Konkurencia- és energiakezelés integrálása eszközmeghajtókba

Vezeték nélküli szenzorhálózatok

Energiahatékonyság

- Beágyazott eszközökben fontos a hatékony energiagazdálkodás
- OS-ek nagy részében ennek ellenére csak explicit módon, alkalmazásszinten van energiakezelés
  → Komplex, redundáns kód → több hiba
- Cél: OS-szintű automatikus energiakezelés
- Általánosságban igaz:
  - Az egyszerű megoldások ritkán hatékonyak
  - Már kevés ismeret az alkamazásról is sokat segítet

Konkurencia és energiafogyasztás

- WSN-eket szeretnénk hónapokig vagy akár évekig is működtetni
- Kutatások rámutattak, hogy a konkurencia és az energiafogyasztás között szoros kapcsolat áll fenn
  - Együtt kellene kezelni
    → Energiahatékony ütemezés

ICEM
Integrated Concurrency and Energy Management

- OS szintű automatikus energiakezelés a TinyOS-hez
- 3 sematikus konkurencia osztály:
  - → virtualized
  - → dedicated
  - → shared
- Split-phase (aktív) zárolási mechanizmus
  - → **power-lock**

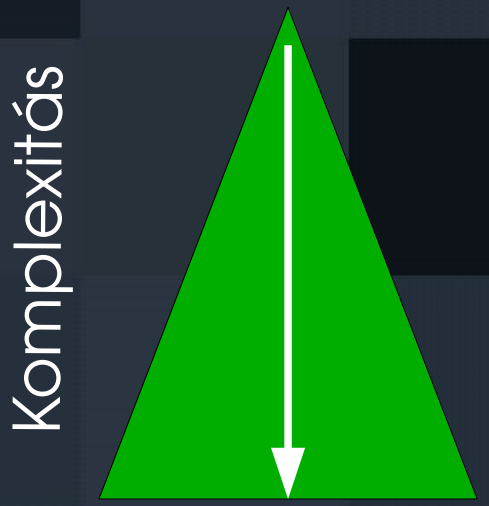
ICEM példa

- Mintavételezés → adatok BS-re küldése
- Csomag-„löketek” (package-burst)
- termelő/fogyasztó kapcsolat a logikai komponensek között
- Split-phase zárolás a visszahívások miatt precíz energiairányítást tesz lehetővé
- Az idő 99%-ában „aludni” kell

→ 1,6%-os energiahatékonyság veszteség az explicit megvalósításhoz képest, de nagy memória overhead!

ICEM konkurencia osztályok

- Virtualizált
- Dedikált
- Osztott



Virtualized class

- Kliens számára legegyszerűbb használni
  - Egy funkcionális/adat interfész
  - Több felhasználó implicit konkurenciával
  - Kérések bufferelése és ütemezése

    → kliens számára transzparens absztrakció

    → de késleltetés lép fel a buffer miatt
  - Kliensenkénti állapot ismert

    → ez alapján ütemezhetünk
  - A késleltetés miatt kompromisszum

    → egyszerűség vs. explicit konkurencia
  - Alkalmazásszintű interfészeknél használt

Dedicated class

- Egy dedikált kliens
  - → Használható alacsony szintű komponenseknél
    - → legalacsonyabb szintű hardverfüggetlen absztrakció
- Nincs szoftveres konkurencia
  - → de az MCU által támogatott hardveres konkurencia kezelhető
- Funkcionális interfészt mellett egy explicit energia kontroll interfészt is nyújthat
  - → de egyes esetekben implicit is kezelhető az energia a funkcionális interfészen keresztül

Shared class

- Explicit konkurencia kezelés
  - Klienseknek versengeniük kell egymással
    - → Aktív zárolás használata (power-lock)
  - Lehetőség a klienseknek atomi műveletek végrehajtására
  - Egy műveletnek több erőforrás kizárólagos használatát is lehetővé teszi
    - → bufferelt ütemezés, mint a virtualizált osztálynál
    - → de itt zárolási kérések kerülnek ütemezésre
- Két változat a zár vizsgálatát tekintve
    - → Checked (pl.: bus driver)
    - → Unchecked (pl.: I/O pin driver)

Driver energia menedzsment

- **Dedikált driver**
  - explicit energiakezelés

    → StdControl (single-phase)

    → SplitControl (split-phase)

    → AsyncStdControl (async. single-phase)
- Alacsony szintű hardver absztrakció

  → AsyncStdControl
- Alkalmazás szinten

  → SplitControl vagy StdControl

Driver energia menedzsment 2.

- Virtualizált és osztott driverek magukban foglalják a konkurencia- és energiakezelést
  - **Virtualizált driver**

    → „interfész specifikus bufferelés”

    → driver specifikus energia menedzsment

    → központi pozíció az irányításban

    - **az általa reprezentált driver minden energiaszükségletéről tud**
  - **Osztott driver**

    → konkurenciakezelésre egységes interfész: **lock**

    → a zárral az energiakezelést is irányítható

    → A zár az energiakezelés aktív irányítója lesz

Split-phase power locks

- **„Energia zár”** (power lock)
  - Aktív szinkronizációs mechanizmus
    - → energia kezelés
    - → konfiguráció kezelés
    - → konkurencia kezelés
  - Ha nincs zárolás ki lehet kapcsolni az eszközt
  - Minden kliensnek lehetőséget nyújt, hogy az eszközön a számára szükséges konfigurációt beállítására
  - A TinyOS felépítése miatt nem lehet blokkoló zár
    - → osztott fázisú (split-phase) zár: callback használata

ICEM Component Library

- Az ICEM komponenseinek 3 típusa lehet:
  - Irányító (arbiter)
  - Energiakezelő (power manager)
  - Konfiguráló (configurator)
- TinyOS-be illeszkedő komponensek az újrahasznosíthatóság végett
- A 3 komponensből komplett energia zárak építhetők

Irányító komponens

- Arbiters
  - Központi komponens
    - → Split-phase interfészt szolgáltatása
    - → Ütemezési irányelv meghatározása
    - DefaultOwner
      - → Ha senki nem használja a zárat
      - → Nyomonkövethető a tétlenség → energiakezelés
    - Configuration
      - → Egyedi eszközbeállítások kezelése
      - → Fordításidőben kioptimalizálható, ha nincs használatban
  - Round-robin és FCFS (first come first served)

Energiakezelő komponens

- Power managers
  - DefaultOwner implementálása
  - Egy explicit energiakezelő implementálása
    - → StdControl (single-phase)
    - → SplitControl (split-phase)
    - → AsyncStdControl (async. Single-phase)
  - Energiakezelési irányelv meghatározása
    - zárolásnál szükség van az eszközre → ON
    - feloldásnál ki lehet kapcsolni az eszközt → OFF
    - **→ azonnali kikapcsolás (immediate)**
    - **→ késleltetett kikapcsolás (deferred)**

Konfiguráló komponens

- Configurators
  - Zár engedélyezési folyamat részei
  - Configuration interfész implementálása az irányító (arbiter) komponensek számára
  - Eszközspecifikus beállítások alkalmazása
    - ADC (Atmega128, MSP430)
    - SPI (MSP430)
    - USART (MSP430)
    - I2C (MSP430)

Sleep Energy Management

- Energia zárak a perifériákat kezelik

  → Szükség van az MCU energiaszintjeinek kezelésére is

- Low Power Modes (LPM) és LPL

  → Mikrokontroller által támogatott energiagazdálkodási mechanizmusok használata

  → A legoptomálisabb kiszámítása komplex művelet

    → MCU sleep funkcionalitásának kihasználása

    → **dirty bit**: ha be van állítva, optimális LPM újraszámítása

    → **override hook**: felülbírálási lehetőség

Hatékonyság mérése

- Microbenchmark
  - RAM és ROM overhead: 350-550 bájt
  - CPU cycle overhead: ~300-350 + ~400 ciklus
  - Warm-up idők figyelembevétele szükséges
- Hatékonyság
  - Az ideálistól (manuális, explicit energiakezelés) amortizáltan ~1,6%-al marad el
  - Elérhető akár tízszeres alkalmazás komplexitás csökkenés (LLOC tekintetében) 500 → 50 sor