

Pannon Egyetem

Villamosmérnöki és Információs Tanszék



Digitális Áramkörök

(Villamosmérnök BSc /
Mechatronikai mérnök MSc)

14. hét: Beágyazott rendszerek alapjai.
Mikrovezérlők.

Előadó: Dr. Vörösházi Zsolt

voroshazi.zsolt@virt.uni-pannon.hu

Kapcsolódó jegyzet, segédanyag:

- <http://www.virt.uni-pannon.hu>
 - Oktatás → Tantárgyak → Digitális Áramkörök (Villamosmérnök BSc / Mechatronikai mérnök BSc/MSc).
- Fóliák, óravázlatok (.ppt)
- Feltöltésük folyamatosan

Ajánlott és felhasznált irodalom

- Fodor Attila, Dr. Vörösházi Zsolt: Beágyazott rendszerek és programozható logikai alkatrészek (TÁMOP 4.1.2) Egyetemi jegyzet (2011)



http://www.tankonyvtar.hu/hu/tartalom/tamop425/0008_fodorvoroshazi/Fodor_Voroshazi_Beagy_0903.pdf

(1. Beágyazott rendszerek fejezetrész)

■ Microcontrollers

 <https://en.wikipedia.org/wiki/Microcontroller>

 <https://www.elprocus.com/microcontrollers-types-and-applications/>

 <https://www.elprocus.com/difference-between-avr-arm-8051-and-pic-microcontroller/>

 <https://www.elprocus.com/difference-between-arduino-and-raspberry-pi/>



Beágyazott rendszerek

Bevezetés

Beágyazott Rendszerek

- A beágyazott rendszer (**Embedded System**) a (számítógépes) **hardver-** és **szoftver**elemeknek kombinációja, amely kifejezetten egy adott funkciót, *specifikus* (vezérlési) feladatot képes ellátni, szemben az általános célú számítógép rendszerekkel.
- A beágyazott rendszerek olyan számítógépes eszközöket tartalmazhatnak, amelyek alkalmazás-orientált célberendezésekkel (**ASIC, ASSP, FPGA, MCU, MPU, DSP**, stb.), vagy komplex alkalmazói rendszerekkel (akár OS) szervesen egybeépülve akár azok **autonóm** működését is képesek biztosítani.
- A **programozható** beágyazott rendszerek olyan programozói interfésszel vannak ellátva, amelyek általában sajátos szoftver (firmware) fejlesztési stratégiákat és technikákat követelnek meg.

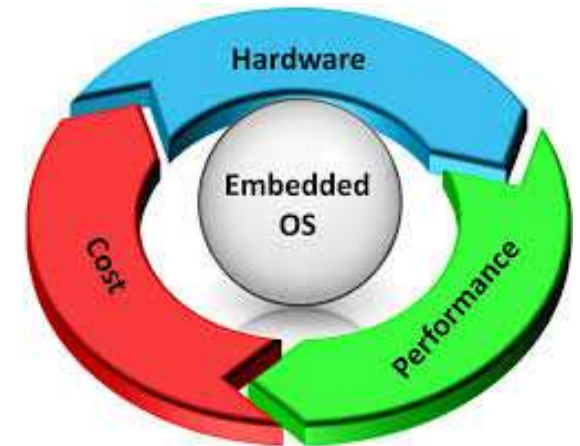
Néhány fontos alkalmazási terület

- Autóipari alkalmazások: beágyazott elektronikus vezérlők
 - Biztonságkritikus: központi elektronikai vezérlő (ECU), motorvezérlés, fékrásegítő, sebességváltó, blokkolásgátló vezérlés (ABS), kipörgésgátló (ESP) légszák
 - Utas központú (komfort) rendszerek: szórakoztatás, ülés/tükör ellenőrzés stb.
- Repülőgép-ipari és védelmi alkalmazások
 - Repülésirányító rendszerek (fedélzeti navigáció, GPS vevő), hajtómű vezérlés, robotpilóta
 - Védelmi rendszerek, radar rendszerek, rádió rendszerek, rakétavezérlő rendszerek
- Gyógyászati berendezések:
 - Orvosi képfeldolgozás
 - Jelmonitorozás (PET, MRI, CT)
- Hálózati/ telekommunikációs rendszerek (modem, router stb.)
- WSN: Vezeték nélküli szenzorhálózatok (motes)
- IoT: Intelligens, vagy smart rendszerek
- Háztartási gépek, ill. fogyasztói elektronika
 - mobiltelefon, PDA, PNA, digitális kamera, nyomtató stb.



Általános követelmények

- Dedikált funkció
 - Jól körülhatárolt (alkalmazás specifikus) funkció(k) támogatása
- Szigorú követelmények
 - Alacsony költség (**C**ost)
 - Gazdaságosság (**E**conomy) - lehetőleg minimális alkatrészből épüljön fel
 - Gyors működés (**S**peed)
 - Alacsony disszipáció (**P**ower)
- Valós idejű (real-time) működés és válasz
 - a környezetet folyamatos monitorozása, és beavatkozás
- Hardver-, és szoftver részek elkülönült, de együttes tervezése (co-design), tesztelése (co-simulation), ellenőrzése (co-verification)



Alapkövetelmények:

- **Idő:** Egy bekövetkező esemény kezelését a beágyazott rendszer egy *meghatározott* időn belül kezdje el.
- **Biztonság:** olyan rendszer vezérlése, amely hibás működés esetén egészségkárosodás, és komoly anyagi kár nélkül kezeli a bekövetkező eseményt.

E filozófia mentén a beágyazott rendszerek két *alcsoportját* lehet definiálni:

- **Valós idejű rendszer (v. idő kritikus):** melynél az időkövetelmények betartása a legfontosabb szempont,
- **Biztonságkritikus rendszer:** melynél a biztonsági funkciók sokkal fontosabbak, mint az időkövetelmények betartása.

Megjegyzés: A valóságban nem lehet ilyen könnyedén a beágyazott rendszereket csoportosítani, mert lehetnek olyan valós idejű rendszerek is, melyek rendelkeznek a biztonságkritikus rendszerek bizonyos tulajdonságaival. Szabványok és a törvények szabályozzák azt, hogy milyen alkalmazásoknál kell kötelezően biztonságkritikus rendszert alkalmazni (pl. ADAS ISO 26262).

Valós-idejű rendszerek

A követelmények szigorúsága alapján kétféle valós-idejű (real-time) rendszert különböztethetünk meg:

- **hard real-time rendszer:** *szigorú* követelmények vannak előírva, és a *kritikus* folyamatok meghatározott *időn* belül kell, hogy feldolgozásra kerüljenek,
- **soft real-time rendszer:** a követelmények kevésbé szigorúak, és a *kritikus* folyamatokat a rendszer mindössze nagyobb *prioritással* dolgozza fel.

Ütemezés (scheduling)

A (valós-idejű) operációs rendszerek (**OS/RTOS**) számára is kritikus feladat az ütemezés és az *erőforrásokkal való optimális gazdálkodás*. Mivel minden rendszer, valamilyen periféria segítségével kommunikál a környezetével, ezért fontos a perifériák valós-idejű rendszer követelményeinek megfelelő módon történő kezelése: a válaszidő betartásához az eseményt lekezelő *utasítás sorozatot* végre kell hajtani. Az utasítássorozat lefutása *erőforrásokat* igényel, melyeket az operációs rendszernek kell biztosítani, hogy hozzá tudja rendelni az időkritikus *folyamatokhoz*.

A processzorok **ütemezésének** következő szintjeit lehet megkülönböztetni:

- Hosszú-távú (long term) ütemezés vagy munka ütemezés,
- Közép-távú (medium term) ütemezés,
- Rövid-távú (short term) ütemezés.

Ütemezés szintjei

Az operációs rendszerek magja (kernel) tartalmazza az ütemezőt.

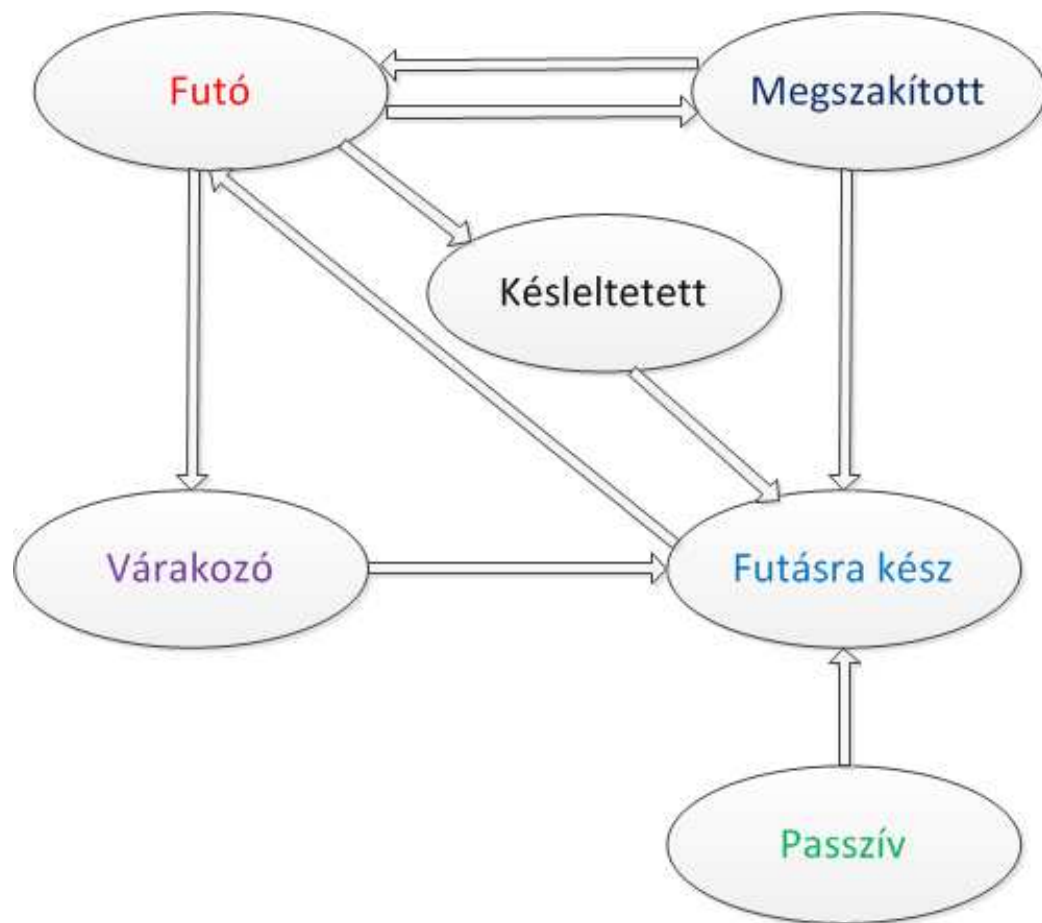
- **A hosszú-távú ütemezés** feladata, hogy a *háttértáron* várakozó, még el nem kezdett munkák közül meghatározza, melyek kezdjenek el futni, a munka befejeződésekor ki kell választania egy új elindítandó munkát. A hosszú-távú ütemezést végző algoritmusnak ezért *ritkán* kell futnia.
- **A közép-távú ütemezés** az időszakos *terhelésingadozásokat* hívatott megszüntetni, hogy a nagyobb terhelések esetében ne legyenek időtúllépések. A középtávú ütemező algoritmus ezt úgy oldja meg, hogy bizonyos (nem időkritikus) folyamatokat *felfüggeszt*, majd *újraaktivál* a rendszer terhelésének a függvényében. Folyamat felfüggesztése esetén a folyamat a *háttértáron* tárolódik, az operációs rendszer elveszi a folyamattól az erőforrásokat, melyeket csak a folyamat újraaktiválásakor ad vissza a felfüggesztett folyamatnak.
- **A rövid-távú ütemezés** feladata, hogy kiválassza, hogy melyik futásra kész folyamat *kapja meg a processzort*. A rövidtávú ütemezést végző algoritmus *gyakran* és *gyorsan* fut le, ezért az operációs rendszer mindig a *memóriában* tartja az ütemező kódját.

Ütemezés – további fogalmak

Az ütemezéssel és a programokkal kapcsolatban a következő alapfogalmak értelmezhetők:

- **Task:** Önálló részfeladat.
- **Job:** A task-ok kisebb, rendszeresen végzett részfeladatai.
- **Process:** A legkisebb futtatható programegység, egy önálló ütemezési entitás, amelyet az OS önálló programként kezel. Van saját (védett) memória területe, mely más folyamatok számára elérhetetlen. A task-okat folyamatokkal implementálhatjuk.
- **Thread:** Saját memóriaterület nélküli ütemezési entitás, az azonos szülőfolyamathoz tartozó *szálak* azonos memóriaterületen dolgoznak.
- **Kernel:** Az operációs rendszer alapvető eleme, amely a task-ok kezelését, ütemezést és a task-ok közti kommunikációt biztosítja. A kernel kódja hardver-függő (device driver), valamint hardverfüggetlen rétegekből együttesen épül fel.

Task állapotok



- **Passzív (Dormant):** Passzív (nyugvó) állapot, amely jelentheti az inicializálás előtti vagy felfüggesztett állapotot.
- **Futásra kész (Ready):** A futásra kész állapotot jelöli. Fontos a task prioritási szintje és az is, hogy az éppen aktuálisan futó task milyen prioritási szinttel rendelkezik, ezek alapján dönti el az ütemező, hogy elindítja e a taskot.
- **Futó (Running):** A task éppen tevékenyen fut.
- **Késleltetett (Delayed):** Ez az állapot akkor lép fel, mikor a task valamilyen *időintervallumig* várakozni kényszerül. Rendszerint szinkron időzítő (*timer*) szolgáltatás hívása után következik be.
- **Várakozó (Waiting):** A task egy meghatározott eseményre várakozik. (Ez rendszerint valamilyen I/O periféria művelet szokott lenni.)
- **Megszakított (Interrupted):** A task-ot megszakították, vagy a megszakítás kezelő rutin éppen megszakítja a folyamatot (IRQ, INT).

Ütemezési algoritmusok

Az ütemezési algoritmusoknak két fő típusa van:

- **Kooperatív** (=nem preemptív): A működési elve és alapötlete, hogy egy adott program vagy folyamat *lemond* a processzorról, ha már befejezte a futását vagy valamilyen I/O műveletre vár. Ez az algoritmus addig működik jól és hatékonyan, amíg a szoftverek megfelelően működnek (nem kerülnek végtelen ciklusba) és lemondanak a processzorról. Ha viszont valamelyik a program/folyamat nem mond le a processzorról vagy kifagy, akkor az egész rendszer stabilitását képes lecsökkenteni. A kooperatív algoritmus ezért soha nem fordulhat elő valós-idejű beágyazott operációs rendszerek esetében.
- **Preemptív**: az operációs rendszer részét képező *ütemező algoritmus* vezérli a programok/folyamatok futását. A preemptív multitask esetén az operációs rendszer elveheti a folyamatoktól a *futás jogát* és átadhatja más folyamatoknak. A valós idejű operációs rendszerek ütemezői minden esetben preemptív algoritmusok, így bármely program vagy folyamat leállása nem befolyásolja számottevően a rendszer stabilitását.

Task-ok közötti kommunikáció

Mivel a rendszer működése közben a task-ok egymással párhuzamosan futnak ezért gondoskodni kell arról, hogy egyazon I/O perifériát, erőforrást vagy memória területet két vagy több task ne használjon egyszerre, mert abból hibás rendszerműködés alakulna ki.

A következő ismert módszerek állnak rendelkezésre:

- **Mutex (kölcsonös kizárás):** ún. „locking” mechanizmus (csak a task amelyik zárolta, oldhatja fel)
- **Szemafor (semaphore):** „signaling” mechanizmus (egyik task jelez a másiknak, hogy végzett, és átveheti az erőforrást) ~ 1 bit információ
- **Események (event flags):** melyek több bit információ kicserélésére is alkalmasak.
- **Postaláda (mailbox):** amely akár komplexebb adatstruktúra átadására is szolgálhat.
- **Sor (queue):** amely több mailbox tömbjében lévő tartalom átadására szolgál.
- **Cső (pipe vagy FIFO):** amely direkt, folyamatos (akár streaming) kommunikációt tesz lehetővé két task között.

(Beágyazott) Operációs rendszerek

Többféle csoportosítás lehetséges:

- Általános célú, vagy **beágyazott OS**
- Valós-idejű (idő-kritikus), vagy nem-időkritikus
- Nyílt forráskódú, vagy licenszelhető, stb.

Általános célú processzorok operációs rendszerei (OS):

- MS-DOS, Linux, Windows, stb.

Beágyazott processzorok *valós-idejű* operációs rendszerei (RTOS):

- Linux
- Android
- Micrium uC/OS
- QNX
- RTLinux
- **Windriver VxWorks (RT)**
- Windows Embedded, IoT, stb...



Processzorok osztályozása

- Integráltság szerint:
 - uP/CPU: hagyományos mikroprocesszorok + fizikailag különálló memória + külső I/O periféria chipek (chipset)
 - **uC/MCU**: mikrokontrollerek: egyetlen chipen integrálva a processzor, a memória (ált. flash), és néhány I/O periféria
 - System-on-a-Chip (SoC) : egychipes rendszer
 - Kis méret és költség, alacsony disszipált teljesítmény
- Utasítás készlet szerint:
 - RISC vs. nem RISC (CISC) ISA – utasításkészletű architektúrák
- Utasítás / Adat memória hozzáférés szerint:
 - Von Neumann (közös) vs. Harvard architektúrák (elkülönült)

Néhány architektúra típus: Intel 8051, ARM, AVR, PIC, MIPS, IBM PowerPC, x86 (32/64), Sun SPARC, stb.

Technológiák és stratégiák

Élenjáró *technológiák* a beágyazott rendszerek tervezéséhez és megvalósításához – processzáló egységek csoportosítása:

- **(DSP):** Digitális jelfeldolgozó processzor alapú rendszerek
- **(MCU):** Mikrovezérlő-alapú rendszerek
- **(ASIC/ASSP):** Alkalmazás specifikus (berendezés orientált) integrált áramköri technológián alapuló rendszerek
- **(FPGA):** Programozható logikai kapuáramkörök technológián alapuló rendszerek
- **(MPU/GPU):** Mikroprocesszor, vagy grafikus processzor

- **SoC: System-on-a-chip:** olyan egychipes rendszer, amely a fentieket akár integrálva is tartalmazhatja!

Fejlesztési *stratégiák:*

- HW/FW/SW co-design: HW/FW/SW részek együttes tervezése
- HW/FW/SW co-verification: HW/FW/SW részek együttes ellenőrzése és tesztelése

I/O Perifériák

- Aszinkron soros kommunikációs interfészek: **RS-232**, RS-422, RS-485, stb.
- Szinkron soros kommunikációs interfészek: **I²C**, **SPI** stb.
- Univerzális soros busz: **USB**
- Multimédia kártyák: (SD) Smart Cards, (CF) Compact Flash stb.
- Hálózat: Ethernet (1GbE / 10 GbE / 100 GbE)
- Ipari hálózati un. „Field-bus” protokollok: **CAN**, **LIN**, **PROFIBUS**, **IO Link**, stb.
- Időzítő-ütemezők: PLL(s), Timers, Counters, Watchdog timers (WDT)
- Általános célú I/O-k (General Purpose I/O - **GPIO**): LED-ek, nyomógombok, kapcsolók, LCD kijelzők, stb.
- Analóg-Digitális/Digitális-Analóg (ADC/DAC) konverterek
- Debug portok: **JTAG**, **ISP**, ICSP, BDM, DP9, stb.

Mikrovezérlők

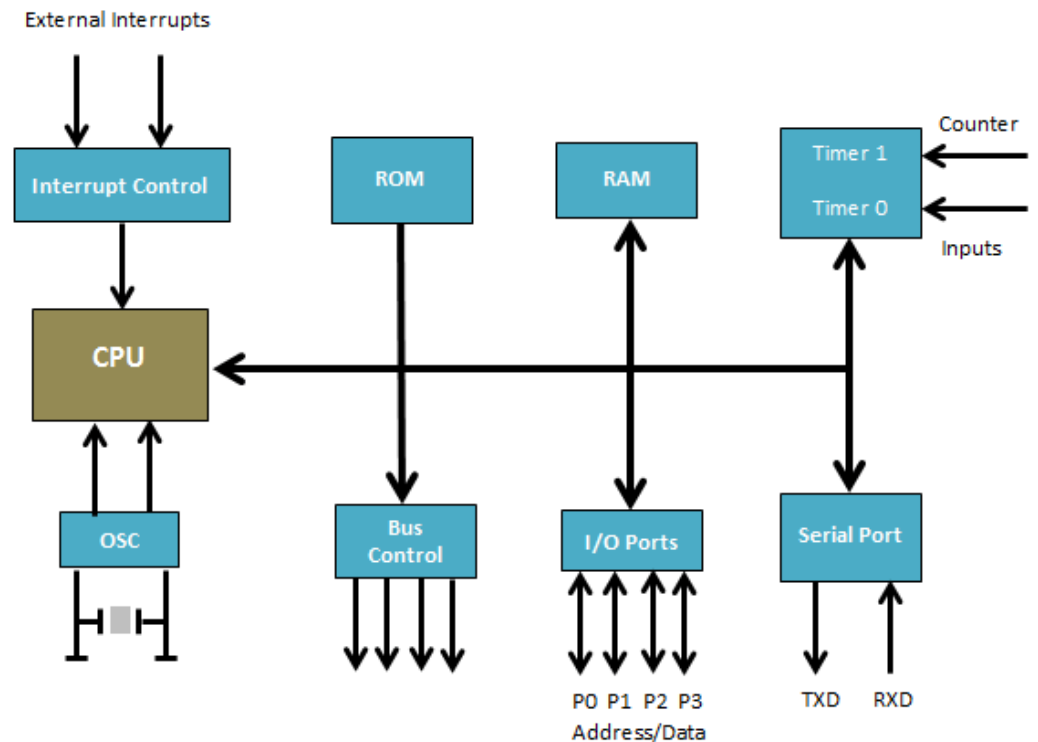


Mikrovezérlők

- Mikrovezérlő = mikrokontroller (MCU – Micro Controller Unit, vagy **UC**, vagy **μ -controller**)
 - *Egy olcsó, kis méretű és fogyasztású „számítógép” egy integrált áramkörön*
 - **Beágyazott rendszerek** alapvető építő eleme
 - *SoC = System-on-a-Chip, egychipes számítógép, feldolgozó egysége lehet az MCU (DSP, FPGA is akár)*
 - *Alkalmazásuk: lásd Beágyazott rendszerek.*

Mikrovezérlők általános felépítése

- Processzor (CPU/MPU) mag(ok)
- Memória (Harvard architektúra):
 - ROM/Flash/(EE)PROM: program (utasítás) memória (KB)
 - RAM: adat memória (KB)
- Interrupt/(WD)Timer/Counter
- OSC: oszcillátor (órajel)
- I/O perifériák, portok
 - Analóg, Digitális, vagy Mixed-signal funkciók
 - Manapság: IoT funkciók (érezékelők, aktuátorok, kommunikációs IF-ek)
 - ADC/ DAC konverterek



Mikrovezérlők tulajdonságai

■ Előnyök:

- Olcsó!
- Egyszerű felépítés
- Kis disszipáció (mW, μ W, nW – órajel/üzemmód függő)
- Könnyű integrálhatóság, beépíthetőség, széleskörű interfész támogatottság
- CPU: Speciálisan DSP műveleteket is támogathat (pl. Microchip dsPIC)

■ Hátrányok:

- Kis sebesség/ órajel (~10 - 100+ MHz)
- Relatív kis pontosság
 - (8-,16-,32-bites adat, cím, vezérlő, utasításbuszok)
- Korlátozott mértékű funkcionalitást biztosít
 - Komplex feladatokhoz több, vagy nagyobb számítási-tárolási kapacitású eszköz kell

Fontosabb gyártók

 Atmel®

- Atmel (ma Microchip): AVR8, AVR32, Intel 8051

 ARM®

- ARM: Cortex-M proc. magok

 MICROCHIP

- Microchip: PIC, dsPIC,

 TEXAS INSTRUMENTS

- Texas Instruments: MSP, C2000

 RENESAS

- Renesas: RL78-16, RX-32

 freescale
semiconductor

- Freescale (ma NXP),

 NXP

- NXP Semiconductors: LPC sorozat

 ST

- ST Microelectronics: STM8, STM16, STM32

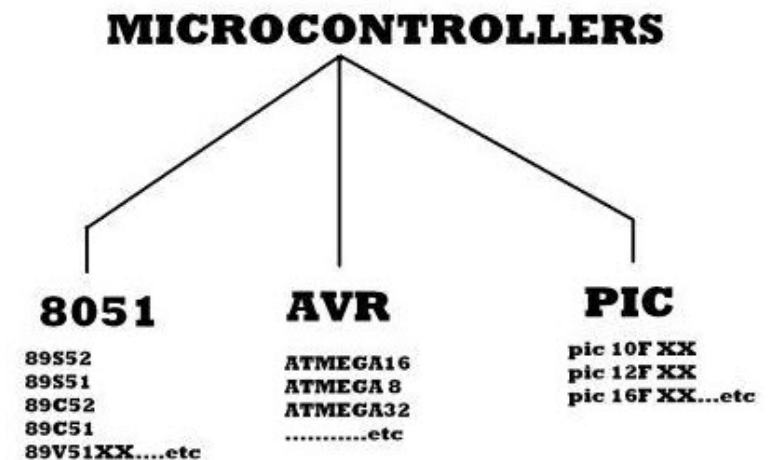
life.augmented

 CYPRESS
EMBEDDED IN TOMORROW

- Cypress: PSoC családok

Mikrokontrollerek osztályozása

- Kategóriák (gyártónként keveredhetnek):
 - **Busz-szélesség?** 8-/16-/32-bites
 - **Utasítás készlet (ISA)?**: RISC vs. CISC
 - RISC: csökkentett utasítás készlet
 - CISC: kibővített/komplex utasítás készlet
 - **Memória elvek?**
 - Neumann vs. Harvard elv
 - Belső vs. külső memória



Kategória: Busz szélesség

- 8-bit: belső busz 8-bites, ALU műveletek.
 - Pl: Intel 8031/8051, PIC1x, Motorola MC68HC11... családok
- 16-bit: 2x-es pontosság, belső busz, ALU
 - Pl: Intel 8051XA, PIC2x, Intel 8096, Motorola MC68HC12... családok
- 32-bit: 4x-es pontosság, belső busz, ALU
 - Pl: Intel/Atmel 251, PIC3x... családok

Kategória: Utasítás készlet (ISA)

ISA = Instruction Set Architecture

- CISC = Komplex/kibővített utasítás készlet:
 - Nagyobb architektúra, sok utasítással.
 - 1 utasításban több elemi utasítás van, (1 utasítás – több órajelig)
 - változó utasítás hossz -> nehezebb dekódolni, majd végrehajtani.
 - Komplex művelet kevesebb CISC utasítás sorral írható le!
- RISC = Csökkentett utasítás készlet:
 - csak a feladatra optimalizált, kevés számú utasítás van
 - azonos utasítás hossz -> könnyebb dekódolás (1 utasítás – 1 órajelig)
 - Komplex művelet sok elemi RISC utasítás sorral írható csak le.

Kategória: Memória

■ Szervezés elve:

- Von Neumann (Princeton) elv: adat/utasítás fizikailag közös memóriában
- Harvard elv: adat/utasítás fizikailag elkülönült memóriában

■ Hol található ?

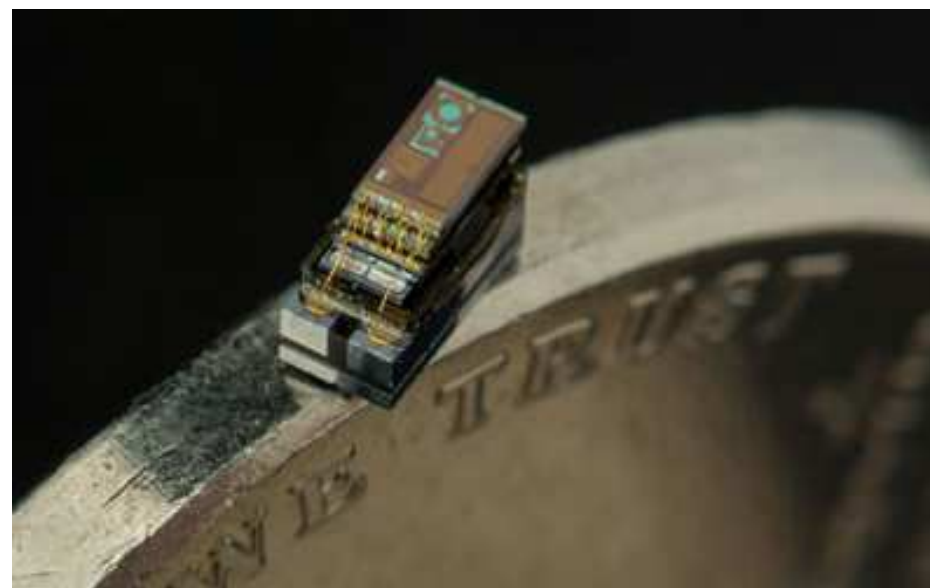
- Belső (on-chip) – beágyazott memória vezérlő (pl. Intel 8051)
- Külső (off-chip) memória vezérlő: nincs belső / on-chip memória (pl: Intel 8031 – nincs program memória)

MCU architektúrák (magok)

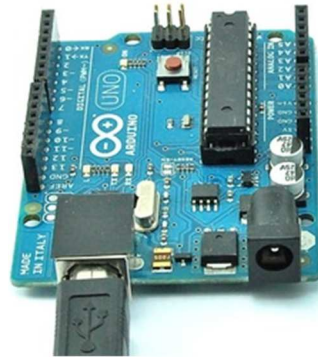
- ARM: harvard RISC
 - Manapság a legnépszerűbb, skálázható magokat gyártanak, több gyártó is integrálja a magokat (pl. STMicro)
- AVR (Atmel): Harvard RISC
 - Népszerű Atmega családok (pl. Arduino kártyák)
- Intel 8051: Harvard CISC
 - Népszerű, beágyazott MCU mag több gyártónál (pl Cypress)
- Renesas: RX-32 - Harvard CISC
- Microchip:
 - PIC16/18 ... - Harvard RISC
 - Gyorsabb, könnyebb programozhatóság

„Világ legkisebb számítógépe”

- 2018. jún (University of Michigan, USA)
- **M³ : Michigan Micro Mote: smart-sensor**
 - Hőmérséklet, nyomás,
 - Képkalkotó szenzor (160x160 pixel)
 - 1 mm² felületű!
 - 2 nA disszipáció (standby mód)
 - CPU + MEM + PWR
RF, battery



Arduino vs. Raspberry Pi



■ **Arduino** – Atmel/Microchip MCU alapú fejlesztő kártya

- Kisebb órajelű (~x10 MHz) MCU mag, kis belső memória, kis bitszélesség (8-, 16 bit)
- Nincs külső memória, nincs OS kezelése, nem real-time eszköz.
- Jó bővíthetőség: „shield”-ek
- Főként egyszerű szabványokat, GPIO-kat kezel, van ADC.
- Olcsó, népszerű, rengeteg szenzor illeszthető, de kisebb komplexitású fejlesztési célokra. Ára: \$5-15 (platform függő)



■ **Raspberry Pi** – ARM alapú általános célú sz.gép, fejlesztő kártya („single board computer)

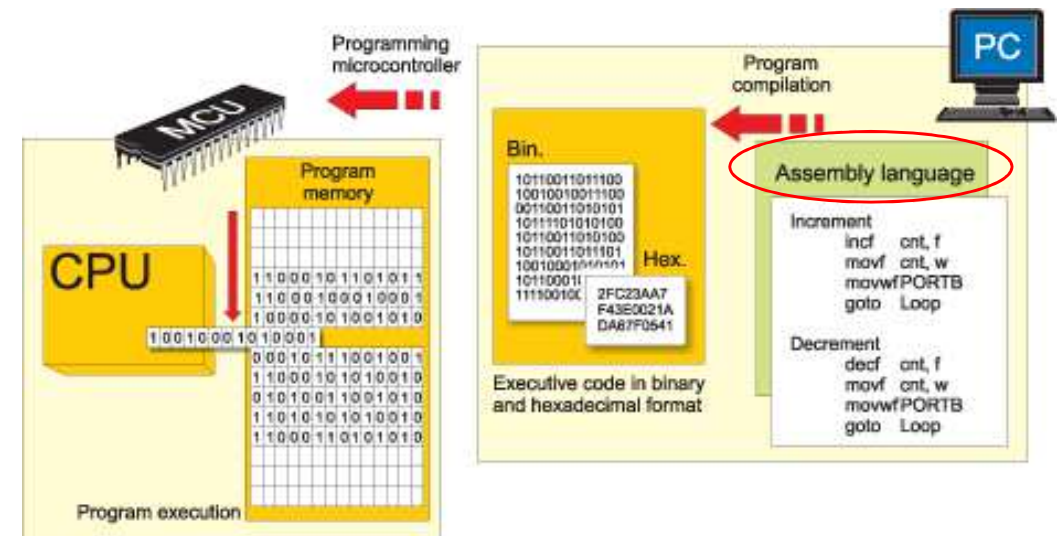
- Dedikált, nagy órajelű CPU magok (ARM 32/64 bit ~x100 MHz, memória (DDR3), GPU mag, HDMI stb.
- Bővíthetőség: SDCard (OS boot), WIFI, BLE, CamIF, de nincs ADC.
- OS/RTOS (HW-es) kezelése Nagyobb komplexitás, több funkció, de drágább.
- Ára: \$ 30- 50 (platform függő)

Mikrovezérlők programozása

■ Programozási nyelvek (compiler függő):

- ASM – assembly (régén, hagyományos)
- C (C++)
- Python, ...
- Egyéb: Interpreter FW elérhetővé tehet más nyelveket is: BASIC...

ASM → MCU



■ Integrált fejlesztő környezetek (IDE), pl.:

- Texas Instruments - Code Composer Studio
- Arduino IDE
- Microchip/PIC – MPLAB
- ARM – Keil MDK / ARM DS-5, stb.

C/C++ → MCU

