KOLOKWIUM 1

**1. Różnice między systemami wbudowanymi, a zwykłymi systemami komputerowymi.**

W Przeciwieństwie do systemów komputerowych systemy wbudowane są zaprojektowane do wykonywania ograniczonej często specyficznej ilości zadań, systemy komputerowe mają bardziej uniwersalne zastosowanie. Systemy wbudowane opierają się na mikrokontrolerach lub specjalnych układach scalonych, a systemy komputerowe na potężniejszy procesorach i mikroprocesorach. Bazują na prostszych systemach operacyjnych.  
**2. Różne podejścia do tworzenia oprogramowania dla systemów wbudowanych.**

Bare Metal – bez systemu operacyjnego

Z prostym SO

Z pelnym SO linux  
**3. Przykłady platform sprzętowych nadających się do realizacji systemów wbudowanych, pracujących pod kontrolą systemu Linux.**

- RPi

- Orange Pi

- Bannan Pi

- Nano Pi  
**4. Powody, dla których w przemysłowych systemach wbudowanych chętnie stosujemy specjalizowane wersje systemu Linux (a nie zwykłe dystrybucje).**

- Małe zużycie pamięci i mocy obliczeniowej

- przemysłowe wersje często mają długoterminowe wsparcie

- Specjalizowane dystrybucje mogą być dostosowane do konkretnych wymagań branżowych

**5. Środowiska pozwalające na stworzenie obrazu systemu Linux dedykowanego dla systemu wbudowanego.**

**-** Buildroot

- OpenWRT  
**6. Środowisko Buildroot - podstawy użytkowania, wybór konfiguracji, składników systemu, wybór konfiguracji jądra.**

**systemu, wybór konfiguracji jądra.**

Praca ze środowiskiem odbywa się z użyciem programu „make”, dla konfiguracji obrazu

korzystamy z interfejsa użytkownika (make menuconfig), gdzie jest dostępna wyszukiwarka

opcji przez „/”. Ponieważ liczba opcji jest bardzo duża, możemy skorzystać z domyślnej

konfiguracji (make nazwa\_platformy\_defconfig) które są dostępne w katalogu /configs.

Jeżeli mówić o ręcznej konfiguracji, to niezbędne są biblioteki i aplikacje systemowe (libc, init),

bootloader i system plików. Dla przespieszenia kompilacji można korzystać z External

toolchain.

Wybór konfiguracji jądra zależy od narzędzia, dla którego konfigurujemy system.  
**7. Możliwe sposoby rekompilacji Linuxa w systemie Buildroot, ich wady i zalety, wybór właściwego sposobu rekompilacji w różnych warunkach.**

**-** Pełna rekompilacja – bardzo wolna ale zapewnia spójność i optymalizacje, wymagan przy dużych zmianach lub przy pierwszej kompilacji – make clean all

- rekompilacja pojedynczych pakietów –szybsza ale nie zapewnia spójności – używana przy małych zmianach

- rekompilacja jądra szybka ale pozwala wprowadzać zmiany tylko w jądrze uzywana przy zmianach w jądrze. -make  **8. Emulator QEMU - właściwości i możliwości wykorzystania do uruchamiania i testowania Linuxa dla systemów wbudowanych.**

Qemu posiada szerokie wsparcie dla różnych architektór oraz wsparcie dla urządzeń peryferyjnych i elastyczność w konfiguracji co czyni go idealnym narzędziem do uruchamiania i testowania linuxa pod różnymi architekturami w celu przygotowania systemu do wdrożenia na system wbudowany

**9. Wybór systemu plików dla Linuxa używanego w systemie wbudowanym. Różnice między pracą z rzeczywistym systemem plików, a "ramdyskiem startowym" (initramfs).**

Przy pracy z rzeczywistym systemem plików mamy możliwość zapisywania danych na stałe w przypadku ramdysku nie ma takiej możliwości, ale dzięki temu że taki system jest przywracany do 0 przy każdym uruchomieniu powoduję o ze jest od bardziej odporny na psucie przez użytkownika

**10. Zagrożenia związane z wgraniem błędnego (np. błędnie skonfigurowanego lub błędnie zbudowanego) systemu operacyjnego do systemu wbudowanego. Sposoby zapobiegania ryzyku nieodwracalnego, lub trudno odwracalnego zablokowania systemu. Sposoby realizacji i uruchamiania "systemów ratunkowych".**

**Zagrożenia:**

-Nieosiągalność urządzenia: Urządzenie może stać się nieosiągalne przez standardowe metody komunikacji, jeśli nowy system operacyjny nie uruchomi się poprawnie.

-Utrata danych: Błędna konfiguracja lub błędy w systemie mogą prowadzić do utraty danych.

-Zagrożenie bezpieczeństwa: Błędy w konfiguracji systemu mogą otworzyć możliwości ataku

-Zmniejszona wydajność

-Utrata funkcjonalności

Zapobieganie:

- Testowanie przed wdrożeniem

- systemy ratunkowe

Realizacja systemów ratunkowych:

- Bootloader z opcją recovery - menu startowe z opcją uruchomienia systemu ratunkowego

- oddzielna partycja dysku na system ratunkowy.

**11. Typowe interfejsy do podłączania urządzeń peryferyjnych do systemów wbudowanych. Możliwości ich obsługi "z przestrzeni użytkownika" (czyli bez tworzenia dedykowanych sterowników działających w przestrzeni jądra).**

Typowym interfejsem jest GPIO sterowane przez sysfs lub bibliotekę libgipod. **12. Możliwości obsługi interfejsu GPIO - dedykowane dla RPi i przenośne między różnymi systemami wbudowanymi. Cechy "starego" systemu opartego na sysfs i "nowego", wykorzystującego libgpiod.**

Obsługa przez sysfs – łatwa w użyciu możliwość korzystania na różnych dystrybucjach linuxa bez specjalnego oprogramowania, ale ograniczona wydajność w związku z operacjami na plikach

Przez libgpiod – lepsza wydajność i większe możliwości obsługi dzięki API łatwe przenoszenie kodu miedzy platformami **13. Środowisko OpenWRT - cechy szczególne, różnice w stosunku do Buildroota.**

Pozwala na instalowanie oraz odinstalowywanie pakietów trakcie działanie systemów bez konieczności rekompilacji. OpenWrt daje możliwość korzystania z wersji prekompilowanej. OpenWrt używa zapisywalnego systemu plików. Zwykle nakładka overlayfs. **14. Obsługa instalowalnych pakietów w OpenWRT -podstawowe operacje.**

Obsługa pakietów odbywa się przez program „opkg”, który może instalować (install),

odinstalować (remove) pakiety, wyświetlić listę plików w pakiecie (files), pobrać listy

dostępnych pakietów (update), wyświetlić listę wszystkich pakietów (list) lub instalowanych

(list-installed). **15. Różnice między pakietami w Buildroocie i w OpenWRT.**

W openWRT możemy doinstalować pakiety w trackie działania systemu dzięki opkg a w buldroocie należy rekompilować system po instalacji pakietów **16. Sposoby modyfikacji konfiguracji OpenWRT.**

Podobnie jak w BR mamy make menuconfig i możliwość dodawania własnych pakietów. Także

możemy określić zależności między pakietami i położenia ich żródeł. Nową opcją jest bardziej

selektywny wybór, co chcemy usunąć. **17. Różnice między dostępnymi trybami rekompilacji w Buildroocie i w OpenWRT.**

Make clean w BR niszczy konfigurację systemu, jednak w OWRT jest bardziej selektywny

wybór, co chcemy usunąć podczas rekompilacji: istnieją make clean, make dirclean, make

distclean. **18. Metody przenoszenia programów (skryptów i skompilowanych aplikacji) między stacją roboczą a uruchamianym systemem wbudowanym.**

Przez serwer http w którym umieszczamy pliki do przeniesieni a i za pomocą programu wget pobieramy z serwera na urządzeniu docelowym lub przez ssh(scp). **19. Praca z SDK w środowisku OpenWRT.**

Można zbudować SDK za pomocą menuconfig. SDK pozwala rekompilować pojedyncze systemy zamiast cały system. **20. Czy w typowym systemie wbudowanym pracującym pod kontrolą systemu Linux możemy użyć partycji lub pliku wymiany aby wirtualnie zwiększyć pojemność pamięci RAM? Proszę uzasadnić odpowiedź. Ewentualnie proszę uwzględnić możliwe warianty.**

Tak możemy w sytuacjach kiedy mamy ograniczoną ilość ram. Możliwe warianty to partycja lub plik wymiany

**21. Jakie jest zastosowanie "systemu ratunkowego" w systemie wbudowanym działającym pod kontrolą systemu Linux? Jak możemy taki system zaimplementować? Proszę opisać przykładową implementację na wybranej przez siebie platformie sprzętowej.**

System ratunkowy zapewnia alternatywne minimalne środowisko do diagnostyki naprawy lub przywrócenia głównego systemu**.** Możemy to zrealizować za pomocą bootloadera z wyborem systemu lub też wydzielić partycje na system ratunkowy ewentualnie posiadanie nośnika z sytemem ratunkowym **22. Czy możemy stworzyć instalowalny pakiet z naszą aplikacją dla OpenWRT bez pobierania i rekompilacji pełnych źródeł OpenWRT? Jeśli tak, to jak możemy to zrobić? Proszę opisać podstawowe etapy przygotowania środowiska pracy i budowania takiego pakietu.**

Tak możemy to osiągnąć korzystając z SDK. Należy pobrać i skonfigurować sdk a następnie zainstalować niezbędne zależności i następnie pisząc odpowiedni makefile następnie za pomocą polecenia make packege/,projekt./compile skompilować naszą paczke. **23. Opracowujemy w Buildroocie oprogramowanie dla systemu wbudowanego, korzystającego z nieulotnej pamięci masowej, dla której nie jest dostępny (jeszcze?) sterownik dla systemu Linux. System jest wyposażony w firmowy bootloader (o zamkniętym kodzie źródłowym), będący w stanie załadować i uruchomić jądro Linuxa. Proszę opisać (uzasadniając), jak powinniśmy skonfigurować BR w takim przypadku.**

Powinniśmy skonfigurować BR do budowania sytemu z initramfs i włączyć odpowiednie ocpcje dla initramfs w jądrze linuxa. Używając initramfs można uniknąć dostępu do nieulotnej pamięci masowej w początkowym etapie projektu. **24. Oprogramowanie dla systemu wbudowanego, tworzone za pomocą środowiska Buildroot, jest testowane za pomocą emulatora QEMU. Zależy nam na tym żeby kolejne wersje naszej aplikacji, napisanej w języku C przetestować bez restartowania emulowanej maszyny. Jak możemy to zrealizować? Proszę podać trzy różne scenariusze przekazania nowej wersji aplikacji do emulowanego systemu. W jakim przypadku nie da się przetestować kolejnej wersji bez restartu tego systemu?**

Można to zrealizować za pomocą zdalnego sytemu plików NFS lub korzystając z SSh do przekazywania plików lub też za pomocą serwera http i programu wget. Niektóre zmiany takie jak na poziomie sytemu operacyjnego wymagają reatartu.

**25. Pakiet aplikacji w Buildroot używa publicznego repozytorium git do przechowywania kodu źródłowego. Chcemy wprowadzić modyfikacje, dostosowujące tę aplikację do naszych potrzeb. Jak możemy to przeprowadzić?**

Należy sklonować repo i na nowej gałęzi wprowadzi pożądane modyfikacje a następnie umiećścić smodyfikowany kod. Następnie należy zmienić pakiet w buildroocie i zrekompilować system. **26. Proszę wymienić cztery wybrane powody, dla których standardowa dystrybucja Linuksa nie nadaje się do wykorzystania w przemysłowym systemie wbudowanym.**

Standardowa dystrybucją jest często zbyt cieżka, zbyt wymagająca sprzętowo/obliczeniowo dla prostych systemów wbudowanych, pzez swoje szerokie zastosowanie nie jest zoptymalizowana do wykonania konkretnego specyficznego zadania dla danego systemu wbudowanego, do tego jest trudna w modyfikacji aktualizacji i utrzymania. **27. Do zbudowanego w środowisku BR Linuksa działającego na przemysłowym systemie wbudowanym chcemy dodać naszą nową aplikację napisaną w języku C. Czym będzie się to różnić od uruchomienia tej aplikacji na naszym PC?**

Środowisko wykonawcze w systemie wbudowany rożni się od tego na PC często jest dużo prostsze i nie posiada wszystkich bibliotek oraz zależności co na PC, aplikacje do systemów wbudowanych mogą wymagać większych optymalizacji niż na PC. **28. Jakiej funkcji niedostępnej z poziomu powłoki systemu wymaga pełna obsługa urządzeń podłączonych do interfejsów I2C i SPI? Proszę wymienić przynajmniej trzy działania (łącznie dla I2C i SPI) do których konieczne jest użycie tej funkcji.**

Wymaga dostępu do wywołań systemowych. Jest to niezbędne w sytuacjach np:

- inicjalizacji urządzeń

- czytania i zapisu do rejestrów urządzenia

- przerwania i obsługi zdarzeń

**29. Dlaczego projekt OpenWRT może udostępniać prekompilowane obrazy Linuksa dla typowych platform sprzętowych, a w przypadku środowiska Buildroot nie jest to stosowane?**

BR jest stworzony do szerszych zastosowań dlatgo prekompilowane obrazy są nie praktyczne ze względu na szeroką możliwość konfiguracji **30. Proszę podać cztery wybrane powody, dla których może być przydatne testowanie w emulatorze QEMU systemu Linux, przygotowywanego dla systemu wbudowanego.**

- Testowanie w qemu pozwala na szybsze usprawnianie aplikacji dzięki braku marnowania czasu na wgrywanie systemu na sprzęt

- Qemu umożliwia testowanie oprogramowania na szerokiej gamie architektur systemów dzięki czemu nie musimy używać wielu urządzeń.

- łatwiejsze debugowanie

- pozwala na automatyzacje testów **31. Po zbudowaniu w środowisku Buildroot Linuksa dla naszego systemu wbudowanego, okazało się, że nie jest w nim dostępny sterownik dla używanej w nim kamery USB (mimo że jest dostępny w źródłach jądra). Proszę opisać szczegółowo w punktach jak rozwiążecie ten problem (uwzględniając dodatkowe problemy związane ze specyfiką BR).**

- Uruchomienie make linux-meucofig w BR

- znalezienie opcji sterowników do USB Vide Class

Zaznaczenie odpowiednich opcji

- włączenie wsparcia dla V4L2

- upewnienie się ze w make menucofig są zaznaczone wszystkie pakiety potrzebne do projektu

- rekompilacja systemu

**Tomasz Winiarski**

**1. Ograniczenia czasowe systemów.**

Ostre – każde przekroczenie ograniczeń jest katastroficzne, na najgorszy przypadek;

Łagodne – przekroczenie degraduje jakość ale nie niszczy, na wartości przeciętne

Sztywne – pojedyncze przekroczenie są tolerowane, liczne – pogarszają jakość.

**2. Cechy systemu czasu rzeczywistego.**

Integralne połączenie z instalacją - związku systemu czasu rzeczywistego z kontrolowanym środowiskiem lub instalacją. System taki jest zintegrowany z fizycznymi urządzeniami lub procesami, które monitoruje i nad którymi sprawuje kontrolę w czasie rzeczywistym.

wiele zastosowań -

terminowość działania - gwarantowaniu wykonania zadań w określonym, ściśle zdefiniowanym czasie.

**3. Co to VoIP?**

Pakiety wysyłane i odtwarzane w stałym tempie, dopuszczalna strata niewielkiej częsci

pakietów.

**4. Problemy implementacji systemów czasu rz.**

Ograniczone zasoby sprzętowe, trudna weryfikacja projektu, konieczność weryfikacji i

walidacji, dominuje C.

**5. Wymagania do sterownika wbudowanego.**

Układy wejść-wyjść procesowych, niezawodność działania, możliwość pracy ciągłej, ograniczony

pobór mocy.

**6. Realizacja przekaźnikowa: wady i zalety.**

Z: izolacja galwaniczna, odporność na zakłócenia

W: duży rozmiar, koszt, zyżycie energii, ograniczone funkcje, trudna modyfikacja

Zalety dla innych: mały pobór energii, małe rozmiary, niski koszt produkcji i t.p.

Wady mikrokontrolera: umiarkowana szybkość; Zalety: elastyczny i modyfikowany

**7. Obsługa I/O karty.**

Odbywa się przez rejestry karty (dodawanie liczb od 0 do 15 do Bazy)

**8. Wielozadaniowy system czasu rzeczywistego.**

Cykliczne odpytywanie, wykorzystanie przerwań, praca wielozadaniowa, ograniczone

wywłaszczanie i t.p.