

NEUMANN JÁNOS INFORMATIKAI KAR

# DIPLOMAMUNKA

A kezdőlapra mindenféle címer kell. Van L<sup>A</sup>T<sub>E</sub>Xminta rá?

OE-NIK HARASZTI GÁBOR  
0 0

**OE-NIK HARASZTI GÁBOR**

Budapest  
2012

# Tartalomjegyzék

<b>1. Rövid tartalmi összefoglaló a téma területéről, a feladatról</b>	<b>3</b>
<b>2. A megoldandó probléma megfogalmazása</b>	<b>4</b>
<b>3. A probléma fontossága, felvezetése</b>	<b>5</b>
<b>4. Az irodalom alapján a lehetséges megközelítési módok és megoldások áttekintése és elemzése</b>	<b>6</b>
4.1. Irodalmi áttekintés . . . . .	6
<b>5. A megoldási módszer kiválasztása, a választás indoklása</b>	<b>12</b>
<b>6. A részletes specifikáció leírása</b>	<b>13</b>
<b>7. A tervezés során végzett munkafázisok és tapasztalataik leírása</b>	<b>14</b>
<b>8. A megvalósítás leírása</b>	<b>15</b>
<b>9. Tesztelés</b>	<b>16</b>
<b>10. Az eredmények bemutatása, értékelése, hasonló rendszerek eredményeivel összevetése</b>	<b>17</b>
<b>11. A megvalósítás elemzése, alkalmazásának és továbbfejlesztési lehetőségeinek számbavétele</b>	<b>18</b>
<b>12. A szakdolgozat tartalmi összefoglalója magyarul és angol nyelven</b>	<b>19</b>
Irodalomjegyzék . . . . .	20
<b>13. Mellékletek</b>	<b>22</b>

## **1. fejezet**

**Rövid tartalmi összefoglaló a téma területéről, a feladatról**

## **2. fejezet**

### **A megoldandó probléma megfogalmazása**

## **3. fejezet**

### **A probléma fontossága, felsezetése**

## 4. fejezet

# Az irodalom alapján a lehetséges megközelítési módok és megoldások áttekintése és elemzése

A munka megkezdése előtt célszerű tájékozódni a probléma lehetséges megoldásairól a szakirodalomban, ami után a megoldási alternatívák már ki tudnak rajzolódni és lehetséges lesz szakmailag megalapozott javaslat megfogalmazására.

### 4.1. Irodalmi áttekintés

Galambos Péter cikkéből [1] megtudhatjuk, hogy habár sokáig megfelelt a robotikában a helyi kontrollerekről vezérelni a robotokat, a manapság szükséges számítási kapacitás igénynek ez már nem felel meg. Az új kihívások szükségessé teszik külső erőforrások bevonását is a munkába. A kapacitás bővítésnek több módja is van a lokális szerverektől a publikus felhő szolgáltatásokig bezárólag. A cikk bemutatja a hagyományos IT területtől némileg eltérő robotikai alkalmazások szükségleteit a felhő szolgáltatásokban.

Először ismertetésre kerülnek a főbb paradigmák a robotika kontextusában:

**Cloud computing:** azaz felhő számítástechnika, hozzáférés nagy, jól skálázható hálózati számítási kapacitásokhoz (végrahajtási, memória és disk) távolról. Ez az infrastruktúra lehet privát, publikus és hibrid.

**Fog computing:** azaz "kód" számítástechnika, olyan számítógép eszközöket jelent, amik a robot kontrollerek mellett helyben, mintegy gateway-ként teszik lehetővé azok illesztését a felhő szolgáltatásokhoz, interfészt jelentve a kétféle, nagyon különböző rendszer között, valamint képesek helyi, nagy tömegű, például a robot szenzorokból érkező adatok elő-

feldolgozására.

**Mist computing:** azaz pára számítástechnika **Meg kell találni a megfelelő magyar szavakat a Fog és Mist szavakra!**, a beépített eszközök számítási kapacitását jelenti, tipikusan előfeldolgozzák és szűrik a szenzorok adatait, mielőtt továbbítanák a kód és felhő szintjére.

A cikk továbbá kitér a felhőbe kiszervezett számítástechnika előnyeire és hátrányaira, az elfogadható késleltetés és adatforgalom tükrében. Minden esetben a tervező mérnök kezébe adva a döntés jogát az egyes szolgáltatások helyének megtervezésére az igények szerint. Bevezeti továbbá a **puppet robot**, azaz báb-robot fogalmát, mely, mely mint fizikai egység már csak az érzékelésért és beavatkozásért felelős szemben a hagyományos gyártási forgatókönyvekkel.

A folytatásban tisztázásra kerül egy nagyon fontos aspektusa a témának, az architektúráis kihívások és kulcs technológiák a felhő robotikában. a szerző élesen elkülönít kétféle szolgáltatás típust – vagy inkább képességet, robotikai értelemben az SaaS inkább Skills as a Service-t jelent – az állapot-információk szerint, mégpedig:

**Context-free:** azaz állapot független szolgáltatásoknak nincs szükségük több információra, mint például a pillanatnyi állapot, vagy az alkalmazás belső állapota. Működésükhöz elegendő a meghívásukhoz szükséges adat. Ezek a szolgáltatások tipikusan egyszerűen implementálhatóak, akár RESTful microservice-ben, akár felhős funkcióban – mint amilyen az AWS Lambda, vagy az Azure Functions – és könnyen skálázhatóak maradnak.

**Contextful:** azaz állapot függő szolgáltatásoknak szükségük van információra a tárgyrendszerről, annak fizikai struktúrájáról, pillanatnyi belső állapotáról. Ezek a szolgáltatások viszont már nem skálázódnak jól, tipikusan szükséges hozzájuk dedikált ROS [3] példány. (A ROS pillanatnyilag csak egy robot környezetet támogat)

Az ELSEVIER-ben megjelent cikk [2] definiálja a mindenhol jelen lévő gyártási rendszert a felhő segítségével, illetve az ICMS rendszereket – Interoperable Cloud Manufacturing System / Átjárható Felhő Gyártási Rendszer –. A felhő új üzleti modellt és lehetőségeket hoz a gyártásba, létrehozva a mindenhol jelen lévő gyártást, segítve az SME-ket – Small and Medium-sized Enterprises / Kis és közép vállalkozások – a kezdeti nagy beruházási költségekkel járó robotikai beruházásokban, csökkentve a modern gyártási környezet kialakításához szükséges költségeket.

A gyártási rendszerek új generációjának a létrehozása lehetővé válik a felhő szolgáltatások által, ami biztosítja a megfelelő számítási kapacitást, képességeket és erőforrá-

sokat. Más szemszögből nézve, a felhőben rendelkezésre áll az a kapacitás, amit költséges lenne egyenként fenntartani és kihasználatlan is lenne, ezért okos gondolat ezen erőforrásokat megosztani több gyártó rendszer között is, ideértve nem csak a számítási kapacitást, hanem azt a tudást is, ami egy-egy feladat hatékony megoldásához szükséges és valaki már elkészítette hozzá a szoftvert.

A cikk foglalkozik a felhő lehetőségeinek azokkal az aspektusaival is, amik megkönnyíthetik az ember-gép együttműködést – például kollaboratív robotok –. Minden ember másmilyen, mind fizikailag, mind mozdulatait tekintve. A felhő számítási kapacitása így jobban képes alkalmazkodni az ember jelenléte miatt szükségessé vált változó környezethez alkalmazkodni.

Fontos részlet – főleg a gyártás számára –, hogy a robotrendszerek energia hatékony módon dolgozzanak, amiben szintén nélkülözhetetlen lehet a felhő szerepe, de fontos lehet olyan helyzetekben is, amikor az energiavételezés lehetőségei korlátozottak. Az energia-optimalizált működés fontos szerepet játszik a zöld-felhasználásban, nem csak a gyártás területén.

Az IJCA cikke [4] a ROS alapú felhő-robotikával foglalkozik. Először is fogalmakat tisztáz. Az operációs rendszer és a robotikai alkalmazások között definiál egy réteget, a "Robotic middleware"-t, mely az alkalmazások elől elfedi a hardware és OS rétegeket, így egyszerűvé téve a robotikai fejlesztéseket és olcsóbbá téve azt. A ROS [3] – Robot Operating System – egy nyílt forráskódú "meta-operációs rendszer", ami biztosítja a

- Hardware abstrakciót
- Alacsony szintű eszköz vezérlést
- Üzenet továbbítást
- Csomag menedzsmentet

A cikk leírja a ROS alapfogalmait, részeit. Alapvető egység a Node, ami reprezentálja a ROS-on belül a felületet a programok felé. Egy ROS példány több Node-ot is indíthat. Az egyes Node-ok tartalmazzák azután a kétféle ROS szolgáltatás típust és a message-eket.

**message.** A message-ek az üzeneteket reprezentálják ROS-on belül.

**service.** A service-ek message párokat jelképeznek, kérés/válasz-ként, más rendszerekbe a függvények felelnek meg nekik.

**topic.** A Node-ok fel tudnak iratkozni topic-okra és maguk is tudnak kiajánlani topic-okat – megvalósítva az informatikában jól ismert publisher/subscriber mintát –. Egy topic-ba több node is írhat (publish) és több node is olvashat (subscribe) belőle.



A ROS-t számos nyelvből el lehet érni, , többek között C++, Python, lisp, Java, stb. környezetekből. Illetve rendelkezik egy csomag kezelővel, amivel finomhangolható, hogy mely szoftverkomponensek legyenek felhasználva az adott ROS példányban. A ROS elsődleges célja, hogy támogassa a kód újrafelhasználást a robotikai kutatásokban és fejlesztésekben [3]. A ROS támogatja a moduláris, **tool-based** fejlesztés filozófiáját a robotikában.

A Springer ROS és Docker alapú kiber-fizikai – CPS, Cyber-Physical Systems – architektúra tervezéssel foglalkozó cikke [5] bevezetést nyújt általánosságban az ICT – Information and Communication Technologies – rendszerek tervezési aspektusaiba, legfőképpen az ipari és CPS felhasználások tekintetében, érintve a tervező rendszerektől az Ipar 4.0-n át az IoT rendszerekig. Kitér a FOSS – Free and Open-Source Systems – rendszerek felhasználhatóságára, a ROS hasznára, mint a kód újrafelhasználhatóság és magas szintű integráció lehetőségére.

A cikk foglalkozik a könnyűsúlyú virtualizációval, ezen belül is a Docker konténer technológiájával. Ez a szoftver nem kívánja meg, hogy minden egyes ROS példányhoz külön operációs rendszert is installálni kelljen, hanem egyugyanazon operációs rendszeren képes futtatni több ROS példányt is, ezzel erőforrásokat megtakarítva az adott gép hardware-én.

Végül a cikk áttekinti a CPS rendszerek tervezésének hardware aspektusait, kezdve az egyszerű Raspberry Pi-től az ipari PLC-n át – Programmable Logic Controller – a PC-ig. Operációs rendszernek – a PLC kivételével – Linux-ot ajánl, valamint a ROS-t, mint middleware-t.

A Springer Rosbridge-el foglalkozó cikke [6] bemutatja a Rosbridge-et, mint a ROS egy köztes réteg absztrakciós szintjét, interfészt a nem ROS felhasználók felé. Lényegében a Rosbridge létrehoz a ROS szolgáltatásaihoz egy roxy-t, amit más környezetekből fel tudunk használni, ráadásul akár az internetről is, mert a tűzfalbarát WebSocket-et használja a kliensek felé.

A cikk bemutatja a ROS-t is. Legfelül vannak az egymással üzeneteket váltó Node-ok, amik szervizekkel és topic-okkal kommunikálnak egymással. A szervizeket úgy kell elképzelni, mint távoli funkció hívásokat, míg a topic-ok olyan folyamatos adatfolyamok, melyekbe "bárki" beleírhat és melyek üzeneteire "bárki" feliratkozhat.

A Rosbridge egy következő absztrakciós szint a ROS-hoz képest. Képes igény szerint

elindítani és leállítani ROS node-okat, a kommunikációt a kliensekkel pedig JSON formában bonyolítja. Lényegében bármely klienssel képes kommunikálni, amelyik "megérti" a JSON formátumot, Web alkalmazásokkal, vagy például Java, .NET szoftverekkel is. Nem csak HTML5 WebSocket-ekkel képes kommunikálni a klienseivel, hanem akár hagyományos POSIX IP socketekkel is. Kiadtak hozzá egy Javascript könyvtárat is a ROSJS-t, amit könnyen integrálni tudunk a Javascript-ben megírt programokhoz.

A Cloudroid-al foglalkozó cikk [8] betekintést enged egy nyílt forráskódú felhő keretrendszerbe [11], mely lehetőséget biztosít a meglévő robotikai szoftverek számára, hogy felhőbe konvertálhatók legyenek, biztosítva számukra a teljes átalakítási transzparenciát az Internet-en elért felhő szolgáltatás felhasználása mellett, a megfelelő QoS – Quality of Service – biztosításával. A biztonságot növelő megoldása, hogy ha hálózati hiba keletkezik képes a kliensnek legenerált Stub visszakapcsolni lokális hívásra, hogy ne akadjon meg végzetesen az elkezdett folyamat.

A Cloudroid mögöttes technológiája a Rosbridge, annak absztrakciójára épül, hasonlóan, mint a Rapyuta publikus robotikai felhő szolgáltató rendszere, mindketten megvalósítják a PaaS – Platform as a Service – infrastruktúra szolgáltatást. Bevezet négy mechanizmust:

**Öntartalmazó VM csomag:.** Amikor egy ROS csomagot migrálunk a felhőbe, akkor az nem lenne képes kommunikálni a ROS Master-el, így ilyenkor automatikusan becsomagolódik egy Docker konténerbe, ami tartalmazni fogja a működéshez szükséges összes többi ROS csomagot.

**Felhő bridging:.** Az adoptált Rosbridge A ROS protokollját konvertálja JSON formátumúvá és WebSocket alapú protokollá..

**Igény szerinti szolga példányosítás és szétoztás:.** A Cloudroidban a szervizből két rész létezik: az interfész és a szolga, azaz egy Docker példány ami tartalmazza a ROS csomagot. Mivel a ROS egy robot kiszolgálására készült, ezért a Cloudroid minden kliens számára más-más Docker példányt készít.

**Szolgáltatás Stub automatikus generálása:.** A Cloudroid-on futó szolgáltatások elérhetőek bármely manuálisan megírt WebSocket klienssel. Ezen túl a Cloudroid képes automatikusan legenerálni a szolgáltatás eléréséhez egy stub-ot, amivel kényelmesen elérhető lesz a szolgáltatás a kliensekről.

A Robot Web Tools-al foglalkozó cikk [9] a nevezett, Rosbridge-re épülő Javascript keretrendszerrel foglalkozik. A ROSJS mellett ez a másik feltörekvő, bár kicsit más készletet tartalmazó, Rosbridge-re épülő keretrendszer. Alapvetően három részből tevődik össze az RWT, a *roslibjs* egy kliens könyvtár, a *ros2djs* és a *ros3djs* pedig vizualizációs

könyvtár.

A *roslibjs* felhasználható bármely helyen mindenféle vizualizációs függőség nélkül. Egy kliens könyvtár, ami képes kommunikálni a ROS topic-okkal és szervizekkel. Tartalmaz továbbá eszközöket az alapvető robotikai feladatok ellátására, például transzfomációk, URDF – Unified Robot Description Format – parser és alapvető mátrix, vektor számítások.

A *ros2djs* és a *ros3djs* vizualizációs könyvtárak a ROS-al összefüggő adatokhoz – az EaselJS és three.js továbbfejlesztései –, például robot modellek, vagy térképek, lézer letapogatások és pontfelhők számára. Mindkét könyvtár HTML5 `<canvas>` elemre rajzol.

Az RWT-ben külön figyelmet fordítottak a streaming adatok megfelelő kezelésére, például pontfelhők folyamatos megjelenítésére. Az RWT vizualizációs része használja a HTML5 `<video>` és `<img>` elemeit a megjelenítésükhöz VP8 codec felhasználásával.

Az IEEE konferencia előadás anyaga [10] egy nagyszabású, katasztrófa keresési és mentési feladatokat ellátó nagyon összetett rendszer tervezéséről szól, amit már csak a felhasznált felhős és egyéb technológiák száma is bizonyít. Az IoT-től – Internet of Things – kezdődően, az RFID-n át – Radio-Frequency Identification –, a különböző felhő technológiákig, SaaS, PaaS, IaaS. Az esettanulmány a már említett technológiák – úgy mint, ROS, virtualization, Rapyuta – felhasználásával épít ki egy komplex, katasztrófa környezetben használható, felhő alapú rendszert.

## **5. fejezet**

### **A megoldási módszer kiválasztása, a választás indoklása**

## **6. fejezet**

### **A részletes specifikáció leírása**

## **7. fejezet**

### **A tervezés során végzett munkafázisok és tapasztalataik leírása**

## **8. fejezet**

### **A megvalósítás leírása**

## **9. fejezet**

### **Tesztelés**



## **10. fejezet**

**Az eredmények bemutatása, értékelése,  
hasonló rendszerek eredményeivel  
összevetése**

## **11. fejezet**

### **A megvalósítás elemzése, alkalmazásának és továbbfejlesztési lehetőségeinek számbavétele**

## **12. fejezet**

**A szakdolgozat tartalmi összefoglalója  
magyarul és angol nyelven**

# Irodalomjegyzék

- [1] Péter Galambos: Cloud-, Fog-, and Mist Computing in Service of Advanced Robot Applications, *ide be kell írni majd, hogy GP cikke hol jelent meg!*, 1969, pp.111-222.
- [2] Xi Vincent Wang, Lihui Wang, Abdullah Mohammed, Mohammad Givchci (Department of Production Engineering, KTH Royal Institute of Technology, Stockholm, Sweden): Ubiquitous manufacturing system based on Cloud: Robotics, em ELSEVIER, Robotics and Computer-Integrated Manufacturing, 45.szám, 2017, pp.116-125
- [3] Robot Operating System (<http://www.ros.org/about-ros/>), utoljára megtekintve: 2018.11.02.
- [4] Adarsha Kharel, Dorjee Bhutia, Sunita Rai, Dhruva Ningombam: Cloud Robotics using ROS, *International Journal of Computer Applications* ® (IJCA) (0975? 8887), *National Conference cum Workshop on Bioinformatics and Computational Biology*, NCWBCB- 2014, pp.18-21. *Jó így a hivatkozás?*
- [5] Pablo González-Nalda, Ismael Etxeberria-Agiriano, Isidro Calvo: A modular CPS architecture design based on ROS and Docker, em ©Springer-Verlag France, 2016, pp.950-955.
- [6] Christopher Crick, Graylin Jay, Sarah Osentoski, Benjamin Pitzer, Odest Chadwicke Jenkins: Rosbridge: ROS for Non-ROS Users ©*Springer International Publishing Switzerland*, 2017, pp.493-503
- [7] Docker lightweight container engine (<https://www.docker.com/products/docker-engine>), utoljára megtekintve: 2018.11.02.
- [8] Ben Hu, Huaimin Wang, Pengfei Zhang, Bo Ding, Huimin Che: Cloudroid: A Cloud Framework for Transparent and QoS-aware Robotic Computation Outsourcing, *Hol jelent meg? Oldalszám?* 2017,
- [9] Russell Toris, Julius Kammerl, David V. Lu, Jihoon Lee, Odest Chadwicke Jenkins, SarahOsentoski, Mitchell Wills, Sonia Chernova: Robot Web Tools: Efficient Messaging for Cloud Robotics, *hol jelent meg? Mikor és oldalszámok?*
- [10] Carla Mouradian, Sami Yangui, Roch H. Glitho: Robots as-a-Service in Cloud Computing: Search and Rescue in Large-scale Disasters Case Study, 15<sup>th</sup> *IEEE Consumer Communications and Networking Conference, Las Vegas, USA* 12-15 January 2018*Konferencia anyagra jó így a hivatkozás?*

- [11] Cloudriod cloud robotic platform Website (<https://github.com/cyberdb/Cloudroid>),  
utoljára megtekintve: 2018.11.02.

## **13. fejezet**

### **Mellékletek**