# Golyó a keréken demonstrációs eszköz

**Dobány Imre, Dr. Kopják József**

Dobány Imre, Kandó Kálmán Villamosmérnöki Kar, Óbudai Egyetem, 1034 Budapest, Bécsi út 94-96., imre@dobany.hu

*Absztrakt:*
*A „golyó a keréken" problémában egy golyót kell egyensúlyozni a biciklikerék tetején. Ez egy általánosan ismert iskolapélda az irányítástechnika szakterületén [1], hasonlóan a jól ismert „fordított inga" vagy a „reaction wheel" problémákhoz.*

*A SISO szabályozás bemente a golyó pozíció, kimenete a motor feszültség. A modell alkotás Newton és Lagrange módszerrel készült, amelyek után a motor paraméter identifikáció és a szabályozó tervezés követte PID, állapot visszacsatolás, állapot becslővel és LQ [2] szabályozókkal Scilab, Xcos és Octave-CLI fejlesztőkörnyezetben. A motorhoz HALL szenzoros hajtás került megvalósításra Arduino-CLI-Nano fejlesztő környezetben. Mivel a legegyszerűbb, legköltséghatékonyabb oktatási összeállításra törekedtem, ezért Linux-os operációs rendszeren, akár PC-n vagy például Raspberry PI számítógép platformon futtatható a szabályozó Octave szkript. A számítógép UART interfészen kommunikál az Arduino Nano kártyával, ahol analóg pozíció érzékelés történik Tsample = 20-100msec időzítéssel, amit a Linux delegált CPU-core-ja elegendő pontossággal kiszolgál. Az Arduino végzi a BLDC motor hajtást a kapott motor kapocsfeszültség alapján.*

*Az összeállítással megmutatható a modellalkotás, motor paraméter identifikáció, szabályozó tervezés menete, továbbá vizsgálható a szabályozók érzékenysége és robusztussága.*

*Kulcsszavak: Newton modell, Lagrange modell, instabil szabályozás, állapotvisszacsatolás, LQ szabályozó*

# 1. Bevezetés

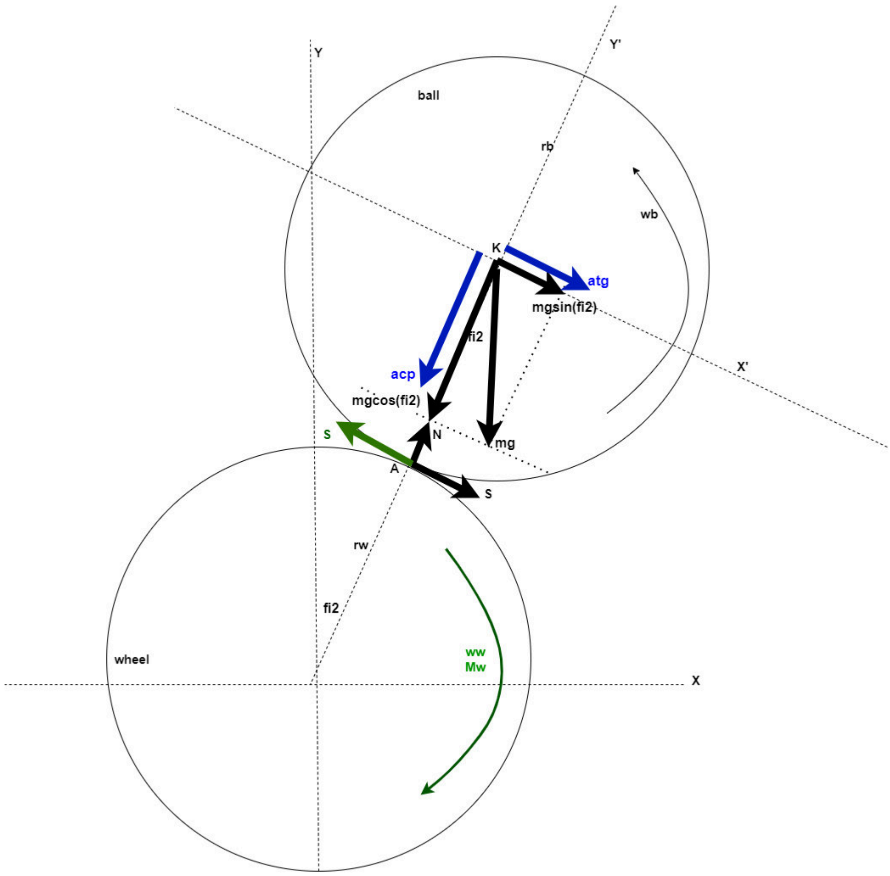
A cikk célja, hogy megmutasson egy automatizálási feladat megoldást a kezdetektől a fizikai megvalósításig modern, de a végletekig leegyszerűsített anyagi erőforrást igénylő eszközökkel, ily módon alkalmassá téve a mindenki számára hozzáférhető oktatási demonstrációs eszközként.
Először a kísérleti elrendezés és a rendszer fizikai tulajdonságai kerülnek röviden bemutatásra, amelyek a későbbi szabályozó tervezéshez szükségesek. Ezt követi a modellalkotás Newton és Lagrange módszerrel a fizikai tanulmányok iskolapéldájaként.
A rendszer modell linearizálása után a szabályozó tervezés bemutatása követi, amelyben a manapság használatos programokkal elvégezhető matematikai műveletsorok kerülnek bemutatásra, míg a mélyebb magyarázatot a megjelölt irodalomjegyzékben megtalálhatjuk. A következő fejezetekben a szabályozó megvalósítás valamint golyó pozíció érzékelés és beavatkozáshoz szükséges motor hajtás kerül megemlítésre.
A cikk végén a szabályozó tervezéshez szükséges fizikai paraméterek meghatározásához (paraméter identifikáció) szükséges lépések kerülnek bemutatásra.
Az összeállítás továbbfejlesztési lehetőségeket rejt magában, mint például felhasználói böngészőn keresztüli html interfész kialakítását, ahol szabályozó típust (PID, állapotvisszacsatolás, LQ), mintavételi időt, jittert, alapjelet, stb. lenne lehetőség állítani, így vizsgáhatóvá válna a különböző szabályozók érzékenység és robosztusságának vizsgálatára. Az eszköz valamely egyszerűsített, lekicsinyített másolata alkalmas lehet csináld magad (DIY) egységcsomag kialakítására, amely motiváló lehet a tanulók számára.
Az összeállításról videó elérhető az alábbi linken:
https://drive.google.com/file/d/1aYekoKIV0XY-9lRHnn9m6GJVegpmdNTa/view?usp=drive_link

*1-1. ábra Golyó a keréken összeállítás*

# 2. Modell alkotás

Matematikai lineáris modellre van szükségünk, hogy a szabályozót meg tudjuk valósítani. Fizikában a mechanika és az elektrodinamika területén a Newton és Lagrange modellezés használatos.

## 2.1 Rendszervázlat, paraméterek és terminológia




*2-1. ábra Golyó a keréken rendszermodell [1]*

| Jellemző | Érték | Megjegyzés |
|---|---|---|
| $g$ | 9,81 m/s^2 | Gravitációs gyorsulás |
| $U_a$, $U_k$ | 0-24V | Kapocsfeszültség |
| $R_a$ | 300mOhm | Armatúra ellenállás |
| La | 30mH | Armatúra induktivitás (1kHz) |
| $K_u$, $k_1$ | 2,093Vs/rad | Villamos konstans |
| $K_m$, $k_2$ | 2,518Nm/A | Nyomaték konstans |
| $I_w$ | 0,30618kgm^2 | Tehetetlenségi nyomaték - kerék |
| $T_v$ | 33,33 * 10^-3 | Villamos időállandó |
| $T_m$ | 263 * 10^-3 | Mechanikai időállandó |
| $r_w$ | 0,27m | Kerék sugár |
| $r_{we}$ | 0,03m | Kerék perem távolság |
| $r_v$ | 2,3*10^-3 | Csapágy súrlódás, nyomaték együttható |
| $\varphi_w$ | - | Kerék szögelfordulás |
| $w_w$ | 0-120rpm | Kerék szögsebesség-Hall mérés |
| $M_m$ | - | Kerék motor nyomaték |
| $M_L$ | - | Terhelő nyomaték |
| $I_b$ | 0.0000731kgm^2 | Golyó tehetetlenség |
| $r_b$ | 0,0254m | Golyó sugár a peremen |
| $r_{bf}$ | 0,03m | Golyó sugár |
| $m_b$ | 0,203kg | Golyó tömeg |
| $\varphi_2$ | -0,45rad -- 0,45 rad | Golyó pozíció szögelfordulás |
| $w_2$ | - | Golyó pozíció szögsebesség |

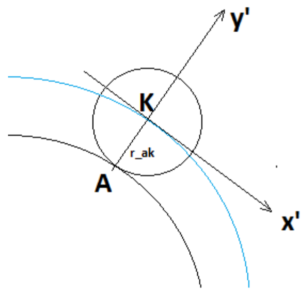
*2-1. táblázat Terminológia és paraméterek*

## 2.2 Mozgásegyenletek felírása (Newton modell)

Az egyenletek rendezéséhez a Maxima [11] programot használtam, melyek a 6. fejezetben leírtak szerint elérhetőek.

### 2.2.1 Kinetikai feltételek

A modellhez elengedhetetlen, hogy a golyó a keréken ne csússzon meg, azaz a tapadás mindvégig fennálljon, ami a tiszta gördülés esetét jelenti. Ekkor a találkozási pont nyugalomban van, nincs relatív elmozdulás, tehát úgy modellezhető, mintha a golyó és a kerék össze lenne ragasztva. Más kérdés, hogy a golyó közben forog a kerületén ható erő miatt, ami majd forgató nyomatékot okoz. Ezen ok miatt az (1) egyenlet használható.




*2-2. ábra Kinetikai feltétel*

A golyó tömegközéppontjának transzlációs és rotációs mozgása közötti kapcsolatot az alábbi egyenlettel fejezhetjük ki. [10]

$$\underline{V}_K = \underline{V}_A + \underline{w}_b \times \underline{r}_{AK} \qquad (1)$$

$$\left(((r_w + r_b)w_2);\ 0;\ 0\right) = \left((r_w w_w);\ 0;\ 0\right) + \left(0;\ 0;\ w_b\right) X \left(0;\ r_b;\ 0\right)$$

$$\left(r_w + r_b\right)w_2 = r_w w_w - w_b r_b$$

A tiszta gördülés, tapadás addig áll fenn, amíg a tapadási erő, 2.1 ábrán feketével jelölt S erő nagyobb vagy egyenlő, mint a súrlódási erő.

$\mu N \leq S$

A μ értékét az anyagok közötti tulajdonságok határozzák meg. A fenti azonosságot tapasztalati úton fogom ellenőrizni. A differenciálegyenlet felírásához, majd, mint látni fogjuk közvetlenül erre nincs szükség.

### 2.2.2 Mozgásegyenletek-golyóra

A golyó mozgásegyenletének felírásakor kísérő-triéderes koordináta rendszert használok a 2-1. ábra szerint (x'; y').
Egy tömegközéppontra 6 mozgásegyenletet tudunk felírni:

- 3-at, (x; y; z) irányban a transzlációs mozgásra és
- 3-at, ($x_2$; $y_2$; $z_2$) irányban a rotációs mozgásra.

Jelen esetben, mivel sík problémáról van szó

- 2 db egyenlet a transzlációs mozgásról szól (x; y)
- 1 db egyenlet a rotációs, forgómozgásról szól ($z_2$)

transzlációs egyenletek

$$m\,g\,cos\,\varphi_2 - N = m\,a_{cp}$$

$$m\,g\,sin\,\varphi_2 + S = m\,a_{tg}$$

$$a_{cp} = \left(\dot{\varphi}_2\right)^2\left(r_w + r_b\right)$$

$$a_{tg} = \ddot{\varphi}_2\left(r_w + r_b\right)$$

rotációs egyenlet

$$S\,r_b = I_b\ddot{\varphi}_b$$

### 2.2.3 Mozgásegyenletek - kerékre

A kerék koordináta rendszerét az 2-1. ábra szerint választom. Mivel transzlációs mozgást nem végez a kerék, ezért egyenletei sincsenek. Mivel a kerék forog, ezért rotációs egyenlete van.

$$M - Sr_w = I_w\,\ddot{\varphi}_w$$

Az 'M' nyomatékba a motor csapágy súrlódást ($M_L$) negatív előjellel a későbbiekben bele kell számolni.

$$M = M_w - M_L$$

$M_w = \frac{k_2\left(U_A - k_1 w_w\right)}{R_A}$ és $M_L = r_r w_w$

### 2.2.4 Összevont mozgásegyenletek

$$\left(I_w + I_b\frac{r_w^2}{r_b^2}\right)\dot{w}_w - I_b\frac{r_w}{r_b^2}\left(r_w + r_b\right)\dot{w}_2 = \frac{k_2\left(U_A - k_1 w_w\right)}{R_A} - r_r w_w \qquad (2)$$

$$- I_b \frac{1}{r_b^2}\left(r_w \dot{w}_w - \left(r_w + r_b\right)\dot{w}_2\right) + m_b\left(r_w + r_b\right)\dot{w}_2 = m_b g\left(r_w + r_b\right) \sin \varphi_2 \quad (3)$$

## 2.3 Modell Lagrange szerint

A Lagrange modell szerint a szakasz Lagrange függvénye [1][2]

$$L = T - V$$

ahol 'T' a mozgási energia és 'V' a helyzeti energia, 'L' pedig a Lagrange-függvény.

Esetünkben tehát:

$T_w = \frac{1}{2} I_w w_w^2$ – kerék mozgási energiája

$T_b = \frac{1}{2} I_b w_b^2 + \frac{1}{2} m_b v_b^2$ – Golyó mozgási energiája

$V_b = m_b g\left(r_w + r_b\right) \cos \varphi_2$ – Golyó helyzeti energiája

$$L = T_w + T_b - V_b$$

$$L = \frac{1}{2} I_w w_w^2 + \frac{1}{2} I_b \frac{1}{r_b^2}\left(r_w w_w - \left(r_w + r_b\right) w_2\right)^2 + \frac{1}{2} m_b\left(r_w + r_b\right)^2 w_2^{\ 2} - m_b g\left(r_w + r_b\right) \cos \varphi_2 \quad (4)$$

Így a Lagrange változók:

$$L\left(\varphi_w; \varphi_2; w_w; w_2\right)$$

A Lagrange-egyenlet nem konzervatív rendszer esetén:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i^* \quad (5)$$

ahol $Q_i^*$ a nem konzervatív erőkből álló általános erők eredője.

Tehát a feladat (4) egyenlet (5) egyeletbe helyettesítése és megoldása. Az (5) egy egyenletrendszer, amelyben Qi*($\varphi_w$; $w_w$; $\varphi_2$; $w_2$). A motor egyenlet felhasználásával:

$Q_w^* = M_w - M_L = \frac{K_M(U_a - K_U W_w)}{R_a} - r_r w_w,$

$Q_2^* = 0,$

mivel a pozíciónak nincs $Q^*$ általános ereje. A motor modelljében ($M_w$) az armatúra áram dinamikája elhanyagolásra került.

### 2.3.1 Lagrange parciális derivált $\varphi_w$; $w_w$ szerint.

$$\left(I_w + I_b \frac{r_w^2}{r_b^2}\right)\dot{w}_w - I_b \frac{r_w}{r_b^2}\left(r_w + r_b\right)\dot{w}_2 = \frac{k_2(U_A - k_1 w_w)}{R_A} - r_r w_w \qquad (6)$$

### 2.3.2 Lagrange parciális derivált $\varphi_2$; $w_2$ szerint.

$$-I_b \frac{r_w}{r_b^2}\dot{w}_w + \left(I_b \frac{r_w + r_b}{r_b^2} + m_b\left(r_w + r_b\right)\right)\dot{w}_2 = m_b g\left(r_w + r_b\right)\sin\varphi_2 \qquad (7)$$

### 2.3.3 Newton Lagrange modell azonosság

Az (2), (3) és (6), (7) egyenleteket páronként összehasonlítva, látható, hogy az egyenletek megegyeznek, tehát igazoltuk, hogy a különböző modellek azonos eredményt adnak.

### 2.3.4 Differenciális egyenletrendszer

Az $\dot{\underline{x}} = \underline{f}(\underline{x}) + \underline{g}(\underline{x})U_A$ rendszeregyenlethez a konstansokra egyszerűbb jelölést bevezetve:

$\dot{w}_w = (\alpha_w \sin\varphi_2 - \beta_w w_w) + \gamma_w U_a,$

$\dot{w}_2 = (\alpha_2 \sin\varphi_2 - \beta_2 w_w) + \gamma_2 U_a,$

$\dot{\varphi}_2 = w_2.$

Ahol a konstansok:

$\alpha_w = \frac{I_b r_w m_b g}{I_b I_w + m_b r_b^2 I_w + m_b I_b r_w^2},$

$\beta_w = \frac{\left(I_b + m_b r_b^2\right)\left(k_U k_M + r_V R_a\right)}{R_a\left(I_b I_w + m_b r_b^2 I_w + m_b I_b r_w^2\right)},$

$\gamma_w = \frac{k_2\left(I_b + m_b r_b^2\right)}{R_a\left(I_b I_w + m_b r_b^2 I_w + m_b I_b r_w^2\right)},$

$\alpha_2 = \frac{I_w r_b^2 + I_b r_w^2}{\left(r_w + r_b\right)\left(I_b I_w + m_b r_b^2 I_w + m_b I_b r_w^2\right)},$

$$\beta_2 = \frac{I_b r_w \left(k_U k_M + R_a r_r\right)}{R_A \left(r_w + r_b\right)\left(I_b I_w + m_b r_b^2 I_w + m_b I_b r_w^2\right)},$$

$$\gamma_2 = \frac{I_b r_w k_M}{R_a \left(r_w + r_b\right)\left(I_b I_w + m_b r_b^2 I_w + m_b I_b r_w^2\right)}.$$

A feladat az, hogy az $\dot{\underline{x}} = \underline{f}(\underline{x}) + \underline{g}(\underline{x})\cdot U_A$ nem lineáris egyenlet a linearizálás után $\dot{\underline{x}} = \underline{\underline{A}}(\underline{x}) + \underline{b}\, U_A$ alakú legyen, ahol $\underline{\underline{A}} = \frac{\partial f}{\partial x} \,|(\underline{X}_e, U_A)$ és $\underline{b} = \frac{\partial g}{\partial U_A} \,|(\underline{X}_e, U_A)$. Az $\underline{X}_e$ az egyensúlyi helyzethez tartozó paramétereket tartalmazza $\dot{\underline{x}} = 0$ esetén. $\underline{\underline{A}}$ rendszermátrix és $\underline{b}$ konstans vektor, amire a bemeneti $U_A$ paraméteren keresztül hat. Az egyensúlyi pont körül linearizálást végzünk deriválással. A nem lineáris tagot $sin\, \varphi_2 \sim = \varphi_2$ lineáris taggal helyettesítve közelítjük, ami 5 fokig gyakorlatban megfelelő. Levezethető, hogy tetszőleges $U_A$ esetén lehetséges az egyensúlyi helyzetben tartani a golyót.

$$\frac{\partial f}{\partial \underline{x}}\left(\underline{x}_e\right) = \left[\frac{\partial f_1}{\partial w_w}\ \frac{\partial f_1}{\partial w_2}\ \frac{\partial f_1}{\partial \varphi_2};\ \frac{\partial f_2}{\partial w_w}\ \frac{\partial f_2}{\partial w_2}\ \frac{\partial f_2}{\partial \varphi_2};\ \frac{\partial f_3}{\partial w_w}\ \frac{\partial f_3}{\partial w_2}\ \frac{\partial f_3}{\partial \varphi_2}\right]\left(\underline{x}_e\right)$$

Tehát az $\dot{\underline{x}} = \underline{\underline{A}}(\underline{x}) + \underline{b}\, U_A$ egyenlet az alábbiak szerint alakul:

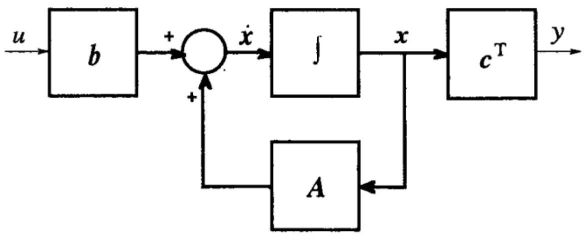
$$\left(\dot{w}_w;\ \dot{w}_2;\ \dot{\varphi}_2\right) = \left[-\beta_w\ 0\ \alpha_w;\ \ -\beta_2\ 0\ \alpha_2;\ 0\ 1\ 0\right]\left[w_w;\ w_2;\ \varphi_2\right] + \left[\gamma_w;\ \gamma_2;\ 0\right]U_A$$

,

$$C^T = (0;\ 0;\ 1),$$

$$D = 0.$$

Így kialakult az állapotteres szakasz modell.




*Ábra 2.3-1*
*Állapotteres modell [3]*

# 3. Szabályozó tervezés

A szabályozó tervezést a szakasz vizsgálatával szükségszerű kezdeni.

## 3.1 Szakasz vizsgálat

Az bow_sys = (A, b, C, D) rendszer mátrixok átviteli függvény részlettörtes alakra alakítást például Matlab program segítségével is végezhetjük, 'zpk(bow_sys)' függvény hívással:

```
                8.5542 s
    ----------------------------
  (s+56.44) (s+3.278) (s-3.282)
```

Látható, hogy van egy jobb oldali labilis, nem lengő pólus és egy jobb oldali kicsi nem lengő zérus, ami nem minimálfázisú rendszert jelent, tehát a bemeneti jel hatására lassú elszálló jelleget mutat a rendszer, ami egy késleltetést jelent.
A szakasz a Scilab 'rank' függvénnyel vizsgálva irányíthatónak és megfigyelhetőnek bizonyult [3].

```
AB = [B, A*B, A*A*B;]              #irányíthatósági mátrix
--> rank(AB)
 ans  =   3.                       #irányítható a rendszer
--> CA=[C; C*A; C*A*A;]
--> rank(CA)
 ans  = 3.          #megfigyelhető, később a pozíció becslő tervezéshez fontos
```

## 3.2 Tervezési sarokpontok

A tervezésnél szempont az alacsony költség és az alkatrészek beszerezhetősége. Ezért már most kialakulnak sarokpontok:

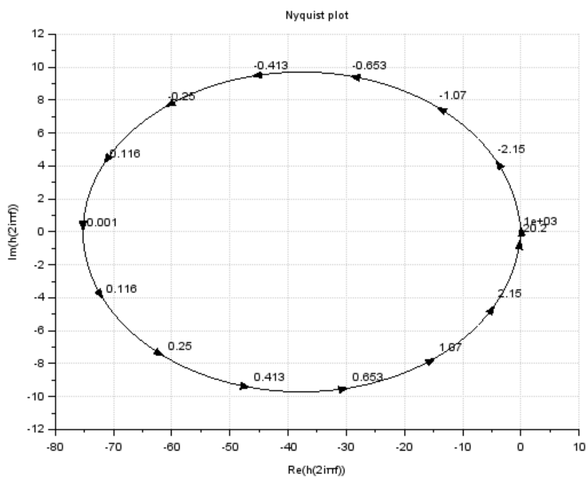
- Szabad felhasználású licensz, alacsony árú, egyszerű HW, például Arduino Nano + inverter: BTS7960-M, UART, Ts = 20-100msec,
- Maximum kilengés < 5fok,
- 6step hajtás, 1 szektor-12.6 mm fent a kerék tetején 2.5fok,
- TpositionADC = 2.5msec,
- Tszakasz = 100msec,
- Linux használat, softRT, Scilab, Octave az egyszerű kontroller implementacio érdekében lehet PC vagy SoC, pl Raspberry Zero 2W.

A fentiek figyelembe vételével a szakasz vizsgálatot követően a szabályozó paraméterek számítását hangolását, majd szimulációját Scilab, Xcos és Matlab programmal végeztem.

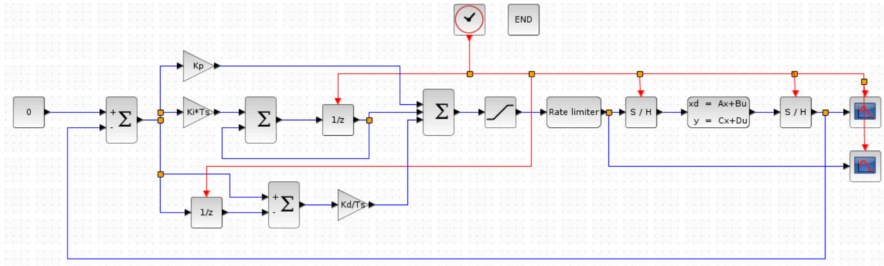
## 3.3 PID szabályozó

Mivel jobb oldali zérus van, azon polinom kiejtést nem hasznos végezni, mert megmarad a belső instabilitás (nem irányítható, nem megfigyelhető)[3].
Tehát el kell fogadni a jobb oldali zérus által okozott ellentétes kilengést (minimálfázisú) és késleltetést. Mivel ez kicsi a szakasznál, e-14 nagyságrendű, lehet vele próbálkozni a gyakorlatban.
A jobb oldali labilis pólus kezelésére pólus kiejtést új zérus bevezetéssel nem hasznos végezni, mert megmarad a strukturális instabilitás (nem megfigyelhető, nem irányíthatóvá válik). A stabilizáláshoz a szabályozó és szakasz nyílt hurkú Nyquist görbének az óra járásával ellentétesen kell 1-szer (1 jobb oldali pólus esetén) kerülnie -1-et [3]. A PID szabályozó ideális paramétereit a Matlab program segítségével könnyen megtalálhatjuk.

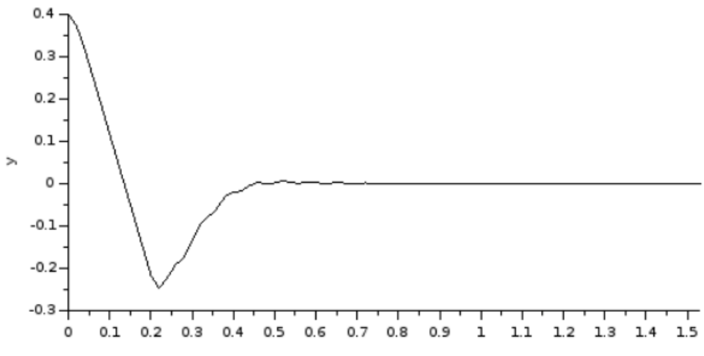
```
>> Gd = c2d(sys, Ts);       % Create discrete-time plant
>> pidTuner(Gd, 'pid')
 Cpid_discr =
                   1              1
  Kp + Ki * Ts * ------ + Kd * ----- * z-1
                  z-1            Ts
   with Kp = 534, Ki = 5.14e+03, Kd = 12.1, Ts = 0.02
Sample time: 0.02 seconds
Discrete-time PID controller in parallel form.
```



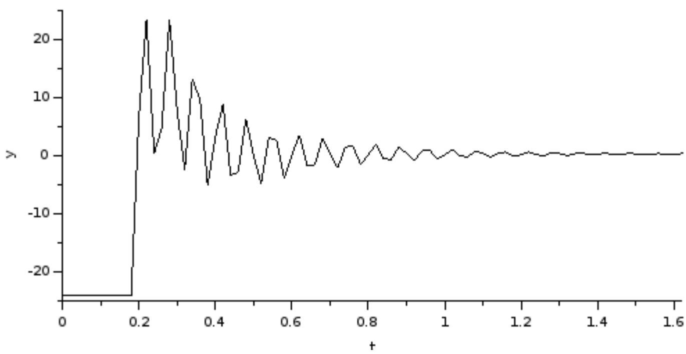

*Ábra 3-1*
*Nyquist diagramm, PID szabályozó és szakasz, nyílt hurok*




*Ábra 3-2*
*PID szabályozó szimulációs blokkvázlat*



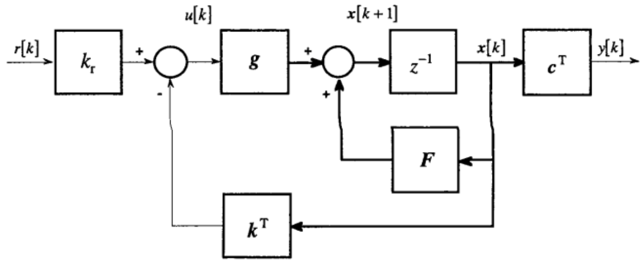

*Ábra 3-3*
*PID szabályozó, pozíció idő függvény*
*x tengelyen idő t[sec], y tengelyen golyó pozíció[rad]*




*Ábra 3-4*
*PID szabályozó, motor feszültség*
*x tengelyen idő t[sec], y tengelyen motor kapocsfeszültség [V]*

## 3.4 Állapot visszacsatolás

A szakasz diszkretizáció [3], amit például az Octave [7] programmal végezhetünk:
*BoW_ssd=c2d(BoW_ss, Ts);    # Ts: sample time*



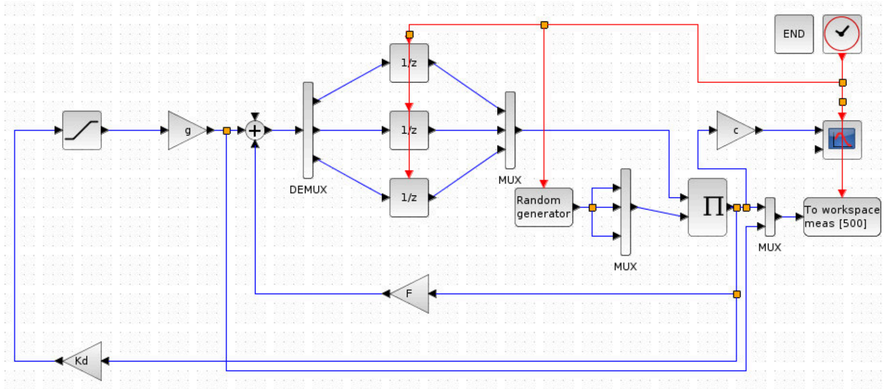

*Állapot visszacsatolás [3]*

A visszacsatoló $k^T$ vektor meghatározása pólusáthelyezéses iterációval végezhető [3], amire most a cikk nem tér ki. Egy másik lehetőség az LQ algoritmus [3][4] alkalmazása, ahol lineáris vektor térben keressük a bemenet és a kimenet közötti optimumot paraméter súlyozással, költség függvénnyel. A súlyok meghatározása iterációs feladat, de jellegében a (8) szerint alakul [12].

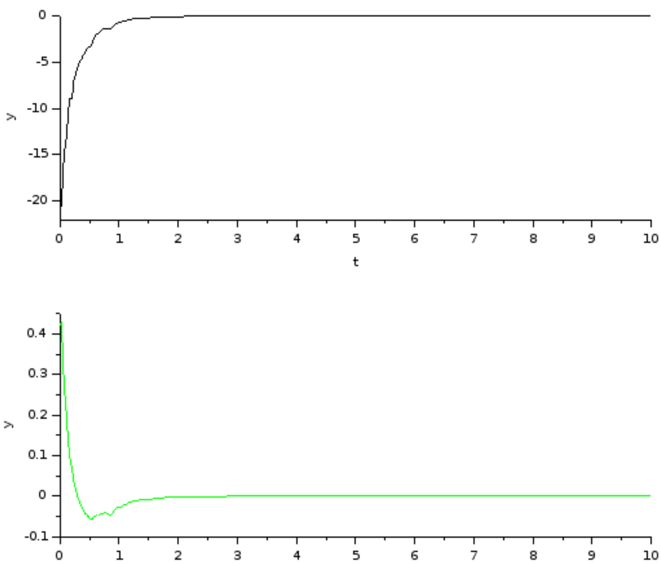
$$q_i = \frac{1}{(\max(x_i))^2} \quad r_j = \frac{1}{(\max(u_j))^2} \tag{8}$$

*q1 = 0.009118906527810399;    #omegaw súly*
*q2 = 0.0914991210900477;    #omega2 súly*
*q3 = 131.3122540004697;    #fi2 súly*
*R = 0.25;    #Uk súly*
*Q = diag([q1, q2,q3]);*
*Big = blkdiag(Q,R);    #közös nagy Q R matrix*
*#// Now we calculate C1 and D12*
*nstates = 3;*
*[w,wp] = fullrf(Big);    #[5]*
*C1=wp(:,1:nstates);    #kiválasztjuk az összes sort és az 1->nstates oszlopokat*
*D12=wp(:,end);    #[C1,D12]'*[C1,D12]=Big //mindegyik sor utolsó eleme-->oszlop matrix*
*P = ss(A,B,C1,D12);    #foly. idejű szakasz*
*[Kd,X] = lqr(P, Q, R);*

Tehát a szabályozó a $K_d$ vektor visszacsatoló tagból áll.



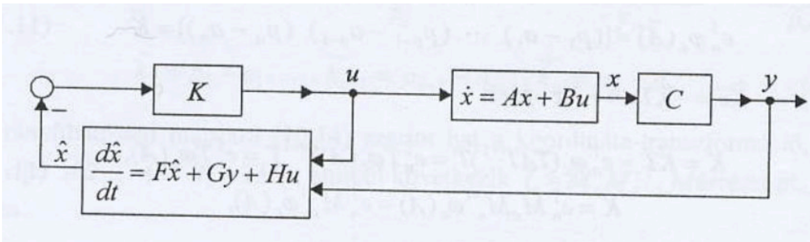

*Ábra 3.4-1*
*Állapot visszacsatolás, Scilab Xcos szimuláció*




*t*
*Ábra 3.4-2*
*Állapot visszacsatolás, szimuláció idő diagram,*
*x tengelyeken idő t[sec], felső ábra y tengelyen motor kapocsfeszültség [V],*
*alsó ábra y tengelyen golyó pozíció[rad]*

## 3.5 Állapot becslő

Az állapot visszacsatolás akkor működik, ha az állapot változókat vissza tudjuk mérni, azonban legtöbbször nem ez a helyzet, mint ahogy ebben az esetben sem. Csak a golyó pozícióról van információnk, viszont mivel ismerjük a rendszer modellt, ezért lehetőség van megbecsülni a többi állapot változót [3][4].

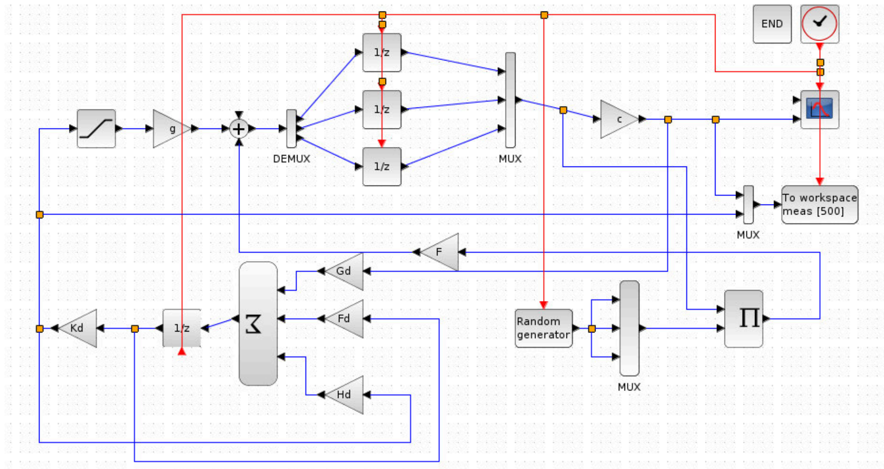



*Ábra 3.5-1*
*Állapot becslő [4]*

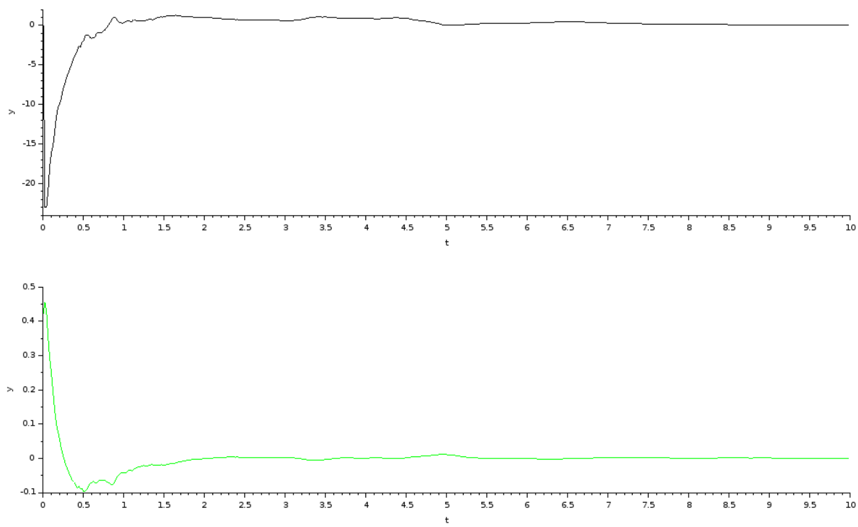
A becslő a szakasz modell egy másolata, amire szintén lehet az LQ eljárást alkalmazni.

*FT = F';*
*cT = c';*
*Big_est = blkdiag(Q,R);*
*[w_est, wp_est] = fullrf(Big_est);*
*c1_est = wp_est(:,1:nstates);*
*d12_est = wp_est(:,end);*
*P_est = ss(FT, cT, c1_est, d12_est, Ts);*
*[K_estd,X_estd] = lqr(P_est, Q, R);*
*eFTcTKestd=eig(FT+cT*K_estd);*
*Gd = K_estd';*
*Fd = F - Gd * c;*
*Hd = g;*

Tehát a szabályozó:
sum_zn1 = Fd * sum_zn1 + Gd * x(3) + Hd * u;
u = Kd * sum_zn1;
Ahol ‘x(3)’ a mért golyó pozíció, ‘u’ pedig a kiadott motor kapocsfeszültség.



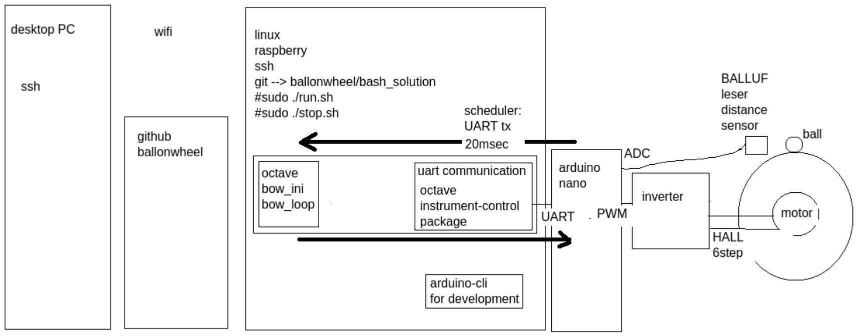

*Ábra 3.5-2*
*Állapot becslő Scilab, Xcos szimuláció*




*Ábra 3.5-3*
*x tengelyeken idő t[sec], felső ábra y tengelyen motor kapocsfeszültség [V],*
*alsó ábra y tengelyen golyó pozíció[rad]*

# 4. Szabályozó megvalósítás

A szakaszról a korábbiakban kiderült, hogy adott szabályozási pontosság (kilengés) mellett egy gyengébb erőforrással rendelkező szabályozó is elláthatja a feladatot. A mindenki asztalán megtalálható PC erre alkalmasnak tűnik némi megkötéssel. Nem lesz valós idejű, de ha nem sűrű és nagy a *dTs* (jitter), akkor a robosztus szabályozó kezelni tudja a problémát.




*Ábra 4-1*
*Rendszer összeállítás*

| *Feladat* | *Eszköz* | *Megjegyzés* |
|---|---|---|
| Érzékelő és beavatkozó processzora | Arduino Nano | Ts=20msec, 115200 baud, 3 bájt frame hossz, ADC 8bit, PWM 8bit |
| Szabályozó processzora | RasperryPi-Zero2w | 4 mag, 1Ghz, 512kRAM 1mag foglalva a szabályozónak |
| Távolság érzékelő | BALLUFF BOD 21M-LA04-S92 | 2.5msec, analóg 10V, lézer, 500mm, 500um |
| Állandó mágnesű szinuszmezős szinkron hajtás motor | 36V 300W 46 mágnes, 3*17 tekercs, 22col | Hat lépéses blokk kommutáció HALL szenzorok alapján |
| Inverter | 2db * BTS7960-M | 48V, 43A, 1usec deadtime |

*Tábla 4-1*
*Rendszer összeállítás alkatrészei*

A RaspberryPi-Zero2w, Linux-Debian környezet alkalmasnak tűnt a feladat megvalósítására, ahol egy mag van a szabályozó részére lefoglalva. A Raspberry-n Octave szkript-en fut a szabályozó algoritmus, amiben az UART kommunikációhoz az “instrument-control” csomag [7] megfelelő.
A BALLUFF távolság érzékelőt és a motor hajtás invertert egy Arduino Nano kezeli, ami az UART-tx interfészen ütemezi a szabályozót, majd az UART-rx vonalon várja a szabályozó választ, amit az inverter-nek továbbít duty cycle, kapocsfeszültség formájában. Az Arduino elvégzi a hat lépéses motor kommutációt a kapott HALL szenzorok jelei alapján.
A Raspberry Zero2w rendelkezik wifi interfésszel, tehát SSH programmal csatlakozhatunk az eszközhöz, aminek segítségével könnyen elérhetővé válik például az Octave install, git verzionálás, html szerver, GCC cross compile az Arduino CLI [8], tehát a teljes fejlesztő környezet díjmentes, open-source és kis erőforrásigényű, konzolból elérhető. Természetesen a Raspberry lecserélhető egy megfelelően erős asztali számítógépre is.
A rendszer tovább fejleszthető a realtime Linux [13] irányába is, az igényektől függően. Komolyabb real time igények esetén, mint például állandó mágnesű szinkron motor mezőorientált nyomaték szabályozása, a mikrokontrolleres környezet célravezetőbb a perifériák gyors kezelése érdekében.

# 5. Motor paraméter identifikáció

A szabályozó megvalósítást a motor paraméterek (Ls, Rs, Iw, Km, Ke) meghatározása megelőzte, amely a 1. táblázatban található. A motor szemrevételezése és típusa alapján egy 36V-os 300W-s szinuszmezős állandó mágnesú szinkron HUB hajtás motor (PMSM) kerül alkalmazásra.
A ‘Ke = U peak / w mech’ értéke a BEMF üresjárási kapocsfeszültségből és a motor ismerete alapján, (46 mágnes, tehát 23 póluspár és 3*17 állórész tekercs) meghatározható. A ‘Km = 3/2 Ke’ a PMSM motoroknál ismert [9]. Az Rs, Ls értékek RLC méréssel kerültek meghatározásra, míg Iw inercia arányossággal történt: $\Sigma M = I_w * \beta_{w1}$ , $\Sigma M = I_w * \beta_{w2}$
Tehát meg kell mérni a kerék fordulatszám felfutási idejét álló helyzetből maximális értékre terhelés nélkül, majd ismert terheléssel. A felfutási idő valamint a terhelő súlyok ismeretében a kerék tehetetlenségi nyomatéka jó közelítéssel meghatározható. A terhelő súlyokat a kerék peremén a lehető legjobban elosztva kell elhelyezni [10].

$I_w$ / d $I_w$ = $M_{üres}$ / dM, ahol d $I_w$ = $I_{w\_terhelt} - I_w$ és dM = $M_{terhelt}$ - $M_{üres}$

A súrlódási erő ($r_v$) kis értékre lett becsülve, amit bizonyára tovább lehetne vizsgálni, mint ahogy a “cogging” nyomatékból származó indítónyomaték hullámzást is. Ezen hibák a szabályozó nagyobb kitéréseit okozzák.

# 6. Verzionálás, működtetés

A szoftver elérhető és működtethető az alábbiak szerint:
>git clone https://github.com/ballonwheel/ballonwheel.git
>cd ballonwheel/bash_solution_only_octave
>sudo ./run.sh
Opcionális kijelző(karakteres) használata:
>sudo octave bow_octave_disp.m
>sudo ./stop.sh
Egyenletek kifejtése
> cd ballonwheel/maxima
> maxima -batch=bowSymbol.mc

**Következtetések, célkitűzések**

A kifejlesztésre került eszköz jól használható kiállításokon irányítástechnikai demonstrációs célra, valamint szakmai gyakorlati órákra a fejlesztés különböző lépései a modellalkotás, szabályozó tervezés és megvalósítás, hangolás, motor identifikáció és motor hajtás ismereteinek elsajátítására. A kialakított rendszer esetleg alapja lehet egy asztali, kisebb változat is vagy "DIY" egységcsomag amely motiváló lehet a diákok számára.

**Köszönetnyilvánítás**

Köszönetet mondok Dr. Kopják Józsefnek (IoT tanszék) a támogatásáért, hasznos ötletekért és javaslatokért az eszköz és a cikk elkészítésében.

**Referenciák**

[1] Control of the Ball on the Wheel System, R.W. van Gils, DCT 2007.010. Technische Universiteit Eindhoven, Department Mechanical Engineering, Dynamics and Control Technology Group, Eindhoven, February, 2007

[2] Lindert, Sven & Höfler, Christian & Schlacher, Kurt. (2018). Nichtlineare Regelung mit einem Raspberry PI: Automatisierung des Versuchsaufbaus „Ball auf Rad" mit einem Raspberry PI und offener Hard- und Software. e & i Elektrotechnik und Informationstechnik. 135., 10.1007/s00502-018-0613-8.

[3] Keviczky László, Bars Ruth - Hetthéssy Jenő, Barta András, Bányász Csilla, Szabályozástechnika, Szent István Egyetem, Győr, 2006

[4] Lantos Béla, Irányítási rendszerek tervezése,

11. Fejezet, Szabályozások tervezése állapottérben, 2016, ISBN: 9789630587280

[5] Openeering, LQ, Online:

http://www.openeering.com/sites/default/files/Inverted_Pendulum.pdf

[6] Matlab, Online:

https://www.mathworks.com/help/control/ref/pidtuner-app.html

[7] Octave, Instrumentation tool, Online:

https://octave.sourceforge.io/instrument-control/package_doc/

[8] Arduino cli: Online: https://arduino.github.io/arduino-cli/0.35/

[9] Dr. Halász Sándor, Villamos hajtások, 1993, (7.14, pp 293)

[10] Gilbert, Sólyom, Kocsányi, „Merev testek kinematikai leírása," Fizika

Mérnököknek, Műegyetemi kiadó, 1999, (pp 62)

[11] Maxima: Online https://maxima.sourceforge.io/

[12] Bauer Péter, Repülőgépek egyszerű referenciajel követő szabályozóinak tervezése LQ Servo módszerrel Matlab/Simulink környezetben

BME Közlekedésautomatikai Tanszék, 2009. (pp 6)

[13] RTAI, Online: https://www.rtai.org/