Sardsüsteemid

Erkki Moorits
Cybernetica AS, Navigatsioonisüsteemide osakond

Teemad

- Sissejuhatus
- MMU
- Erinevad arhitektuurid
- Erinevad mikrokontrollerid
- Kernel
- Energia sääst

Embedded System – sardsüsteem

- Üldiselt mõne kindla funktsiooniga süsteem
  - Võib olla tegemist nii 8 bitise kui ka 64 bitise süsteemiga
- Väga tihti on tegemist reaalaja süsteemiga
- Tunduvalt piiratumate võimalustega kui “tavaline” arvuti
  - Suhteliselt piiratud arv operatistioonisüsteeme/kerneleid ja arendusvahendeid
- Süsteemide lõikes väga erinev energiatarve
- Paljudel juhtudel on võimelised töötama ka väljaspool toatemperatuure

Deeply Embedded System - (süvasardsüsteem?)

- Tugevalt integreeritud sardsüsteem
- Sardsüsteemi üks alamosa, kuid siiamaani pole head eristust süvasardsüsteemi ja sardsüsteemi vahel
  - See eristus sõltub pigem kasutajate/arendajate ringkonnast
- Süvasardsüsteemi põhilised erinevused sardsüsteemist:
  - Kasutaja liides on olematu või siis väga raskelt juuredepääsetav, näiteks läbi seriaalliidese
  - Süvasardsüsteemid on tavaliselt 8 või 16 bitistel kontrolleritel ja harvem 32 bitistel

Deeply Embedded System - (süvasardsüsteem?) jätk

- Süvasardsüsteemi põhilised erinevused sardsüsteemist (jätk):
  - Tavaliselt on tegemist väga väikese energiatarbega süsteemiga
  - Väga paljudel juhtudel töötab ka väljaspool toatemperatuure
  - Üks võimalik eristuse variant – süvasardsüsteemil pole võimalik Linuxit kasutada
- Kuid – enamus arendusmeetodeid mis toimivad süvasardsüsteemil toimivad ka sardsüsteemil ja tavalisel arvutil
- Järgnevates loengutes on eeldatud, et tegemist on süvasardsüsteemidega

Sardtarkvara loomise erinevus „tavalisest“ tarkvarast

- Puudub igasugune kaitse ohtlike mälulekete vastu
  - Paljudel mikrokontrolleritel pole MMU'd (Memory Management Unit)
- Põhiprogrammist saab otse erinevate mäluväljade ning registrite poole pöörduda, katkestustel on võimalik suhelda otse põhiprogrammiga
- Erinevate programmerimiskeelte valik pole nii lai kui seda on PC'l

Sardtarkvara loomise erinevus „tavalisest“ tarkvarast jätk

- Sardtarkvara funktsionaalsus sõltub väga suurel määral riistvarast
- Erinevalt tavalisest PC programmidest kasutatakse väga palju kindla pikkusega muutujaid näiteks:
  `uint8_t, int16_t`
- Sardtarkvara kirjutamiseks ja testimiseks kulub reeglina rohkem aega kui „tavalisele“ tarkvarale
  - Vigadel on tihtipeale komme „ära kaduda“, näiteks võivad olla sõltuvuses temperatuurist
- Sardtarkvara loomiseks kuluva aja ennustamine nõuab väga suurt kogemust sarnaste sardsüsteemide vallas

Sardsüsteemide tarkvara tüüpilised probleemid

- Inimresurss
  - Kuna sardtarkvara kirjutamine on väga riistvara lähedane töö, siis peab olema väga hea ülevaade riistvarast ja kasutatavast elektroonika osast
- Limiteeritud mälu
  - Eriti oluline on see väikeste mikrokontrollerite puhul
- Protsessori jõudlus
- Süsteemi voolutarve
  - Voolutarve on väga oluline just patareitoitega seadmetel
- Toote hind

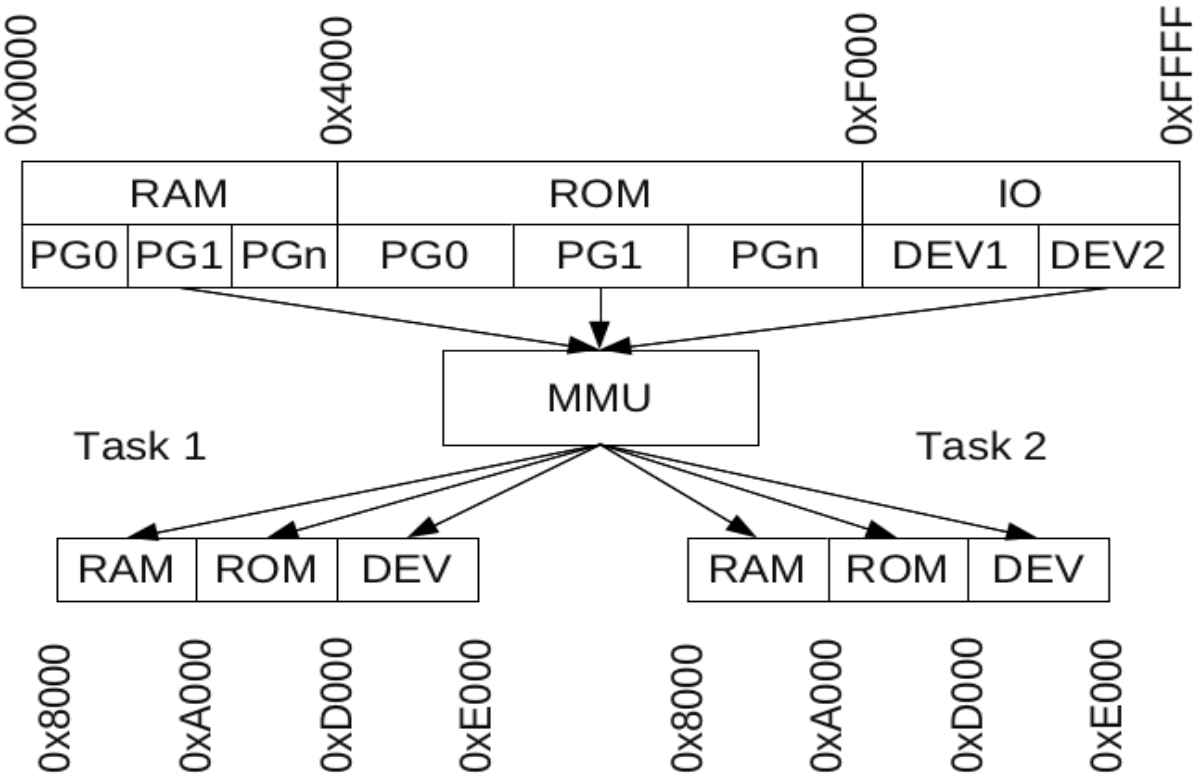
Sardsüsteemide tarkvara tüüpilised probleemid jätk

- Sardtarkvarast vigade otsimisele ja tarkvara testimisele läheb minimaalselt kahekordne koodi kirjutamise aeg
- (Sard-)süsteemis kasutatav operatsioonisüsteem koos oma teekidega
- Tarkvara loomisel ei pruugi alati toimida ülalt-alla meetodid

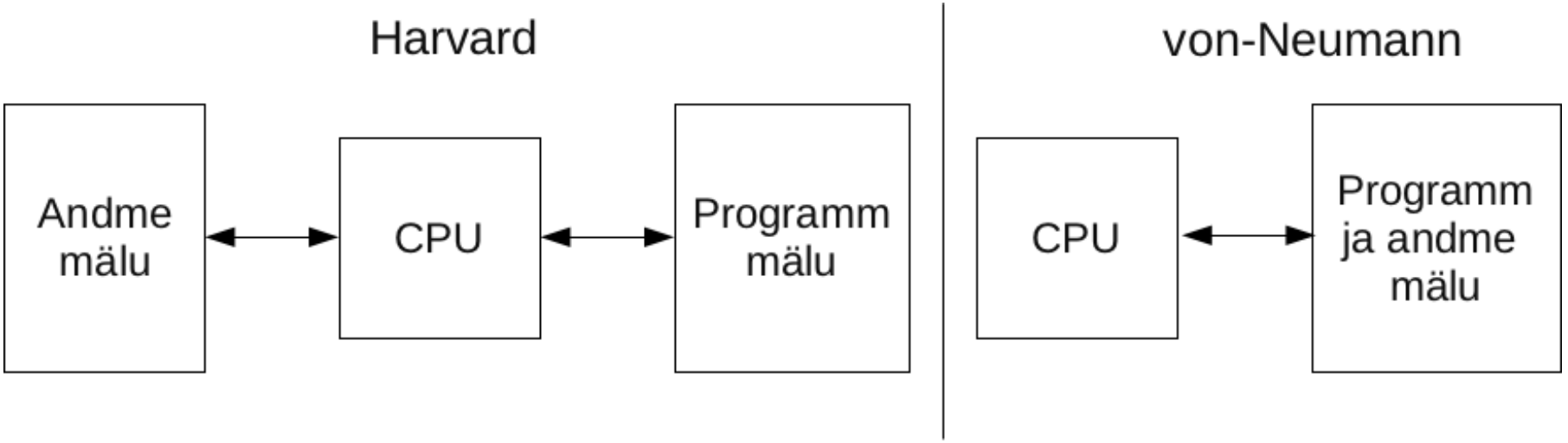
Mikrokontrollerite riistvara

MMU – Memory Management Unit

- MMU on vajalik nii mälupiirkondade kaitsmiseks kui ka virtuaalse aadressi “tõlkimiseks” füüsiliseks aadressiks



Kaks erinevat arhitektuuri



- Harvardi arhitektuuriga arvutil on võimalik pöörduda sama-aegselt mõlema mälu poole
- Von Neumann arhitektuur on oluliselt lihtsam, kuid puhtal pudelikaelaks võib olla mälu poole pöördumine

Harvardi ja von-Neumanni erinevused

- Harvardi arhitektuuriga mikrokontrollerid võivad olla mõningatel juhtudel kiiremad kui von-Neumani omad
  - Eraldi programm ja andmed
- Harvardi arhitektuuriga mikrokontrolleritel on (üldiselt) võimalik ainult ROM'is olevat programmi täita
  - Bootloader?
  - Tarkvara uuendamine?
  - MMU

Harvardi ja von-Neumanni erinevused jätk

- GCC ja C teegid on algselt tehtud von-Neumanni arhitektuuri jaoks ja seetõttu käsitletakse mõndasid Harvardi arhitektuuriga seonduvaid asju teisiti
  - Näiteks Harvardi arhitektuuriga mikrokontrolleritel on olemas käsud nagu printf_P või programm-mälust lugemise käsud (pgm_read_byte)
- von-Neumanni arhitektuurile on (natukene) lihtsam programmi kirjutada

Põhilised mikrokontrollerite tüübid

- 8 bitised
  - Atmel'i poolt toodetud 8 ja 8/16 bitised AVR mikrokontrollerid
  - Microchip'i poolt toodetud 8 bitised PIC mikrokontrollerid
- 16 bitised
  - Texas Instruments'i MSP430 tootepere
- 32 bitised
  - Atmeli AVR32 tootepere
  - Erinevad ARM mikrokontrollerid

Tüüpilised mikrokontrollerite lisad

- I/O liidesed
- Ostsillaatorid
- Riistvaralised loendurid ja taimerid
  - Tavaliselt suudavad loendurid ka PWM modulatsiooni genereerimisega toime tulla
- Seriaal liidesed
  - UART/USART, SPI, TWI/I2C
- AD muundid ja/või analoog komparaatorid
  - Enamustel juhtudel on 10 bitine AD muundi
  - Tavaliselt võimaldavad kuni 200 ksp/s (ühe kanali peale)
- JTAG või mõni muu debugimise liides

Mitte nii tavalised mikrokontrollerite lisad

- DMA
  - Võimaldab tunduvalt lihtsamalt analoog mõõtmisi teha
- USB
- CAN ja/või LIN
- Integreeritud traadita ühenduse võimalus
  - Tavaliselt on toetatud ZigBee kuid on näha olnud ka teisi protokolle
- DA muundi
  - Sobilik mootorite juhtimiseks
- AES ja DES krüpto lisad

AVR mikrokontrollerid I

- Loodud 1996 Atmeli poolt
- Harvardi arhitektuuriga
- Mikrokontrolleri loomisel on silmas peetud C kompilaatorite eripära
  - Sobib päris hästi mikrokontrolleritel kasutatava C keele õppimiseks
- Sobib väga paljutele rakendustele, kuid:
  - Ei ole just kõige sobilikum esmaseks assembleri õppimiseks
  - Analoog mõõtmistega tegelemine on keerukas

AVR mikrokontrollerid II

- Kõik vajalik tarkvara on tasuta saadaval internetis:
  - GCC pordist on C, C++ ja Ada kompilaator ning GNU linkur olemas
  - Programmatorite PC tarkvarast on Atmelil mitmeid erinevaid programme AVRStudio'ga kaasas ja vabavarana on Avrdude
  - Debuggeritest ja simulaatoritest on olemas AVRStudioga kaasasolev debugger aga on olemas ka GDB AVR'i port ja AVR'i vabavaraline simulaator
- Väga palju erinevaid programmatoreid ja katseplaate

AVR mikrokontrollerid III

- TinyAVR
  - 6–32-väljaviiguga korpuses, 0.5 kB – 8 kB flashmälu
  - Suhteliselt vähe integreeritud seadmeid
- MegaAVR
  - 28–100-väljaviiguga korpuses, 4 kB – 256 kB flashmälu
  - Korrutamise käsud ja suhteliselt palju erinevaid integreeritud lisaseadmeid
- XMEGA
  - 44–100-väljaviiguga korpuses, 16 kB – 384 kB flashmälu
  - Korrutamise käsud, DMA, "Event System", krüpto funktsioonid, DAC ja palju muid integreeritud lisaseadmeid

PIC mikrokontrollerid I

- Harvardi arhitektuuriga
- Populaarsed hobiprojektides
- Väga palju tasuta arendustarkvara
  - Kuid puudub GCC kompilaator
  - Peamiselt on arenduse tarkvara Windowsi all
- Väga palju erinevaid programmatoreid
- Peamiseks programmerimise keeleks on assembler
  - Sobib väga hästi assembleri algõppeks
- Suhteliselt keerukas kasutada suuremates ja keerukamates projektides
  - Katkestustele vastamine on külaltki keerukas

PIC mikrokontrollerid II

- PIC10
  - Minimaalselt lisaseadmeid
  - Väga väike voolutarve
  - 6-8 väljaviiguga korpus
- PIC12
  - Suhteliselt vähe lisaseadmeid
  - 8-14 väljaviiguga korpus
- PIC16
  - Piisavalt erinevaid lisaseadmeid
  - 14-44 väljaviiguga korpus
  - Kasutatakse üpriski palju hobiprojektides

PIC mikrokontrollerid III

- PIC18
  - Palju erinevat lisaseadmeid
  - 18-100 väljaviiguga korpus
  - Erinevalt teistest 8 bitistest PIC seeria mikrokontrolleritest on peamiseks arenduse keeleks C

MSP430 mikrokontrollerid

- Von-Neumani arhitektuuriga
- Arendatud 90'ndate alguses Texas Instruments'i poolt
- Väikse energiatarbega
  - Algselt olid mõeldud vee- ja energiamõõtjatele
- Väga lihtsalt kasutatav analoog muundi
- Olemas TI enda poolt pakutav arendus keskkond ja samaväärne GNU arenduskeskkond
- Ei ole kõige parem valik mikrokontrollerite õppimiseks
- Suhteliselt vähe vabavara programmatoreid

AVR32 ja PIC32 mikrokontrollerid

- ARM mikrokontrolleri analoogid Atmel'i ja Microchip'i poolt
- Mõlemad on Harvardi arhitektuuriga
- Mõlematel on GCC tugi olemas
  - Microchip'il on mitteametlik GCC tugi
- Väga palju erinevaid integreeritud seadmeid
  - Kuigi neil on integreeritud AD muundi ei pruugi see olla kõige täpsem
- Suhteliselt keerukad ning algtaseme õppe jaoks mitte eriti sobivad

ARM mikrokontrollerid

- Algselt loodud x86 asenduseks, kuid lihtsa ehituse ja väikse voolutarbe tõttu pigem sobivad sardsüsteemidesse
- ARM7 on von-Neumanni arhitektuuriga, ARM9 on Harvardi arhitektuuriga
- Suuremad kontrollerid ei ole eriti sobivad täpseteks analoogmõõtmisteks
- Väga lai arendustööristaade valik
  - On olemas GCC ja GNU linkeri port
- Ei ole kõige sobilikum algajatele

Kernel / OS

Kerneli ülesanded

- Rakenduste juhtimine
  - Mäluhaldus
  - Rakendustele tööaja eraldamine
    - Planeerimine, ajajaotus, saalimine (swap), …
- Teenused
  - Kommunikatsioon
    - Rakenduste vaheline, internet, …
  - Failisüsteem
  - Muud päringud
    - RTC, …
  - Füüsiline s/v, …

Kernelid üldiselt

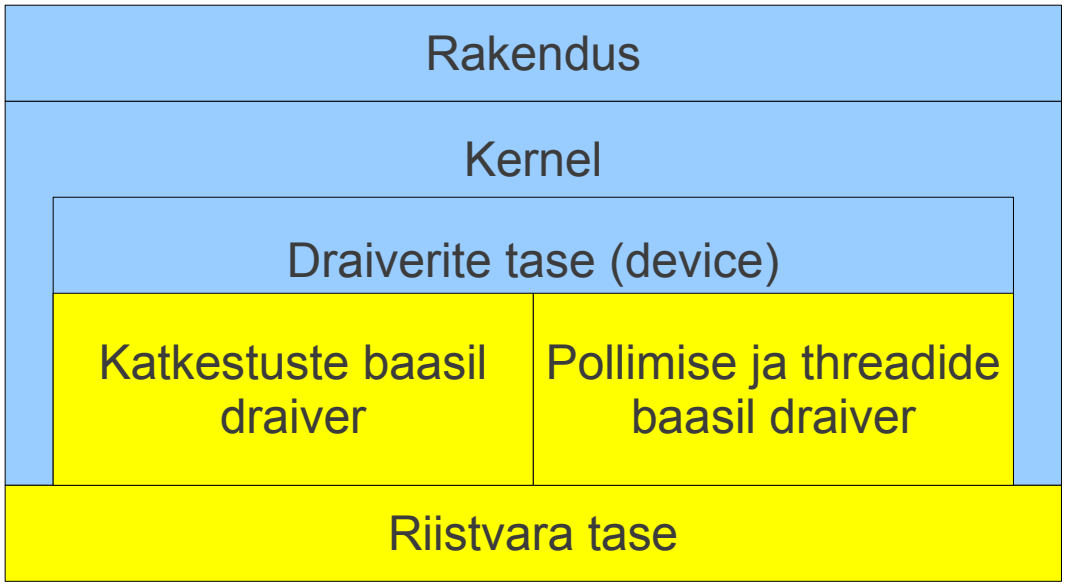
- Kerneleid saab üldiselt jaotada ressurssi nõudluse järgi kolme gruppi:
  - 8/16 bitiste kontrollerite kernelid
  - Väiksemate 32 bitiste kontrollerite kernelid, mis on üldiselt ilma MMU'ta ja vähese mäluga
  - Protsessorite millel on MMU, palju mälu ning ei pea eriti energiat kokku hoidma
- Arenduses on alati parem kasutada kernelit, millel on rohkem draivereid ja lisaprogramme kaasas
  - Tihtipeale läheb kerneli portimine ühelt arhitektuurilt teisele tunduvalt lihtsamini kui teisele kernelile näiteks võrgu stacki külge panek

Sard OS'i kerneli variandid

- Kernel koos draiveritega
- Kernel ilma draiveriteta – kasutaja peab ise draiverid kirjutama
- „Tavaline“ kernel RT kerneli peal

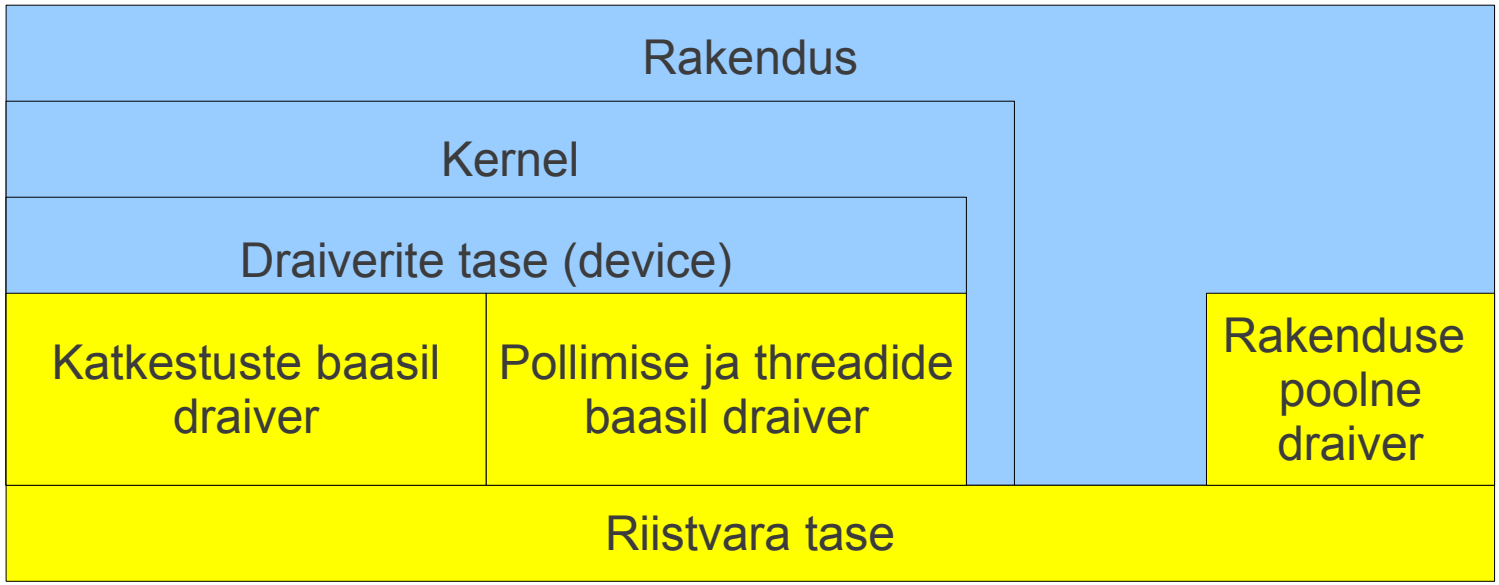
Standartse OS'i ülesehitus

- Kernel on eraldatud rakendusest
- Rakendusel ei ole võimalik otse riistvaraga suhelda
  - Riistvaraga suhtlemine on kerneli privileeg
- Reeglina ei ole mõeldud real-time rakenduste jaoks
  - Võib oma ülesandeid täita erineva kiirusega



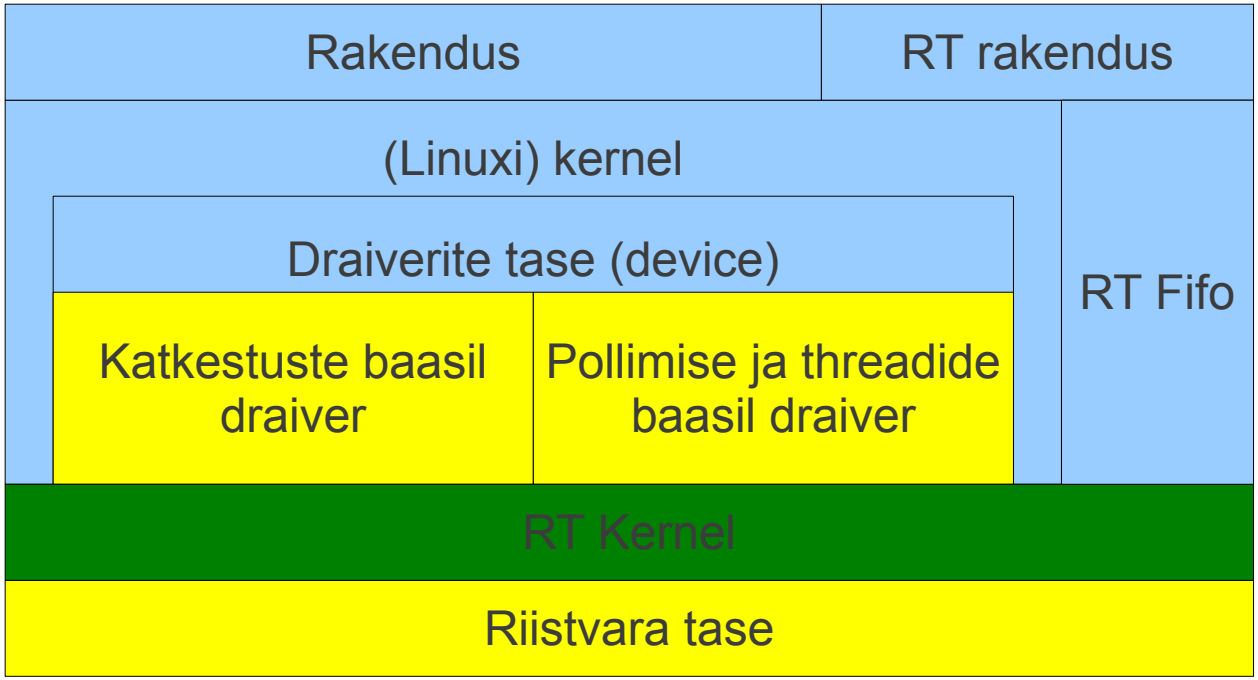
Sard OS'i ülesehitus

- Rakendus võib olla suhteliselt tihedalt seotud kerneliga
- Rakendusel võib olla kas läbi kerneli või siis ka otsene juurdepääs riistvarale



Üks võimalike Real-Time OS'i variante

- RT kernel tegeleb otse riistvara juhtimisega ja ühtlasi tagab ka RT rakendustele kiire täitmise
- RT rakendustel on võimalik suhteliselt lihtsalt riistvaraga suhelda



Real-time kernel vs „tavaline“ kernel

- RT kernel (hard real-time)
  - Võimaldab kiiresti reageerida välistele signaalidele
  - Rakendustel määratletud deadline'd
  - Reageerimise ajad mikrosekundi kandis
  - Miinuseks on suurem keerukus või kõrge hind
- „Tavaline“ kernel (soft real-time)
  - Oluliselt suurem arv rakendusi kui RT kernelile
  - Palju kasutajaid
  - Parem riistvara tugi
  - Reageerimise ajad millisekundi kandis (õigemini mõnest millisekundist mõnesajani)
  - Tunduvalt odavam, kuid võib tahta rohkem ressurssi

... kuid reaalselt

- Soft-realtime kernelid on piisavalt head enamustele toodetele
  - Soft-realtime ei sobi paljudel juhtudel meditsiinis, lennunduses, autodes, robotites
- Kui just pole tegemist mingite väga kiirete sündmustega ei ole RT kerneli väga suuri eeliseid tavalise kerneli ees

- Draiverite tasemel on võimalik teha soft-realtime kerneliga realtime rakendusi
- 8 bitiste kontrolleritega pole eriti keerukas teha hard-realtime süsteemi

Preemptive kernel

- Võimaldab igale protsessile oma ajapilu
  - Kui üks protsess võtab kõik protsessori resursi ära siis ei mõjuta see teisi protsesse
- Suhteliselt lihtne programme või kerneli osasid kirjutada

Cooperative kernel

- Iga protsess peab ise andma protsessori järje teisele protsessile üle
  - Üks vigane programm mõjutab kogu süsteemi tööd
- Korralikku cooperative kernelit või programmi on suhteliselt raske teha
- Korralikult tehtuna on võimalik väga täpselt määrata kui palju saab iga protsess protsessori aega
- Tavaliselt käib protsessori aja üleandmine sellise käsuga mis sisaldab endas **sleep** käsku
- NB! Kui on olemas **Yield** käsk siis on tavaliselt võimalik ümber teha preemptive kerneliks

Sardsüsteemides levinumad kernelid

- NutOS
- FreeRTOS
- Contiki
- eCos
- Linux

NutOS

- AVR, AVR32, ARM, H8300h ja m68k mikrokontrolleritele/protsessoritele
- Cooperative kernel
- Kerneliga kaasas draiverid
- Determineeritud katkestute latentsused
  - Ei kehti mitme samaaegse katkestuse korral
  - Draiverite tasemel võimaldab väga hästi teha hard real-time rakendust
- Kerneliga kaasas üpris palju lisaprogramme, näidiseid ja teeke
- Arendatakse spetsiaalse riistvara jaoks

FreeRTOS

- Üle 20 erineva arhitektuuri (AVR, AVR32, MSP430, PIC, ARM.. jne)
- Cooperative või preemptive kernel
- Draiverid näidisprogrammidega kaasas, kuid kernelil endal pole draivereid
- Portimine on üpriski lihtne
- Paralleelselt arendatakse ka OpenRTOS'i (ilma GNU litsentsita) ja SafeRTOS'i (IEC 61508 sertifitseeritud)
- Arendatakse üldise riistvara jaoks, st. arendajad ise ei tooda riistvara

Contiki

- Peamiselt väikestele kontrolleritle (AVR, MSP430)
  - Üks vähesema mälunõudlusega kernel
- Kasutab peamiselt protothreads'e kuid võimalik kasutada preemptive threade
- Palju draivereid on kerneliga kaasas
- Kasutatakse väga palju ülikoolide juures uurimistööde läbiviimisel (peamiselt smart dust)
  - Contiki on see kernel kust on võetud väga paljude teistele väikestele kernelitele võrgu stacki

eCos

- Võimsamatele mikrokontrolleritele/protsessoritele
  - ARM, CalmRISC, FR-V, FR30, H8, IA32, 68K/ColdFire, Matsushita AM3x, MIPS, NEC V8xx, PowerPC, SPARC, SuperH
- Võimalik kasutada mitmeid erinevaid *schedulere*
- Draiverid on kerneliga kaasas
- Hea kasutada seal kus Linuxit ei saa kasutada
- Arendatakse üldise riistvara jaoks

Linux

- Pordid olemas praktiliselt kõikidele võimsamatele protsessoritele
- Standartselt kaasas üpris hea scheduler
- Võimalik kasutada real-time kerneli lisasid
- Draiverid kerneliga kaasas
- Problemaatiline kasutada väikese energiatarbega süsteemides

Energia säästmine kontrolleril

- Energia säästmine tööajal
  - Madala kontrolleri taktiga
  - „Kergete“ progarmeerimiskeeltega
  - Kontrolleri madala energia tarbega režiimiga
  - Operatsioonisüsteemi vastava režiimiga

Kontrolleri takt

- Kontrolleri takt ja energiatarve on omavahel lineaarses sõltuvuses
  - Tavaliselt antakse kas voolutarve/MHz'le või voolutarve/MIPS'le. Näiteks MSP430F551X on 200uA/MIPS, kuid AT32UC3 on 300uA/MHz
- Väikse energiatarbe saavutamiseks võib tuua kontrolleri takti võimalikult madalaks
  - Takti võimalikult madalaks toomine toimib hästi ainult lihtsate programmidega – keerukamad ei pruugi kõikide katkestustega toime tulla

Energia sääst ja programmerimiskeeled

- Interpreteeritavad keeled (Java) ja ka mõningatel juhtudel „turvalised“ keeled (Ada) nõuavad töötamiseks rohkem energiat
  - Mida pikemalt kontroller on aktiivses režiimis seda rohkem ta energiat kulutab
- Kõrgemataasemelised keeled (Java) ei suhtle ise riistvaraga ja seetõttu peab kernel protsessori režiimidega tegelema
  - Java puhul saaks riistvaraga suhtlemise JNI'ga teha

Mikrokontrolleri madala energia tarbega režiimid

- Enamustel mikrokontrolleritel on võimalik kasutada madala energiatarbega režiime
  - Näiteks 8 bitisel AVR mikrokontrolleril on kuus erinevat madala enerigatarbega režiimi, 16 bitisel MSP430 mikrokontrolleril on viis kuni seitse erinevat madala enerigatarbega režiimi
- Mikrokontrolleril on võimalik lülitada mittekasutatavad liidesed välja
  - Liideste välja lülitamine ei anna väga suurt energia kokkuhoidu, tavaliselt hoiab nii kokku kuni 20%

Operatsioonisüsteemide valmisolek

- Enamus kerneleid ei tegele pidevalt energia säästu režiimide kontrollimise ja vahetamisega
  - Selliseid lahendusi kernelis on suhteliselt keerukas teha
- On võimalik sundida kernelit/mikrokontrollerit minema madala enerigatarbega režiimi
  - Võib tekkida probleeme riistvara suhtlemisel

Küsimusi?