet w przykładach*, Wydawnictwo BTC, Legionowo 2011

Automatic software updates on Embedded devices

Abstract

In this article implementation of a system of automatic software updates on embedded device with a single-chip microcontroller was presented. Mechanisms and protocols used in existing solutions working under control of operating systems was analyzed. Necessary optimizations were suggested that allow for implementation of an update mechanism in systems with limited memory resources. Implementations of chosen variants of proposed solutions in the microcontroller with Cortex core were described. Furthermore, this work also emphasizes a question of providing security of automatic software updates, with regard to unauthorized access, as well as resistance to failure of communication network.

***Keywords**: automatic software update, embedded system, a single-chip microcontroller*