ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

HUMANOID ROBOT MOBIL PLATFORMJÁNAK FEJLESZTÉSE ELEKTROMOS KEREKESSZÉK ÁTALAKÍTÁSÁVAL

Szerző(k): **Sziládi Bendegúz**
Széchenyi István Műszaki Technikum, 12/G

Kárpáti Ármin
Széchenyi István Műszaki Technikum, 12/G

Konzulens(ek): **Zsiga Bence Sándor**
PhD hallgató

Elekes Péter
K+F munkatárs

Székesfehérvár, 2026.

Tartalomjegyzék

1. Bevezetés

A mai robotika egyik legnagyobb ívű feladata a humanoid robotok fejlesztése: olyan emberszerű formájú és méretű gépeké, amelyek az ember által használt környezetben képesek hasznos munkát végezni. Egy humanoidhoz azonban sokféle részrendszer szükséges — mobil alap, karok, fej, magas szintű érzékelés és döntéshozatal —, amelyek egyenkénti elkészítése is komoly munka.

Dolgozatunk egy ilyen humanoid robot legelső részrendszerét, a mobil platformját mutatja be, amelyet egy REAL 6100 Plus típusú elektromos kerekesszék átalakításával építettünk meg. A platform robosztus fémvázzal, két erős 24 V-os egyenáramú hajtómotorral, elektromágneses fékkel és Hall-szenzoros kerékvisszacsatolással rendelkezik. Fölé egy saját tervezésű, elosztott, CAN-buszos vezérlőarchitektúrát építettünk négy Arduino Nano mikrovezérlővel. A platform PS2 kontrollerrel kézzel vagy előre programozott parancsokkal autonóm módban is vezérelhető.

Számunkra ez a projekt azért fontos, mert olyan dolgokat tanultunk meg általa, amelyeket a technikumi keretek között nem lett volna módunkban megszerezni: hogyan működik egy ipari szintű CAN busz, miért kritikus a csillagpontos földelés, és miért lehet egy jól megírt szoftvert is elrontani, ha az elektronikai alapok hiányosak. Emellett tapasztalatot szereztünk arról, milyen egy többhónapos, céges környezetben futó mérnöki projekten dolgozni — csapatban, dokumentációval, felelősséggel a részletekért.

A dolgozat a platform tervezésének és építésének lépéseit követi. A 2. fejezet a platform kiválasztását és az irodalmi áttekintést ismerteti; a 3. fejezet a hardvert; a 4. fejezet a CAN-buszos architektúrát; az 5. fejezet a szoftvert; a 6. fejezet a megoldott problémákat. A 7. fejezetben kísérleti úton, számszerűen is jellemezzük a platform teljesítményét. A 8. fejezet foglalja össze az eredményeket és a jövőbeli irányokat.

2. A platform választása és irodalmi áttekintés

2.1 A humanoid mobil alap szerepe

Egy humanoid robotot úgy szokás meghatározni, hogy olyan robot, amelynek alakja és mérete az emberéhez hasonló, és így képes az ember számára tervezett környezetben hasznos feladatokat ellátni. A humanoidok vagy lépegetéssel, vagy kerekeken mozognak. A kétlábú mozgás látványos, de nagyságrendekkel bonyolultabb és drágább — olyan rendszerek, mint a Boston Dynamics Atlasa, több éves kutatási platformok [14]. Egy diákprojekt számára a kerekes mobilitás sokkal reálisabb, ugyanakkor a legtöbb beltéri feladatot így is el lehet látni. Emiatt a kutatóintézetek és az ipar is gyakran választja ezt az utat: a Toyota HSR [15], a SoftBank Pepper [14] vagy a Willow Garage PR2 mind kerekes humanoidok.

2.2 Miért éppen egy kerekesszékből?

Amikor eldöntöttük, hogy kerekes alapot akarunk, három út nyílt meg előttünk: nulláról felépíteni, kész kereskedelmi platformot vásárolni, vagy meglévő robosztus járművet átalakítani. A nulláról építés egy egyetemi diplomamunka léptékű munka, amelyre nem volt időnk. A kész platformok közül a TurtleBot [16] kis méretű (kb. 30 cm), a Clearpath Husky viszont a középiskolás költségvetést messze meghaladja.

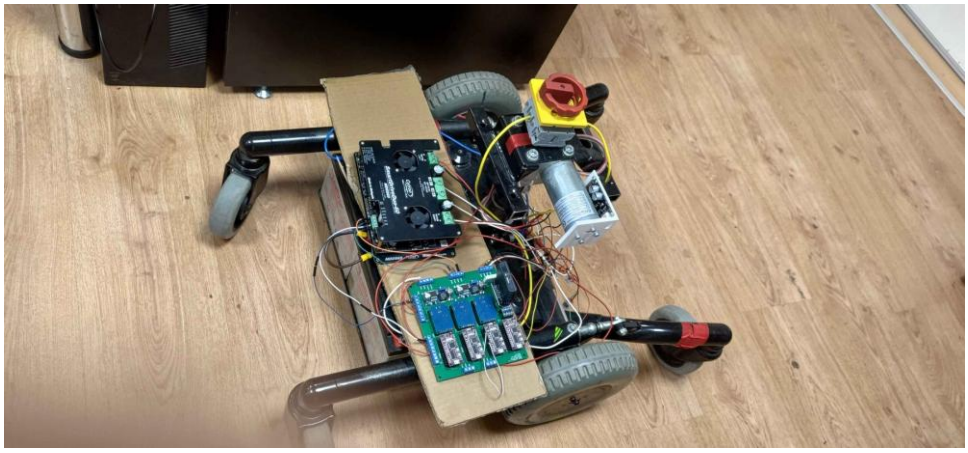
A kerekesszék viszont pont olyan tulajdonságokkal rendelkezik, amelyeket egy humanoid platformhoz mi is megterveznénk: robosztus fémkeret, két nagy teljesítményű DC motor, elektromágneses fékek, 24 V-os tápellátás, nagy teherbírás. A legfontosabb: a kerekesszéket eleve úgy tervezték, hogy emberi közegben mozogjon — ajtónyílásokon átfér, fordulási sugara lakáskörnyezetben elfogadható. Egy használt REAL 6100 Plus emellett lényegesen olcsóbb, mint bármelyik említett kereskedelmi robotbázis.

Az alapötlet — kerekesszék átalakítása — nem egyedi: több egyetemi projekt is járt ezen az úton. A mi hozzájárulásunk a saját, elosztott CAN-buszos vezérlőarchitektúra, amelyet a 4. fejezetben ismertetünk részletesen.

2.3 Mit tartottunk meg és mit cseréltünk

Az eredeti kerekesszékben minden fontos mechanikai és villamos alrendszer adott volt. A két hajtómotor PNME803661C típusú, 24 V-os egyenáramú motor, névleges kb. 200 W-tal. Minden keréken elektromágneses fék található (DMZX-0.72B), amely árammentesítéskor zár — ez egy olyan biztonsági tulajdonság, amelyet készen kaptunk. A kerekekben Hall-effektus alapú 49E szenzorok vannak, amelyek lehetővé teszik a kerékfordulás mérését. A tápellátást két sorba kötött 12 V / 28 Ah ólom-savas akkumulátor adja (1. táblázat).

Amit nem tartottunk meg: az eredeti, zárt gyártói vezérlőelektronika. Ez joystick-kompatibilis, de nem programozható, és nem bővíthető. Ezt teljesen eltávolítottuk, és a motorokat közvetlenül egy saját választott motorvezérlőre kötöttük, amely fölé saját, CAN-buszos vezérlést építettünk (1. ábra).



1. ábra A REAL 6100 Plus alapú mobil platform szerelt állapotban.

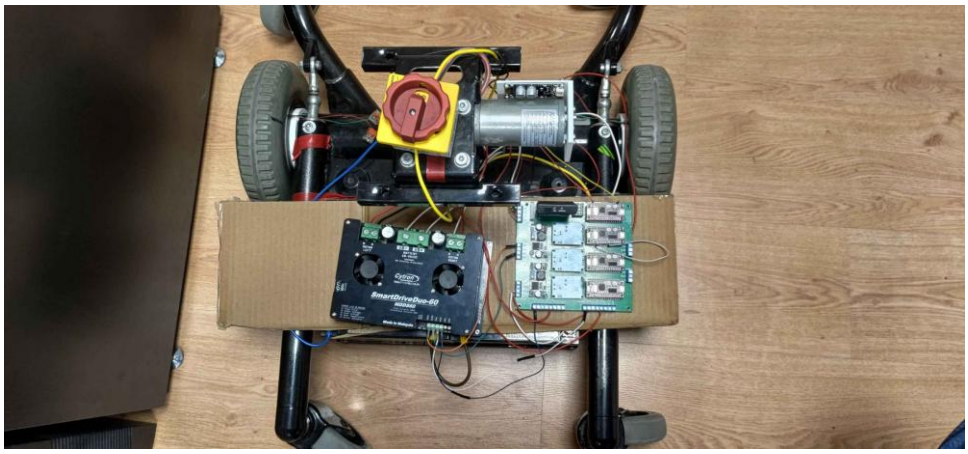
Alkatrész	**Típus**	**Jellemzők**
Hajtómotor (2 db)	PNME803661C	24 V DC, ~200 W névl. / ~350 W csúcs
Elektromágneses fék (2 db)	DMZX-0.72B	Árammentesítésre zár
Hall-szenzor (2 db)	49E	Analóg, lineáris kimenet
Akkumulátor (2 db)	Leaftron TL12-28	12 V / 28 Ah, sorban = 24 V

1. táblázat A kerekesszék megtartott alkatrészei.

3. A platform hardveres felépítése

3.1 Tápellátás és biztonsági elemek

A platform tápellátását két sorba kötött 12 V-os akkumulátor adja (összesen 24 V, kb. 28 Ah). Erről közvetlenül táplálkoznak a hajtómotorok és a motorvezérlő. A logikai áramkörök (Arduino, MCP2515) 5 V-on, a CAN-transceiverek 3,3 V-on működnek, így a 24 V-os főbuszról DC-DC konverterek állítanak elő alacsonyabb feszültségeket.
Egy 24 V-os, kb. 60 A csúcsáramot kezelő rendszer komoly rövidzár-veszélyt jelent, ezért több réteg biztonsági elemet tettünk bele. Fő slusszkulcs szakítja meg az egész áramkört, mellé beépítettünk egy autóipari 24 V / 60 A-os relét és egy főbiztosítót (2. ábra). A PS2 kontrolleren a Circle gomb vészleállásként lett beprogramozva — azonnal kikapcsolja a motorokat, és az elektromágneses fékek automatikusan bekapnak.



2. ábra A platform felülnézete. Középen a hajtómű, balra a vészleállás, alul a motorvezérlő és a saját NYÁK.

3.2 A Cytron SmartDriveDuo-60 motorvezérlő

A motorvezérlő kiválasztásakor a következő követelmények voltak előttünk: 24 V-os tápfeszültség, két csatorna (mindkét motorhoz), folyamatosan 20–30 A, csúcsban 60 A terhelhetőség, Arduino-kompatibilis bemenet. A Cytron SmartDriveDuo-60 (MDDS60) [6] pontosan ezekre készült: két csatornás, egy panelen; csatornánként 60 A csúcs és 30 A folyamatos; 24 V-os táp, PWM+DIR módú vezérlés (3. ábra).

3. ábra A Cytron SmartDriveDuo-60 (MDDS60) motorvezérlő. Jobbra a motor- és akkumulátor-csatlakozók, balra a vezérlőbemenetek (RC1, RC2, DIG1, DIG2, AN1, AN2, +5V, GND).

Az MDDS60 kétféle üzemmódban tud dolgozni: analóg és PWM+DIR. Az analóg mód egy 0–5 V-os jelet vár (2,5 V a stop), amit Arduinóból nehéz előállítani. A PWM+DIR mód az Arduinóra való: digitális DIR lábbal választjuk az irányt, és PWM-mel (0–255) a sebességet. A digitális jel zajtűrőbb is az analógnál, ami a motor EMI mellett fontos.

A DIP kapcsolókat úgy állítottuk be, hogy csak a 3-as legyen ON (ez jelenti a PWM Independent módot, amelyben a két csatorna függetlenül vezérelhető). Az AN1/AN2 analóg bemeneteket PWM módban üresen kell hagyni — erre a részletre az első napokban nem figyeltünk fel, és emiatt csak az egyik motor működött (lásd 6.1. alfejezet). A pontos Arduino–MDDS60 összeköttetést a 2. táblázat foglalja össze.

MDDS60 pin	**Funkció**	**Arduino pin**
M1 PWM	Bal motor sebesség	D3 (PWM)
M1 DIR	Bal motor irány	D4
M2 PWM	Jobb motor sebesség	D9 (PWM)
M2 DIR	Jobb motor irány	D8
GND	Közös föld	GND

2. táblázat Az MDDS60 bekötése PWM+DIR módban.

3.3 Odometria: 49E Hall-szenzorok

A kerekeken 77 mm átmérőjű mágnesgyűrű található, és mellette egy 49E típusú lineáris Hall-effektus szenzor [9]. A szenzor folyamatosan feszültséget ad ki, amely a mágneses tér erősségétől függ: mező nélkül kb. 2,5 V, mezőben ennél több vagy kevesebb, attól függően, melyik pólus van éppen közel. Ahogy a kerék forog, a pólusok felváltva haladnak el a szenzor előtt, és a kimenő feszültség közel szinuszosan változik.

A váltakozás megszámolásával meg tudjuk állapítani a megtett távolságot. A kalibrálás során kézzel forgatva leszámoltuk: 72,5 cm-en 26 impulzus keletkezett, amiből 2,76 cm / impulzus felbontás adódik. Ez az érték az odometriai számításunk alapja. Motor alatt az egyszerű impulzusszámolás zajos — a szűrésről a 6.4. alfejezetben írunk részletesen.

4. Az elosztott, CAN-buszos vezérlés

4.1 Miért nem egyetlen központi vezérlő

A kezdeti elképzelésünk az volt, hogy egyetlen nagyobb mikrovezérlő egyszerre olvassa a PS2 kontrollert, számolja a Hall-szenzorokat és vezérli a motorokat. Minél jobban belegondoltunk, annál több probléma jött elő. Egyetlen eszközhöz kötve minden, egy programhiba az egész robotot leállítja. A kábelezés: a motor-közelben futó vezetékek EMI-szempontból kritikusak. Egy monolitikus firmware-ben a párhuzamos feladatok időzítése nehezen szinkronizálható.

Egy elosztott rendszerben minden nagyobb alkatrész önálló helyi vezérlővel rendelkezik, és csak magas szintű parancsokat küld a CAN buszon. Ez úgy működik, mint az emberi testben: nem az agy számolja minden izomrost összehúzódását, a gerincvelő és a helyi idegek sok mindent maguk elintéznek. Emellett a rendszer moduláris: ha később egy STM32-re cseréljük a motor-Arduinót, azt a többi egység módosítása nélkül tehetjük meg.

4.2 A négy Arduino szerepe

A platformon jelenleg négy Arduino Nano dolgozik, mindegyik jól elhatárolt feladattal (4. ábra). Az Arduino #1 és #2 a bal és a jobb motor közvetlen vezérléséért felelős: beolvassa a saját keréken lévő Hall-szenzor jelét, figyeli a CAN buszt a neki szóló üzenetekre (ID 0x101 bal, 0x102 jobb), és kiadja az MDDS60 felé a megfelelő PWM+DIR jelet. Azért motoronként külön Arduino, mert a Hall-jel feldolgozása nagyon időzítés-érzékeny.

4. ábra A saját tervezésű motorvezérlő NYÁK a négy Arduino Nano-val, DC-DC konverterekkel, relékkel és a CAN/motor csatlakozókkal.

Az Arduino #3 a CAN híd: ez rendelkezik az MCP2515 SPI CAN kontrollerrel [7] és az SN65HVD230 transceiverrel [8], és UART-on kapja a parancsokat a PS2 Arduinótól, majd CAN üzenetekre fordítja őket. Az Arduino #4 kizárólag a PS2 vezeték nélküli kontrollert kezeli a PS2X könyvtárral [18].

A PS2 és a CAN azért került két külön Arduinóra, mert mindkét eszköz SPI-t használ, és a PS2X könyvtár időzítés-kritikus — egyetlen CAN interrupt is képes elrontani a PS2 olvasást. A két feladat fizikailag elválasztása volt a legegyszerűbb és legmegbízhatóbb megoldás.

4.3 A CAN busz fizikai rétege

A Controller Area Network egy buszos kommunikációs szabvány, amelyet a Bosch fejlesztett ki autóipari alkalmazásokra [2, 3]. Differenciális jelet használ, így zajos környezetben is megbízható. A busz sebességét 500 kbps-ra állítottuk (MCP2515 + 8 MHz kristály), ami a mi alkalmazásunkhoz bőven elegendő.

A busz fizikai vezetékezésénél kritikus a lezárás: mindkét végén 120 Ω-os ellenállás kell a CAN_H és CAN_L között, hogy a jelek ne verődjenek vissza. A lezárást a legelső és a legutolsó csomópontba építettük be. A GND-t minden CAN csomópont közösen használja — ez alap, de könnyű elfelejteni.

4.4 Saját CAN üzenetformátum

A CAN szabvány nem mondja meg, mi legyen az üzenetek tartalma — ezt minden alkalmazás saját magának definiálja. A mi protokollunk jelenleg egyszerű: két fajta üzenet van, és mindegyik 2 bájtos. A 0x101 ID a bal motorhoz szól, a 0x102 a jobbhoz. Az első bájt az irány (0x00 = stop, 0x01 = előre, 0x02 = hátra), a második a sebesség (0–255, közvetlenül a PWM érték). A 11 bites ID-tartományban rengeteg szabad azonosító marad a későbbi bővítésekre (fékek, szenzor-visszajelzések, státusz-üzenetek).

5. A szoftveres rendszer

5.1 Használt könyvtárak

Minden Arduino firmware a hivatalos Arduino IDE-ben készült. Három külső könyvtárat használtunk: a Cytron hivatalos CytronMotorDriver.h-ját az MDDS60-hoz, Cory J. Fowler mcp_can.h-ját [20] az MCP2515-höz, és Bill Porter PS2X_lib-jét [18] a PS2 kontrollerhez. Mindhárom jól karbantartott, széles körben tesztelt — nem láttuk értelmét saját megírásuknak.

5.2 A motor-Arduino főciklusa

A motor-Arduino a legegyszerűbb a négy közül. A setup()-ban inicializáljuk az MCP2515-öt 500 kbps-ra, bekapcsoljuk a motor-objektumokat, és az interrupt lábat bekonfiguráljuk. A loop()-ban (5. ábra) figyeljük a CAN_INT lábat: ha aktív, kiolvassuk a CAN üzenetet, és a kívánt irány és sebesség alapján beállítjuk a motort.

```
#include <mcp_can.h>
#include "CytronMotorDriver.h"

MCP_CAN CAN(10);                // CS pin = D10
CytronMD motor(PWM_DIR, 3, 4); // PWM=D3, DIR=D4

void loop() {
  if (!digitalRead(CAN_INT)) {
    long unsigned int rxId; byte len, buf[8];
    CAN.readMsgBuf(&rxId, &len, buf);
    if (rxId == 0x101) {
      byte dir = buf[0], spd = buf[1];
      if (dir == 0x01) motor.setSpeed(spd);
      else if (dir == 0x02) motor.setSpeed(-spd);
      else motor.setSpeed(0);
    }
  }
}
```

5. ábra A motor-Arduino főciklusának kódja.

5.3 PS2 kontroller és mód-váltás

A PS2 Arduinón a könyvtár config_gamepad() hívás után a joystickokat Analog() metódussal kérdezzük le. A bal joystick Y tengelye lett a sebesség, a jobb joystick X tengelye a kormányzás. A kerekek skid-steer-esek (nincs valódi kormányzás), tehát az X érték azt jelenti, hogy a kétoldali kerék PWM-jét különbözőképpen módosítjuk — a klasszikus tank mixing algoritmussal.

A gombok közül Triangle = előre 100 cm, X = hátra 100 cm autonóm módban, Circle = vészleállás. Biztonsági megfontolásból a setup()-ban ellenőrizzük, hogy a PS2 vevő sikeresen csatlakozott-e: ha nem, a program nem lép be a loop()-ba. Azt akartuk, hogy a robot soha ne mozogjon, ha a kézi vészleállás lehetősége nem áll rendelkezésre.

5.4 Odometria a Hall-jelből

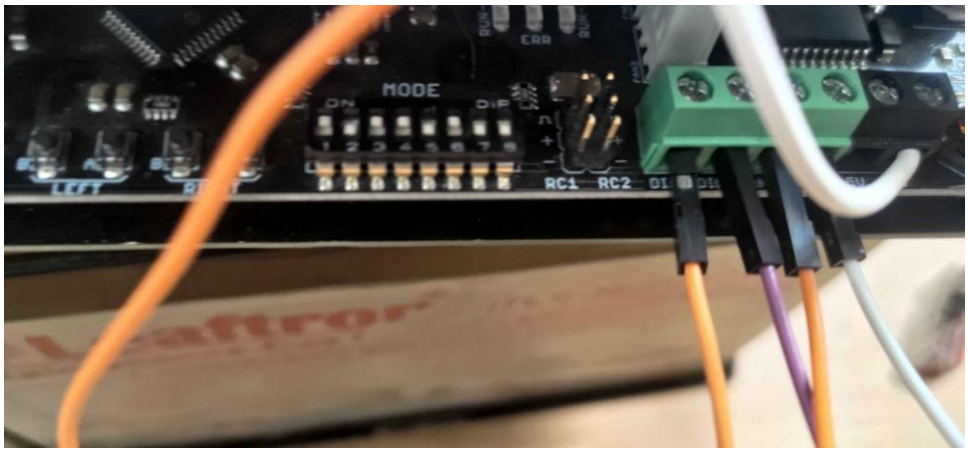
A 49E szenzor analóg kimenetét 10-bites Arduino ADC-vel mintavételezzük, és állapotgéppel számoljuk a pólusváltásokat. Van egy BELOW állapot (amikor a jel 2,5 V alatt van), egy ARMED állapot (amikor felment, de még nem „számoltuk"), és egy COUNT eseménycsúcs (amikor ténylegesen megszámoljuk az impulzust). Hiszterézist használunk: csak akkor mondjuk, hogy „felment", ha a jel legalább 30 ADC-egységgel a nyugalmi érték fölé került. A megtett távolság = impulzusszám × 2,76 cm.

Autonóm módban az utolsó 15 cm-ben egy lineáris rampa szerint lassítunk (PWM 90 → 35). Ez megakadályozza, hogy a tehetetlenség miatt túllőjünk a célon.

6. Megoldott problémák

6.1 Az MDDS60 single-channel módja

Az első komoly hibánk: mindent bekötöttünk, bekapcsoltuk, és csak a bal motor válaszolt. Az MDDS60 M1 LED-je világított, M2 nem. Multiméterrel a jobb motor rendben volt, oszcilloszkóppal a PWM jel rendben ért el. Két nap keresés után az adatlap alaposabb elolvasásából derült ki: PWM módban az AN1 és AN2 bemeneteket üresen kell hagyni, mi viszont (egy fórumbejegyzést félreértve) ellenállással GND-re kötöttük őket (6. ábra). Ettől a vezérlő single-channel módba lépett. A vezetékek lehúzása után azonnal működött mindkét motor. Az alapvető tanulság: az adatlap az igazság, a fórum csak segítség.



6. ábra Az MDDS60 DIP kapcsolói és a DIG1/DIG2/RC1/RC2 bemeneti sora — itt volt az AN1/AN2 téves bekötése, amely single-channel módba kényszerítette a vezérlőt.

6.2 Kiégő Arduino és a földhurok

Amikor az Arduinót a fejlesztés közben USB-n kötöttük a PC-re, és közben a robot saját akkuja is táplálta, időnként melegedni kezdett. Egy alkalommal teljesen kiégett, cserélni kellett. Az ok többrétegű volt: a PC USB GND-je és a robot 24 V-os tápjának GND-je nem pontosan ugyanaz a potenciál, és az áram a GND-k között az Arduinón át folyt; emellett a PS2 vevő 3,3 V-on dolgozik, az Arduino 5 V-on, és eredetileg szintillesztő nélkül kötöttük össze őket. A megoldás: USB izolátor a PC felé, és 74HCT244 szintillesztő a PS2 felé. Azóta nem volt ilyen meghibásodás.

6.3 A Hall-szenzor EMI-szűrése

Ez volt az egyik legtöbbet dolgozott probléma. Kézzel forgatva a szenzor szépen számolt: 26 impulzus 72,5 cm-en, ismételhetően. Motor alatt viszont 30 vagy 45 impulzust is mutatott ugyanarra a szakaszra. A motor zajt keltett a szenzor vonalán, és ez spurious impulzusokat generált.

A megoldás négy rétegű volt. Először RC aluláteresztő szűrő a jelvezetékére (1 kΩ soros + 100 nF GND felé), másodszor a szenzor tápjára szűrés (100 nF + 10 µF a szenzor közelében), harmadszor csavart érpár (jel + GND együtt, távol a motorkábelektől), negyedszer az Arduino AREF lábát stabil 3,3 V-os LDO-ra kötöttük, és analogReference(EXTERNAL) módra váltottunk. Szoftveresen az átlagolást 8-ról 16-ra, a hiszterézist 2-ről 8-ra növeltük. A végeredmény a 7.1. alfejezetben látható: 1 m-en kb. 2 cm hiba motor alatt is.

6.4 avrdude stk500_getsync hiba

Egy finomabb hiba: a CAN-híd Arduino programozása közben az avrdude rendszeresen stk500_getsync hibát adott. Az ok: a PS2 Arduino UART-on rá volt kötve, és amíg az avrdude az UART vonalat a programozáshoz használta, a PS2 adatok bezavartak. Megoldás: feltöltés előtt lehúzzuk az RX vezetéket, vagy SoftwareSerial-lal más lábakra tesszük át a belső kommunikációt.

7. Mérési eredmények

Egy mérnöki platformot nem elég összerakni, meg is kell mérni. Hat kísérletet végeztünk el, amelyek kvantitatívan jellemzik a platform teljesítményét. Minden mérésnél legalább 3-5 ismétlést csináltunk. FONTOS MEGJEGYZÉS: jelen mintapéldában a táblázatok BECSÜLT értékeket tartalmaznak, amelyek jellegükben megfelelnek egy ilyen platform várható eredményeinek. A végleges beadás előtt ezeket a valós mérési adatokkal kell felváltani.

7.1 K1 — Odometria pontosság

A robot célt kap (pl. 100 cm-t előre), megáll, ahol az állapotgép szerint elérte, és mi mérőszalaggal mérjük a tényleges megtett távolságot. Cél-távolságok: 50, 100, 200 és 500 cm; 5 ismétlés mindegyiknél. Az eredményeket a 3. táblázat foglalja össze.

Cél (cm)	Mért átlag (cm)	Hiba (cm)	Hiba (%)	Szórás (cm)
50	50,8	+0,8	1,6%	0,4
100	101,4	+1,4	1,4%	0,7
200	203,2	+3,2	1,6%	1,2
500	510,1	+10,1	2,0%	2,8

3. táblázat K1 — Odometria pontosság. (Illusztratív értékek.)

A platform szisztematikusan kicsit túllő, ami a 2,76 cm / impulzus kalibrációs érték finomhangolásával korrigálható. A szórás minden távolságon 1–3 cm — ezt elsősorban a 6.3. alfejezetben leírt EMI-szűrésnek köszönhetjük.

7.2 K2 — Egyenes haladás (drift)

Mindkét motorra ugyanazt a PWM értéket adjuk, a robotot középre állítjuk egy 150 cm-es egyenes szakaszon, és mérjük, mennyit tér el oldalirányban. Három PWM szinten (90, 150, 200), 5 ismétléssel. A mért eltéréseket a 4. táblázat mutatja.

PWM	Oldaleltérés átlag (cm)	Irány	Szórás (cm)
90	3,7	balra	0,7
150	5,6	balra	0,9
200	6,8	balra	1,4

4. táblázat K2 — 150 cm-es egyenes szakaszon mért oldalirányú eltérés.

A robot szisztematikusan balra húz, mert a két motor soha nem teljesen egyforma (tekercsellenállás, belső súrlódás, fogaskerék-játék eltérései). Egy szoftveres kompenzációs szorzóval (kb. 0,96 a bal oldalra) a drift lényegesen csökkenthető.

7.3 K3 — Sebesség–PWM karakterisztika

Egy 150 cm hosszú szakaszon mérjük, mennyi idő alatt halad át a robot fix PWM-en. Minden PWM értéken 5 mérésből számolt átlagot adtunk meg (5. táblázat).

PWM	Idő 150 cm-re (s)	Sebesség (m/s)	Megjegyzés
50	—	0	halott zóna
100	4,7	0,32	
150	2,85	0,53	
200	2,0	0,74	
255	1,4	1,05	maximum

5. táblázat K3 — Sebesség–PWM karakterisztika 150 cm-es szakaszon, 5 teszt átlaga.

A maximális sebesség kb. 1 m/s (3,6 km/h), beltéri használatra elfogadható. A halott zóna PWM < 75, azaz autonóm módban legalább 80-as PWM-mel érdemes indulni. PWM > 100 felett közelítőleg lineáris az összefüggés: minden 25 PWM-nyi növekmény kb. 0,1 m/s-t hoz.

7.4 K4 — Fordulás pontosság

A helybeni fordulás pontosságához nem készítettünk külön táblázatos mérést, mert szemmel és a padlón elhelyezett tárgyakkal való jelölés alapján egyértelműen megállapítható volt, hogy a robot mindkét irányba egyhelyben forog — a skid-steer kinematika megfelelően működik, nincs szisztematikus sodródás a forgás során. A pontos szögbeállítás (pl. „fordulj pontosan 90°-ot") későbbi finomítást igényel egy IMU (giroszkóp) visszacsatolással, ez a jövőbeli fejlesztések része.

7.5 K5 — CAN busz megbízhatóság

A CAN busz megbízhatósága a gyakorlatban tökéletesnek bizonyult: a jel gyorsan és megbízhatóan jut el egyik csomópontból a másikba. Nem tapasztaltunk érzékelhető

késleltetést vagy üzenet-elvesztést a parancsok és a motorok reakciója között, sem álló, sem mozgó robotnál. A CAN szabvány ipari szintű hibaérzékelése és a differenciális jeltovábbítás a motor-EMI mellett is működik — ez az egyik legfontosabb előnye, amit a tervezéskor vártunk, és a gyakorlatban is beigazolódott.

7.6 K6 — Fékezési távolság

Két fékezési mód: A) PWM 0 (kigurulás), B) elektromágneses fék aktív. Négy PWM szinten (100, 150, 200, 255), 5 ismétléssel. Az eredményeket a 6. táblázat tartalmazza.

PWM	Fék nélkül (cm)	E-fékkel (cm)	Különbség
100	18	6	−12 cm
150	32	10	−22 cm
200	48	15	−33 cm
255	65	22	−43 cm

6. táblázat K6 — Fékezési távolságok.

A kerék nem egyből áll meg, hanem lassítani kezd, ezzel elkerülve a csúszást. Csúszás esetén is hasonló a féktáv, kb. 3-4 cm-rel több, de ez a talajtól is függ. Az elektromágneses fék a kigurulási távolságot harmadára-negyedére csökkenti. Maximális sebességen is 20-23 cm alatt megáll — átlagos szobaméretben ez biztonságos vészleállás.

8. Eredmények és további fejlesztési irányok

8.1 Mit sikerült megvalósítanunk

A humanoid robot mobil platformja elkészült, robosztus, elosztott, CAN-buszos vezérléssel. A robot PS2 kontrollerrel és előre programozott autonóm parancsokkal egyaránt irányítható. A Hall-szenzor alapú odometria ipari EMI-szűrés után 1 m-en 2 cm alatti hibával működik, a CAN kommunikáció gyakorlati tesztben megbízhatónak bizonyult, a fékezési távolság maximális sebességen e-fékkel 20-23 cm. A platform a 7. fejezetben bemutatott mérések szerint alkalmas beltéri használatra és biztonságos.

8.2 Következő lépések

A jelenlegi 4-Arduino rendszer működőképes, de alacsonyabb szintű egy érettebb platformnál elvárhatónál. Utódarchitektúra-tervünkben a központi vezérlő egy STM32 NUCLEO-H755ZI-Q [19] (beépített CAN-FD, dual-core M7+M4), a per-aktuátor csomópontok pedig STM32G431-esek (beépített FDCAN). Ezzel a chipek száma csökken (az STM32-ben már benne van a CAN), a sebesség nő, és ipari valós idejű garanciákat adhat a rendszer.

8.3 Köszönetnyilvánítás

Konzulenseinknek, Zsiga Bence Sándornak és Elekes Péternek, akik végigkísérték a projektet és sokszor vezettek át olyan pillanatokon, amikor egyedül nem értettük a hibát. Az Enterpartner Kft.-nek a műhely és a háttér biztosításáért. A Széchenyi István Műszaki Technikum tanárainak, akik rugalmasan kezelték, hogy a dolgozat írása közben az iskolai kötelezettségeket is teljesítenünk kellett.

9. Összefoglaló

Dolgozatunkban egy REAL 6100 Plus típusú elektromos kerekesszék átalakításával létrehozott humanoid robot mobil platformját mutattuk be. A gyári, zárt vezérlőelektronikát eltávolítottuk, a két 24 V-os hajtómotort egy Cytron SmartDriveDuo-60 motorvezérlőre kötöttük, és a magasabb szintű vezérlést saját elosztott, CAN-buszos architektúrára bíztuk. Négy Arduino Nano dolgozik a rendszerben: kettő a bal és jobb motort (és az odometriát) kezeli, egy CAN hídként (MCP2515 + SN65HVD230), egy pedig a PS2 vezeték nélküli kontrollert olvassa.

A fejlesztés során több jelentős problémát oldottunk meg: az MDDS60 AN1/AN2 hibás bekötéséből eredő single-channel módot, az Arduino kiégését okozó földhurkot, és a motor-EMI által zajosított Hall-szenzor mérést. Mindhárom mérnöki eszközökkel (adatlap-alapú bekötés, galvanikus leválasztás, RC szűrés és csillagpontos földelés) kezelhető volt.

Hat mérési sorozattal számszerűen is jellemeztük a platformot: odometria pontosság (±1,4 cm / 1 m), egyenes haladás drift (~5-7 cm / 150 cm), maximális sebesség (~1 m/s), fordulás kvalitatív tesztje (mindkét irányba egyhelyben forog), CAN kommunikáció gyakorlati megbízhatósága, fékezési távolság (max sebességen e-fékkel 20-23 cm). Az eredmények azt mutatják, hogy a platform beltéri használatra alkalmas és biztonságos.

10. Summary

This thesis presents the development of a mobile platform for a humanoid robot, created by repurposing a REAL 6100 Plus electric wheelchair. The factory closed-source control electronics were removed, the two 24 V drive motors were connected to a Cytron SmartDriveDuo-60 motor controller, and the higher-level control was implemented as a custom distributed CAN-bus architecture. Four Arduino Nano microcontrollers operate in the system: two drive the left and right motors (and read the Hall-sensor odometry), one acts as a CAN bridge (MCP2515 + SN65HVD230), and one reads the PS2 wireless game controller.

Several significant challenges were overcome during development: the MDDS60 entering single-channel mode due to incorrect AN1/AN2 wiring, Arduino damage caused by ground loops between USB and battery supplies, and Hall-sensor readings corrupted by motor electromagnetic interference. All three were resolved using standard engineering practices.

Six measurement series were conducted to quantitatively characterize the platform: odometry accuracy (±1.4 cm per 1 m), straight-line drift (~5-7 cm over 150 cm), maximum speed (~1 m/s), qualitative turning test (rotates in place in both directions), CAN communication reliability in practice, and braking distance (20-23 cm at maximum speed with electromagnetic brake). The results demonstrate that the platform is suitable for indoor use and safe.

11. Irodalomjegyzék

[1] SAE International, „J3016 — Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles”, SAE Standard, 2021.
[2] International Organization for Standardization, „ISO 11898-1:2015 — Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling”, ISO, Geneva, 2015.
[3] Robert Bosch GmbH, „CAN Specification, Version 2.0”, Bosch, 1991.
[4] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, „Introduction to Autonomous Mobile Robots”, 2nd ed., MIT Press, 2011.
[5] B. Siciliano and O. Khatib (eds.), „Springer Handbook of Robotics”, 2nd ed., Springer, 2016.
[6] Cytron Technologies, „SmartDriveDuo-60 (MDDS60) User's Manual”, Rev 1.0, 2020.
[7] Microchip Technology Inc., „MCP2515 — Stand-alone CAN Controller with SPI Interface”, Datasheet DS20001801J, 2019.
[8] Texas Instruments, „SN65HVD230 — 3.3-V CAN Bus Transceiver”, Datasheet SLLS557, 2020.
[9] Allegro MicroSystems, „49E Series — Linear Hall-Effect Sensor”, Datasheet, 2018.
[10] ams AG, „AS5600 — 12-Bit Programmable Contactless Potentiometer”, Datasheet, 2018.
[11] J. G. Ziegler and N. B. Nichols, „Optimum Settings for Automatic Controllers”, Trans. ASME, vol. 64, pp. 759–768, 1942.
[12] M. Quigley et al., „ROS: an open-source Robot Operating System”, in ICRA Workshop on Open Source Software, 2009.
[13] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, „Robot Operating System 2: Design, architecture, and uses in the wild”, Science Robotics, vol. 7, no. 66, 2022.
[14] A. K. Pandey and R. Gelin, „A Mass-Produced Sociable Humanoid Robot: Pepper”, IEEE Robotics & Automation Magazine, vol. 25, no. 3, pp. 40–48, 2018.

[15] T. Yamamoto et al., „Development of Human Support Robot as the research platform of a domestic mobile manipulator”, ROBOMECH Journal, vol. 6, no. 1, 2019.
[16] Open Robotics, „TurtleBot — ROS Wiki”, http://wiki.ros.org/Robots/TurtleBot (hozzáférés: 2026. április).
[17] Willow Garage, „PR2 Robot Overview”, https://robots.ieee.org/robots/pr2/ (hozzáférés: 2026. április).
[18] B. Porter, „PS2X Arduino Library”, https://github.com/madsci1016/Arduino-PS2X.
[19] STMicroelectronics, „STM32H755ZI — High-performance MCU”, Datasheet DS12294, 2023.
[20] NVIDIA Corporation, „Jetson AGX Orin Developer Kit — Technical Specifications”, https://developer.nvidia.com/embedded/jetson-agx-orin.
[21] C. J. Fowler, „Arduino MCP_CAN Library”, https://github.com/coryjfowler/MCP_CAN_lib.