Biró Csaba – Geda Gábor – Kovács Emőd
Eszterházy Károly Főiskola
birocs@aries.ektf.hu; gedag@aries.ektf.hu; emod@ektf.hu

MIDDLEWARE-EK A ROBOTIKÁBAN

Absztrakt: Middleware a számítástechnikában manapság leggyakrabban használt, „legdivatosabb" fogalmak egyike. A robottechnikában elsődlegesen összetett robotirányítási rendszerekben használják. Egy úgynevezett glue szoftver, amelynek segítségével, a fejlesztők egyszerre el tudják érni az összes szoftver és hardver komponenst. Nemcsak a komponensek közötti kommunikációt segíti, hanem a rendszerekbe ágyazott alkalmazás független kód által segítséget nyújt a komponens alapú fejlesztéshez. Az utóbbi időben egyre több paradigma támogatja az elosztott és konkurens rendszerek fejlesztést. A middleware-ek többnyire különböznek egymástól. Egyrészt abban, hogy milyen kompromisszumokat és megszorításokat kell kötni, másrészt, hogy a robotika mely irányába mutatnak. Cikkünkben először megvizsgáltuk, hogy milyen elvárások fogalmazhatók meg, illetve milyen problémák merülnek fel összetett rendszer tervezésénél. Majd áttekintjük és bemutatjuk, hogy jelenleg milyen middleware megoldások *(Miro, Orca, Player/Stage rendszer, PEIS kernel, MARIE, ASEBA stb.)* léteznek a robotika területén, vizsgálva a különböző platformok általános céljait és alkalmazott módszereit.

Bevezetés

A middleware a számítástechnikában manapság leggyakrabban használt fogalmak egyike. Ezzel szemben a szó jelentése nem kiforrott, eltérő értelmezésekkel rendelkezik. Azt lehet mondani, hogy a middleware egy olyan elérhető szoftver réteg, mely a heterogén platformok és protokollok hálózati rétege és az üzleti alkalmazások között helyezkedik el. Leválasztja az üzleti alkalmazásokat bármilyen, a hálózati réteg okozta függőségről, melyet a heterogén operációs rendszerek, hardver platformok és kommunikációs protokollok okoznak.

Problémák összetett rendszerek tervezésénél

Egyik legfontosabb feladat az interfész megtervezése. Egy összetett rendszernél az interfészt úgy kell megalkotni, hogy az ne legyen bonyolultabb, mint az összes alrendszer kombinációja. Ellenkező esetben nem lehet kihasználni a valós idejű tervezés előnyeit. Ugyancsak nagy kihívást jelent egy újrafelhasználható komponensekből épített rendszer esetén megtalálni az egyensúlyt a teljesítmény és az egyszerű újrafelhasználhatóság között.

Fogalmi szinten, egy komplex robotvezérlő rendszer összetevőinek mindegyike besorolható a következő felsorolás valamelyikébe:

- **Kommunikáció:** Egyes összetevők közötti információcsere *(adatok, események, parancsok)*
- **Számítás:** Minden komponens bizonyos számításokat hajt végre, amelyek elvégzéséhez a rendszer biztosítja a megfelelő funkcionalitást.

- **Konfiguráció:** A pontos konfiguráció elengedhetetlen a tervezésnél a rendszer és az egyes komponensek implementációjánál. Szükséges a szoftver életének különböző fázisaiban: fordítási idő, futásidő,…
- **Koordináció:** A várt viselkedés és teljesítmény érdekében rendszer szinten az alkotóelemek tevékenységeit össze kell hangolni. A koordináció magába foglalja:
    - döntéshozatal,
    - ütemezés,
    - alrendszerek és/vagy összeköttetések aktiválása vagy inaktiválása.

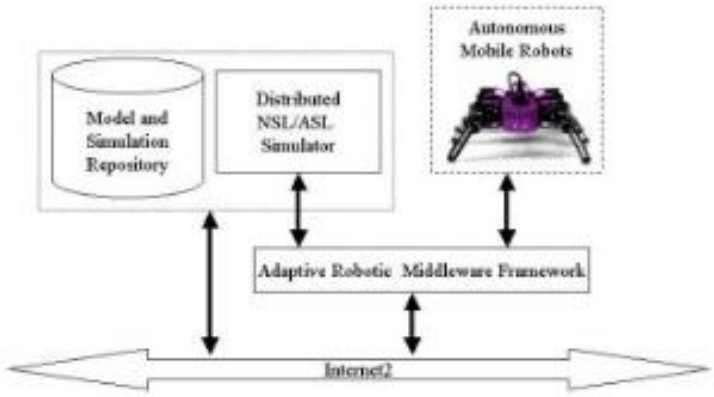
Alrendszerekből optimálisan összeállítani egy „nagy" rendszert még napjainkban is inkább művészet, mint tudomány. Legnagyobb kihívást az jelenti, hogy az alrendszerek amellett, hogy szorosan együttműködnek a nagy rendszerrel továbbra is megtartsák stabilitásukat.

Lehetséges megoldások különböző szoftverkomponensek összeállítására:

- explicit hivatkozásokkal összekapcsolt objektumosztályok létrehozása;
- olyan objektumosztályok létrehozása, amelyek nem tudnak egymásról.

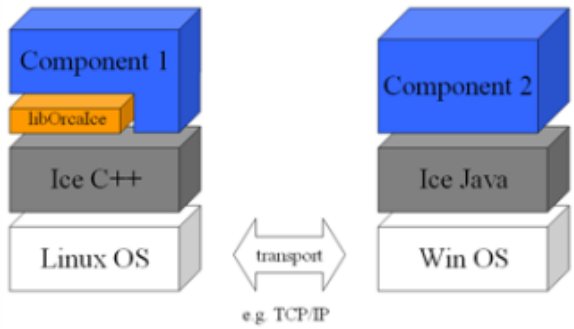
Middleware-ek a robottechnikában

Miro: Elosztott objektum orientált keretrendszer, mobil robotok irányítására. Az alapkomponenseket C++-ban fejlesztették ACE (Adaptive Communications Environment) környezetben. Az ACE egy objektum orientált multi platform keretrendszer, amelyet kifejezetten operációsrendszertől független interprocesszekre, hálózati és valós idejű kommunikációra terveztek. TAO (The ACE ORB) ORB (Object Request Broker) egy CORBA implementáció, amelyet nagy számítási teljesítmény eléréséhez és valós idejű alkalmazások futtatásához terveztek.



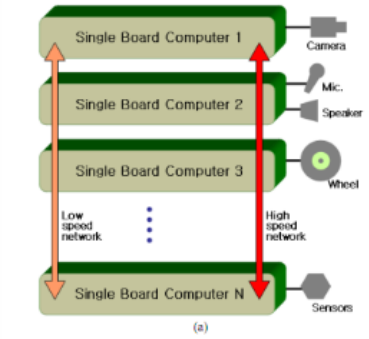
4. ábra MIRO elosztott architektúrája

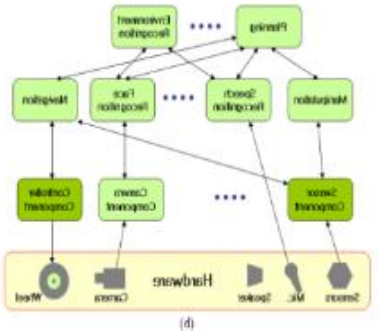
Orca: Olyan middleware keretrendszer, amely a komponens alapú fejlesztés segítését tűzte ki célul. Elsődlegesen olyan alkalmazások fejlesztésére tervezték, amely egy járművön levő elosztott érzékelő-hálózatot képes kezelni. Elosztott komponens alapú robottechnikai rendszerek implementálását teszi lehetővé, továbbá fejlesztők számára engedélyezi új interfészek definiálását és megadja annak lehetőségét, hogy módosítsák a kommunikáció mechanizmusait. Orca első verziója egy nyílt forráskódú CORBA implementáció, de a CORBA-nál jelentkező komplexitási problémák miatt az Ice-t használják. Ice egy egészen új szemléletet vezetett be az objektum orientált middleware-ek között. Biztosít egy az eddigieknél sokkal kisebb és konzisztensebb API-t, amelynek köszönhetően átláthatóbb implementációt tesz lehetővé, fejlett szolgáltatásokkal, és jobb teljesítménnyel.



5. ábra ORCA kommunikációs modellje

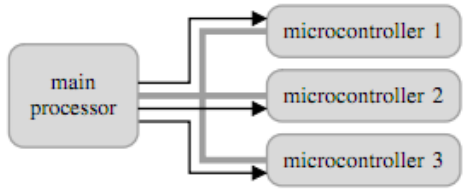
UPnP Robot Middleware: Az UPnP Robot Middleware fejlesztője a KIST (Korea Institute of Science and Technology). Az UPnP (Universal Plug and Play) architektúrát használja fel dinamikus robotok belső és külső szoftverintegrációihoz és irányításához. Az UPnP protokoll támogatja konfigurálás-mentes és egyenrangú hálózatok létrehozását PC-k, egyéb hálózati készülékek és vezeték nélküli eszközök között. Egy UPnP kompatibilis eszköz, képes dinamikusan csatlakozni egy hálózathoz, megszerzi az IP-címet, bejelenti a nevét, kérésre közli a képességeit, más eszközök jelenlétéről értesülni, valamint képes azok képességeiből tanulni. A DHCP és DNS-kiszolgálók szabadon választhatóak és csak akkor használhatóak, ha elérhetőek a hálózaton. Az eszközök automatikusan elhagyhatják a hálózatot anélkül, hogy otthagynának bármilyen felesleges állapotinformációt. Ezt a mechanizmust használják fel a robot komponenseinek konfigurálása. Mindezek tükrében egy robot képes körülötte lévő robotokat szenzorjai (kamerák, szenzorhálózatok, elektromechanikus eszközök) segítségével felfedezni és velük interakcióba lépni. Egy moduláris robot képes több akár komponensén keresztül is csatlakozni belső hálózathoz. Ha minden egyes komponens támogatja az *UPnP*-t, akkor leegyszerűsödik az összetevőinek összekapcsolásának és beállításának folyamata. Az automatikus felfedezés és konfigurációs mechanizmusok rendkívül fontosak dinamikus számítási környezetben. Intelligens komponenseinek köszönhetően képesek belső és külső szoftverkomponensekkel együttműködni.



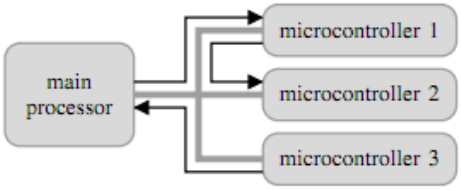


3. ábra Software konfiguráció *4. ábra Hardware konfiguráció*

ASEBA: Egy eseményvezérelt köztesszoftver, amely támogatja az elosztottságot (irányítást és hatékony erőforrás-kihasználást) a multiprocesszoros robotoknál. Több processzorral rendelkező robotok kommunikációjának segítésére tervezték.



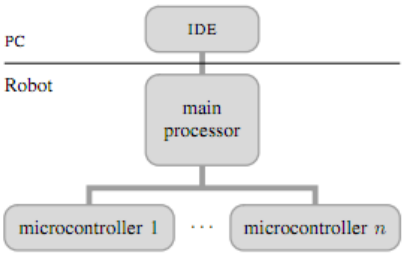
(a) classical architecture, master/slave bus. The main processor initiates all data transfer.



(b) event-based architecture, multi-master bus. Any node can initiate a data transfer.

5. ábra

Robotok a fő processzor mellett rendelkeznek mikrokontrollerekkel. A mikrokontrollerek külön-külön is működtethetnek és ellenőrizhetnek érzékelőket. A mikrokontrollerek képesek egymással kommunikálni, aszinkron üzeneteket küldeni. Minden egyes feladat, amit a mikrokontrollerek végeznek, csökkentik a fő processzor leterheltségét. Az ASEBA azáltal, hogy az egyes feladatokat mikrokontrollerek végzik, (amelyek kommunikálhatnak egymással és a fő processzorral) javít a robotok modularitásán és hatékonyságán.



6. ábra

Mindezeknek köszönhetően lehetővé válik, hogy dedikáljuk a fő processzort magas intenzitást igénylő feladatok elvégzésére (pl. látás). Az ASEBA biztosít egy integrált fejlesztői környezetet (szerkesztővel), saját fordítót és hibakeresőt.

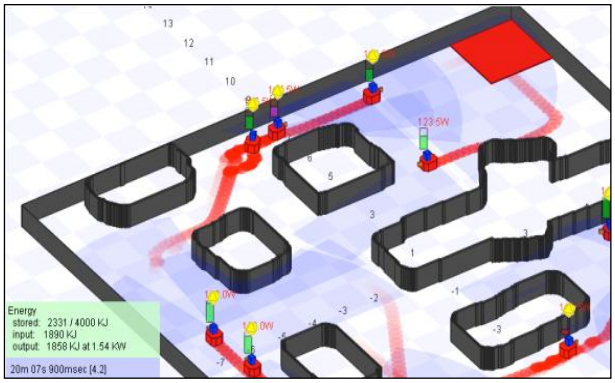
Player/Stage System: Player/Stage rendszer, mint köztesszoftver platform eszközöket és algoritmusokat biztosít mobil robotalkalmazásokhoz.

Player: Egy repository szerver indítókaroknak, szenzoroknak, és robotoknak. Itt minden egyes eszközhöz tartozik egy driver és egy interfész. Az interfészeket a kliens egyrészt arra használja, hogy alkalmazásokat indítson, másrészt pedig, hogy információt kapjon a szenzor működéséről. A drivereken implementálva vannak még az üzenetküldéssel kapcsolatos algoritmusok is.

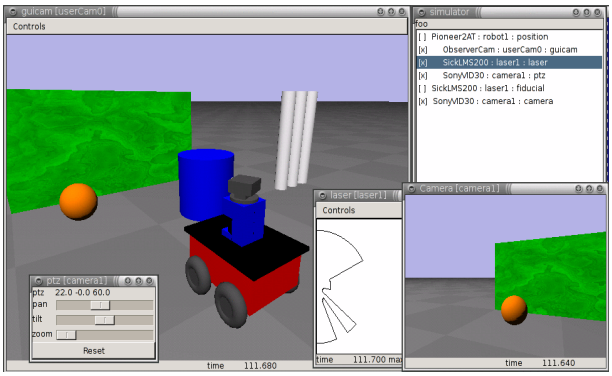
Stage: Egy grafikus alapú szimulátor, aminek segítségével az elkészített modelleket és eszközöket felhasználó által definiált környezetben lehet tesztelni.

A Player/Stage System háromrétegű architektúrája felosztja az alkalmazást három különálló rétegre:

- alkalmazásokhoz fejlesztett kliens oldali szoftver (C, C++,Java, Python),
- interfészek biztosít a különböző eszközökhöz és szolgáltatásokhoz,
- robotok szintje (érzékelők, működtetők).



8. ábra Stage szimulátor



9. ábra Gazebo 3D szimulátor

PEIS kernel: Az ETRI (Electronics and Telecommunications Research Institute, Korea) és CAASS (Centre for Applied Autonomous Sensor Systems, Sweden) kutatóközpontok közös munkájának eredménye. Tervezéskor a PEIS Ecology és az EPEIS (Ecology of Physically Embedded Intelligent Systems) koncepcióját vették alapul. Ezeknek a rendszereknek elsődleges célja, otthonunkban mindennapi feladataink elvégzésében segítséget nyújtó robotok vezérlése. A mobil robotokon lévő érzékelők és működtetők illetve automatizált háztartási gépek számára egyaránt használható egységes kommunikációs és együttműködési modell megalkotása.

ORiN: Egy interfészt biztosít Windows operációs rendszerrel rendelkező számítógépeknek, szabványos módszerek eléréséhez és robotikai rendszerek irányításához. Alapját a http és egyéb webes szabványok képezik, mint például XML és SOAP. Egy könnyen fejleszthető „olcsó rendszer". Az általuk kifejlesztett interfész segítségével egyrészt biztosítják a specifikáció és az implementáció elkülönítését, másrészt lehetőséget adnak arra, hogy ellenőrzött PC-ken lehessen dolgozni a robotikai alkalmazásokon.

MARIE: MARIE (Mobile and Autonomous Robotics Integration Environment). Céljuk egy rugalmas és elosztott komponensrendszer fejlesztése volt, amely egy rendkívül gyors alkalmazásfejlesztés tesz lehetővé.

A MARIE háromrétegű architektúrája:

- Mag
- Komponensek
- Alkalmazások

A mag feladatai:

- kommunikáció szervezése,
- alacsonyszintű üzemeltetési funkciók,
- elosztott számítási feladatok.

A komponens réteg feladata: komponenseket biztosít gyakran használt szolgáltatásokhoz.

Az alkalmazási réteg: támogatja integrált alkalmazások építését és irányítását.

MARIE szolgáltatásai egy rugalmas kapcsolódási felületet biztosítanak a különböző kommunikációs protokollok és alkalmazások felé. Az integrációhoz az ACE kommuni-

kációs keretrendszert használja. Központi komponensén kívül még négy funkcionális részből áll:

- alkalmazói felület,
- kommunikációs felület,
- kommunikációs ütemező,
- alkalmazás ütemező.

RSCA (Robot Software Communication Architecture): Az RSCA a Seoul National University által (QoS szolgáltatásaira építve) fejlesztett middleware. Fő erőssége, hogy valós idejű támogatást biztosít. QoS: Hálózatok és hálózati eszközök képessége az erőforrások meghatározott rend szerinti felosztására, és garantált sávszélesség biztosítására. A QoS-t támogató hálózatokon a magas prioritású üzenetek előnyben részesíthetők alacsonyabb besorolású társaikkal szemben, illetve konkurencia-helyzetben az előbbiek továbbítása az utóbbiak feltartóztatásával garantált sebességen biztosítható.

RT-Middleware: RT (Robot-Technology) a JARA (The Japan Robot Association) a METI (Japanese Ministry of Economy, Trade and Industry) és a AIST (National Institute of Advanced Industrial Science Technology) közös fejlesztése. RT-Middleware egy CORBA alapú implementáció. Kihasználva ennek az elosztott middleware-nek számos specifikációját. A prototípust OpenRTM-ben készítették el.

Felhasznált irodalom

http://www.corba.org/
http://www.cs.wustl.edu/~schmidt/ACE.html
http://www.cs.wustl.edu/~schmidt/TAO.html
http://orca-robotics.sourceforge.net/
Sang Chul Ahn, Jin Hak Kim, Kiwoong Lim, Heedong Ko, Yong-Moo Kwon and Hyoung-Gon Kim UPnP Approach for Robot Middleware
St´ephane Magnenat, Valentin Longchamp, Francesco Mondada ASEBA, an event-based middleware for distributed robot control
http://playerstage.sourceforge.net/
http://www.aass.oru.se/~peis/
Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar A Review of Middleware for Networked Robots
http://marie.sourceforge.net/wiki/index.php/Publications
http://www.openrtm.org/OpenRTM-aist/html-en/What20is20RTM.html
http://en.wikipedia.org/wiki/Robotics_middleware
Nader Mohamed, Jameela Al-Jaroodi, Imad Jawhar Middleware for Robotics
http://wiki.blender.org/index.php/Robotics:Middleware