

XXVI. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)

Fizika szekció

Kolozsvár, 2023. május 18–21.

Egyszerű robotok vezérlése multistabil dinamikai rendszerekkel

Szerzők:

András Barnabás Tamás

Babeş–Bolyai Tudományegyetem, Fizika Kar, Mérnöki Fizika szak, III. év

Témavezetők:

dr. Sándor Bulcsú, egyetemi adjunktus

Babeş–Bolyai Tudományegyetem, Fizika kar

Kivonat

A robotok irányítása napjainkban már nem csak egy mérnöki és informatikai feladat. A hagyományos, inverz és direkt kinematikán alapuló mérnöki és a mesterséges intelligenciát felhasználó informatikai módszerek mellett egyre nagyobb teret hódít magának, egy a fizikában jól ismert, az önszerveződő dinamikai rendszerek elméletét gyakorlatban alkalmazó irányítási módszer.

A kutatásunk során egy korábbi dolgozat továbbfejlesztését tűztük ki célul, úgy elméleti, mint technikai szempontból. Dolgozatunkban egyszerű robotok dinamikáját vizsgáljuk melyeket multistabil, önszerveződő dinamikai rendszerekkel vezérelünk. Kétkeredű robotjaink esetében a kerekeket két-két neuron irányítja, míg a PhantomX típusú hatlábú robot esetében a lábak mozgását tizenkét csatolt, oszcilláló neuronnal vezéreljük. Arra keressük a válaszokat, hogy a paraméterek és kezdőfeltételek függvényében milyen mozgásformák és lépésmintázatok alakulhatnak ki önszerveződő módon ezen robotok esetében illetve, hogy milyen feltételek mellett tudunk átmenni a rendszer egyik stabil atraktoráról egy másikra. A dinamikai rendszereket számítógépen szimuláltuk, majd a kapott dinamikát a robotokon kísérletileg is tanulmányozzuk.

XXVI. ERDÉLYI TUDOMÁNYOS DIÁKKÖRI KONFERENCIA
CONFERINȚA ȘTIINȚIFICĂ STUDENȚEASCĂ DIN TRANSILVANIA (CȘST), EDIȚIA A XXVI-A
26TH TRANSYLVANIAN STUDENTS' SCIENTIFIC CONFERENCE (TSSC)
reál- és humántudományok • științe reale și umane • real and human sciences

KOLOZSVÁR, 2023. MÁJUS 18–21.

Kolozsvári Magyar Diákszövetség • Uniunea Studențească Maghiară din Cluj
400083 Kolozsvár, Petőfi Sándor/Avram Iancu utca 21 • tel./fax: 0755116251
e-mail: etdk@kmdsz.ro, etdk.kolozsvar@gmail.com • honlap/web: <https://etdk.kmdsz.ro>

Témavezetői igazolás

Alulírott *dr. Sándor Bulcsú* igazolom, hogy *András Barnabás Tamás* hallgató a(z) *Egyszerű robotok vezérlése multistabil dinamikai rendszerekkel* című dolgozatot az alulírott szakmai irányításával készítette el, és javaslom az említett tudományos munka bemutatását a XXVI. Reál- és Humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK) *Fizika* szekciójában.

dr. Sándor Bulcsú, egyetemi adjunktus
Babeș–Bolyai Tudományegyetem, Fizika kar

Kolozsvár, 2023. április 28.

Tartalomjegyzék

1. Bevezető	1
1.1. Klasszikus vezérlési módszerek	1
1.1.1. Direkt kinematika	2
1.1.2. Inverz kinematika	4
1.2. Multistabil dinamikai rendszerek	6
2. A robotok felépítése és vezérlése	8
2.1. Romi - a kerekes robot felépítése, saját fejlesztések	8
2.2. A kerekes robot vezérlése	9
2.3. Hexapod - a hatlábú robot felépítése	9
2.4. A hexapod irányításához szükséges fogalmak	11
2.5. A hexapod programozása és saját fejlesztések	12
3. Vezérlés multistabil dinamikai rendszerekkel	14
3.1. Neuronokról fizikus szemmel	14
3.2. Kerekes robot vezérlése neuronokkal	15
3.2.1. Impulzus-vezérlés	17
3.3. Hatlábú robot vezérlése neuronokkal	18
4. Kísérleti eredmények	21
4.1. Kerekes robot neuronokkal	21
4.1.1. Kick control	24
4.2. Hatlábú robot neuronokkal	27
4.2.1. Lépésmintázatok keresése	30
4.2.2. Impulzus-vezérlés	33
Következtetések és továbbfejlesztési lehetőségek	34
5. Melléklet	
5.1. Romi robot technikai leírása	
5.2. A hexapod robot programozása	
5.3. AX-12A szervomotorok bemutatása	

1. Bevezető

Dolgozatunkban egyszerű robotok vezérlését vizsgáljuk multistabil dinamikai rendszerek segítségével. Célunk, egy már létező, önszerveződésre épülő vezérlési model reprodukálása, megértése, majd továbbfejlesztése úgy elméleti, mint technikai szempontból [1, 2].

A dolgozat első felében a robotikában használt klasszikus irányítási módszereket mutatjuk be, bevezetjük a robotikában használatos alapfogalmakat (1.1. fejezet). Ezt követően rövid betekintést teszünk a dinamikai rendszerek világába (1.2. fejezet) és rátérünk arra, hogy miképpen lehet robotokat vezérelni ilyen rendszereket implementálva (3. fejezet). Ismertetjük a kísérletezés során használt kétkerekű és hatlábú robot felépítését, fizikai paramétereit, a rajtuk eszközölt technikai és numerikus változtatásokat (2. fejezet), valamint az irányításukhoz használt dinamikai rendszereket (3. fejezet).

A vezérléshez használt oszcilláló neuronokból álló dinamikai rendszereket numerikusan és kísérletileg robotokon is megvizsgáljuk. Feltérképezzük a megjelenő stabil mozgásmintázatokat, valamint javasolunk két új módszert, melyek segítségével váltani tudunk a létrejövő attraktorok között (4. fejezet).

1.1. Klasszikus vezérlési módszerek

A napjainkban használt robotok általában valamilyen repetitív feladat elvégzésének céljából lettek létrehozva. Ilyen feladatok közé tartozik például a tárgyak szállítása, ezt általában kerekes robotok végzik, illetve manipulálása, amelyhez robotkarokat szoktak használni. A vezérlésük mindkét esetben általában a klasszikus robotika módszereivel van megvalósítva: a robotok egy előre precízen megtervezett mozgásformát vagy útvonalat követnek, így kizárva a tévedés lehetőségét.

A robotkarok felépítésük szerint *szegmensekből* és *csuklókból* állnak. A szegmensek alkotják a robotkar vázát és felelnek a robot mechanikai szilárdságáért. A szegmensek különböző típusú csuklókon keresztül csatlakoznak egymáshoz: a *rotációs csuklók* lehetővé teszik, hogy valamely tengely körül elforduljanak, míg a *transzlációs csuklók* esetében a szegmensek eltolhatóak egy tengely mentén. A csuklók mozgását *aktuárok* végzik (mechanikai elmozdulást megvalósító eszközök). A robotkar végén kap helyet a robot "megfogója", melyet *effektornak* vagy *végberendezésnek* hív a szakirodalom. Egy rögzített talappal rendelkező, néhány szegmensből és csuklóból álló robotkart általánosan

manipulátornak nevezünk.

A kerekes robot esetében, ha a legegyszerűbb felépítést tekintjük, akkor a robot rendelkezik egy vázzal és három keréssel. Ezek közül kettő vezérelhető (forgó aktuátorokkal), míg a harmadik kerék passzív, és ez felel a robot stabilitásáért. Általános esetben a kerekes robotokat *mobil robotoknak* hívja a szakirodalom [3].

Ahhoz, hogy a robotok helyesen el tudják végezni feladatukat, pontosan le kell tudnunk írni annak a testnek a helyzetét, melyet el szeretnénk mozdtítani a manipulátor segítségével, vagy azt a helyet ahová szeretnénk, hogy eljusson a mobil robotunk. Továbbá a nagy pontosságú feladatvégzés érdekében szükséges ismernünk a mobil robot, vagy épp a robotkar effektorának helyzetét is.

Három dimenzióban egy merev test helyzetének pontos leírásához elengethetetlen tudnunk annak *pozícióját* és *orientációját*. A pozíció egy három elemű vektor, $\mathbf{p}=(x,y,z)$, mely tartalmazza az adott koordináta-rendszerben a három tengely menti koordinátát. Az orientáció a merev testhez rögzített koordináta-rendszer tengelyeinek irányát határozza meg. Az orientációt általában három egymást követő elemi forgatási szöggel szokták jellemezni (Euler- vagy Cardano-féle szögekkel). Összesen tehát egy merev test helyzetének a térben történő leírásához hat paramétert kell megadnunk.

1.1.1. Direkt kinematika

A klasszikus robotika módszerei közé tartozik a direkt kinematika, mely megadja egy n csuklóval és n szegmenssel rendelkező robotkar effektorának a helyzetét, ha ismerjük a $\mathbf{q} = (q_1, q_2, \dots, q_n)$ csuklószögeket. Minden szegmenshez hozzárendelhetünk egy koordináta-rendszert. Robotikában az $\{A\}$ és $\{B\}$ koordináta-rendszerek közötti transzformációt a *homogén transzformációs mátrixsal* reprezentáljuk, mely megadja az $\{A\}$ koordináta-rendszerben a $\{B\}$ koordináta-rendszer pozícióját és orientációját, tartalmazza a $\mathbf{t}$ transláció vektort és az $\mathbf{R}$ forgatási mátrixot [1]:

$${}^A T_B = \begin{pmatrix} {}^A \mathbf{R}_B & {}^A \mathbf{t}_B \\ \mathbf{O}_{(1 \times 3)} & 1 \end{pmatrix} \quad (1)$$

Amennyiben a koordináta-rendszer valamelyik tengelye körül forgatunk θ szöggel, a rotációs mátrix megadható, mint egy elemi forgatási mátrix:

$$\mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} \quad (2)$$

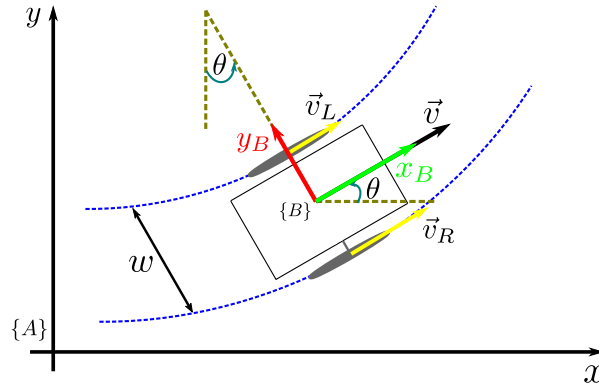
$$\mathbf{R}_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \quad (3)$$

$$\mathbf{R}_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (4)$$

Ha több, egymás utáni elforgatást szeretnénk leírni, elégséges a különböző tengelyekre vonatkoztatott forgatási mátrixok egymást követő alkalmazása. A direkt kinematika feladata tehát az, hogy teremtsen egy függvényi kapcsolatot a csuklószögek és az effektor pozíciója között:

$$\mathbf{p} = \mathbf{K}(\mathbf{q}), \quad (5)$$

ahol $\mathbf{K}$ egy nemlineáris függvény. Ez a kapcsolat a homogén transzformációs mátrixok egymás utáni alkalmazásával vagy egyszerű geometriai műveletekkel megadható, ha ismerjük a csuklók szögét és a robotkar szegmenseinek a hosszát.



1. ábra. A kerekes robot mozgásának szemléltetése. $\{A\}$ a világ koordináta-rendszer, $\{B\}$ a robot testéhez rendelt koordináta-rendszer, θ a robot orientációját megadó szög, $\vec{v}_R$ és $\vec{v}_L$ a jobb- és baloldali kerekek sebessége, $\vec{v}$ a robot haladási sebessége és w a kerekek közötti távolság.

Kerekcs robotok esetében a robot pozíciója és orientációja megadható, ha ismerjük a kerekek sebességét minden időpillanatban ($v_R(t)$ és $v_L(t)$ kétkerekű robot esetén) és feltételezzük, hogy a kerekek nem csúsznak meg. Az 1. ábrát figyelembe véve, kétkerekű robot esetén a feladat megoldásához a következő egyenletek írhatók fel [3]:

$$\begin{cases} \dot{x} = v(t) \cdot \cos \theta \\ \dot{y} = v(t) \cdot \sin \theta \\ \dot{\theta} = \frac{v_R(t) - v_L(t)}{w} \end{cases} \quad (6)$$

ahol $v(t) = \frac{v_R(t) + v_L(t)}{2}$ a robot haladási sebessége az adott időpillanatban, $v_R(t)$ és $v_L(t)$ a jobb- és baloldali kerekek sebessége, x és y a robothoz rendelt koordináta-rendszer középpontjának pozíciója az $\{A\}$ világ koordináta-rendszerben, θ a robot orientációját megadó szög és w a kerekek közötti távolság. A fenti differenciálegyenleteket egy numerikus integrálási módszerrel megoldva, ismerve a kerekek sebességét minden időpillanatban és egy kezdeti állapotot, a robot pozíciója és orientációja megadható [3].

1.1.2. Inverz kinematika

Az inverz kinematika tulajdonképpen a direkt kinematika fordítottja. Egy inverz kinematikai feladat során meg kell határoznunk, hogy milyen csuklózögekre (q) van szükségünk, hogy a robotkar effektorja eljusson egy előre megválasztott p pozícióba (és orientációba, de ennek a tárgyalásától most eltekintünk, mivel az effektort pontszerűnek tekintjük). Ez a feladat azonban, ellentétben a direkt kinematikai feladattal, sok esetben analitikusan nehezen vagy egyáltalán

nem megoldható. A megoldás nehézségét az okozza, hogy az egyenletek nem lineárisak, legtöbb esetben nincs egzakt megoldásuk. Legtöbbször egy problémára több vagy akár végtelen számú megoldás is létezhet. Továbbá vannak olyan esetek is amikor a probléma egyáltalán nem megoldható (például ha a robotkar nem elegendően hosszú, vagy nem rendelkezik elegendő csuklóval ahhoz, hogy elérje a kívánt pozíciót). Az analitikus megoldások esetén fellépő problémák feloldhatók, ha a feladat megoldását numerikusan közelítjük meg.

A fent említett n csuklóval rendelkező robotkar esetén, ha ismerjük az $\mathbf{p}^*$ célpozíciót, numerikusan az inverz kinematikai feladat az alábbi differenciálegyenlettel közelíthető meg (bővebben: [3]):

$$\dot{\mathbf{q}}(t) = \alpha \cdot \mathbf{J}^T \cdot \Delta(\mathbf{K}(\mathbf{q}(t)), \xi_E^*), \quad (7)$$

ahol $\Delta(\mathbf{K}(\mathbf{q}), \xi_E^*) = (\mathbf{p}^* - \mathbf{p})$ az effektor aktuális ($\mathbf{p}$) és célpozíciója ($\mathbf{p}^*$) közötti három elemű eltolásvektor, $\mathbf{J}^T$ a robotkar geometriai Jacobi mátrixának transzponáltja, α egy állandó és $\dot{\mathbf{q}}(t)$ a csuklózögek idő szerinti deriváltjai. Ha a robotkar n csuklóval rendelkezik, akkor a Jacobi mátrixnak a következő alakja van [3]:

$$\mathbf{J} = \frac{d\mathbf{K}(\mathbf{q}(t))}{d\mathbf{q}} = \begin{pmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \frac{\partial z}{\partial q_3} & \cdots & \frac{\partial z}{\partial q_n} \end{pmatrix} \quad (8)$$

A (7)-es egyenlet numerikusan megoldható, ha időben diszkretizáljuk és valamilyen numerikus integrálási módszerrel (például Euler- vagy Runge-Kutta-módszer) addig iteráljuk az aktuális csuklózög értékét, amíg egy hibahatáron belül közel kerül a cél értékhez. Ekkor úgy tekinthető, hogy a robotkar effektora elért a kívánt pozícióra [1, 3].

Mobil robotok esetén inverz kinematikai feladatról akkor beszélhetünk, amikor az a cél, hogy ismerve a robot célpozícióját, adjuk meg a kerekek sebességét és a robot elfordulási szögét minden időpillanatban. Az ilyen problémákra azonban már két dimenzió esetén is végtelen számú megoldásunk van, mivel a robot bármennyi ideig, bármilyen utat bejárva, akár a célponton többször áthaladva is eljuthat végül a kívánt pozícióba. Ezért ilyen esetekben megkötéseket kell alkalmaznunk a feladat megoldása során. Ilyen megkötés lehet az, hogy a legrövidebb út alatt történjen a mozgás vagy, hogy közben a robot érintsen bizonyos pozíciókat vagy, hogy a mozgás során kerüljön ki egy akadályt.

1.2. Multistabil dinamikai rendszerek

Dinamikai rendszerek olyan fizikai, biológiai, kémiai, de akár gazdasági rendszerek, amelyek az időben változnak és általában kevés változó mennyiséggel rendelkeznek. Ezen rendszerek változása az időben lehet diszkrét vagy folytonos. A folytonos idejű dinamikai rendszerek egyszerű elsőrendű, közönséges differenciálegyenlet alakba írhatók fel:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad (9)$$

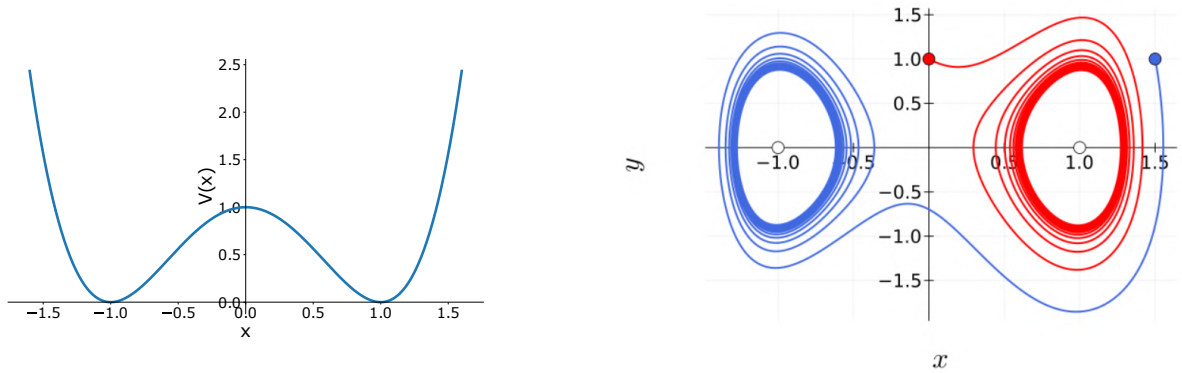
ahol $\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]$ a rendszer változóit tartalmazza, $\mathbf{f}(\mathbf{x}) = [f_1, f_2, f_3, \dots, f_n]$ a változók közötti függvényi kapcsolatokat, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, és n a változók száma, amely megadja a rendszer dimenzióját. A dinamikai rendszer állapotát egy időpillanatban az összes változó együttese jellemzi ($\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]$). A változók értékei minden pillanatban egy pontot határoznak meg az n dimenziós térben, amit *fázistérnek* hívunk, és az általuk meghatározott pont a *fázispont*. A fázispont mozog a fázistérben, miközben a rendszer evolúál az időben, az általa leírt görbét *trajektóriának* nevezi a szakirodalom. Azt az alteret amelybe a fázispont a kezdeti paraméterektől függetlenül aszimptotikusan tart *attraktornak* nevezzük. Az attraktor lehet egy pont (*fixpont*), egy zárt körbe (*határciklus*) vagy egy fraktálszerkezet [10]. Léteznek azonban olyan dinamikai rendszerek is, amelyekben a kezdeti paraméterektől nagyon is függ, hogy a fázispont melyik attraktorra konvergál. Az ilyen dinamikai rendszereket nevezzük *multistabil dinamikai rendszereknek*. A multistabil dinamikai rendszerek esetében a fázistérben egyszerre több attraktor van. Az ilyen esetekben minden attraktor rendelkezik egy *vonzási tartománnyal*. Tehát a kezdeti feltételek függvényében más-más attraktor vonzási tartományába kerülhet a fázispont. Az általunk robotok vezérlésére használt neurális rendszerek esetén is ezért alakulhatnak ki a sokszínű mozgásmintázatok.

A multistabil dinamikai rendszerek esetében lehetséges azonban az is, hogy a fázispontot "leválasszuk" az egyik attraktorról és "átvigyük" egy másikra. Amennyiben két attraktor a fázistérben közel van egymáshoz, vagy a vonzási tartományuk elég nagy, a dinamikai rendszer egy kis perturbálásának hatására a fázispont "átugorhat" egyik attraktorról a másikra [2]. Dolgozatunkban a multistabil dinamikai rendszerek ezen tulajdonságát használjuk fel a robotok mozgásmintázatainak impulzus-vezérléssel ("kick control") történő megváltoztatása során.

Tekintsük példaként a következő adaptív mekhanikai típusú rendszert:

$$\begin{cases} \dot{x} = y \\ \dot{y} = -\alpha \cdot [V(x) - \mu] \cdot y - \frac{dV(x)}{dx} \end{cases} \quad (10)$$

ahol $V(x) = (x - x_1)(x - x_2)$ a rendszer potenciálja (a 2. ábra, bal oldali grafikonja szemlélteti a függvény alakját), valamint α , μ , x_1 és x_2 a rendszer paraméterei [13]. A fenti differenciálegyenlet rendszert numerikusan megoldva a 2. ábrán szereplő fázisportrét kapjuk.



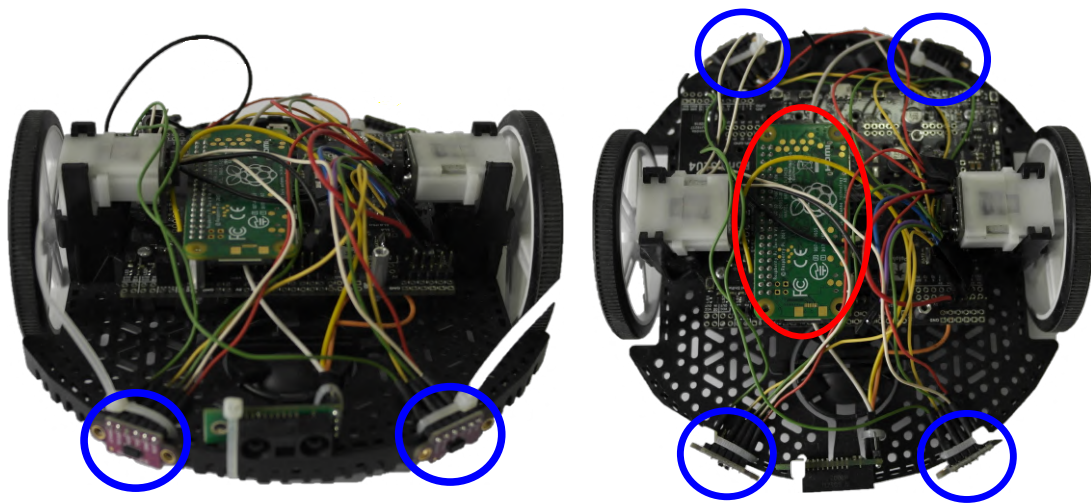
2. ábra. **Bal:** A $V(x)$ függvény egy dupla potenciálgödörrel rendelkező potenciáletteret ír le. Az $x = \{-1, 1\}$ pontokban a potenciálnak minimuma van ($x_1 = -1$ és $x_2 = 1$). **Jobb:** A dinamikai rendszer fázisportréja: pirossal a $[0, 1]$ kezdőállapotból indított fázispont trajektóriája, kékkel az $[1.5, 1]$ kezdőállapotból indított fázispont trajektóriája. Paraméterek: $x_1 = -1$, $x_2 = 1$, $\alpha = 1$ és $\mu = 0.1$ [13].

A fázisportrén jól látható, mit is jelent egy rendszer multistabilitása. Esetünkben a fázistérben két *instabil fixpont* található a $[-1, 0]$ és $[1, 0]$ pontokban, amelyek körül kialakul egy-egy stabil határciklus. A $[0, 1]$ pontból indított "piros" fázispont a "piros" határciklus vonzáskörében van, ezért idővel a "piros" fázispont ehhez a stabil attraktorhoz tart és rákonvergál a stabil határciklusra. A $[1.5, 1]$ pontból indított "kék" fázispont, habár a kezdeti időpillanatokban közelebb van a "piros" határciklushoz, a "kék" határciklus vonzási körzete kinyúlik a "piros" határciklus köré, így a "kék" fázispont idővel a "kék" határciklus által meghatározott stabil attraktorhoz tart. Tehát egy dinamikai rendszer két különböző kezdőállapotból indított fázispontja két különböző dinamikát eredményez, ezért nevezhető multistabil dinamikai rendszernek.

2. A robotok felépítése és vezérlése

2.1. Romi - a kerekes robot felépítése, saját fejlesztések

A mobil robotos kísérleteink során egy **Romi** típusú két hajtókerékkel rendelkező robotot használtunk, melyet a *Pololu Robotics & Electronics* fejleszt. A cég ezt a robotot oktatási eszköznek és egy egyszerű bevezető versenyrobotnak szánta, ezért rengeteg programcsomag és könyvtár elérhető hozzá, ezáltal alapszintű vezérlése, működtetése igencsak megkönnyített. Az összeszereléséhez nem szükséges magas szintű elektronikai és technikai tudás, az alkatrészeket egyszerűen a helyükre kell illeszteni [5].

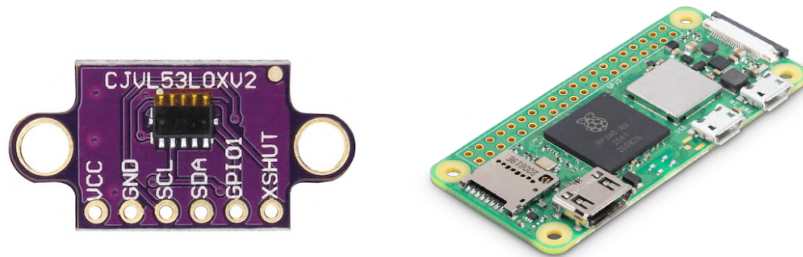


3. ábra. A Romi robot felépítése. **Bal:** a robot szemből nézve; a kékkel bekarikázva látható a két első VL53L0X távolságmérő szenzor. **Jobb:** a robot felülnézetből; pirossal bekarikázva látható a Raspberry Pi Zero 2 W és kékkel bekarikázva a roboton elhelyezett négy távolságmérő szenzor.

A Romi robot a Romi 32U4 vezérlőpanelen keresztül irányítható. A robot talán egyik legfontosabb eleme a *Raspberry Pi Zero 2 W* mini számítógép (lásd 4. ábrán). Ezen keresztül valósul meg a kommunikáció a robot és a felhasználó számítógépe között vezeték nélkül, WiFi-n, *SSH* protokolon. A vezérlőpanel további elemeinek felsorolása a Mellékletben található.

A kísérleteink céljából a robotot kibővítettük négy darab VL53L0X távolságmérő szenzorral (lásd 4. ábrán). A VL53L0X egy új generációs (ToF - Time of Flight - "repülési idő") lézeres távolságmérő modul. A hagyományos technológiáktól eltérően pontos távolságmérést tesz lehetővé a céltárgy reflexiók tényezőitől függetlenül. Akár két méteres abszolút távolság mérésére is képes. A szenzor működése során kibocsájt egy 940 nm hullámhosszú (emberi

szemnek láthatatlan) lézerimpulzust és méri azt az időt, ami ahhoz szükséges, hogy a lézerimpulzus elérje a céltárgyat, onnan visszaverődjön és visszajusson a szenzorhoz [9]. A robot energiaellátását hat darab AA típusú elem biztosítja, melyek elektromotoros feszültsége, teljesen feltöltött állapotban, egyenként 1.5V, összesen pedig 9V.



4. ábra. **Bal:** VL53L0X távolságmérő szenzor. **Jobb:** Raspberry Pi Zero 2 W.

2.2. A kerek robot vezérlése

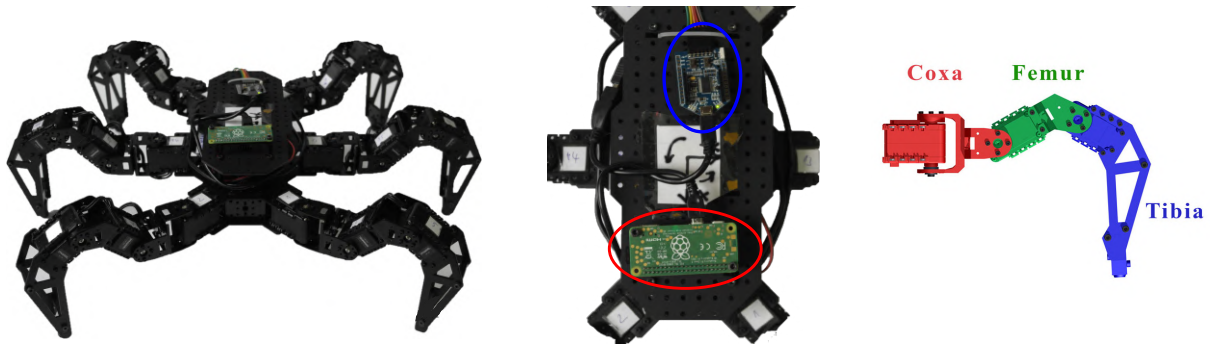
A fejlesztő által létrehozott programcsomagok és könyvtárak lehetővé teszik számunkra, hogy az Arduino sajátos C++ programozási nyelvezete helyett, a Raspberry Pi Zero W 2-t felhasználva a robotot irányítani tudjuk Python nyelvben írt kódok által is. Első lépésként a fejlesztők által megadott könyvtárakat telepíteni kell a raspberry-re, ez ahhoz szükséges, hogy a raspberry értelmezni tudja az általunk írt kódot és kommunikálni tudjon az *Atmel ATmega32U4* AVR chippel. Az általunk *Python* programozási nyelvben megírt programot SSH protokolon tölthetjük fel a robotra, majd a raspberry termináljában futtatjuk azt.

A motorok üzemeltetését két *DRV8838* típusú egyenáramú motormeghajtó végzi. A kerekek elfordulási szögét egy beépített függvény segítségével kérherjük le, mely enkoder értékben téríti vissza a *Romi Encoder Pair Kit* szenzorai által mért értékeket. Ezek a szenzorok egy mágneses korongal ellátott, a *Hall-effektuson* alapuló digitális szenzorok. A gyártó adatlapja szerint a szenzorok által visszatérített enkoder érték, a kerék egy teljes körbefordulása során 1400-al változik. Ahhoz, hogy a kerekek elfordulási szögét megkapjuk ezeket az értékeket kell radiánba átalakítani [6].

2.3. Hexapod - a hatlábú robot felépítése

Kutatásunk során, azt a **PhantomX AX Metal Hexapod MK-III** típusú hatlábú robotot használtuk és fejlesztettük tovább, melyet Herczeg Ágnes és Harkó Csanád a „Hatlábú

robotok irányítása oszcilláló neuronokkal” című *ETDK* dolgozatukban és Harkó Csanád a „*Hatlábú robot kísérleti tanulmányozása: önszerveződő dinamika csatolt neuronok által*” című államvizsga dolgozatában említene [1, 4].



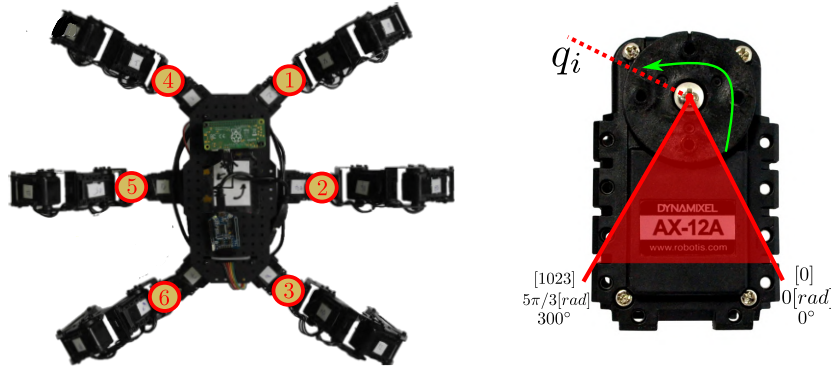
5. ábra. **Bal:** a hatlábú robot szemből nézve. **Középen:** a hexapod "háta" és rajta USB kábelén összekötve a Raspberry Pi Zero 2 W (pirossal bekarikázva) és az FDTI-USB átalakító (késsel bekarikázva). **Jobb:** a robot lábának felépítése és a szegmensek elnevezése (kép forrása: [11]).

A robotot az *Interbotix Labs* cég fejlesztette. A hexapod vázát nagyrészt erős alumínium lemezek alkotják, a lemezeket műanyag merevítő rudak és fém csavarok tartják a helyükön. A merev váz belsejében kap helyet a lítium-polimer akkumulátor (nagy teljesítményű, 2800 mAh-ás *Li-Po*), mely így védve van az esetleges külső mechanikai hatásoktól. A robot vezérlőpanelje egy *Arduino* kompatibilis *ArbotiX-M* típusú *robokontroller* (*robocontroller*), amelyen egy *ATMEGA644p* típusú 8-bites, 16 MHz-es mikrovezérlő kapott helyet. A hexapod mindegyik lába három-három csuklóval rendelkezik, ez így összesen tizennyolc csukló, melyek közül egyikkel a láb a robot testéhez csatlakozik (5 ábra). A csuklók mozgatásáért egy-egy speciális *AX-12A* típusú okos szervomotor felelős (bővebben az okos szervomotorokról a Mellékletben). A robot lábai egyenként egy-egy három szegmensből és három csuklóból álló robotkarnak tekinthetők. A láb szegmenseinek elnevezését a rovar lábak részei ihlették: *coxa*, *femur* és *tibia* (vagyis: csípő, comb és sípcsont). Egyszerűsítés céljából egy láb három csuklója is ezeket a neveket és ezek első betűs rövidítéseit (c, f, t) viseli [4].

A könnyebb és gördülékenyebb kísérletezés érdekében a munkánk során a robotot kibővítettük egy *Raspberry Pi Zero W 2* mini számítógéppel, így a mérési eredményeket WiFi hálózaton keresztül tudja valós időben küldeni.

2.4. A hexapod irányításához szükséges fogalmak

A robot lábainak a számozása megegyezik a 6. ábrán látható értékekkel. Kísérleteink során a tibia csuklók szögét állandó értéken tartottuk, $q^{(t)} = {}^0q^{(t)} = konstans$. A coxa és a femur csukló szögei egy előre meghatározott intervallumban változtathatók: $q^{(c)} \in ({}^{MIN}q^{(c)}, {}^{MAX}q^{(c)})$ és $q^{(f)} \in ({}^{MIN}q^{(f)}, {}^{MAX}q^{(f)})$. A ${}^{MAX}q^{(c)}$ az jelenti, hogy a robot a lábát teljesen előre, míg a ${}^{MIN}q^{(c)}$ azt, hogy a robot a lábát teljesen hátra rakta. A femurok esetében ${}^{MAX}q^{(f)}$ jelenti a teljesen felemelt lábat, míg ${}^{MIN}q^{(f)}$ a teljesen leengedett lábat. Ezeknek a korlátozásoknak egyszerű fizikai okai vannak, így a mozgás során a robot lábai nem érintkeznek egymással. Minden kísérlet elején a robot csuklóit egy előre meghatározott ${}^0q_i^{(t)}, {}^0q_i^{(c)}, {}^0q_i^{(f)}$ kezdőállapotba állítjuk be, ahol i a láb számának felel meg.



6. ábra. **Bal:** A hatlábú robot lábainak számozása. **Jobb:** Az AX-12A szervomotor külső felépítése és elfordulásának pozitív iránya, illetve a tiltott tartomány pirossal jelölve. A servo akadálytalanul 300° fokban képes elfordulni. q_i a szervó aktuális szögét jelöli (az ábra reprodukció: [1]).

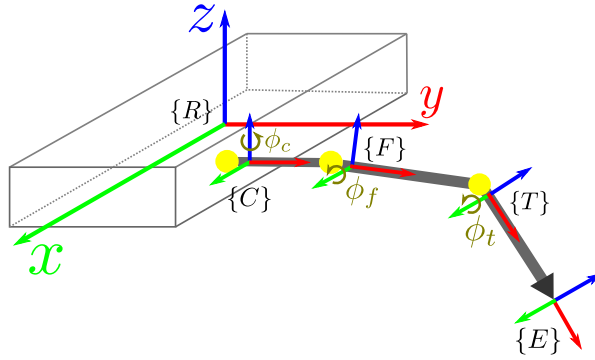
A kezdőállapotok enkoder értékekben a következők:

$$\begin{cases} {}^0q^{(c)} = [450, 512, 573, 573, 512, 450] \\ {}^0q^{(f)} = [670, 670, 670, 350, 350, 350] \\ {}^0q^{(t)} = [250, 250, 250, 774, 774, 774] \end{cases} \quad (11)$$

és a $(0, 1023)$ intervallumban változtathatóak. A csuklópozíciók radiánban a következő képlet segítségével adhatóak meg:

$$q_{rad} = q_{enk} \cdot 51 \cdot 10^{-4} [rad] \quad (12)$$

A robot testéhez rögzítettünk egy koordináta rendszert, $\{R\}$ (ezt személteti a 7. ábra). A koordináta-rendszer origója (O pont) a robot testének közepén található, az OX tengely pozitív



7. ábra. A robot testéhez ($\{R\}$) és lábszegmenseihez rendelt koordináta-rendszerek elhelyezkedése és elnevezése ($\{C\}$ - coxa, $\{F\}$ - femur, $\{T\}$ - tibia, $\{E\}$ - effektor). Illetve a csuklók forgási tengelyei.

iránya megegyezik a robot előre történő haladási irányával, XOY sík egybeesik a robot alsó lemezének síkjával és ha a robot felemeli a lábát, a láb végpontjának mozgása a OZ tengely pozitív irányába történik. A végpont pozíciójának meghatározása érdekében a szegmensekhez is képzeletben koordináta-rendszereket rendelünk, amelyek a csuklók által el vannak forgatva és a szegmensek hossza miatt el vannak tolva egymáshoz képest. Ismerve a szegmensek hosszát, a csuklósögeket, forgatásokat és eltolásokat alkalmazva kiszámíthatjuk a láb végpontjának koordinátáit a robot testéhez rögzített $\{R\}$ koordináta-rendszerben:

$${}^R\mathbf{T}_E = {}^R\mathbf{T}_C(\theta_C, \mathbf{t}_C) \cdot {}^C\mathbf{T}_F(\theta_F, \mathbf{t}_F) \cdot {}^F\mathbf{T}_T(\theta_T, \mathbf{t}_T) \cdot {}^T\mathbf{T}_E(\mathbf{t}_E), \quad (13)$$

ahol ${}^j\mathbf{T}_i(\theta_i, \mathbf{t}_i)$ a homogén transzformációs mátrixa, $\mathbf{t}_i$ egy három elemű vektor, amely két egymást követő koordináta-rendszer eltolását tartalmazza (a szegmensek hosszából meghatározható), és θ_i a két egymást követő koordináta-rendszer közötti elforgatási szög, mely könnyen meghatározható a coxa (C), femur (F) és tibia (T) csuklósögek ismeretében ($q^{(c)}$, $q^{(f)}$, $q^{(t)}$). Ezeket a transzformációkat felhasználva tudjuk meghatározni a láb végpontjának pozíciójának OZ tengely menti komponensét minden időpillanatban. Ha ezeket az adatokat ábrázoljuk az idő függvényében egy hőképen ("heatmap") jól láthatóvá válnak az egyes mozgásokhoz tartozó lépésmintázatok.

2.5. A hexapod programozása és saját fejlesztések

A robotra szánt algoritmusokat az Arduino sajátos, C++-on alapuló, nyelvezetén kell megírni. A robot vezérléséhez és a kommunikációhoz, az esetek nagytöbbségében a fejlesztők által megírt, beépített függvényeket használjuk. A kódot az *ArbotiX-M* robotkontrollerre az *FTDI*

kivezetésein keresztül tölthetjük fel a számítógépről, ehhez azonban szükségünk vagy egy *FTDI-USB* átalakítóra, mely lehetővé teszi, hogy a számítógép USB portja kommunikálhasson a robotkontroller FTDI csatlakozóival (további részletek a hexapod programozásáról a Mellékletben).

A raspberryvel történő kibővítés előtt, a robottal való kommunikáció egy USB kábelén keresztül valósult meg. Ezen keresztül kellett feltölteni a kódokat a robotra és a robot is ezen keresztül küldte a mért adatokat. Ez azt jelentette, hogy a lépésmintázatok vizsgálatakor a robot után kellett sétálni a lappal, hogy a mért adatokat továbbítani és menteni tudjuk a számítógépre. Ez sem kényelmi, sem kísérleti szempontból nem volt megfelelő, mivel akaratunk ellenére is az USB kábelén keresztül kölcsönhatottunk a robottal és így meg volt annak az esélye, hogy megváltoztassuk a lépésmintázatok és nem lesz releváns az adott mérés.

A raspberry közbeiktatásával azonban egy vezeték nélküli kommunikációt tettünk lehetővé a számítógép és a robot között, ehhez csak annyi szükséges, hogy azonos Wi-Fi hálózathoz legyen csatlakoztatva mindkét eszköz. Az általunk megírt Arduino kódot SSH protokolon keresztül feltöltjük a raspberry pi-re, terminál parancsokat használva a raspberry képes a kódot olyan nyelvre lefordítani, melyet a robot mikrovezérlője értelmezhet. Mivel a raspberry és a robot mikrovezérlője egy USB kábelén és az FTDI-USB átalakítón keresztül van összekapcsolva, szintén terminál parancsokat használva, a raspberryn lefordított kódot fel tudjuk tölteni a mikrovezérlőre. Ezt követően a robot elindításához az szükséges, hogy futtasuk a raspberryn egy python programot, amely bemenetként a felhasználótól megkapja a robot konfigurációs paramétereit. A paramétereket *soros kommunikációt* (UART kommunikáció) használva átküldi a robotnak, amely a paraméterek célbaérése után elindul. A raspberryn futó program ezek után "fogadó" üzemmódba vált és a robot által soros kommunikáción keresztül küldött, mért adatokat fogadja és lokálisan menti a raspberry tárhelybe *text* fájl formájában. A kísérletek befejezése után a raspberryre mentett fájlokat a felhasználó SSH protokolon keresztül másolhatja át a számítógépére.

Az említett fejlesztések nagyban megkönnyítették és felgyorsították a sorozatos kísérletezéseket és kiküszöbölték azt a problémát, hogy interakcióba lépünk a robottal a mérés során, ezzel elrontva az adatokat.

3. Vezérlés multistabil dinamikai rendszerekkel

Napjainkra a klasszikus vezérlési módszerek mellett, a robofizika területén rengeteg dinamikai rendszert dolgoztak ki különféle robotok vezérlésére. Talán a legismertebb ilyen dinamikai rendszer, amelyben megfigyelhető valamiféle kollektív viselkedés az a Kuramoto-rendszer, mely csatolt fázisoszcillátorokból áll. Továbbá a multistabil dinamikai rendszerek közé tartoznak az oszcilláló neuronokból álló rendszerek is, melyek köré dolgozatunk témája épül. Ebben a fejezetben bemutatjuk, hogy hogyan is kell elképzelni egy-egy ilyen neurális rendszert és általuk hogyan vezérelhetjük a robotokat.

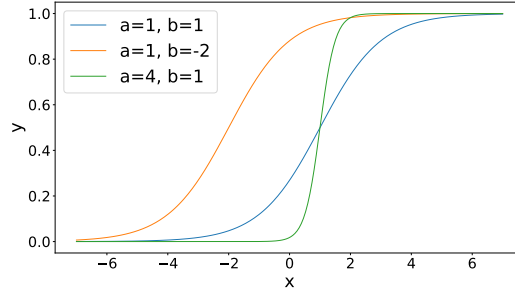
3.1. Neuronokról fizikus szemmel

Annak érdekében, hogy megértsük a neuronokra épülő modelleket, ebben az alfejezetben röviden bemutatjuk, hogyan is viselkednek a neuronok (idegsejtek) a komplex biológiai rendszerekben. Egy olyan összetett szervben, mint például az emberi agy, idegsejtek milliárdjai felelősek a fizikai és kémiai jelek és információk közvetítéséért. Az ilyen összetett neurális hálózatokban egyetlen idegsejt rengeteg másik idegsejttel lehet kapcsolatban. Amikor egyik idegsejt axonja, hozzá csatlakozik a másik idegsejt dendritjéhez kialakul a *szinapszis*. A szinapszis az a két idegsejt közötti kapcsolódási hely, ahol az *ingerület* egyik neuronról áterjed a másikra [2].

Egy neuron dentritje tehát, sok másik neurontól fogad ingerületeket, persze nem egyforma "erősséggel", vagyis különböző súlyfaktorok határozzák meg, hogy az adott neuron mennyire hat kölcsön a többi neuronnal. Ha a többi neurontól érkező ingerek átlépnek egy küszöbértéket (aktivációs küszöbérték), a szakirodalmi megnevezés szerint, egyszerű modellekben ha a neuron *tüzel*, kimenete 1 lesz, máskülönben 0. [7]. A neuronok tüzelése az a folyamat, amely során az idegsejt létrehoz egy akciós potenciált. A dolgozatunkban vizsgált modellek a neuronok ezen tulajdonságát használják fel, mivel az átlagos tüzelési ráta megadható egy aktivációs függvény segítségével [1, 2]:

$$y(x) = \frac{1}{1 + \exp[a \cdot (b - x)]}, \quad (14)$$

ahol az a paraméter a szigmoid függvény meredekségét adja meg, vagyis azt, hogy milyen gyorsan váltson a neuron alacsony aktivitási szintről magas aktivitási szintre (0-ról 1-re). A b paraméter pedig a már fent említett aktivációs küszöbérték, ez határozza meg, hogy milyen membránpotenciál esetén vált aktivitási állapotot az idegsejt [2]. Az egyenletben szereplő



8. ábra. Az aktivációs függvény (y) alakja különböző meredekség (a) és aktivációs küszöbérték (b) esetén.

x változó, az úgynevezett membrán potenciál, melynek értékét a vele kapcsolatban lévő idegsejtből érkező ingerek befolyásolják. A szigmoid függvény viselkedését különböző a , b paraméterek esetén az 8. ábra szemlélteti. Egy több neuront tartalmazó, komplex, szinapszisok által csatolt rendszer időben való változása a következő egyenletekkel írható le [1]:

$$\tau_{xi} \cdot \dot{x}_i = -x_i + \sum_j w_{ij} \cdot y_j(x_j), \quad (15)$$

ahol τ_{xi} az i . neuron sajátideje, w_{ij} az i . és j . idegsejt közötti kölcsönhatás erőssége, $y_j(x_j)$ a j . neuron átlagos aktivitása.

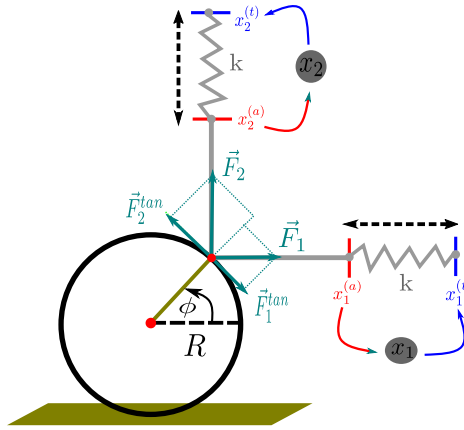
A **homeosztázis** az az állapot, amikor az idegsejtet optimális paraméterek, működési körülmények jellemzik. Az ilyen optimális körülmény megteremtésének érdekében az idegsejt képes egy belső változást, adaptációt végrehajtani, mégpedig megtudja változtatni a már említett aktivációs küszöbértékét (b paraméter) [1]. Tehát a rendszer és a neuronok teljes, összesített dinamikája a következőképpen írható le:

$$\begin{cases} \tau_{xi} \cdot \dot{x}_i = -x_i + \sum_j w_{ij} \cdot y_j(x_j) \\ \tau_{bi} \cdot \dot{b}_i = y(x_i) - \frac{1}{2}, \end{cases} \quad (16)$$

ahol τ_{bi} a neuron aktivációs küszöbértékének sajátideje [1] (bővebben: [12]).

3.2. Kerek robot vezérlése neuronokkal

Az általunk használt Romi robot két aktív, DC motorok által meghajtott és egy passzív (nem vezérelhető) stabilizáló kerékkel rendelkezik. Ebben az alfejezetben bemutatjuk, azt a már ismert modellt, amely neuronok tüzelését, aktivációs rátáját használja arra, hogy a robot kerekeit



9. ábra. A kerék meghajtására használt rugós modell. $x_i^{(a)}$ az aktuális pozíció, $x_i^{(t)}$ a célpozíció, k a rugóállandó és x_i a neuron membránpotenciálja.

vezérelje, így kialakítva sokszínű önszerveződő dinamikát és mozgásformát [8]. A vizsgált modell keretén belül a robot egyetlen kerekét két idegsejt irányítja. A neurális aktivitás forgó mozgássá alakítható, a hagyományos gőzmozdonyok, gőzgépek erőátviteli karjához hasonló módszerrel, amely a túlnyomásos dugattyú mozgását alkította forgó mozgássá (9. ábra).

Esetünkben a neuronok aktivitását két szimulált rugóerővel alakítjuk át forgó mozgássá és így irányítjuk a robot kerekeit. A rendszert úgy kell elképzelni, mintha a kerék egy kerületi pontjához két ideális, k rugalmassági állandójú rugó lenne erősítve, egymásra merőleges irányban (egy függőlegesen és egy vízszintesen). Az elképzelt modell vázlatos felépítését a 9. ábra szemlélteti. A rugókban fellépő rugalmassági erők:

$$F_i = k \cdot (x_i^{(t)} - x_i^{(a)}), \quad (17)$$

ahol $x_i^{(t)}$ a kerék normált célpozíciója és $x_i^{(a)}$ a kerék normált jelenlegi pozíciója. A kerék ϕ elfordulási szögét ismerve, $x_i^{(a)}$ az aktuális pozíció kiszámolható, mint az aktuális szög vetülete a függőleges és vízszintes irányokra:

$$x_1^{(a)} = \cos(\phi); \quad x_2^{(a)} = \sin(\phi) \quad (18)$$

Innen látható, hogy $x_i^{(a)} \in (-1, 1)$. A célpozíciók ($x_i^{(t)} \in (-1, 1)$) pedig megadhatók a két független vezérlő neuron aktivitását (y) felhasználva:

$$x_i^{(t)} = 2y(x_i) - 1 \quad (19)$$

és esetünkben az aktivációs függvényhez a küszöbérték $b = 0$. Tehát a membránpotenciálok dinamikája leírható, az alábbi differenciál egyenlettel:

$$\tau \dot{x}_i = x_i^{(a)} - x_i, \quad i = 1, 2 \quad (20)$$

τ pedig a neuronok sajátideje, ez tipikusan pár száz milliszekundum [8]. A kerékre ható tangenciális erő megadható mint az egyes rugóerők tangenciális komponensének az összege:

$$F^{tan} = F_1^{tan} + F_2^{tan} = F_1 \cdot \sin(\phi) - F_2 \cdot \cos(\phi) \quad (21)$$

A robot irányítása során a kerekeket meghajtó motorok sebességét tudjuk szabályozni, ezért a kerékre ható tangenciális erő (F^{tan}) értékét át kell skáláznunk úgy, hogy általa a kerekek sebességét tudjuk befolyásolni:

$$V = V_{max} \cdot \tanh(F^{tan}), \quad (22)$$

ahol V lesz az aktuális enkoder érték ami bemenetként beállítunk a kerék sebességét szabályozó függvénynek és V_{max} a megengedett maximális sebesség enkoder értéke.

A robot kerekei egymástól függetlenül irányíthatók, ezért a robot másik kereke hasonlóan van vezérelve. Mivel a robot kerekeit egymástól független rendszerek vezérlik a program futása közben a két rendszer között kizárólag csak mechanikai kapcsolat áll fent. A robot teste merev, így a vázon és a környezeten keresztül a két kerék kölcsönhat egymással ezzel befolyásolva a különálló dinamikai rendszerek viselkedését. Olyan komplex viselkedések alakulhatnak ki, amelyek befolyásolhatják a önálló rendszerek esetében kialakuló attraktorokat [8].

3.2.1. Impulzus-vezérlés

A dolgozatunkban továbbiakban vizsgáljuk, hogy a rendszer egy-egy stabil attraktoráról milyen módon tudjuk átmozdítani a rendszert egy másik stabil attraktorra. Az első és legkézenfekvőbb megoldás az, ha a robot környezetével való kölcsönhatás (beragad a kereke, falnak ütközik) során vált az egyik stabil attraktorról a másikra, anélkül, hogy a rendszer valamely paraméterét megváltoztattuk volna.

Egy alternatív megoldás, az ha szimuláció során a dinamikai rendszerünket egy kissé megperturbáljuk. Ez azt eredményezi, hogy a rendszer *fázispontja* a *fázistérben* kissé elmozdul

és egy ilyen kis változtatás, a megfelelő pillanatban elég lehet ahhoz, hogy a rendszer az egyik stabil attraktorról átkerüljön egy másik dinamikát eredményező, stabil attraktor vonzáskörébe.

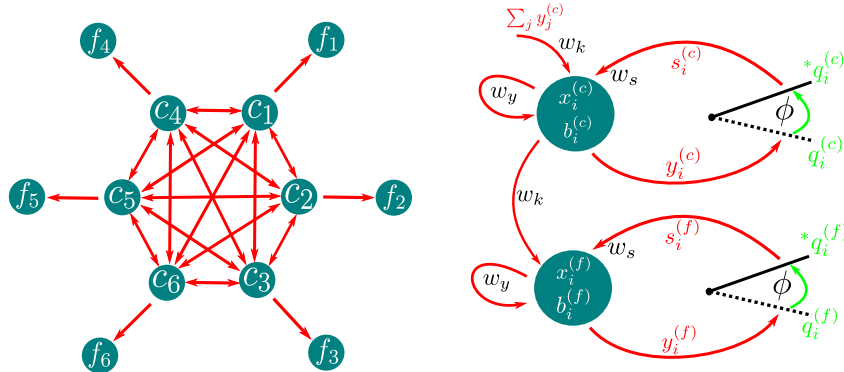
Az általunk vizsgált impulzus-vezérlés ("kick control") során egyszerűen egy adott pillanatban megváltoztattuk a kerekeket vezérlő idegsejtek membránpotenciáljának előjelét, vagyis [8]:

$$x_1 \rightarrow -x_1; \quad x_2 \rightarrow -x_2 \quad (23)$$

Egy ilyen kis változtatás elég ahhoz például, hogy a rendszer az előre haladó mozgásnak megfelelő attraktorról átmenjen a hátrafele történő mozgás attraktorára. Természetesen a "kick control" hatása függ attól, hogy a fázispont a fázistér melyik pontjában van éppen.

3.3. Hatlábú robot vezérlése neuronokkal

A hatlábú robot vezérléséhez egy tizenkét neuronból álló, csatolt rendszert használunk fel, amelyet egy korábbi ETDK dolgozatban javasoltak [1]. A robot csuklóinak a mozgásáért egy-egy neuron felelős, tehát van hat darab coxa és hat darab femur csuklót vezérlő neuronunk. A tibia csuklókat fix pozícióban tartjuk, nem vezéreljük. Egy olyan neurális hálózatot hozunk létre, amelyben a coxa neuronok az összes többi coxa neuronnal kölcsönhatásban vannak és a kölcsönhatás erősségét a w_k paraméter (súlyfaktor) határozza meg. A coxákkal ellentétben a femur csuklók, csak az adott lábon lévő coxa csuklóval vannak kölcsönhatásban, ezt a kölcsönhatást is az w_k csatolási paraméterrel jellemezzük. A neuronok képesek önmagukat is gerjeszteni, ennek a *visszacsatolásnak* az erősségét egy w_y súlyfaktor skálázza. Továbbá a neuronok információt kapnak az aktuális pozíciójukról is, annak érdekében, hogy visszajelzést kapjanak mennyire sikerült elérniük a célpozíciót. Ezt a visszajelzést egy w_s súlyfaktoron és egy $s(q_i)$ függvényen keresztül kapják meg. A csatolásokat a 10. ábra mutatja be.



10. ábra. **Bal:** A coxa (c) és femur (f) neuronok közötti kölcsönhatások és a csatolások iránya. **Jobb:** Egy coxa (c) és a hozzá tartozó femur (f) neuronok kapcsolatának, működésének bemutatása. Mindkét neuron szerepe, hogy a hozzájuk rendelt aktuátor jelenlegi szögét (q_i) eljuttatni a célszög értékre (*q_i), az aktivitásuk segítségével (y_i). A coxa neuron információt kap: w_k súllyal a többi coxa neuron aktivitásáról, w_y súllyal a saját aktivitásáról és w_s súllyal a célpozíció elérésének sikeréről a s_i visszacsatolási függvényen keresztül. Továbbá a femur neuron bemenetei: w_k súllyal a hozzá tartozó coxa neuron aktivitása, w_y súllyal a saját aktivitása és w_s súllyal a s_i visszacsatolási függvény (az ábra reprodukció [1])

Az $s(q_i)$ a y aktivációs függvényhez hasonlóan normált a $[0,1]$ intervallumra. Ezeket a kölcsönhatásokat figyelembe véve a következő csatolt differenciál egyenlet rendszer írható fel a teljes rendszerre [1]:

$$\begin{cases} \tau_x^{(c)} \cdot \dot{x}_i^{(c)} = -x_i^{(c)} + w_y y_i^{(c)} + w_k \sum_{j \neq i} y_j^{(c)} + w_s s_i^{(c)} \\ \tau_b \cdot \dot{b}_i^{(c)} = y_i^{(c)} - \frac{1}{2} \\ \tau_x^{(f)} \cdot \dot{x}_i^{(f)} = -x_i^{(f)} + w_y y_i^{(f)} + w_k y_i^{(c)} + w_s s_i^{(f)} \\ \tau_b \cdot \dot{b}_i^{(f)} = y_i^{(f)} - \frac{1}{2} \end{cases} \quad (24)$$

Esetükben az aktivációs függvény alakja [1]:

$$y_i^{(c,f)} = \frac{1}{1 + \exp[a \cdot (b_i^{(c,f)} - x_i^{(c,f)})]} \quad (25)$$

Az $s_i^{(c,f)}(q_i^{(c,f)})$ visszacsatolási függvény alakja [1]:

$$s_i^{(c,f)}(q_i^{(c,f)}) = \frac{q_i^{(c,f)} - \text{MIN } q_i^{(c,f)}}{\text{MAX } q_i^{(c,f)} - \text{MIN } q_i^{(c,f)}}, \quad (26)$$

ahol $q_i^{(c,f)}$ a csuklók aktuális pozíciója, amely $q_i^{(c,f)} \in [\text{MIN } q_i^{(c,f)}, \text{MAX } q_i^{(c,f)}]$ eleme az általunk beállított, a csuklók által elérhető minimális és maximális értékek által határolt intervallumnak (ezt a 2.4. fejezetben részletesen tárgyaljuk).

A csuklóknak beállított célpozíciót pedig a neuronok aktivitását felhasználva számoljuk ki [1]:

$${}^*q_i^{(c,f)} = MIN q_i^{(c,f)} + y_i^{(c,f)} (MAX q_i^{(c,f)} - MIN q_i^{(c,f)}). \quad (27)$$

Látható, hogy ebben az esetben is átskáláztuk az értékeket a 2.4. fejezetben említett minimális és maximális értékek közé, hogy a robot lábai ne érhessenek el a dinamika során nemkívánt pozíciót. Továbbá, egy ilyen skálázás azt is eredményezi, hogy ha a neuron aktivitása alacsony a robot lába a minimális érték fele mozdul el, magas aktivitás esetén pedig a maximális érték irányába [1].

4. Kísérleti eredmények

4.1. Kerekes robot neuronokkal

A 3.2 fejezetben bemutatott modellt implementáltuk a *Pololu Robotics & Electronics* által fejlesztett Romi kétkerekű robotra. A (20). egyenletet mindkét kereket irányító neuronok membránpotenciálja esetén felírhatjuk. Az egyenleteket diszkretizáltuk és Euler integrálási módszerrel megoldottuk [8]. A diszkrét idejűvé redukálás után az egyenleteink a következő alakot öltik:

$$\begin{cases} {}^{(R,L)}x_1(t + \Delta t) = {}^{(R,L)}x_1(t) + [({}^{(R,L)}x_1^{(a)} - {}^{(R,L)}x_1)\frac{1}{\tau}] \cdot \Delta t \\ {}^{(R,L)}x_2(t + \Delta t) = {}^{(R,L)}x_2(t) + [({}^{(R,L)}x_2^{(a)} - {}^{(R,L)}x_2)\frac{1}{\tau}] \cdot \Delta t \end{cases} \quad (28)$$

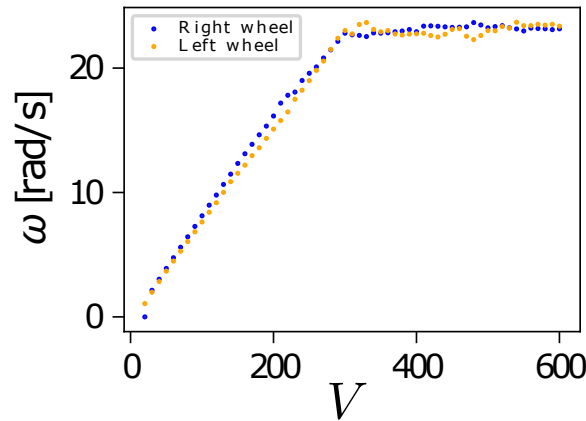
ahol a Δt a program egy ciklusának lefutásához szükséges idő, az $x_i^{(a)}$ pozíciók a (18) egyenlet segítségével határozhatók meg és az (R, L) felső index azt jelöli, hogy melyik kereket irányító neuronokról beszélünk (R - right, L - left).

Az általunk Python programozási nyelvben megírt, a robot vezérlésére használt program igencsak egyszerű felépítéssel rendelkezik:

- szükséges könyvtárak importálása
- az modellhez szükséges állandók és a rendszer kezdőértékeinek definiálása (${}^{(R,L)}x_i^{(0)}$, a , k , V_{max} és τ)
- létrehozunk egy *while()* loopot, mely a ciklikusságot biztosítja. Ebben a ciklusban történik a robot tényleges vezérlése, mely a robot kikapcsolásáig folyamatosan fut.
- a kerék elfordulási szögének lekérése (enkoder értékben)
- ${}^{(R,L)}x_i^{(a)}$ aktuális pozíciók kiszámítása (18. képlettel)
- ${}^{(R,L)}x_i(t + \Delta t)$ értékek kiszámítása Euler módszerrel
- új ${}^{(R,L)}x_i^{(t)}$ célpozíciók számolása a (19). képlet segítségével
- F_i erők számolása (17). képletet használva
- a F^{tan} tangenciális erő kiszámítása (21). képlet segítségével

- az V sebességek kiszámítása enkoder értékekben (22. képlet)
- új forgási sebességek beállítása
- mért és számolt adatok kiírása (kerekek elfordulási szöge, jelenlegi és célpozíciók, beállított forgási sebességek enkoder értékekben).

Először vizsgáljuk meg, milyen sebesség tartományban lehet a roboton levő motorok sebességét beállítani a beépített függvény bemeneti paraméterének (V) függvényében. Ennek érdekében 20 és 600 között 10-es lépésekkel változtattuk a kerék sebességét szabályozó függvény bemenetét. Egy adott érték esetén 20 másodpercig hagytuk a kereket forogni, miközben mértük a szögsebességet és es ebből egy átlagos szögsebesség értéket számoltunk (11. ábra).

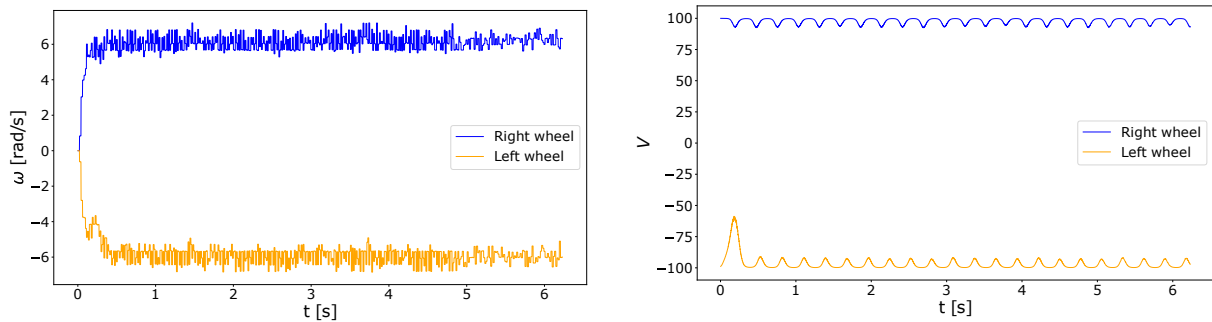


11. ábra. A Romi robot kerekeinek mért átlagos szögsebessége, a kerekek sebességét szabályozó függvény bemeneti paraméterének (V) függvényében.

A bemeneti paraméter függvényében a kerekek szögsebessége lineárisan nő, amíg el nem érjük a 300-as bemeneti értéket. A 300-as értéknél nagyobb esetekben, ugyan a függvény elfogadja a kapott értéket, a kerekek szögsebessége nem nő tovább, az általunk elért $\omega_{max} = 23 \pm 0.5$ [rad/s] értéknél. Ez az érték kissé változhat és a kísérletek azt mutatják, hogy nagyban függ a robotot ellátó akumlátorok töltöttségétől.

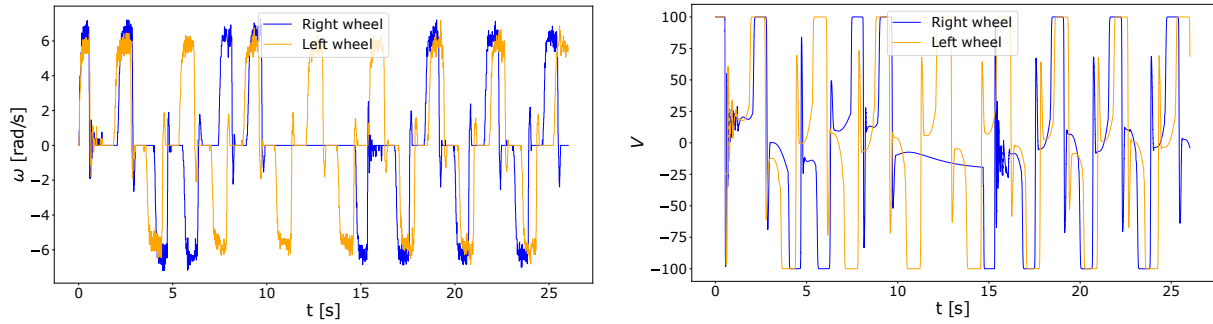
A robot irányítására felhasznált neurális modell, mivel egy multistabil dinamikai rendszer több stabil attraktort rejt magában, a különböző attraktorok pedig eltérő mozgásdinamikát eredményeznek. Kísérleteink során véletlenszerű, ${}^{(R,L)}x_i^{(0)} \in [-1, 1]$ kezdőértékekből indítottuk a dinamikai rendszer megoldását. A rendszert jellemző paramétereket (a és k) változtatva kerestük a lehetséges mozgásformákat.

Egy lehetséges attraktor, amely számunkra kedvező és a robot irányítása szempontjából megfelelő dinamikát eredményez, az amelyik a kerekek előre vagy hátrafele történő forgatására készíti a robotot. Mivel ha mindkét kerék előre, vagy hátra forog egy haladó mozgást tudunk elérni. Illetve ha a két kerék ellentétes irányba, vagy azonos irányba de eltérő sebességgel forog a robot kanyarodását is elő tudjuk idézni. Egy ilyen előre/hátra-felé történő forgó mozgást mutat be a 12. ábrán látható két grafikon. A 12. ábra **bal** oldali grafikonján jól látható, hogy a véletlenszerű kezdőfeltételből indított rendszernek kis időre van szüksége, amíg a jobb és a bal kerék fázispontja rákonvergál a stabil attraktorra. Ebben az esetben a jobb kerék fázispontja (right wheel) az előre felé forgó attraktorra konvergált rá (a sebesség értéke pozitív ha a kerék előre forog), míg a bal oldali kerék fázispontja (left wheel) a hátra forgó attraktorhoz tart (a sebesség negatív ha a kerék hátrafele forog). A robot esetében ez egy helyben történő, saját tengelye körüli forgást eredményez, mivel a két kerék azonos sebességgel ellentétes irányba forog. A **jobb** oldali grafikon, a kerekek sebességét szabályzó V paraméter dinamikáját szemlélteti az időben.



12. ábra. **Bal:** A kerekek sebessége az idő függvényében. A jobb oldali kerék (right wheel) előre forog (sebessége pozitív), míg a bal oldali kerék (left wheel) hátrafele forog (sebessége negatív). **Jobb:** A kerekeket vezérlő függvény (V) időbeni dinamikája. ($V_{max} = 100$, $\tau = 100ms$, $a = 4$, $k = 5$)

Egy másik lehetséges attraktor *kaotikus* dinamikának felel meg. A kezdeti paraméterektől függően, ha a rendszer fázispontja ehhez az attraktorhoz konvergál a robot mozgása és a kerekek forgási sebessége, illetve forgási iránya előrejelezhetetlen lesz. A robot egy szabálytalan mozgást végez, miközben előre, hátra halad, vagy éppen forog.

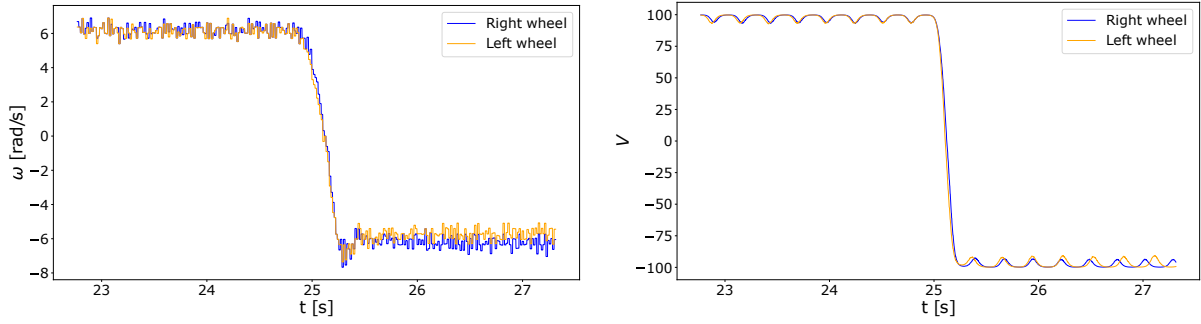


13. ábra. **Bal:** A kerek sebessége az idő függvényében kaotikus mozgás esetén. A robot kerekjei semmilyen periódikus mozgást nem végeznek. A sebességértékek előrejelezhetetlenül váltják egymást. **Jobb:** A kaotikus viselkedés jól kivehető a V paraméter időbeli változásából is ($V_{max} = 100$, $\tau = 2000ms$, $a = 4$, $k = 30$).

4.1.1. Kick control

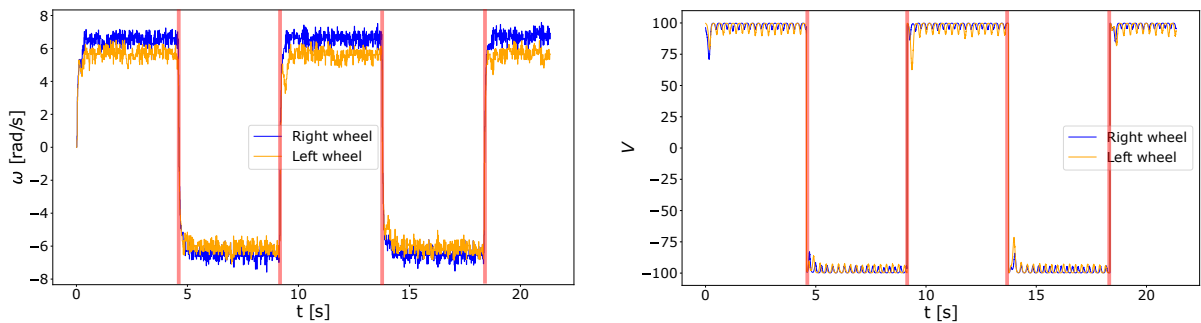
A rendszert kissé megperturbálva a kellő időpillanatban a fázispont átugorhat egyik stabil attraktorról a másikra. Esetünkben az előre forgató attraktorról a hátra forgató attraktorra, vagy fordítva. Ez a perturbáció többféle lehet, származhat a környezettel való kölcsönhatásból vagy egy belső pillanatszerű numerikus változtatásból. Dolgozatunkban ezeket az eseteket vizsgáljuk.

A környezettel való kölcsönhatást a robot a kerekein keresztül "érzi", mivel mérjük az aktuális kerék pozíciót és ez bemenetként szolgál a dinamikai rendszerünknek. Ha a kerék beakad, vagy a talaj miatt nehezebben tud fordulni, az aktuális pozíció mérése által ezek a hatások befolyásolják a dinamikai rendszer megoldását és ezzel a fázispont helyzetét a fázistérben. Egy általunk vizsgált környezeti hatás az, amikor a robot nekiütközik egy előtte levő akadálnak, például a falnak. Az ütközés megtörténte után a robot tovább szeretne menni egyenesen, de az akadály miatt nem tud, ezért a robot kerekje megcsúsznak és kipörögnek. A kipörögés során a talajjal való súrlódás miatt lelassulnak és már nem érik el a célpozíciót olyan gyorsan mint azelőtt. Az aktuális pozíció mérése által módosul ebben az esetben a rendszer megoldása és megtörténhet, hogy a fázispont "átugrik" az előre forgató attraktorról a hátra forgató attraktorra és a robot elindul hátrafele, így eltávolodva a faltól [8]. Egy ilyen esetet a 14. ábra szemléltet.



14. ábra. **Bal:** A kerekek sebessége az idő függvényében, miközben visszafordul a faltól. **Jobb:** A V vezérlő paraméter az idő függvényében ($V_{max} = 100$, $\tau = 100ms$, $a = 4$, $k = 5$). A robot nekimegy a falnak, a padlóval való súrlódás következtében a kerekek forgása lelassul és ezáltal megváltozik a rendszer dinamikája. Mivel a jobb és a bal kereket két különálló dinamikai rendszer vezérli, a kerekek forgási sebessége és iránya nem teljesen egyidőben és azonos módon változik meg. Azonban a robot teste által a kerekek kölcsönhatnak, így ösztönözve egymást az azonos dinamika követésére [8].

Egy másik lehetőség az attraktor váltásra az lehet, ha numerikusan perturbáljuk meg a rendszerünket. Erre azt a megoldást használjuk, hogy egy adott időpillanatban, egy Δt időintervallum erejéig megváltoztatjuk a membránpotenciál (x) előjelét, ahogy azt a (23). képlet is mutatja. Ha az impulzus megfelelő időpillanatban érkezik a kerék fázispontja átugrik az előre forgató attraktorról a hátra forgató attraktorra [8]. Ezt szemlélteti a 15. ábra.



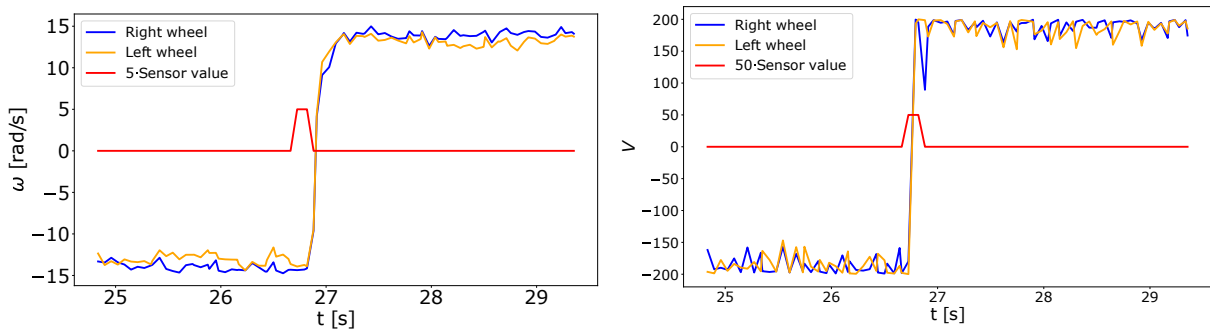
15. ábra. **Bal:** A kerekek sebessége az idő függvényében. **Jobb:** A kerekek sebességét megadó V paraméter az idő függvényében ($V_{max} = 100$, $\tau = 100ms$, $a = 4$, $k = 20$). A függőleges piros vonalak jelzik, hogy a mozgás során mikor történt a "kick". Jól látható, hogy ebben az esetben a jobb és a bal kerékhez tartozó fázispontok egyszerre ugranak át egyik stabil attraktorról a másikra, mivel a "kick" mindkét rendszer esetében azonos időben történt. A robot előre mozog, majd hátra és ez ismétlődik a "kick" hatására [8].

A kísérletezés során felszereltünk a robotra négy darab VL53L0X távolságmérő szenzort, az 3. ábrán látható módon. Ezekkel az volt a célunk, hogy a robot máshonnan is információt tudjon szerezni a környezetéről, ne csak a kerekeken keresztül. Annak érdekében, hogy a távolságmérő szenzorok adatait felhasználva tudjuk a rendszert átvinni egyik stabil attraktorról a másikra, az

eredetileg használt (20). egyenletet a következőképpen módosítottuk:

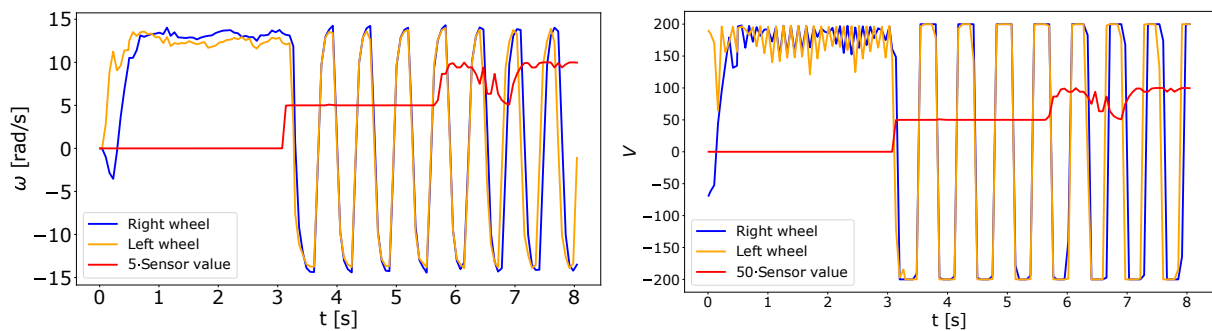
$$\tau \dot{x}_i = x_i^{(a)} - x_i + s, \quad i = 1, 2 \quad (29)$$

ahol az $s = s_1 + s_2 + s_3 + s_4$ ($s \in [0, 4]$) a négy szenzorból érkező adat összege és $s_i = \frac{1}{1 + \exp[\gamma \cdot (\beta - d_i)]}$, ahol γ , β állandó paraméterek (esetünkben $\gamma = -10$ és $\beta = 10$) és d_i az i . szenzor által mért távolság. Ezt a rendszert implementálva a robotra a 16. ábrán bemutatott eredményeket értük el. A mozgás kezdetén a robot tolat, mivel a szögsebességek értéke negatív. Az akadály messze van, ezért az s paraméter (*Sensor value*) nulla. Amikor a robot elegendően közel ér az akadályhoz (közelebb, mint 10 cm, mivel $\beta = 10$) az s paraméter értéke hirtelen megugrik. Ez a rendszer esetében azt eredményezheti, hogy a kerekek fázispontja átugrik a hátra forgató attrakorról az előre forgató attraktorra. Ezt szemlélteti a 16. ábra. Látható, hogy az s paraméter hirtelen megváltozásával a V vezérlő paraméter szinte azonnal előjelet vált. A robot tehetetlensége miatt a kerekek sebessége ezt időben egy kissé eltolva teszi meg.



16. ábra. **Bal:** A robot kerekeinek szögsebessége és az s (*sensor value*) paraméter (piros görbe) az idő függvényében. Kezdetben a robot kerekei hátrafele forognak (a szögsebességek értéke negatív). Az akadály messze van, ezért az s paraméter nulla. Amikor a robot elegendően közel ér az akadályhoz az s paraméter értéke megugrik és a robot mozgásirányt vált. **Jobb:** A sebességet kontrolláló V paraméter és az s paraméter (piros görbe) az idő függvényében. ($V_{max} = 200$, $\tau = 100ms$, $a = 4$, $k = 5$).

A *kick control* irányítások esetén fontos az, hogy a rendszer fázispontját a megfelelő időpillanatban perturbáljuk meg ahhoz, hogy át tudjon "ugrani" egy másik attraktorra. A távolságmérő szenzoros perturbáció esetén is ez a helyzet. Amennyiben a szenzoroktól származó jel nem a megfelelő pillanatban érkezik, a robot az akadály előtt egy oszcilláló mozgást kezd el végezni. Ez látható 17. ábrán. Az előre-hátra mozgás addig folytatódik, amíg a robot valamilyen irányba, az oszcillálás következtében el nem távolodik az akadálytól.



17. ábra. Az s bemenő jel nem a megfelelő időben perturbálta a dinamikai rendszert, ezért a robot oszcilláló mozgást kezd el végezni az akadály előtt. **Bal:** A robot kerekeinek szögsebessége és az s (*sensor value*) paraméter (piros görbe) az idő függvényében. **Jobb:** A sebességet kontrolláló V és az s görbék (piros) az idő függvényében. ($V_{max} = 200$, $\tau = 100ms$, $a = 4$, $k = 5$).

4.2. Hatlábú robot neuronokkal

Kísérleteink során a PhantomX hatlábú robot esetén is hasonló képpen kellett eljárunk mint a Romi robot esetében. Ahhoz, hogy a 3.3. fejezetben említett neurális modell dinamikáját felhasználva vezérelni tudjuk a robotunkat, numerikus integrálási módszert alkalmazva meg kell oldjuk a (24). összefüggésben szereplő csatolt differenciálegyenlet rendszert minden egyes coxa és femur neuron esetén. Herczeg Ágnes és Harkó Csanád a „*Hatlábú robotok irányítása oszcilláló neuronokkal*” című *ETDK* dolgozatukban [1] és Harkó Csanád a „*Hatlábú robot kísérleti tanulmányozása: önszerveződő dinamika csatolt neuronok által*” [4] című államvizsga dolgozatában említett Euler integrálási módszert helyettesítettük egy negyedrendű Runge-Kutta módszerrel. Erre azért volt szükség, mivel a kísérletezések során észrevettük, hogy néhány olyan esetben, ahol a numerikus szimulálás fixpontot eredményez a rendszer esetében, a robot nem megy be ebbe a fixpontba, hanem egy nagyon gyors oszcilláló mozgást végez. Mint kiderült ez az Euler integrálási módszer pontatlanságának tudható be. Runge-Kutta módszert alkalmazva ez a gyors oszcillálás megszűnt és a fázispont a robot esetében is bement a fixpontba (a robot nem mozdult).

A dinamikai rendszer megoldásáért és a robot vezérléséért egy Arduino programozási nyelvben megírt algoritmus a felelős, melynek főbb részei a következők:

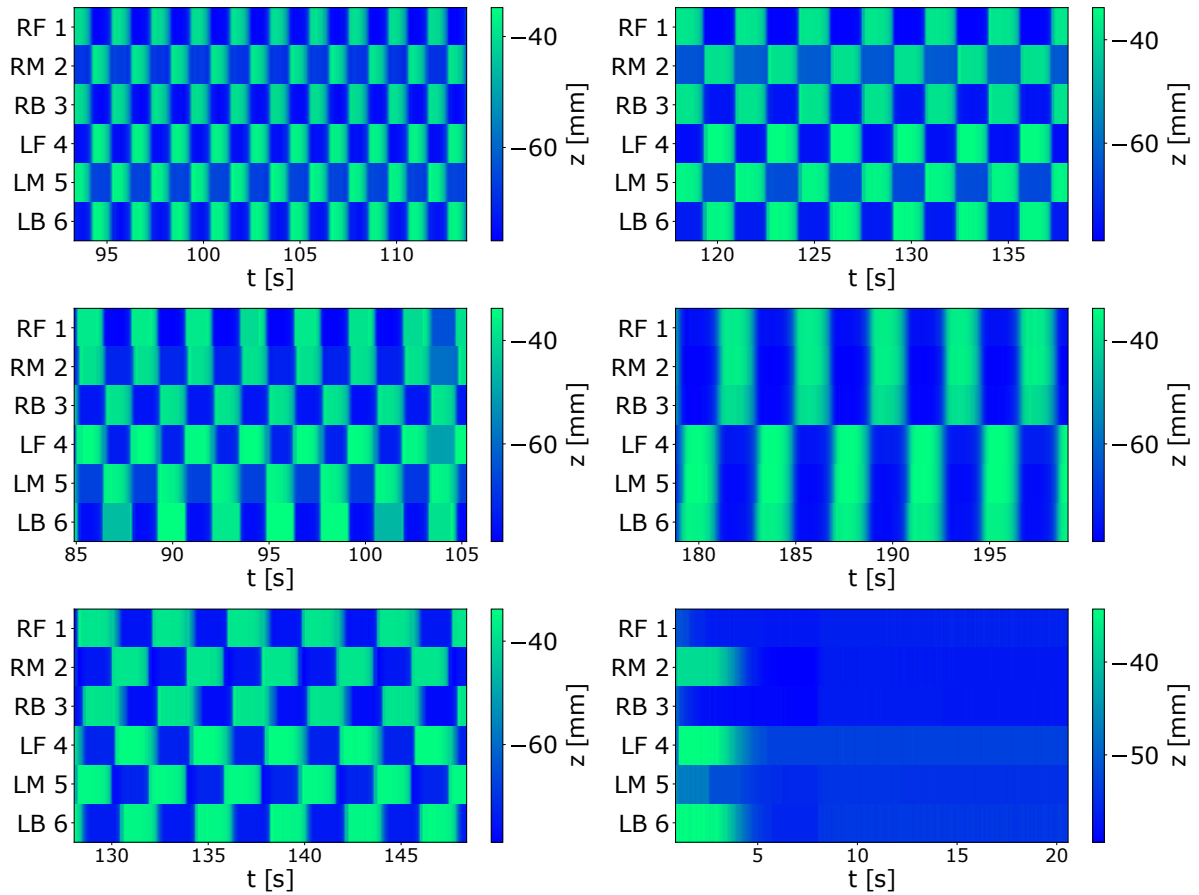
- szükséges könyvtárak importálása.
- objektumok létrehozása.
- globális változók és a számítások során használt állandók (mint például: $MIN q_i^{(c,f)}$),

$MAX q_i^{(c,f)}$, a szervok maximális nyomatéka, $\tau_x^{(c,f)}$, τ_b saját idők).

- *setup()* függvény: a robot indítását követő szükséges beállítások itt történnek (mint például: a soros kommunikáció sebességének beállítása, a robot kezdeti pozíciójának beállítása, a kommunikáció létrehozása a raspberry pi mini számítógéppel, a maximális szervo nyomaték értékek beállítása).
- a *loop()* függvény: itt történik a robot tényleges vezérlése, dinamikai rendszer integrálása, $y_i^{(c,f)}$ aktivációs érték kiszámítása (a (25). képlet segítségével), a $q_i^{(c,f)}$ aktuális csuklópozíciók lemerése, az $s_i^{(c,f)}$ értékek kiszámítása ((26). képlet), $*q_i^{(c,f)}$ célpozíciók kiszámítása ((27). képlet) és beállítása, a mért adatok soros kommunikáción való közvetítése és a ciklus megtételéhez szükséges Δt idő lemerése, amely a következő ciklusban lesz a Runge-Kutta módszer időlépése.
- a *loop()* függvény addig ismétlődik, amíg le nem állítjuk a robotot.

A hatlábú robottal való kísérleteink során w_y (milyen mértékben hasson a neuron aktivitása önmagára) és w_s (milyen mértékben kap a neuron visszacsatolást a célpozíció elérésének sikeréről) paraméter párost változtatva vizsgáltuk a kialakuló különböző lépésmintázatokat. A kezdeti értékeket és a többi paramétert lerögzítettük: ${}^0x_i^{(c)} = [2.0, -2.0, 2.0, -2.0, 2.0, -2.0]$, ${}^0x_i^{(f)} = [-1.0, 1.0, -1.0, 1.0, -1.0, 1.0]$, ${}^0b_i^{(c)} = [4.0, 0.0, 4.0, 0.0, 4.0, 0.0]$, ${}^0b_i^{(f)} = [1.0, 4.0, 1.0, 4.0, 1.0, 4.0]$, $\tau_x^{(c)} = 0.05$, $\tau_x^{(f)} = 0.025$, $\tau_b = 0.5$, $a = 2$, $w_k = -1$, ahol ${}^0x_i^{(c,f)}$ a membránpotenciálok kezdőértékei, ${}^0b_i^{(c,f)}$ az aktivációs küszöbérték kezdőértékei, $\tau_x^{(c,f)}$ a coxa és femur neuronok sajátideje, τ_b az aktivációs küszöbérték sajátideje, w_k a neuronok közötti kölcsönhatás erőssége (taszító kölcsönhatás, mivel negatív).

A (w_y, w_s) paraméter-párost változtatva rengeteg különféle mozgásmintázatot lehet felfedezni. Ezen lépésmintázatok közül néhány eredményez számunkra kedvező és keresett egyenletes haladó mozgást, de az esetek többségében a robot egyáltalán nem is halad. A 18. ábra szemléltet néhány gyakran tapasztalt mintázatot. A lépésmintázatokat hő térképen ábrázoltuk, a grafikonon szereplő értékek a lábak végpontjának Oz tengely irányú távolsága a robot testétől (2.4. fejezetben bemutatott (13). transzformációs egyenletet használva).



18. ábra. Lépésmintázatok hő térképen megjelenítve. **Első sor, bal oldal:** Háromlábú "tripod" lépésmintázat $(w_y, w_s) = (2.0, 0.0)$. **Első sor, jobb oldal:** Tripod lépésmintázat $(w_y, w_s) = (3.0, 0.0)$. **Második sor, bal oldal:** Blokkolt mintázat $(w_y, w_s) = (2.4, 0.2)$. **Második sor, jobb oldal:** Tánc mintázat $(w_y, w_s) = (0.0, 2.4)$. **Harmadik sor, bal oldal:** Példa más típusú lépésmintázatra $(w_y, w_s) = (3.2, 0.2)$. **Harmadik sor, jobb oldal:** Fixpont $(w_y, w_s) = (0.4, 0.0)$. **A lábak jelölése:** RF - jobb első, RM - jobb középső, RB - jobb hátsó, LF - bal első, LM - bal középső, LB - bal hátsó.

A **tripod** lépésmintázat, a hatlábú rovarok mozgására emlékeztet, a járás stabilitásának szempontjából ez a legmegfelelőbb választás. A robot mozgása során egyszerre három láb van a levegőben úgy, hogy a lábak végpontjait összekötve egy háromszöget kapunk (például: 1, 3, 5 számú lábak). A maradék három láb (például: 2, 4, 6 számú lábak) a földön maradnak, ezekkel a lábakkal tolja a robot előre a testét. Tripod mintázat több paraméter-páros esetén is kialakulhat, például a 18. ábra első sorában szemléltetett $(w_y, w_s) = (2.0, 0.0)$ és $(w_y, w_s) = (3.0, 0.0)$ értékekre. Azonban látható, hogy a nagyobb w_y paraméterrel rendelkező mozgás esetén a lépések periódusa megnő. A kísérletek azt mutatják, hogy ha egy adott lépésmintázat esetén a w_y paraméter értékét növeljük, azzal egyidőben növeljük a lépés periódusát is, ugyanakkor a robot lába erőteljesebben ütközik a földhöz.

A **blokkolt** állapot, ugyancsak egy gyakran előforduló lépésmintázat. Egy ilyen mintázatot szemléltet a 18. ábra második sorának bal oldali hő térképe. Az ilyen mintázatok esetében páronként két-két láb azonos fázisban kezd mozogni. A példa esetében, jól látjató a hő térképen, hogy az 1. és 2. láb illetve az 5. és 6. láb kerültek azonos fázisba, úgymond beblokkolnak. A maradék két láb a 3. és a 4. ezektől a pároktól eltolva mozog. Ez a lépésmintázat is több (w_y , w_s) paraméter-párosra előfordulhat és az is változhat, hogy mely lábak blokkolódnak be. Ha a lábak megfelelő módon blokkolnak be, a robot elég stabil haladó mozgásra képes.

A **tánc** lépésmintázat esetén az azonos oldalon lévő lábak fázisban vannak, míg az ellentétes oldalakon lévő lábak ellenfázisban vannak. Ez azt eredményezi, hogy amikor az egyik oldalon a lábak előre/hátra illetve fel/le mozognak, a másik oldalon a lábak ezekkel ellentétes elmozdulást produkálnak. Ezt szemlélteti a 18. ábra második sorának, jobb oldali hő térképe. A tánc elnevezés onnan származik, hogy ebben az esetben a robot teste nem halad előre, csak az Oz tengelye körül ide-ode fordul (oszillál) és az Ox tengelye körül billeg jobbra és balra.

A **más típusú** lépésmintázatokból van a legtöbb. Ezek azok a mintázatok amelyek még nem kaptak elnevezést. A 18. ábrán látható mintázat közel áll a tripod mintázathoz, de a kis eltérés miatt a robot elég esetlenül halad. A "más típusú" lépésmintázatok esetében a robot legtöbbször csak nagyon keveset halad előre, a lábak szabálytalan váltakozása miatt.

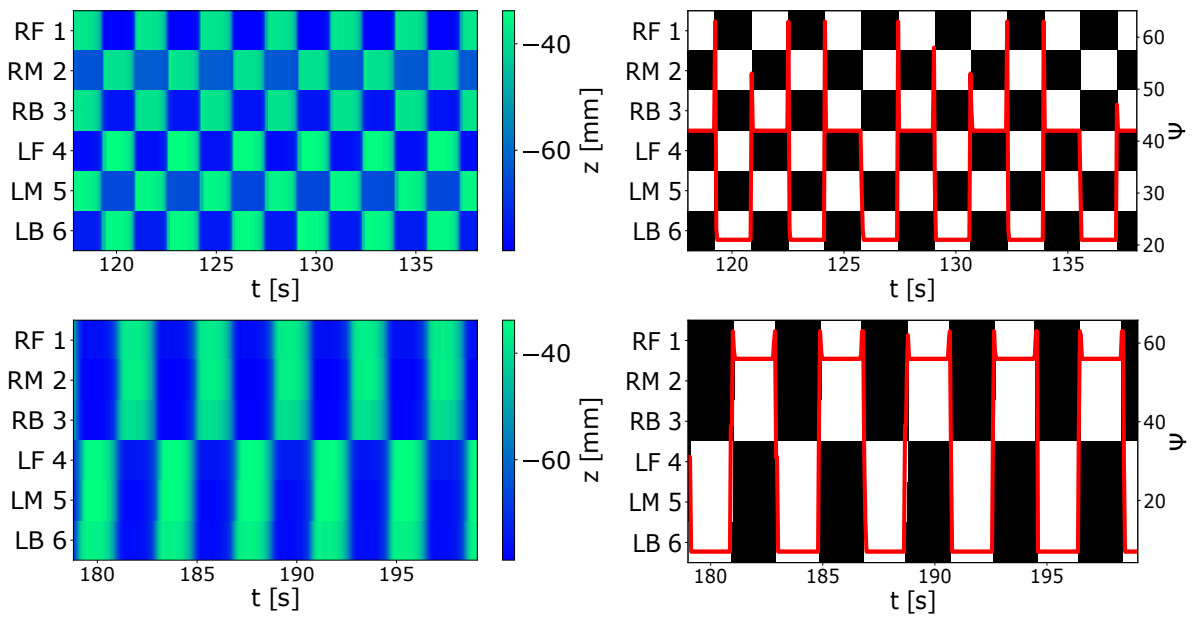
Fixpontok esetén a robot semmiféle mozgást nem mutat. Kezdetben megpróbál elindulni, de a lábak nagyon hamar megmerevednek és a robot beáll egy végleges pozícióba, általában minden láb a földön van és a robot teste vízszintes. Egy ilyen esetet szemléltet a 18. ábra harmadik sorának, jobb oldali hő térképe.

4.2.1. Lépésmintázatok keresése

A (w_y , w_s) paraméterpáros összes lehetséges eseteinek feltérképezése nehéz feladat, valamilyen módon különbséget kell tegyünk az egyes lépésmintázatok között. Erre a feladatra sajnos a szemünk nem a legmegfelelőbb, mivel nem tudunk egyszerre hat lábat figyelni és a szemünk hajlamos mintázatokot társítani olyan mozgásdinamikákhoz, amelyekben semmilyen szabályosság nem lelhető fel. A már említett hő térképek, már több információt mondanak el egy egy lépésmintázatról, de ezeknek az összehasonlítása megint csak nehézkes és ugyancsak a szemünkre kell hagyatkoznunk.

A mintázatok megkülönböztetéséhez, a hő térkép adatait binárisá tesszük. Ez azt jelenti, hogy azokat az értékeket, amelyek a lábak vépontjának és a robot teste között mért maximális

távolság felénél kisebb (azaz a láb fel van emelve) 0-ás értékre állítjuk, míg 1-re azokat az értékeket, melyek az említett küszöb értéknél nagyobbak (vagyis a láb a földön van). Így a hőterképet alkotó mátrix elemei egyesek és nullások lesznek. Ezt követően minden oszlop elemét megszorozzuk a $[2^1, 2^2, 2^3, 2^4, 2^5, 2^6]$ oszlopvektor megfelelő elemével. Tehát a mátrix első sora 2^1 -vel, a második 2^2 -vel szorzódik és így tovább. A szorzások elvégzése után az újonnan kapott mátrix oszlopainak elemeit összeadjuk. Így minden oszlop, Δt időlépés esetén kapunk egy számot, amelyet Ψ -vel jelölünk. A 19. ábra két "egyszerű" mintázat esetén szemlélteti a fent említetteket.



19. ábra. **Első sor. Bal:** Egy tripod mintázat hőterképe $(w_y, w_s) = (3.0, 0.0)$. **Jobb:** A tripod mintázat bináris hőterképe. A fekete szín jelzi, ha a láb a földön van és fehér ha a levegőben. A piros vonal mutatja a Ψ mérőszám változását az időben (jobb oldali skálán). A mérőszámok, amelyek jellemzik a lépés periódusát: (21, 42). **Második sor. Bal:** Egy tánc mintázat hőterképe $(w_y, w_s) = (0.0, 2.4)$. **Jobb:** A tánc mintázathoz tartozó bináris hőterkép. A mérőszámok, amelyek jellemzik a lépés periódusát: (7, 56).

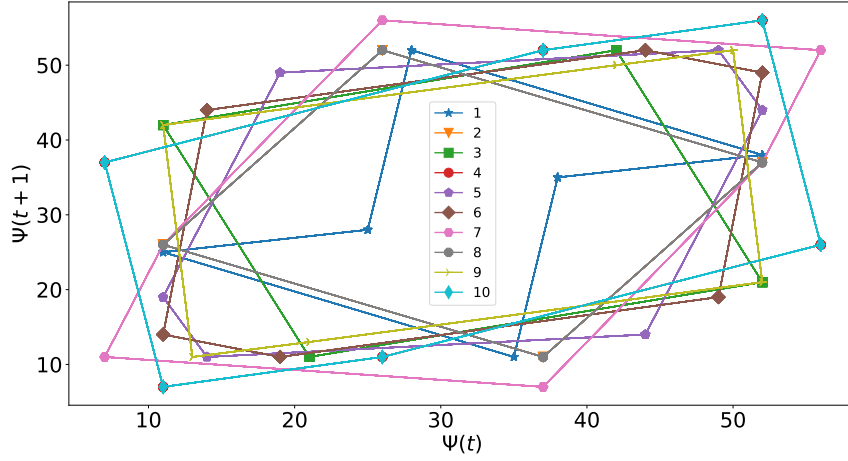
A grafikonokon jól látható, hogy a mérőszám értékek jól követik a lépések dinamikáját. Ebből kifolyólag, ha a lépésmintázat periódikus, akkor a mérőszámok is valamilyen periódusban kellene kövessék egymást és mi ezeket a periódikus ismétlődéseket keressük. Ugyanakkor az is kivehető a grafikonokon, hogy azokban a pillanatokban amikor a lábak állapotot váltanak (éppen emelkednek vagy ereszkednek) pillanatszerű kiugrások törnek meg a mérőszámok periodicitását. Ezért az ismétlődések vizsgálatakor, az erre használt algoritmusnak beállítottuk, hogy kissé legyen "elnéző" az ilyen pillanatszerű periodicitás-sértésekkel szemben.

Az algoritmus nagyon jól működik abban az esetben, ha a lábaktól érkező $s_i^{(c,f)}$

visszacsatolást kikapcsoljuk azáltal, hogy a w_s paramétert nullára állítjuk. Ilyen esetben ugyanis, a környezet nem befolyásolja a robot mozgását, a fázispont nem megy át egyik stabil attraktorról a másikra. Ha a kísérletet ilyen feltételek mellett végeztük, a rendszert véletlen kezdőértékekből indítva $(^{(0)}x_i^{(c,f)}, ^{(0)}b_i^{(c,f)}) \in [-5.0, 5.0]$, $(w_y, w_s) = (3.0, 0.0)$, az algoritmus 30 darab kísérletből 28 esetben talált periódikus változást a mérőszámokra. Amennyiben a w_s paraméter már nem volt nulla ez az érték drasztikusan lecsökkent. Ez a csökkenés annak tudható be, hogy a külső behatások miatt a lépésmintázatok, ha egy nagyon kicsit is (ami szabad szemmel nem látható a hőtésképen), de változnak és ez elegendő ahhoz, hogy az algoritmus ne találjon periódikus változást a mérőszámokban. A megtalálási arányt lehet növelni, ha még toleránsabbá teszük az algoritmust, de ekkor fenn áll az a lehetőség, hogy a nem kívánt, veletlenszerű fluktuációk a lábak állapotának változásakor is a periódikus számsor része lesznek.

Tovább rontja a módszer hatékonyságát az, ha a robotot hagyjuk a talajon járni. Ekkor ugyanis, a talajjal való kölcsönhatás során a lábak nem mindig ugyanúgy érik el a célpozíciót, beakadnak, nem tudja a robot eléggé előre vagy hátra tolni. Ilyen esetben a módszer akkor is sok esetben csődöt mond, ha a lábaktól érkező $s_i^{(c,f)}$ visszacsatolást kikapcsoltuk ($w_s = 0$), tehát a környezet direkt módon nem befolyásolja a vezérlő dinamikai rendszert.

Ahhoz, hogy a periódikus mérőszám ismétlődéseket össze tudjuk hasonlítani több lépésmintázat esetén, az ismétlődő számsorokat ábrázoltuk. Olyan módon, hogy a grafikon Ox tengelyére került a sorozat $\Psi(t)$ eleme, és az őt követő $\Psi(t+1)$ elem az Oy tengelyre. Ezt addig ismételtük amíg végig értünk a számsoron. A periódicitás miatt a grafikonon véges számú pont látható. Azok a mintázatok egyeznek meg, amelyek esetében ezek a zárt görbék fedik egymást, egymásra tevődnek. Ezt tíz darab periódikus mérőszám (Ψ) sorozat esetén a 20. ábra mutatja be. A 4.-10. és 2.-8. számsorozatokhoz tartozó görbék páronként fedik egymást, ez azt jelenti, hogy ezek a lépésmintázatok megegyeznek.

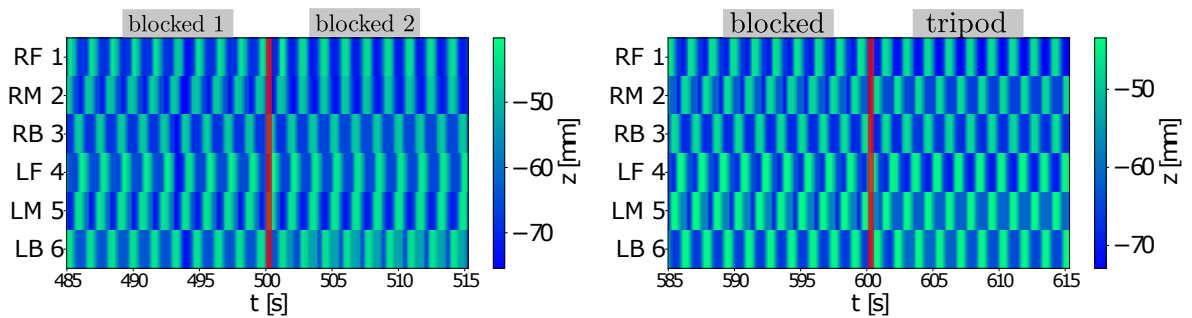


20. ábra. $\Psi(t + 1)$ mérőszámok ábrázolása a $\Psi(t)$ függvényében. A periodicitások miatt a görbéken lévő pontok mutatják a számsorokban előforduló mérőszám (Ψ) értékeket. A grafikonon 10 számsort ábráztunk, de ebből csak 8 látható, mivel 2-2 számsor grafikonja fedi egymást (4.-10. és 2.-8.). Ez azt jelenti, hogy azonos lépésmintázatokról van szó.

4.2.2. Impulzus-vezérlés

A Romi kétkerekű robothoz hasonlóan, a hatlábú robot esetén is továbbfejlesztettük az algoritmust, hogy a környezet hatását helyettesítve egy numerikus perturbáció segítségével vigyük át a rendszert egyik stabil attraktorról a másikra. A hatlábú robot esetében a "kick" azt jelentette, hogy az 1-es számú láb coxa és femur neuronjának a membrán potenciál értékének előjelét egy pillanatban megváltoztattuk:

$$x_1^{(c)} \rightarrow -x_1^{(c)}; \quad x_1^{(f)} \rightarrow -x_1^{(f)} \quad (30)$$



21. ábra. Lépésmintázat változások kick control esetén. A piros függőleges vonal jelzi a "kick" megtörténésének időpontját. **Bal:** A robot az 1-es számmal jelzett blokkol (*blocked 1*) mintázatban mozgott, a perturbáció következtében lépésmintázat változás történt és a robot átváltott a *blocked 2* lépésmintázatba. **Jobb:** Kick hatására lépésminta változás egy blokkolt állapotból a tripod mintázatba. $(w_y, w_s) = (2.0, 0.0)$.

A kétkerekű robot egyszerűbb dinamikai rendszeréhez hasonlóan, a hexapod bonyolultabb neuron hálózata esetén is ez a perturbáció elegendő volt ahhoz, hogy a robotunk egyik stabil attraktoról átugorjon egy másikra és ez lépésmintázat változást eredményezzen. Két ilyen lépésmintázat változást szemléltet a 21. ábra.

Következtetések és továbbfejlesztési lehetőségek

Kísérletezéseink során bebizonyosodott, hogy egyszerű multistabil dinamikai rendszereket használva sokféle, komplex lépésmintázatokat, mozgásformákat lehet előállítani, bármiféle szokásos robotikai módszer alkalmazása nélkül. A rendszerek önszerveződő mivoltukból adódóan képesek reagálni a környezet behatásaira kizárólag a csuklók és a kerekek pozíciójának mérésén keresztül. Ilyenkor látszólag a robotok megoldják az előttük álló problémákat: a kerekes robot vissza fordul a faltól, a hexapod a falnak ütközés után a tánc mintázatba megy át, így tovább nem erőltetve magát a fal eltaszításával.

A klasszikus robotikában a környezettel való interakcióhoz szenzorokat használnak. Dolgozatunkban arra is rámutattunk, hogy a dinamikai rendszerekkel történő vezérlés nem zárja ki a szenzorok használatát. Javasoltunk egy olyan módosítást az egyenletekhez, hogy a szenzoroktól érkező adat elérje a kívánt hatást a robot mozgása során, például a távolságmérő szenzorok használatával a kerekes robot esetében. Külső vezérlő jel beavatkozása is lehetővé tehető az impulzus-vezérlés (*kick control*) használatával. Ilyen esetekben, egy általunk jól időzített numerikus perturbáció lehetőséget ad arra, hogy a különböző mozgásmintázatokhoz tartozó attraktorok között változtathassuk a robot rendszerének fázispontját.

Dolgozatunkban arra is adtunk egy egyszerű, ámbár kezdetleges megoldást, hogy hogyan lehet különbségeket tenni a lépésmintázatok között. Hiszen a robot pontos irányításához jól kell ismernünk a kialakuló dinamikákat.

A jövőben tovább szeretnénk fejleszteni a lépésmintázatok keresésének algoritmusát, hogy az (w_y, w_s) paraméterek terét valamivel nagyobb mértékben feltérképezzük és beazonosíthassuk a különböző mintázatokhoz tartozó paraméterbeállításokat. Továbbá meg szeretnénk vizsgálni azt, hogy melyek a megfelelő időpillanatok a "kick" alkalmazására, hogy a számunkra megfelelő mintázatot kapjuk.

Hivatkozások

- [1] Herczeg Ágnes és Harkó Csanád. „Hatlábú robotok irányítása oszcilláló neuronokkal”. *XXV. Reál- és Humántudományi Erdélyi Tudományos Diákköri Konferencia*. 2022.
- [2] Sándor Bulcsú. „The world of dynamical systems: multistability, neural dynamics and robotic locomotion”. *PhD thesis, Goethe University Frankfurt*. 2017.
- [3] Peter Croke. *Robotics, Vision and Control, Fundamental Algorithms in MATLAB*. Springer International Publishing AG, 2017.
- [4] Harkó Csanád. „Hatlábú robot kísérleti tanulmányozása: önszerveződő dinamika csatolt neuronok által”. *Államvizsga dolgozat*. 2022.
- [5] Pololu Robotics & Electronics. *Pololu Romi 32U4 Control Board User's Guide*. URL: <https://www.pololu.com/docs/0J69/all> (elérés dátuma: 2023. ápr. 12.)
- [6] Pololu Robotics & Electronics. *Romi Encoder Pair Kit*. URL: <https://www.pololu.com/product/3542> (elérés dátuma: 2023. ápr. 13.)
- [7] McCulloch és Pitts. „Logical Calculus Of The Ideas Immanent In Nervous Activity”. *Bulletin of Mathematical Biophysics*. 1943, 115–133. oldal.
- [8] Bulcsú Sándor és mások. „Kick control: using the attracting states arising within the sensorimotor loop of self-organized robots as motor primitives”. *Frontiers in neurorobotics*. 2018.
- [9] STMicroelectronics. *VL53L0X - World smallest Time-of-Flight ranging and gesture detection sensor*. URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/948120/STMICROELECTRONICS/VL53L0X.html> (elérés dátuma: 2023. ápr. 12.)
- [10] Tamás Tél és Márton Gruiz. *Kaotikus dinamika*. Nemzeti tankönyvkiadó, Budapest, 2002.
- [11] TrossenRobotics. *Phantom X Getting started guide*. URL: <https://web.archive.org/web/20200919115810/https://learn.trossenrobotics.com/10-interbotix/crawlers/phantomx-hexapod/172-phantomx-hexapod-metal-assembly-guide.html> (elérés dátuma: 2023. ápr. 22.)
- [12] Hendrik Wernecke, Bulcsú Sándor és Claudius Gros. „Attractor metadynamics in terms of target points in slow-fast systems: adiabatic versus symmetry protected flow in a recurrent neural network”. *Journal of physics Communications*. 2018.
- [13] Hendrik Wernecke, Bulcsú Sándor és Claudius Gros. „How to test for partially predictable chaos”. *Scientific Reports*. 2017.

5. Melléklet

5.1. Romi robot technikai leírása

A robot működtetéséért a Romi 32U4 vezérlőpanel a felelős, mely egy *Atmel ATmega32U4* AVR 8-bit-es, 16MHz órajelű mikrovezérlő köré van építve és egy Arduino-kompatibilis bootloaderrel van ellátva. A mikrovezérlő 32 kilóbájt írható (programozható) és 2.5 kilóbájt statikus (SRAM) memóriával van ellátva. A vezérlőpanel továbbá rendelkezik:

- négy darab +5V-os csatlakozóval és hét GND csatlakozóval
- harmincnyolc ki-/bemeneti digitális csatlakozóval (I/O pins)
- tizenhét analóg csatlakozóval (ADC pins)
- egy I^2C csatornával
- egy soros (UART) csatornával
- egy *SPI* csatornával

A vezérlőpanelen továbbá helyet kapott [5]:

- két mini, műanyag hajtóműves motor (mini plastic gearmotor)
- két H-híd, melyek a két motor vezérléséhez szükségesek
- egy *Romi Encoder Pair Kit*, mely a motorokhoz van kapcsolva és ezáltal a kerék elfordulási szöge mérhető (enkoder értékben)
- egy nagy teljesítményű 5V-os feszültségszabályozó (step-down), amely képes akár 2A állandó áram szolgáltatására, valamint egy sokoldalú tápkapcsoló és elosztó áramkör
- egy három tengelyű giroszkóp és gyorsulásmérő, hogy a robot meghatározza orientációját és a környezetből ható erőket (típus: LSM6DS33)
- három beépített nyomógomb, melyeken keresztül a felhasználó léphet interakcióba a robottal
- beépített LED-ek és csipogó (buzzer), ezek lehetőséget adnak a robotnak hogy visszajelzéseket adjon.

A két DRV8838 motorvezérlő az ATmega32U4 chip 9, 15-ös (jobb oldali motor esetén) és 10, 16-os (bal oldali motor esetén) digitális pinjeihez vannak csatlakoztatva. Ezeken a pineken keresztül vezérelhetők a kerekek. Például, jobb oldali kerék esetén, ha a 15-ös pin *LOW* (alacsony, 0V) értéken van a kerék előre forog, míg *HIGH* (magas, +5V) érték esetén hátrafelé. A 9 pinen keresztül *PWM* (Pulse Width Modulation) jelet használva tudjuk szabályozni a motor forgási sebességét. A bal oldali kerék esetében is ugyanígy működik a vezérlés.

5.2. A hexapod robot programozása

A robot vezérlőpanelje egy *Arduino* kompatibilis *ArbotiX-M* típusú *robokontroller* (*robocontroller*), amelyen egy *ATMEGA644p* típusú 8-bites, 16 MHz órajelű mikrovezérlő kapott helyet. Az *ATMEGA644p* mikrovezérlő 64 kilobájt írható/programozható és 4 kilobájt statikus (SRAM) memóriával rendelkezik. A vezérlőpanelen helyet kapott 28 ki-/bemeneti csatlakozó (I/O pins), amelyek közül 8 darab analóg bemenetként is használható. Továbbá találunk még három darab *TTL DYNAMIXEL* csatlakozót, ez teszi a vezérlőpanelt kompatibilissé a *TTL AX/MX DYNAMIXEL* típusú szervomotorokkal.

Az a tény, hogy a vezérlőpanel kompatibilis az *Arduino IDE* kódfejlesztő környezettel nagyban megkönnyíti a dolgunkat. A fejlesztőcsapat rengetek nyílt forráskódú könyvtárat és példákat hozott létre, annak érdekében, hogy a robot könnyen irányítható legyen. Miután a könyvtárakat a "sep-by-sep" útmutatót követve telepítettük az *Ardiono IDE*-ben, a mintaprogramok könnyen elérhetők a "Példák" (Examples) menüpontja alatt. A robotra szánt algoritmusokat az *Arduino* sajátos, *C++*-on alapuló, nyelvezetén kell megírni. Egy ilyen kód megírásának fontosabb lépései:

- beimportáljuk a szükséges könyvtárakat, köztük a szervomotorok működtetéséhez szükségeset is
- létrehozuk az objektumokat és a globális változókat
- a *setup()* függvényben elvégezzük a kezdeti beállítási folyamatokat (ez a függvény csak egyszer fut le a robot indítását követően)
- a *loop()* függvényben végezzük a robot irányítását, az okos szervomotorok által mért adatok lekérését és továbbítását soros (UART) kommunikációt használva (ez a függvény egy végtelen ciklus, addig fut amíg le nem állítjuk a robotot)

- a kód végén létrehozuk a vezérléshez, számításokhoz és a kommunikációhoz szükséges függvényeket

5.3. AX-12A szervomotorok bemutatása

Az előzőekben úgy említettük az AX-12A típusú szervomotorokat, mint okos szervomotorok. Ebben a fejezetben megismerkedünk ezekkel a szervomotorokkal, bemutatjuk miben rejlik az okosságuk és látni fogjuk miben térnek el a hagyományos, egyszerű léptetőmotoroktól.

A hagyományos léptetőmotorokkal ellentétben az AX-12A típusú motorok rendelkeznek belső szenzorokkal és egy beépített mikrokontrolerrel, így van saját memóriájuk. A szenzorok által információt kaphatunk a motor: hőmérsékletéről, aktuális pozíciójáról, a tápfeszültségéről, a terheléséről. A beépített mikrovezérlő kétféle memóriával rendelkezik: *EEPROM* és *RAM*. A két memóriatípusban összesen 50 byte-nyi adat fér el, melyek 1 vagy 2 byte méretű regiszterekre vannak osztva. Ezek a regiszterek kétfélék lehetnek:

- **R** típusú - ezek olyan regiszterek, melyek módosítása nem megengedett, értéküket csak olvasni lehet
- **RW** típusú - ezen regisztereknek a módosítása és az értéküknek a kiolvasása is megengedett

Például a szervomotorokhoz rendelhető ID egy 1 byte-os **RW** típusú regiszterben van tárolva, ami azt jelenti, hogy 0-tól 255-ig bármilyen értéket beállíthatunk neki. Továbbá ez azt is jelenti, hogy egyidőben, egy robotkontrollerrel akár 256 darab különböző ID-vel rendelkező szervomotor is vezérelhető. A szervomotor jelenlegi pozíciója (present position), amelyet a beépített szenzor mér, egy 2 byte-s **R** típusú regiszterben van tárolva, vagyis értéke a (0, 1023) intervallumból származhat és a regiszter értéke a felhasználó által nem változtatható.

Az **EEPROM** memóriában kapnak helyet, a szervomotor hosszútávú beállításai. Ezek a beállítások akkor sem vesznek el, ha a motort áramtalanítjuk, a bekapcsolás után a regiszterek ugyanazon értékeket fogják tartalmazni, mint a kikapcsolás előtt. Ilyen regiszterekben vannak eltárolva például a következő beállítások: modell szám, szoftver verzió, ID, a motor kommunikációs sebessége, a motor válaszüzeje, a mozgási tartomány korlátai, hőmérséklet határ, minimális és maximális tápfeszültség határ, maximálisan kifejtendő nyomaték.

A **RAM** memóriában kapnak helyet az ideiglenes beállítások, ezek a regiszterek a motor újraindítását követően visszaállnak a fejlesztők által az EEPROM memóriában mentett alapértelmezett (default) beállításokra. Ilyen regiszterekben van tárolva például: a nyomaték engedélyezése, a célpozíció, a mozgási sebesség, a nyomaték limit, a jelenlegi pozíció és sebesség, a jelenlegi tápfeszültség, az aktuális hőmérséklet.

Az aktuátorok (szervomotorok) és a robotkontroller között a *TTL Level Multi Drop Bus* segítségével valósul meg, ezt azt jelenti, hogy a aktuátorok a robotkontroller ugyanazon kommunikációs csatornájához (bus) vannak csatlakoztatva és az ID-k használatával teszünk különbséget az egyes aktuátorok között. A használt protokoll pedig egy *Half Duplex Asynchronous Serial Communication* kommunikációs protokoll. Ez azt jelenti, hogy a robotkontroller és az aktuátor között lehetséges a kétirányú kommunikáció, de nem azonos időpillanatban (half duplex). Továbbá a kommunikáció aszinkron módon valósul meg (asynchronous serial communication), vagyis a kommunikáció két résztvevője nem minden időpillanatban van szinkronban egy közös órajelet figyelve. Ez az következményt okozza, hogy a küldött adatcsomagoknak mindig azonos "kinézetük" kell legyen, hogy a fogadó fél értelmezni tudja. Esetünkben az adatcsomagok felépítése: 8 bit (ez tartalmazza a tényleges információt), 1 stop bit (ez jelzi az adatcsomag végét) és nincs paritás.

A Dynamixel AX-12A típusú szervomotorok rendelkeznek saját, beépített biztonsági rendszerrel, amely megakadályozza a szervomotor mechanikájának vagy elektronikájának a károsodását. Amennyiben a servo vezérlőelektronikája valamilyen furcsaságot érzékel a motor működésében (például túlterhelődik, vagy túlmelegedik) azonnal leállítja a szervomotort és egy piros LED kezd el villogni a motor külsején, ezzel jelezve a felhasználónak, hogy valamilyen probléma adódott. Ilyekor az aktuátort áramtalanítani kell és kis idő múlva újra használható. A motor elektronikája a következő problémákat képes érzékelni: parancs hiba, bemeneti feszültség hiba, túlterhelés, túlmelegedés, adatcsomag hiba, megengedett tartományon kívüli hiba, pozíció hiba.

Az AX-12A aktuátorkor mechanikai felépítésüket tekintve minőségi, erős műanyagból vannak elkészítve, úgy a külső borítás, mint a belső forgató mechanika. A szervomotorok további technikai jellemzői:

- baud rate (kommunikációs sebesség): 7843 [bps] $\sim$ 1 [Mbps]
- tömeg: 53,4 [g]
- méret: 32x50x40 [mm]
- szögfelbontás: 0,29 [°] (ilyen pontossággal képes beállítani a kívánt pozíciót)
- megengedett elfordulási intervallum: (0° - 300°)
- maximális nyomaték: 1.5 Nm (12 V, 1.5 A táplálás esetén)
- maximális sebesség: 59 fordulat/perc (12 V táplálás esetén)
- bemeneti feszültség: 9 - 12 V (ajánlott: 11.1 V)
- csatlakozó típusa: TTL, három bemenettel rendelkezik, kettő a táplálásnak (VDD és GND) és egy az adatforgalomnak (DATA)

Az 6. ábrán az általunk használt szervomotor látható, annak külső műanyag borítása, az elfordulás szögének pozitív iránya és a szervomotor által lefedhető szögintervallum.

6. Saját eredmények

Feladatok, fejlesztések és eredmények a Romi robot esetében:

- Megvalósítottam a robot programozását, hogy a bemutatott és előzőleg már ismert, csatolt neuronokból álló, dinamikai rendszer segítségével vezérelni lehessen (2.2. és 4.1. fejezetek).
- Kiegészítettem a robotot felépítését négy darab VL53L0X távolságmérő szenzorral (2.1. fejezet).
- A távolságmérő szenzorokból érkező információt sikerült felhasználni a robotot vezérlő neurális hálózat befolyásolására és ezzel impulzus-vezérlést (*kick control*) végrehajtani a roboton. Ezzel is adva egy módszert arra, hogy hogyan lehet robotok esetében a neurális vezérlést ötvözni a külső szenzorokkal (3.2.1. és 4.1.1. fejezetek).

Feladatok, fejlesztések és eredmények a Hexapod robot esetében:

- Kiegészítettük a robot felépítését egy Raspberry Pi 2W miniszámítógéppel, melynek segítségével a robotot vezeték nélkül tudjuk irányítani és lokálisan menteni a kísérlet során mért adatokat. Ez a fejlesztés nagyban megkönnyítette a kutatómunkát és lehetővé tette, hogy a kísérletezések során ne kelljen interakcióba lépni a robottal, így nem befolyásoljuk a mért adatokat (2.3. és 2.5. fejezetek).
- Elkészítettem azokat a programokat, amelyek segítségével a Raspberry, a felhasználó és a robot kommunikálni tudnak egymással a mérések során (2.5. fejezet).
- Javasoltunk egy egyszerű módszert amely segítségével különbségeket lehet tenni a robot mozgása során kialakuló lépésmintázatok között (4.2.1. fejezet).
- Az impulzus-vezérlést (*kick control*) a hatlábú robot esetében is megvizsgáltuk. Sikeresen rámutattunk arra, hogy még az ilyen bonyolult felépítésű neurális rendszerek esetében is, könnyen elérhető az, hogy a rendszer dinamikája megváltozzon egy megfelelő numerikus perturbáció hatására. Ez az eredmény megnyitotta azt a lehetőséget is, hogy a későbbiekben a hexapod robot esetében is külső szenzorok segítségével befolyásoljuk a neurális hálózatot (4.2.2. fejezet).

Én voltam a felelős a robotokkal végzett kísérletekért és az adatok feldolgozásáért is.