1. Wstęp

System czasu rzeczywistego RTS (ang. *Real Time System*) to taki system, którego wynik przetwarzania zależy nie tylko od jego logicznej poprawności, ale także od chwili, w której taki wynik się pojawi. Systemy czasu rzeczywistego są stosowane wszędzie tam, gdzie konieczna jest bieżąca reakcja na sygnały płynące z zewnętrznego świata. W ostatnim okresie obserwujemy istotny wzrost znaczenia takich systemów. Powodem tego zjawiska jest istotny spadek kosztów i wzrost możliwości sterowników mikroprocesorowych, które są instalowane w nowych urządzeniach. O ile zasoby sprzętowe wcześniejszych sterowników były zbyt skromne, aby pomieścić system operacyjny, obecnie stało się to możliwe, a nawet ze względu na komplikację układów interfejsowych konieczne. Różnorodne sterowniki są obecne w licznych urządzeniach powszechnego użytku, takich jak sprzęt audiowizualny, samochody, telefony, sterują instalacjami przemysłowymi, robotami, samolotami, centralami telefonicznymi. Mówimy, że sterowniki takie są wbudowane w urządzenia, którymi sterują, stąd pojęcie systemów wbudowanych (ang. *embedded*). Systemy wbudowane są zwykle systemami czasu rzeczywistego. Składają się one ze sprzętu i oprogramowania. Fundamentem, na którym jest budowane takie oprogramowanie, jest system operacyjny czasu rzeczywistego RTOS (ang. *Real Time Operating System*). Kroki, które powinny być wykonane, aby potrzebny system komputerowy mógł być zrealizowany, to specyfikacja wymagań, wybór platformy sprzętowej i systemu operacyjnego, posadowienie systemu operacyjnego na wybranej platformie sprzętowej, wytworzenie oprogramowania aplikacyjnego oraz testowanie i weryfikacja. Niniejsza książka obejmuje przede wszystkim fazę tworzenia oprogramowania aplikacyjnego dla systemów czasu rzeczywistego.

Książka jest poświęcona tworzeniu oprogramowania aplikacyjnego w środowisku systemu operacyjnego czasu rzeczywistego zgodnego ze standardem POSIX 1003, a w szczególności systemu QNX6 Neutrino.

O ile sprzętowa część systemu wbudowanego jest konstruowana z powtarzalnych komponentów (na rynku dostępna jest znaczna liczba różnorodnych sterowników), to oprogramowanie jest silnie zindywidualizowane. W praktyce dla każdego zastosowania oprogramowanie aplikacyjne jest tworzone od nowa. Dlatego wytworzenie oprogramowania i jego testowanie oraz weryfikacja stanowią najbardziej kosztowny i czasochłonny etap przedsięwzięcia inwestycyjnego. Oprogramowanie takie, poza nielicznymi wyjątkami, jest tworzone w środowisku pewnego systemu operacyjnego czasu rzeczywistego. Współczesne aplikacje czasu rzeczywistego są często aplikacjami złożonymi z wielu procesów intensywnie komunikujących się ze sobą i z otoczeniem, nierzadko rozproszonych i poddanych ograniczeniom czasowym. Tak więc tworząc taką aplikację, w ogólnym przypadku należy rozwiązać następujące zagadnienia:

– Zapewnić spełnienie wymaganych ograniczeń czasowych.
– Rozwiązać kwestię komunikacji i synchronizacji lokalnych procesów i wątków.
– Rozwiązać problem komunikacji pomiędzy węzłami systemu rozproszonego.

Metody rozwiązania wymienionych zagadnień są najważniejszym tematem tej książki.

Trafny wybór systemu operacyjnego decyduje często o wiarygodności całego systemu, możliwości spełnienia wymagań czasowych, łatwości tworzenia przyszłej aplikacji i jej przenośności. Można więc bez dużej przesady powiedzieć, że częstokroć decyduje on o powodzeniu całego przedsięwzięcia. Obecnie jest dostępnych wiele systemów operacyjnych czasu rzeczywistego. Niektóre z nich podano w tabeli 2.1 w rozdziale 2. Wymienić tutaj można zarówno systemy komercyjne (np. Solaris, LynxOS, VxWorks, Windows CE, QNX Neutrino), jak i systemy open source (np. RT Linux [20] czy eCOS [7]). Systemy te różnią się znacznie właściwościami. Najważniejsze kryteria, względem których porównuje się RTOS, to: łatwość instalacji i konfiguracji, dostępne platformy sprzętowe, wiarygodność, własności czasowe, wsparcie architektur rozproszonych, jakość dokumentacji, zgodność ze standardami, dostępność kodu źródłowego, dostępność wsparcia technicznego, a także cena. Znaczenie poszczególnych kryteriów zależy od konkretnych uwarunkowań. Jest oczywiste, że inne kryteria będą istotne w przypadku urządzenia wbudowanego w kuchenkę mikrofalową, a inne w przypadku sondy kosmicznej. Poszczególne kryteria trudno też wyrazić ilościowo, sprowadzić do wspólnego mianownika i wyodrębnić najlepszy RTOS. Tym niemniej takie próby są podejmowane. Niezależna organizacja Dedicated Systems [6] zajmująca się ewaluacją oprogramowania przeprowadziła porównanie ważniejszych systemów czasu rzeczywistego, w tym systemu QNX Neutrino v6.1, VxWorks AE1.1 i Windows CE.NET. Z uzyskanych rezultatów wynika, że system QNX6 Neutrino góruje pod wieloma względami nad konkurentami. Jeżeli nawet do takich testów podejść z rezerwą, to i tak nie da się zaprzeczyć, że system QNX6 jest obecnie jednym z najbardziej zaawansowanych technologicznie systemów operacyjnych czasu rzeczywistego.

W tym miejscu należy podkreślić istotną rolę standardów. Jeżeli aplikacja dostosowana jest do pewnego otwartego, szeroko uznanego standardu, to są większe szanse, że przetrwa ona zmianę platformy sprzętowej i programowej. Może też łatwiej współpracować z innymi aplikacjami. W dziedzinie systemów czasu rzeczywistego najszerzej stosowanym standardem jest POSIX 1003.1. Najistotniejsze informacje o tym standardzie podano w podrozdziale 2.4, a także w pracach [15], [16]. Pełną lub częściową zgodność z tym standardem deklarują takie systemy jak Solaris, LynxOs, VxWorks, Irix, a także RTLinux [20] czy eCOS [7]. System QNX6 Neutrino wykazuje prawie pełną zgodność ze standardem POSIX 1003.1 [20]. Tak więc aplikacja napisana dla QNX6 Neutrino ma szansę działać (po mniejszych lub większych modyfikacjach) także na innych platformach. Fakt ten jest dodatkowym argumentem przemawiającym za tym, że warto zainwestować w naukę programowania aplikacji czasu rzeczywistego w systemie QNX6 Neutrino.

System QNX6 Neutrino jest spadkobiercą długiej linii rozwojowej. Jego historia jest dokładniej opisana w podrozdziale 2.5. Pierwsza wersja systemu QNX powstała w roku 1982 dla procesora Intel 8088. QNX zaprojektowano od podstaw jako system wielozadaniowy, rozproszony i spełniający wymagania czasu rzeczywistego. Zyskał on sobie opinię systemu niezawodnego i mającego niewielkie zapotrzebowanie na zasoby. Pod koniec lat 80. powstał QNX w wersji 4. W porówna-

niu do wersji poprzedniej, wprowadzono znaczące zmiany. Dostosowano system do standardu POSIX 1003.1. Rozwijano też konsekwentnie technologię systemu rozproszonego. W roku 2001 pojawiła się pierwsza edycja nowej wersji systemu oznaczona jako QNX6 z nowym mikrojądrem Neutrino. Jego architektura jest opisana w rozdziale 3. System od podstaw zaprojektowano jako wieloplatformowy. Wspierane procesory to Intel x86, MIPS, PowerPC, SH-4, ARM, StrongARM i xScale. System spełnia wymagania stawiane współcześnie systemom czasu rzeczywistego i, jak już wspomniano, jest zgodny ze standardem POSIX 1003, a także zapewnia wsparcie dla maszyn wieloprocesorowych SMP oraz zawiera wiele mechanizmów tolerowania awarii. Tak więc jego cechy powodują, że jest on w stanie sprostać większości wymagań stawianych systemom czasu rzeczywistego i systemom wbudowanym.

Naukę tworzenia aplikacji czasu rzeczywistego trudno jest prowadzić w oderwaniu od konkretnego systemu operacyjnego. Nie jest to łatwe, gdyż współczesne RTOS są bardzo skomplikowane, a ich dokumentacja bardzo obszerna. W przypadku systemu QNX6 Neutrino liczba wywołań systemowych przekracza 1500, a dokumentacja składa się z 24 podręczników. Stąd odpowiedni dobór materiału, tak aby zostały uwzględnione tylko rzeczy najistotniejsze, ma istotne znaczenie. Dokonany przez Autora dobór materiału wynika z jego wcześniejszych doświadczeń w tworzeniu aplikacji czasu rzeczywistego. W książce opisano 128 najważniejszych wywołań systemowych. Omawiane zagadnienia ilustrują 53 przykładowe programy. Jednak z powodu ograniczonej objętości książki wiele istotnych zagadnień, na przykład tworzenie administratorów zasobu, pominięto. Kolejnym aspektem nauczania metod tworzenia aplikacji czasu rzeczywistego jest dostępność „ćwiczebnego" systemu operacyjnego. Pod tym względem godne uwagi są systemy open source, takie jak RTLinux czy eCOS. Jednak systemy te są stosunkowo mało zaawansowane szczególnie pod względem narzędzi konfiguracyjnych i możliwości tworzenia systemów rozproszonych. W przypadku systemu QNX6 Neutrino mamy do czynienia z sytuacją pośrednią. Firma QNX Software Systems [37] udostępnia bezpłatnie 90-dniową wersję testową systemu zawierającą pakiet Momentics (jest to zintegrowane środowisko wspomagające tworzenie systemów wbudowanych). Dodatkowo uczelnie mogą się starać o bezpłatną licencję edukacyjną lub bezterminową licencję dla zastosowań niekomercyjnych.

W rozdziale 1 omówiono podstawowe definicje dotyczące systemów czasu rzeczywistego. W rozdziale 3 przedstawiono architekturę systemu QNX6 Neutrino. Rozdział 4 zawiera informacje o instalacji, konfiguracji i podstawach obsługi systemu. W rozdziale 5 podano ogólne informacje o procesach, wątkach i szeregowaniu. Rozdziały 6 i 7 poświęcono zarządzaniu procesami i wątkami. Kolejne rozdziały (8, 9, 10, 11 i 14) zawierają dość obszerny opis metod komunikacji i synchronizacji międzyprocesowej wykorzystującej łącza, komunikaty, pamięć dzieloną, semafory i sygnały. Rozdział 9 obejmuje komunikację w systemach rozproszonych. Z kolei w rozdziale 12 omówiono metody odmierzania czasu, a w rozdziale 13 zaprezentowano timery i zdarzenia. Rozdział 15 poświęcono obsłudze przerwań, a rozdział 16 obsłudze typowej karty pomiarowej PCL718 dołączonej do szyny komputera. Zaprezentowane tam przykłady stanowią wprowadzenie do programowania urzą-

dzeń wejścia-wyjścia z wykorzystaniem przerwań. Książkę kończy rozdział 17 traktujący o programowaniu portu transmisji szeregowej.

Intencją Autora było dostarczenie studentom wyższych szkół technicznych podręcznika do przedmiotu „*Systemy czasu rzeczywistego*" prowadzonego na studium podstawowym lub magisterskim kierunków Automatyka i Robotyka, Elektronika i Telekomunikacja lub Informatyka. W szczególności dotyczy to sytuacji, gdyby przedmiot tak prowadzono opierając się na systemie operacyjnym QNX6 Neutrino. Większość przedstawionych tu mechanizmów jest objęta standardem POSIX 1003.1, a więc książka może być pomocna w studiowaniu innych, zgodnych z tym standardem, systemów. W książce są omówione także ogólne pojęcia dotyczące systemów czasu rzeczywistego i programowania współbieżnego, niezwiązane z żadnym konkretnym systemem operacyjnym. Wymienić tu można ogólne zagadnienia dotyczące procesów i wątków, szeregowania, sekcji krytycznej, licznych metod komunikacji międzyprocesowej, synchronizacji wątków, zakleszczeń, miar efektywności systemów RTS, obsługi układów interfejsowych. Aby treść tej książki była dla Czytelnika zrozumiała, zakłada się podstawową znajomość architektury komputerów oraz znajomość języka C na podstawowym poziomie.

Na zakończenie Autor pragnie wyrazić podziękowania tym, którzy przyczynili się do powstania tej książki. Wszystkich tych osób nie sposób tu wymienić. Autor pragnie podziękować profesorowi dr. hab. Ewarystowi Rafajłowiczowi z Instytutu Informatyki, Automatyki i Robotyki Politechniki Wrocławskiej za życzliwe słowa zachęty i wsparcie finansowe w wydaniu tej ksiażki udzielone w ramach Fundacji na rzecz Nauki Polskiej (FNP). Podziękowanie należy się także panu Czesławowi Bilowi z firmy Quantum, Korporacja Transferu Technologii[1], za pomoc w uzyskaniu licencji systemu QNX6 Neutrino, sprzętu i różnych pomocnych informacji. Słowa wdzięczności należą się też Recenzentowi, którym zgodził się być prof. dr hab. inż. Krzysztof Sacha. Jego wnikliwe uwagi przyczyniły się do uniknięcia wielu błędów i nieścisłości za które Autor musiałby się wstydzić.

[1] *http://www.qnx.com.pl.*