# Böngésző alapú felhasználói felület Panda robothoz

# Browser based user interface for Panda robot

***BOCSI Kristóf[1], B.Sc., fejlesztőmérnök, NACSA János[1,2], Ph.D., tud. főmunkatárs***

[1] SZTAKI, Mérnöki és Üzleti Intelligencia Kutatólaboratórium
H-1111 Budapest, Kende utca 13-17.; Telefon: +36 1 279 6000; Fax: +36 1 466 7503
bocsi.kristof|nacsa.janos@sztaki.hu; www.sztaki.hu
[2] Széchenyi István Egyetem, Járműipari Kutatóközpont
H-9026 Győr, Egyetem tér 1.; Telefon: +36 96 613 680; jkk.sze.hu

## Kivonat

*A kollaboratív robotok [8] egyik ígéretes típusa a hét csuklós Panda robot. A robot hivatalos kezelőfelülete támogatja az egyszerű használatot és minimális programozást igényel. Ugyanakkor hiányoznak olyan robotvezérlési funkciók, melyek minden teach pendant-en elérhetők. Ezeket a funkciókat hivatott pótolni a SZTAKI-ban fejlesztett pandaGUI nevű grafikus felhasználói felület.*

**Kulcsszavak:** kollaboratív robot; felhasználói felület

## Abstract

*One of the newest collaborative robot is the Panda robot with seven joints. The official HMI of the robot provides an easy usage and minimal programming. Ont he other hand functions known from other teach pendants are missing. This functions were added in the pandaGUI developed in SZTAKI.*

**Keywords:** collaborative robot; user interface

## 1. BEVEZETÉS

A Franka Emika Panda robotja egy hét csuklós kollaboratív robot. (1. ábra) [1] A robot programozására három hivatalos módszer van biztosítva. Egy magas szintű grafikus programozó felületet biztosít a Franka Desk nevű hivatalos böngésző alapú kezelőfelület [2]. Ez a robot hivatalos kezelőfelülete könnyen használható és a robot programozást mindenki számára elérhetővé teszi. Ugyanakkor sok olyan funkció hiányzik belőle, ami a legtöbb ipari robot kezelőfelületén megtalálható. Az egyes mozgások célpontjai egyszerűen csak kézi betanítással adhatók meg, hiányzik a robot koordináta értékkel vagy iránygombokkal történő közvetlen mozgatása. A felületen kicsit nehézkes az egyszeri parancs kiadása, mert minden mozgáshoz létre kell hozni, majd futtatni egy grafikus programot.


1. ábra *Panda robot tanítása [3] és a „Desk” programozói felülete [2]*

Alacsonyabb szintű programozást tesz lehetővé [6] a franka_ros és a libfranka. Előbbi egy ROS [7] metapackage aminek segítségével a robot ROS alapú programozása megvalósítható. A libfranka egy C++ könyvtár a robot programozására. Lényege, hogy a program 1kHz-es frekvencián valós időben küldi ki az aktuális időlépésben elérendő cél értéket (ez lehet Descartes koordináta rendszerbeli pozíció vagy sebesség, az egyes csuklók pozíciója vagy sebessége, valamint a motorok nyomatéka) a robot vezérlésének, a szükséges kinematikai számítások a vezérlés oldalán történnek.

A jelen projekt során létrehozott pandaGUI nevű grafikus kezelőfelület célja a hivatalos felület kiegészítése a hagyományosan megszokott funkciókkal, illetve a SZTAKI mintarendszereiben levő más kollaboratív robotokhoz már használt működtetés biztosítása [5]. A fenti két programozási lehetőség közül a projekt első fázisban a libfranka alapú megoldás készült el.

## 2. A MŰKÖDTETŐ PROGRAM FELÉPÍTÉSE

A program két fő részből áll, ahol a C++-ban írt konzol alkalmazás a robot vezérlését valósítja meg, a webes felület pedig, a felhasználóval történő interakcióért felelős. A két rész websocketen keresztül kommunikál egymással. Kommunikáció közben a konzol működik szerveroldalként. Az elküldött üzenetek mindig JSON-ok amiket az ID elemük alapján azonosítunk. A konzol felől a grafikus felületnek küldött üzenet lehet például a robot pozíciója, vagy az esetleges hibaüzenetek. A grafikus felület felől a konzolnak küldött üzenetek (pl. a mozgás parancsok).

Amikor a robot nem mozog, a konzol program három szálon fut (2. ábra). Ezek közül kettő a webes felülettel történő kommunikációért felelős. Az egyik szálon a bejövő a másokon a kimenő kommunikáció van megvalósítva (9002-es és 9003-as port). A bejövő üzenetek feldolgozása a fő szálban történik. A program indítás után megpróbál kapcsolódni a robot vezérléséhez. Ezután elindítja a kommunikációért felelős szálakat, majd egy shell-script lefuttatásával a böngészőben (Mozilla Firefox) megnyitja a grafikus felületet. Ha beérkezik egy mozgás parancs a megfelelő robot mozgató függvény egy új szálon fog futni. Ez a szál megkapja az ún. real time priority-t, hogy a valós idejű kommunikáció megvalósítható legyen.

A böngésző alapú grafikus felület felépítése a klasszikus weblap design-t követve egy HTML egy JavaScript és egy CSS fájlból tevődik össze.A weblap betöltés után megpróbál websocket kapcsolatot létesíteni a 9002-es és a 9003-as porton található szerverrel. Ha a kapcsolódás sikeres a robot pozíció kijelzése átvált az alap 0.0 értékekről a tényleges pozícióra. Ezután a robot működtethető a grafikus felület segítségével.

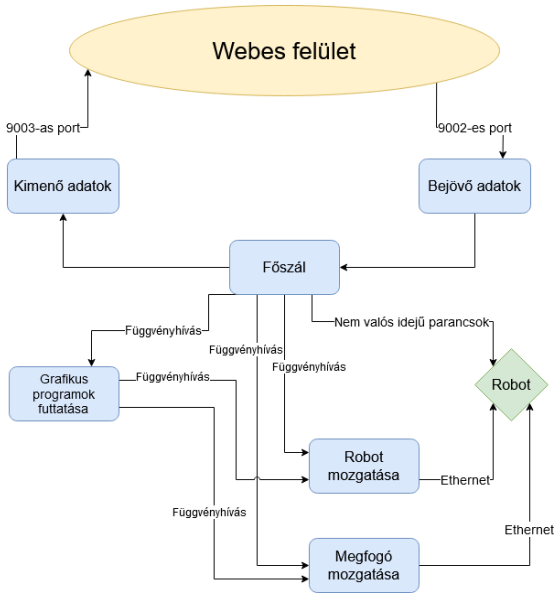
A felület használatát lineáris mozgás esetében a 3. ábra mutatja. Látható az ábrán, hogyha hibás egy bemenet, akkor erre egy megjelenő hibaüzenet figyelmeztet. Helyes bemenetek (jelen esetben elérhető értékek) megadása esetén a „Start” gomb megnyomásával indul a mozgás. Mozgás közben a „Start” gomb inaktív, új mozgása csak akkor indítható, ha az előző véget ért. Ha egy bemeneti érték (pl. a Z érték üresen marad), akkor annak az aktuális értéke lesz figyelembe véve (vagyis a példában a mozgás csak az X-Y síkban történik). Sebességek és gyorsulások esetén az alapértelmezett érték egy megfelelően kicsi – előre programozott - érték.

A programban a „Start” gomb megnyomása után a megadott paraméterekkel rendelkező JSON [9] el lesz küldve a C++ konzolnak, mely üzenetet a konzol fő szála dolgoz fel. Ezután egy új szálban elindul a lineáris mozgást megvalósító *movel*() függvény a kapott paraméterekkel. Ha a mozgás véget ér, a konzol egy visszajelzést küld a webes felületnek. Ezután a mozgás indító gombok újra aktívak lesznek. Ha a mozgás valamilyen hibával áll le, akkor a mozgás vége üzenet mellett az adott hibaüzenetet is megkapja a grafikus felület, melyeket a „Hibák” fülön jelent meg. Ütközés esetén egy külön ablak ugrik fel a grafikus felületen.

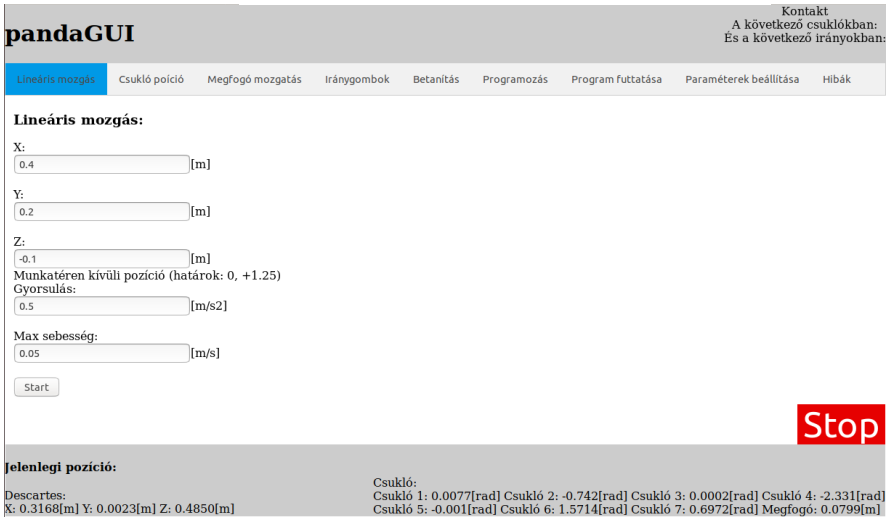
A felület tetején és alján levő információk állandóan láthatóak, a különböző funkció csoportok pedig a következő fülekbe vannak elrendezve (3. ábra):

- A robot lineáris mozgatása egy tetszőleges 3D pontba tetszőleges gyorsulással és sebességgel.
- Az egyes csuklók szögeinek beállítása (egyenként és egyszerre is).
- A megfogó kalibrálása, a megfogó adott szélességre mozgatása adott sebességgel, adott szélességű tárgy megfogása adott sebességgel és erővel.
- A robot koordináta tengelyekkel párhuzamos mozgatása iránygombok segítségével
- Egyszerű robotprogram létrehozása pontok kézi betanításával

– Blokkokból felépített grafikus program létrehozása (a Desk-hez hasonló módon)
– A robot merevségének beállítása (mennyire tudjon a programozott ponttól elmozdulni külső erő és nyomaték hatására)
– Kontakt és ütközés érzékeléséhez erő és nyomaték határértékek beállítása (a kontakt határérték átlépése esetén a felhasználó visszajelzést kap, de a mozgás nem áll le, az ütközési határérték átlépése esetén a mozgás egy hibaüzenettel leáll)




2. ábra *A konzol program belső szálai*




3. ábra *A felhasználói felület (lineáris mozgás esetében)*

Az összes további egyszerű parancs kiadása esetén a felület a 3. ábrán láthatóhoz hasonló. A paraméterek szöveges bemenetként adhatók meg, majd a parancs a start gombbal adható ki. Ha valamelyik érték a megadott határértékeken kívül van (pl. az adott pont nem érhető el) az adott bemenet mellett egy hibaüzenet jelenik meg és a parancs nem lesz elküldve. A parancs elküldésekor a paraméterek egy JSON megfelelő elemeibe lesznek beírva. A JSON ID mezője a parancs típusát határozza meg.

## 3. GRAFIKUS PROGRAMOZÓI FELÜLET

A „Desk” koncepció értékei egy grafikus programozó felület kialakításában kerültek átmentésre. Jelen változatban természetesen csak töredék számú blokkal a Desk-hez képest: (Lineáris mozgás, Beállás adott csukló pozíciókra, Megfogó mozgatása –akár mozgás közben is -, Ciklus)

Egy adott blokk hozzáadása esetén megjelenik egy felugró ablak ahol az adott mozgás paraméterei adhatók meg. A megadott paraméterek ellenőrzése az egyszeri parancsok kiadásánál leírtakkal megegyezik. Üresen maradt érték esetén itt is a robot aktuális pozíciója lesz alapértelmezettként használva így ez a funkció is használható betanításra. Az éppen szerkesztett program a grafikus felület oldalán egy tömbben van eltárolva. A tömb minden eleme egy JSON, ami egy lépést reprezentál. Ezeknek a JSON-oknak a felépítése az egyszerű parancsok elküldésére használt JSON-okhoz hasonló. A program elmentése a „Program mentése” gombbal történik. Ekkor a program lépésenként el lesz küldve a konzolnak. A konzol a JSON-okat egy szöveges fájlba menti. A fáj minden sora egy lépést tartalmaz.

Program futtatása esetén a grafikus felület elküldi a futtatandó program nevét a konzolnak, ami a megfelelő fájlt soronként olvassa. Minden sor esetén meghívódik a megfelelő robot mozgató függvény egy új szálban. A következő sor akkor lesz olvasva, ha az előző mozgás véget ért.

A Franka Desk-ben nehézkes a ciklusok szervezése, erre a pandaGUI a következő megoldást adja. A ciklus kezdete és vége egy-egy lépésként van jelölve a programban. Ha a kiolvasott sor a ciklus kezdetét jelöli a program az utána következő lépéseket nem lefuttattja, hanem egy vektorba menti, amíg el nem ér a ciklus vége lépésig. Ezután ennek a vektornak az elemeit futtatja.

A Panda robot nagy előnye a kész betanítás egyszerűsége. A pandaGUI-ban az adott pozíció a „Pozíció mentése” gombbal menthető el. Ekkor a C++ konzol a megfelelő nevű fájlhoz hozzáad egy sort. Ez lényegében „egy csukló pozícióra állás” parancs a robot aktuális pozíciójával. Mivel a kézi betanítással és grafikus programozással létrehozott programok felépítése megegyezik, a betanítással elkészített programok később grafikus programként szerkeszthetők.

*A pandaGUI és a Franka Desk összehasonlítása* 1. táblázat

| Szempont | pandaGUI | Franka Desk |
|---|---|---|
| **Mozgási cél magadása** | Számszerűen és kézi betanítással | Csak kézi betanítással |
| **Iránygombos mozgás** | Van | Nincs |
| **Mozgásparancs kiadása** | Egyszeri parancs vagy program létrehozása | Minden mozgáshoz programot kell létrehozni |
| **Hibakezelés** | Indítás óta fellépett hibák kijelzése | Csak az utolsó hibaüzenet |

# 4. ÖSSZEFOGLALÁS

Bemutatásra került az újszerű Panda kollaboratív robot programozhatóságát a hagyományosabb megoldásokhoz közelítő pandaGUI és annak belső felépítése. A megoldás ötvözni tudja a Desk felület újszerűségét a megszokott teach-pendant funkciókkal. A további tervekben a robot erőszenzorait aktívan használó kollaboratív alkalmazások fejlesztése és ipari kamera illesztése a robothoz szerepelnek.

# KÖSZÖNETNYILVÁNÍTÁS

A bemutatott kutatást részben az Európai Unió H2020-as EPIC (No. 739592) és a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal INEXT - Kutatások az ipari digitalizáció által nyújtott potenciál minőségi kiaknázására (ED_18-22018-0006) projektjei támogatták. Nacsa János külön köszöni a „Felsőoktatási Intézményi Kiválósági Program – Digitális ipari technológiák kutatása a Széchenyi István Egyetemen" projekt (TUDFO/47138-1/2019-ITM) támogatását.

# IRODALOMJEGYZÉK

[1] JSON Wikipédia: hu.wikipedia.org/wiki/JSON
[2] Ionescu T.B., Schlund S., Schmidbauer C. (2020) Epistemic Debt: A Concept and Measure of Technical Ignorance in Smart Manufacturing. In: Advances in Human Factors and Systems Interaction. AHFE 2019. Advances in Intelligent Systems and Computing, vol 959. Springer, Cham
[3] Franka Emika: Panda Press Kit, 2019, s3-eu-central-1.amazonaws.com/franka-de-uploads/uploads/2019/04/Franka-Emika-PressKit-2019.zip
[5] Ipar 4.0 kutatási és innovációs kiválósági központ, https://ipar40kutatas.hu/
[6] Franka Control Interface Documentation; frankaemika.github.io/docs/
[7] About ROS (Robot Operating System); www.ros.org/about-ros/
[8] Thomas, C & Matthias, Bjoern & Kuhlenkötter, Bernd. (2016). Human-Robot Collaboration – New Applications in Industrial Robotics