Architektury systemów czasu rzeczywistego

RTOS

Wykonał:
Bartłomiej Bugański

System operacyjny czasu rzeczywistego (ang. *Real-Time Operating System - RTOS*) to komputerowy system operacyjny, który został opracowany tak, by spełnić wymagania narzucone na czas wykonywania zadanych operacji. Systemy takie stosuje się jako elementy komputerowych systemów sterowania pracujących w reżimie czasu rzeczywistego - system czasu rzeczywistego.

Ogólnie można przyjąć założenie, że zadaniem systemu operacyjnego czasu rzeczywistego oraz oprogramowania pracującego pod jego kontrolą i całego sterownika komputerowego jest wypracowywanie odpowiedzi (np. sygnałów sterujących kontrolowanym obiektem) na skutek wystąpienia pewnych zdarzeń (zmianie sygnałów z czujników sterownika). Biorąc to pod uwagę, podstawowym wymaganiem dla systemu operacyjnego czasu rzeczywistego jest określenie najgorszego (najdłuższego) czasu, po jakim urządzenie komputerowe wypracuje odpowiedź po wystąpieniu zdarzenia.

Budowa

Systemy RTOS od „zwykłych” systemów odróżnia sposób szeregowania zadań. Te najpowszechniejsze systemy jak np. Linux, Windows, stosują bardzo sprawiedliwą politykę podziału zasobów. Umożliwiają każdemu zadaniu wykonanie swojej pracy. Procesy czasu rzeczywistego nie są specjalnie uprzywilejowane, ani też nie są zachowywane ich priorytety. Zazwyczaj informacja o priorytetach jest gubiona, kiedy usługi systemowe są wykonywane w kontekście wątków użytkownika. Priorytety te, mogą być np. zmniejszane wraz z każdym impulsem zegarowym. Doprowadza to, do powstawania wszelkiego rodzaju opóźnień. Dodatkowo, ciężko tutaj przewidywać dokładnie kolejność wykonania czy zakończenia poszczególnych zadań. Kolejną sprawą jest, że zwykłe systemy operacyjne poniekąd faworyzują tzw. użytkowników terminalowych i programy interaktywne, jak np. edytory tekstu. Jeśli dany proces zostaje wstrzymany na operacji wejścia wyjścia, to często jego priorytet jest zwiększany. Poza tym, program szeregujący zadania bardzo często oddaje czas procesora zadaniom o niskim priorytecie, co sprawia, że wykonanie dowolnego programu jest uzależnione od obciążenia całego systemu oraz zachowania innych procesów. Bardzo często również stosowane są wszelkiego rodzaju optymalizacje podnoszące ogólną wydajność jak np. mające na celu zwiększenie transferu odczytu danych z dysku poprzez wybranie tego zadania, którego żądania wymagają najmniejszego przesunięcia głowic. Istotnym jest również fakt, że procesy wykonujące się w trybie systemowym nie mogą zostać wywłaszczone.

Budowa cd.

Systemy czasu rzeczywistego, są pod tymi względami o wiele bardziej restrykcyjne. Dla nich można powiedzieć, że czas to pieniądz. Priorytety są przypisane do określonych zadań i w oparciu o nie przydzielane są zasoby. Szukają wszelkich rozwiązań, które eliminują możliwe opóźnienia oraz niedeterministyczny przebieg wykonania. Nie jest dopuszczalna sytuacja, aby krytyczne zadanie musiało czekać ponieważ jakiś proces właśnie kopiuje olbrzymie porcje danych lub uruchamiany jest serwer XWindow. Dodatkowo, w takich systemach, praktycznie rezygnuje się z pamięci wirtualnej. Pomimo wszelkich dobrodziejstw jakie ona oferuje, jak np. możliwość uruchomienia programów, których sumaryczna ilość wymaganej pamięci przekracza ilość dostępnej fizycznie w systemie, niesie ona ze sobą poważne niebezpieczeństwo. Jeśli dany proces kilkakrotnie nie trafi w odpowiednią jej stronę, zostanie wygenerowane opóźnienie, którego skutki mogą być katastrofalne.

Budowa cd.

aby edytować style orca tekstu

poziom

poziom

arty poziom

ąty poziom

BSP - Board Support Package. Jest to zbiór sterowników dla sprzętu oraz innych urządzeń.

Każdy system czasu rzeczywistego składa się z jądra (ang. *kernel*) oraz grupy dodatkowych komponentów zwiększających jego możliwości oraz udostępniających dodatkowe usługi. Najważniejszą częścią jest program szeregujący zadania oraz zarządzający zasobami, który dodatkowo musi być tak zaprojektowany, aby było możliwe spełnienie ograniczeń czasowych.

Jak widać na rysunku, najniższe dwie warstwy to sprzęt oraz zbiór sterowników (BSP). Później jest jądro systemu oraz szereg modułów jak np. system plików, podsystem wejścia - wyjścia, dodatkowe sterowniki, protokoły sieciowe oraz komponenty wspierające tworzenie aplikacji. Na samym szczycie powyższej hierarchii,

Systemy operacyjne czasu rzeczywistego dzielą się na dwa rodzaje:

- Twarde - takie, dla których znany jest najgorszy (najdłuższy) czas odpowiedzi, oraz wiadomo jest, że nie zostanie on przekroczony.

- Miękkie - takie, które starają się odpowiedzieć najszybciej jak to możliwe, ale nie wiadomo jest, jaki może być najgorszy czas odpowiedzi.

Do najbardziej znanych systemów czasu rzeczywistego, zaliczamy:

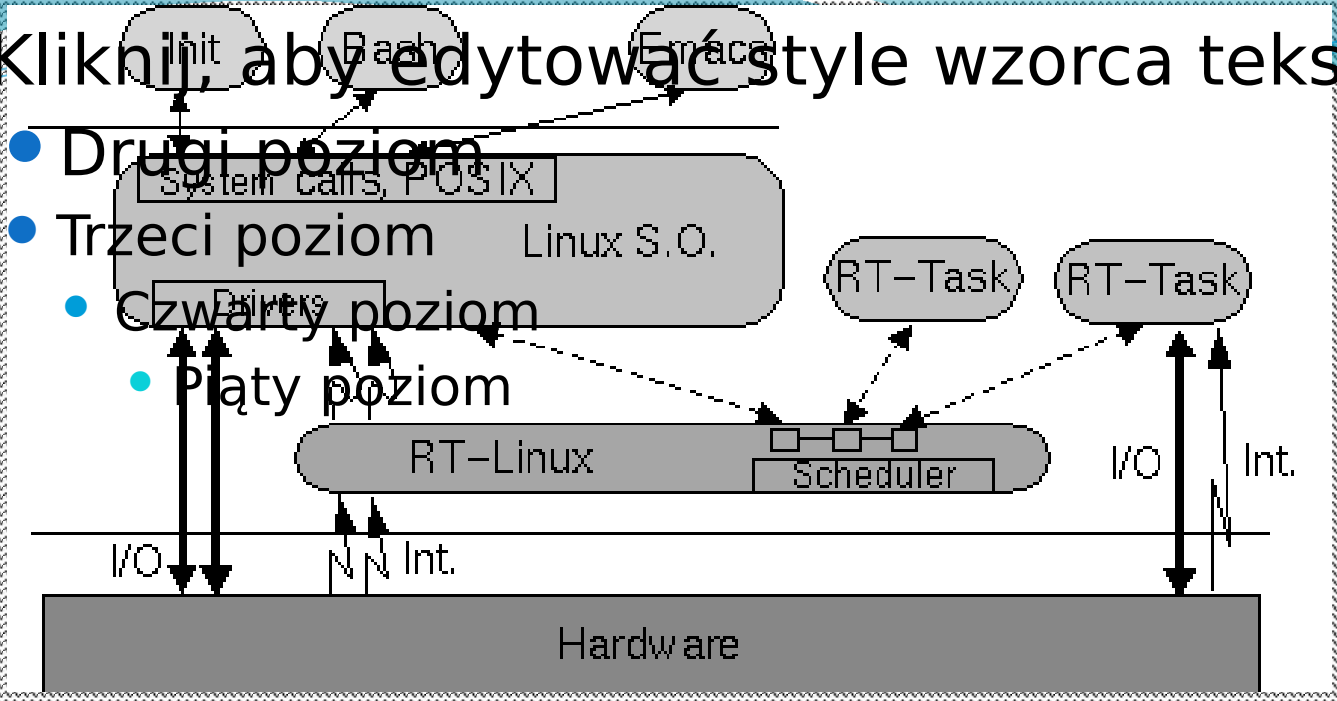
- RTLinux
- QNX (Neutrino)
- Windows Embedded

RTLinux

RTLinux, to rygorystyczny system czasu rzeczywistego.

Oddziela mechanizmy systemu operacyjnego czasu rzeczywistego od systemu operacyjnego ogólnego zastosowania. Działa traktując zwykłe jadro Linuksa jako zadanie pod kontrola niewielkiego i prostego systemu operacyjnego czasu rzeczywistego. W istocie, Linux jest zadaniem tła dla RTLinuksa, wykonywanym jedynie wtedy, gdy żądne z zadań czasu rzeczywistego nie ubiega sie o procesor. Z założenia zadanie Linuksa nigdy nie może zablokować przerwan i zapobiec wywłaszczeniu siebie.

- Kliknij, aby edytować style wzorca tekstu
  - Drugi poziom
  - Trzeci poziom
    - Czwarty poziom
      - Piąty poziom



RTLinux separuje mechanizmy jadra czasu rzeczywistego i mechanizmy jadra zwykłego systemu . Tak wiec każdy z osobna może być optymalizowany niezależnie. Jest tak zaprojektowany, że wyeliminowane są sytuacje, w których musi czekać na zwolnienie jakichkolwiek zasobów przez Linuksa. RTLinux nie alokuje pamięci, nie dzieli sekcji krytycznych ani nie synchronizuje żadnych struktur danych, z wyjątkiem sytuacji niezbędnych do współdziałania obydwu systemów.

Mechanizmy komunikacyjne używane do wymiany danych pomiędzy zwykłymi procesami a zadaniami czasu rzeczywistego są nieblokujące po stronie RTLinuksa. Nigdy nie występuje przypadek, że zadanie czasu rzeczywistego czeka na zakolejkowanie lub pobranie danych z kolejki. Jedna z kluczowych zasad projektowych RTLinuksa jest, aby pozostawić go jak najmniejszym i jak najprostszym. Im mniej spraw do „załatwienia” po stronie RTLinuksa i im więcej po stronie Linuksa, tym lepiej. Tak wiec startem systemu, inicjalizacja urządzeń, ładowaniem modułów, systemem plików i dynamicznym przydzielaniem zasobów zajmuje sie zwykły system. Zadaniem RTLinuksa jest dostarczenie bezpośredniego dostępu do sprzętu dla wątków czasu rzeczywistego, szeregowanie, dostarczanie mechanizmów odmierzania czasu i technik komunikacji miedzyprocesowej.

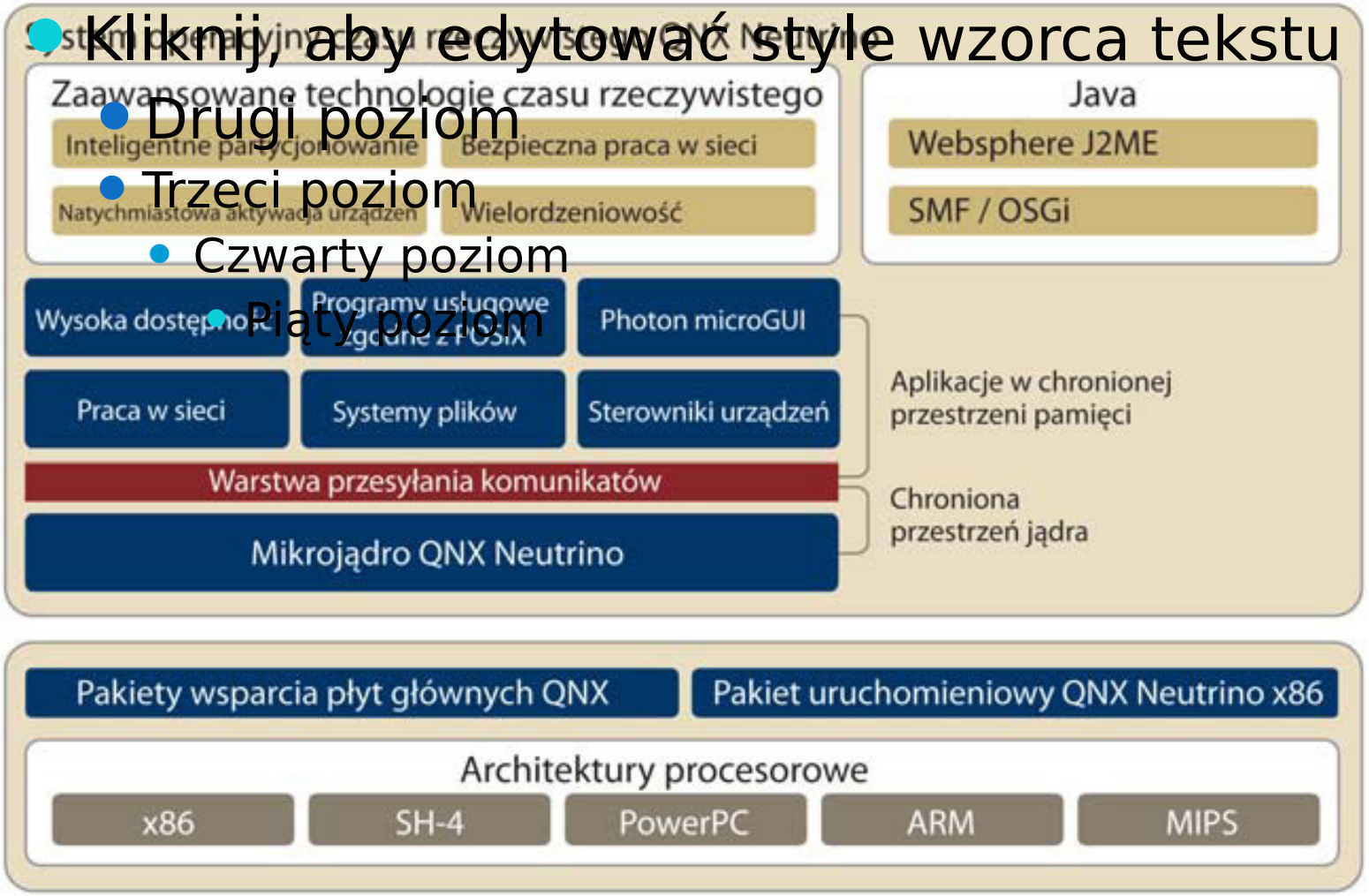
QNX (Neutriono)

QNX, to zdaniem wielu (np. AMD, IBM, Cisco Systems) najlepszy i jednocześnie najbardziej zaawansowany oraz przyszłościowy, rygorystyczny system operacyjny czasu rzeczywistego. Architektura mikrojądra z ochroną pamięci zapewnia aplikacjom wbudowanym maksimum niezawodności, nieporównywalną skalowalność oraz wydajność czasu rzeczywistego.

System operacyjny czasu rzeczywistego QNX Neutrino zbudowany jest w oparciu o architekturę mikrojądra. Dzięki temu wszystkie sterowniki, aplikacje, stosy protokołów i systemy plików działają bezpiecznie poza jądrem, w chronionym obszarze pamięci. Jeżeli którykolwiek z komponentów ulega awarii, jest on automatycznie ponownie uruchamiany bez wpływu na pozostałe komponenty lub samo jądro systemu. Żaden inny komercyjnie dostępny system operacyjny czasu rzeczywistego nie oferuje tak wysokiego poziomu zabezpieczenia awarii i przywracania.

Architektura systemu QNX

- Kliknij, aby edytować style wzorca tekstu
  - Drugi poziom
    - Trzeci poziom
      - Czwarty poziom
        - Piąty poziom



Windows Embedded

Microsoft Windows Embedded Systems, to grupa systemów operacyjnych, które znajdują zastosowanie przede wszystkim we wszelkiego rodzaju urządzeniach współczesnej elektroniki i automatyki. Na czele tej grupy stoją dwie najnowsze produkcje o nazwach: Windows XP Embedded oraz Windows CE .NET.

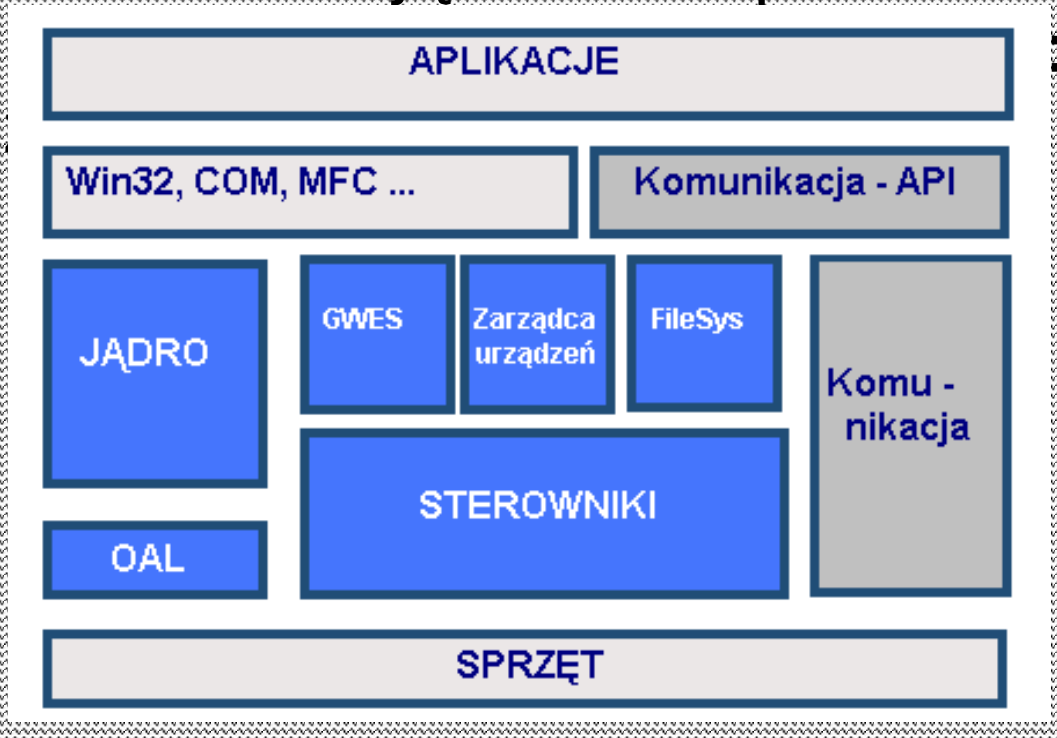
Microsoft Windows CE .NET jest otwartym, w pełni skalowalnym, 32 – bitowym, wielozadaniowym i wielowątkowym rygorystycznym systemem czasu rzeczywistego. Jest w taki sposób zaprojektowany, aby można go było łatwo przenosić pomiędzy różnymi platformami sprzętowymi. Ogólnie wymagane jest, aby platforma docelowa była zaopatrzona przynajmniej w układ pamięci, procesor oraz chip odmierzający czas. Wszelkie dodatkowe urządzenia, mogą być dodawane w fazie projektowania lub „w locie”.

Architektura systemu

Ogólnie architektura całego systemu czasu rzeczywistego korzystającego z Windows CE .NET może być logicznie podzielona na 4 warstwy:

- warstwa sprzętowa (ang. *hardware layer*) – ogólny zbiór urządzeń, z jakich składa się dany system;
- warstwa OEM (ang. *OEM layer*) – zawiera wspomniany już pakiet BSP oraz jest charakterystyczna dla konkretnych rozwiązań sprzętowych. Do niej, zalicza się także jądro systemu, które korzysta ze specyficznych dla danego procesora funkcji. Oczywiście przenoszenie jądra na różne platformy nie dokonują już dostawcy OEM, tylko Microsoft lub specjalna grupa jego partnerów (ang. *Porting Partners*).

- warstwa systemu operacyjnego (ang. *operating system layer*) – zawiera wszelkie komponenty, z jakich składa się system Windows CE .NET.
- warstwa aplikacji (ang. *application layer*) – to warstwa rozszerzająca podstawowe możliwości : o nowy, zlokali



Podsumowanie

Pojawienie się systemów operacyjnych tego typu wiąże się z m.in. zapotrzebowaniem techniki wojskowej na precyzyjne w czasie sterowanie rakietami. Obecnie systemy operacyjne tego typu są wykorzystywane powszechnie w przemyśle cywilnym, sterują również urządzeniami takimi jak na przykład: centrale telefoniczne, marsjańskie ladowniki NASA oraz samochodowy ABS.

Dziękuje za uwagę