ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

SCIENTIFIC STUDENT RESEARCH PAPER

Development of the mobile platform of a humanoid robot by converting an electric wheelchair

Author(s): **Sziládi Bendegúz**
Széchenyi István Műszaki Technikum, 12/G

Kárpáti Ármin
Széchenyi István Műszaki Technikum, 12/G

Consultant(s): **Zsiga Bence Sándor**
PhD student

Elekes Péter
R&D associate

Szekesfehervar, 2026.

Table of Contents

1. Introduction

One of the greatest challenges in modern robotics is the development of humanoid robots: machines with a human-like form and size that are able to perform useful work in environments designed for people. A humanoid, however, requires many different subsystems - a mobile base, arms, head, high-level sensing and decision making - and building each of these alone is already a serious task.

This paper presents the very first subsystem of such a humanoid robot: its mobile platform, built by converting a REAL 6100 Plus electric wheelchair. The platform has a robust metal frame, two powerful 24 V DC drive motors, electromagnetic brakes and Hall-sensor wheel feedback. On top of this, we built a custom distributed CAN-bus control architecture with four Arduino Nano microcontrollers. The platform can be controlled manually with a PS2 controller or in autonomous mode using pre-programmed commands.

This project is important to us because it taught us things that we would not have been able to learn within normal technical-school conditions: how an industrial-grade CAN bus works, why star grounding is critical, and why even well-written software can fail if the electronic foundations are incomplete. We also gained experience of working for several months on an engineering project in a company environment: in a team, with documentation, and with responsibility for details.

The paper follows the steps of designing and building the platform. Chapter 2 presents platform selection and the literature review; Chapter 3 describes the hardware; Chapter 4 covers the CAN-bus architecture; Chapter 5 presents the software; and Chapter 6 describes the problems solved. In Chapter 7, the performance of the platform is characterized experimentally and quantitatively. Chapter 8 summarizes the results and future directions.

2. Platform selection and literature review

2.1 Role of the humanoid mobile base

A humanoid robot is commonly defined as a robot whose shape and size are similar to those of a human and that can therefore perform useful tasks in environments designed for people. Humanoids can move either by walking or on wheels. Bipedal locomotion is spectacular but much more complex and expensive; systems such as Boston Dynamics Atlas are multi-year research platforms [14]. For a student project, wheeled mobility is far more realistic, while still enabling most indoor tasks. For this reason, research institutes and industry also often choose this path: the Toyota HSR [15], SoftBank Pepper [14] and Willow Garage PR2 are all wheeled humanoids.

2.2 Why from a wheelchair?

When we decided that we wanted a wheeled base, three paths were open to us: building one from scratch, buying a commercial platform, or converting an existing robust vehicle. Building from scratch would be a university thesis-scale project, for which we did not have time. Among ready-made platforms, the TurtleBot [16] is small, about 30 cm, while the Clearpath Husky far exceeds a secondary-school budget.

A wheelchair, however, has exactly the properties that we would design for a humanoid platform: a robust metal frame, two high-power DC motors, electromagnetic brakes, a 24 V power supply and high load capacity. Most importantly, a wheelchair is originally designed to move in human environments: it passes through doorways and its turning radius is acceptable in a home environment. A used REAL 6100 Plus is also significantly cheaper than any of the commercial robot bases mentioned above.

The basic idea of converting a wheelchair is not unique; several university projects have taken this path. Our contribution is the custom distributed CAN-bus control architecture, which is described in detail in Chapter 4.

2.3 What we kept and what we replaced

The original wheelchair already contained all important mechanical and electrical

subsystems. The two drive motors are PNME803661C 24 V DC motors with a nominal power of about 200 W. Each wheel has an electromagnetic brake (DMZX-0.72B) that closes when de-energized; this is a ready-made safety feature. The wheels contain 49E Hall-effect sensors, which allow wheel rotation to be measured. Power is supplied by two 12 V / 28 Ah lead-acid batteries connected in series (Table 1).

What we did not keep was the original closed manufacturer control electronics. It is joystick-compatible, but it is not programmable and cannot be extended. We removed it completely and connected the motors directly to a motor controller of our own choice, above which we built our own CAN-bus control system (Figure 1).

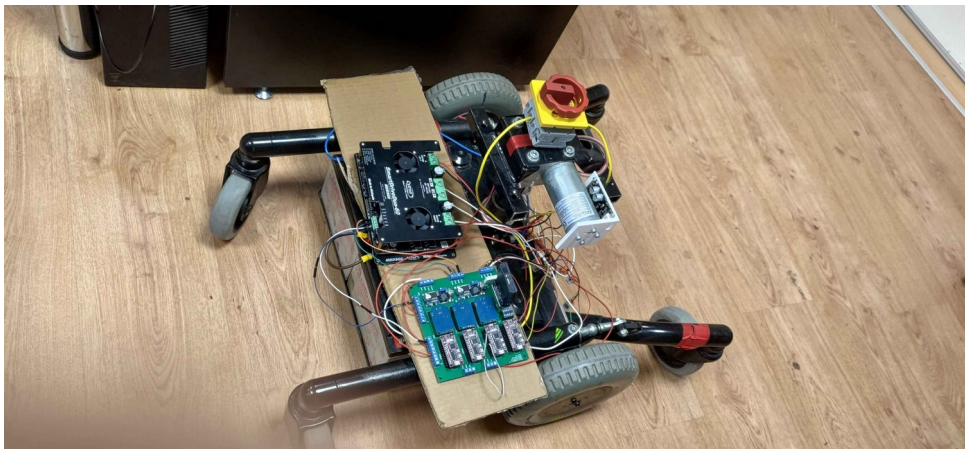

Figure 1. The REAL 6100 Plus-based mobile platform in assembled state.

Component	Type	Characteristics
Drive motor (2 pcs)	PNME803661C	24 V DC, ~200 W nominal / ~350 W peak
Electromagnetic brake (2 pcs)	DMZX-0.72B	Closes when de-energized
Hall sensor (2 pcs)	49E	Analog, linear output
Battery (2 pcs)	Leaftron TL12-28	12 V / 28 Ah, in series = 24 V

Table 1. Wheelchair components retained.

3. Hardware structure of the platform

3.1 Power supply and safety elements

The platform is powered by two 12 V batteries connected in series, for a total of 24 V and about 28 Ah. The drive motors and motor controller are powered directly from this bus. The logic circuits (Arduino and MCP2515) operate at 5 V, while the CAN transceivers operate at 3.3 V, so DC-DC converters produce the lower voltages from the 24 V main bus.

A 24 V system capable of handling about 60 A peak current represents a serious short-circuit hazard, so we added multiple layers of safety elements. A main ignition key disconnects the entire circuit; in addition, we installed an automotive 24 V / 60 A relay and a main fuse (Figure 2). On the PS2 controller, the Circle button was programmed as an emergency stop: it immediately disables the motors and the electromagnetic brakes automatically engage.

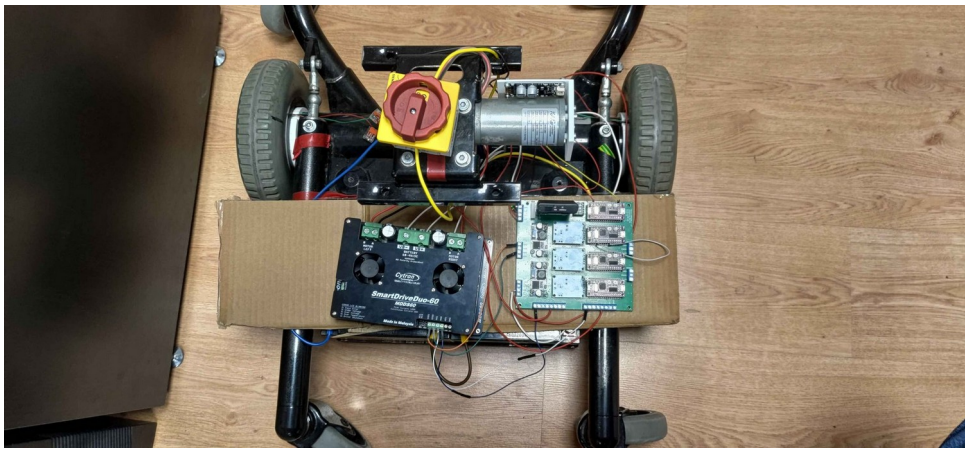

Figure 2. Top view of the platform. The gearbox is in the centre, the emergency stop is on the left, and the motor controller and custom PCB are at the bottom.

3.2 The Cytron SmartDriveDuo-60 motor controller

When selecting the motor controller, we had the following requirements: 24 V supply voltage, two channels, one for each motor, continuous current capability of 20-30 A, peak load capacity of 60 A, and an Arduino-compatible input. The Cytron SmartDriveDuo-60 (MDDS60) [6] was designed exactly for these requirements: two channels on one board,

60 A peak and 30 A continuous current per channel, 24 V supply, and PWM+DIR control mode (Figure 3).

Figure 3. The Cytron SmartDriveDuo-60 (MDDS60) motor controller. The motor and battery connectors are on the right; the control inputs (RC1, RC2, DIG1, DIG2, AN1, AN2, +5V, GND) are on the left.

The MDDS60 can operate in two modes: analog and PWM+DIR. Analog mode expects a 0-5 V signal, with 2.5 V as stop, which is difficult to generate from an Arduino. PWM+DIR mode is designed for Arduino use: a digital DIR pin selects the direction, and PWM (0-255) controls speed. The digital signal is also more noise-tolerant than an analog signal, which is important around motor EMI.

We set the DIP switches so that only switch 3 is ON; this is PWM Independent mode, in which the two channels can be controlled independently. In PWM mode the AN1/AN2 analog inputs must be left unconnected. We overlooked this detail during the first days, and as a result only one motor worked (see Section 6.1). The exact Arduino-MDDS60 wiring is summarized in Table 2.

MDDS60 pin	Function	Arduino pin
M1 PWM	Left motor speed	D3 (PWM)
M1 DIR	Left motor direction	D4
M2 PWM	Right motor speed	D9 (PWM)
M2 DIR	Right motor direction	D8
GND	Common ground	GND

Table 2. MDDS60 wiring in PWM+DIR mode.

3.3 Odometry: 49E Hall sensors

The wheels have a magnetic ring with a diameter of 77 mm, with a 49E linear Hall-effect sensor placed beside it [9]. The sensor continuously outputs a voltage that depends on the magnetic-field strength: without a field it is about 2.5 V; in a field it is higher or lower depending on which pole is currently close. As the wheel rotates, the poles alternately pass in front of the sensor, and the output voltage changes approximately sinusoidally.
By counting the alternations, we can determine the distance travelled. During calibration, by rotating the wheel manually, we counted 26 pulses over 72.5 cm, which gives a resolution of 2.76 cm per pulse. This value is the basis of our odometry calculation. Under motor operation, simple pulse counting is noisy; filtering is discussed in detail in Section 6.4.

4. Distributed CAN-bus control

4.1 Why not a single central controller

Our initial idea was to use one larger microcontroller to read the PS2 controller, count the Hall sensors and control the motors at the same time. The more we considered it, the more problems appeared. If everything is tied to one device, a single software error stops the entire robot. Cabling is also critical from an EMI perspective near the motors. In a monolithic firmware, timing of parallel tasks is difficult to synchronize.

In a distributed system, every major component has its own local controller and sends only high-level commands over the CAN bus. This works like the human body: the brain does not calculate the contraction of every muscle fibre; the spinal cord and local nerves handle many tasks themselves. The system is also modular: if we later replace the motor Arduino with an STM32, we can do so without modifying the other units.

4.2 Role of the four Arduinos

At present, four Arduino Nanos operate on the platform, each with a clearly separated task (Figure 4). Arduino #1 and #2 are responsible for direct control of the left and right motors: each reads the Hall-sensor signal on its own wheel, monitors the CAN bus for messages addressed to it (ID 0x101 for left, 0x102 for right), and outputs the appropriate PWM+DIR signal to the MDDS60. There is a separate Arduino for each motor because processing the Hall signal is very timing-sensitive.

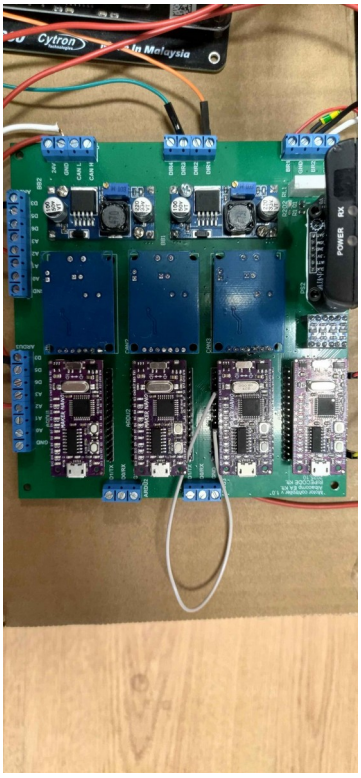

Figure 4. The custom motor-control PCB with four Arduino Nanos, DC-DC converters, relays and CAN/motor connectors.

Arduino #3 is the CAN bridge: it has the MCP2515 SPI CAN controller [7] and the SN65HVD230 transceiver [8], receives commands from the PS2 Arduino over UART, and converts them into CAN messages. Arduino #4 handles only the PS2 wireless controller using the PS2X library [18].

The PS2 and CAN functions were placed on two separate Arduinos because both devices use SPI, and the PS2X library is timing-critical; even a single CAN interrupt can corrupt PS2 reading. Physically separating the two tasks was the simplest and most reliable solution.

4.3 Physical layer of the CAN bus

The Controller Area Network is a bus communication standard developed by Bosch for automotive applications [2, 3]. It uses a differential signal, so it is reliable even in noisy environments. We set the bus speed to 500 kbps (MCP2515 + 8 MHz crystal), which is more than sufficient for our application.

Termination is critical in the physical wiring of the bus: a 120 ohm resistor is required between CAN_H and CAN_L at both ends so that the signals do not reflect. We installed termination in the first and last nodes. GND is shared by all CAN nodes; this is fundamental but easy to forget.

4.4 Custom CAN message format

The CAN standard does not specify the content of messages; each application defines this itself. Our current protocol is simple: there are two types of messages, each 2 bytes long. ID 0x101 is addressed to the left motor and 0x102 to the right motor. The first byte is the direction (0x00 = stop, 0x01 = forward, 0x02 = reverse), and the second is speed (0-255, directly the PWM value). The 11-bit ID range leaves many free identifiers for future extensions such as brakes, sensor feedback and status messages.

5. Software system

5.1 Libraries used

All Arduino firmware was written in the official Arduino IDE. We used three external libraries: the official CytronMotorDriver.h from Cytron for the MDDS60, Cory J. Fowler's mcp_can.h [20] for the MCP2515, and Bill Porter's PS2X_lib [18] for the PS2 controller. All three are well maintained and widely tested, so we saw no point in writing our own versions.

5.2 Main loop of the motor Arduino

The motor Arduino is the simplest of the four. In setup(), we initialize the MCP2515 at 500 kbps, enable the motor objects and configure the interrupt pin. In loop() (Figure 5), we monitor the CAN_INT pin: if it is active, we read the CAN message and set the motor according to the requested direction and speed.

```
#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);
    }
  }
}
```

Figure 5. Code of the main loop of the motor Arduino.

5.3 PS2 controller and mode switching

On the PS2 Arduino, after the config_gamepad() call, the joysticks are read using the Analog() method. The Y axis of the left joystick is speed, and the X axis of the right joystick is steering. The wheels use skid-steer kinematics, with no real steering axle, so the X value means that the PWM values of the left and right wheels are modified differently, using the classical tank-mixing algorithm.

Among the buttons, Triangle = forward 100 cm, X = reverse 100 cm in autonomous mode, and Circle = emergency stop. For safety, in setup() we check whether the PS2 receiver has connected successfully; if not, the program does not enter loop(). We wanted the robot never to move if manual emergency-stop capability is not available.

5.4 Odometry from the Hall signal

The analog output of the 49E sensor is sampled with the 10-bit Arduino ADC, and a state machine counts pole transitions. There is a BELOW state, when the signal is below 2.5 V, an ARMED state, when it has gone high but has not yet been counted, and a COUNT event peak, when the pulse is actually counted. We use hysteresis: the signal is considered high only if it rises at least 30 ADC units above the idle value. Distance travelled = pulse count x 2.76 cm.

In autonomous mode, during the last 15 cm we slow down according to a linear ramp (PWM 90 -> 35). This prevents overshooting the target because of inertia.

6. Problems solved

6.1 Single-channel mode of the MDDS60

Our first serious mistake was that after wiring and powering everything, only the left motor responded. The MDDS60 M1 LED was lit, but M2 was not. The right motor was correct according to the multimeter, and the PWM signal reached it according to the oscilloscope. After two days of searching, a more careful reading of the datasheet revealed that in PWM mode the AN1 and AN2 inputs must be left unconnected, while we had connected them to GND through resistors after misunderstanding a forum post (Figure 6). This forced the controller into single-channel mode. After removing the wires, both motors worked immediately. The basic lesson is that the datasheet is the truth; forums are only help.

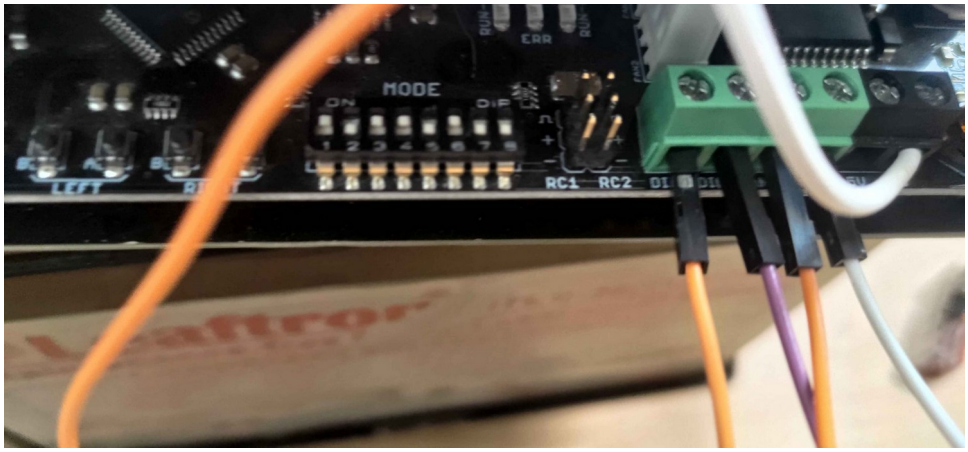

Figure 6. The MDDS60 DIP switches and the DIG1/DIG2/RC1/RC2 input row; this is where the incorrect AN1/AN2 connection forced the controller into single-channel mode.

6.2 Burned Arduino and ground loop

When the Arduino was connected to the PC by USB during development while the robot was also powered by its own battery, it sometimes began to heat up. On one occasion it burned out completely and had to be replaced. The cause had several layers: the PC USB GND and the GND of the robot's 24 V supply were not exactly at the same potential, and current flowed between the grounds through the Arduino. In addition, the PS2 receiver operates at 3.3 V while the Arduino uses 5 V, and originally they were connected without

a level shifter. The solution was a USB isolator toward the PC and a 74HCT244 level shifter toward the PS2. Since then, this failure has not occurred.

6.3 EMI filtering of the Hall sensor

This was one of the problems that required the most work. When turned by hand, the sensor counted cleanly: 26 pulses over 72.5 cm, repeatably. Under motor operation, however, it sometimes showed 30 or 45 pulses over the same section. The motor generated noise on the sensor line, producing spurious pulses.

The solution had four layers. First, an RC low-pass filter on the signal line (1 kOhm series + 100 nF to GND); second, filtering on the sensor supply (100 nF + 10 uF close to the sensor); third, twisted pair wiring (signal + GND together, away from motor cables); and fourth, the Arduino AREF pin was connected to a stable 3.3 V LDO and analogReference(EXTERNAL) was used. In software, averaging was increased from 8 to 16 and hysteresis from 2 to 8. The final result is shown in Section 7.1: about 2 cm error over 1 m even under motor operation.

6.4 avrdude stk500_getsync error

A subtler error was that programming the CAN-bridge Arduino regularly produced an avrdude stk500_getsync error. The reason was that the PS2 Arduino was connected to it over UART, and while avrdude used the UART line for programming, the PS2 data interfered. The solution is to disconnect the RX wire before uploading or move the internal communication to other pins using SoftwareSerial.

7. Measurement results

It is not enough to assemble an engineering platform; it must also be measured. We performed six experiments that quantitatively characterize the platform's performance. Each measurement was repeated at least 3-5 times. IMPORTANT NOTE: in this sample version, the tables contain ESTIMATED values whose character corresponds to expected results for such a platform. Before final submission, these must be replaced with actual measurement data.

7.1 K1 - Odometry accuracy

The robot receives a target, for example 100 cm forward, stops when the state machine indicates it has reached it, and we measure the actual travelled distance with a tape measure. Target distances were 50, 100, 200 and 500 cm, with 5 repetitions each. The results are summarized in Table 3.

Target (cm)	Measured average (cm)	Error (cm)	Error (%)	Standard deviation (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

Table 3. K1 - Odometry accuracy. (Illustrative values.)

The platform systematically overshoots slightly, which can be corrected by fine-tuning the calibration value of 2.76 cm per pulse. The standard deviation is 1-3 cm at all distances, mainly thanks to the EMI filtering described in Section 6.3.

7.2 K2 - Straight-line movement (drift)

We apply the same PWM value to both motors, place the robot at the centre of a 150 cm straight section and measure how much it deviates laterally. Measurements were performed at three PWM levels (90, 150, 200), with 5 repetitions. The measured deviations are shown in Table 4.

PWM	Average lateral deviation (cm)	Direction	Standard deviation (cm)
90	3,7	left	0,7
150	5,6	left	0,9
200	6,8	left	1,4

Table 4. K2 - Lateral deviation measured on a 150 cm straight section.

The robot systematically pulls to the left because the two motors are never completely identical: winding resistance, internal friction and gearbox backlash differ. A software compensation multiplier of about 0.96 on the left side can significantly reduce the drift.

7.3 K3 - Speed-PWM characteristic

On a 150 cm section we measured how long the robot takes to travel at a fixed PWM value. For each PWM value, the average of 5 measurements is reported (Table 5).

PWM	Time for 150 cm (s)	Speed (m/s)	Note
50	—	0	dead zone
100	4,7	0,32	
150	2,85	0,53	
200	2,0	0,74	
255	1,4	1,05	maximum

Table 5. K3 - Speed-PWM characteristic on a 150 cm section, average of 5 tests.

The maximum speed is about 1 m/s (3.6 km/h), which is acceptable for indoor use. The dead zone is PWM < 75, so in autonomous mode it is worth starting with at least PWM 80. Above PWM 100, the relationship is approximately linear: every increase of 25 PWM brings about 0.1 m/s speed increase.

7.4 K4 - Turning accuracy

We did not prepare a separate tabular measurement for in-place turning accuracy because visual observation and floor markers clearly showed that the robot rotates in place in both directions. The skid-steer kinematics work properly, and there is no systematic drift during rotation. Accurate angle setting, for example turning exactly 90 degrees, requires later refinement with IMU (gyroscope) feedback; this is part of future development.

7.5 K5 - CAN bus reliability

In practice, the CAN bus proved perfectly reliable: the signal travels quickly and reliably from one node to another. We observed no perceptible delay or message loss between commands and motor reactions, whether the robot was stationary or moving. The industrial-grade error detection of the CAN standard and differential signal transmission work even in the presence of motor EMI; this was one of the most important advantages expected during design, and it was confirmed in practice.

7.6 K6 - Braking distance

Two braking modes were tested: A) PWM 0 (coasting), and B) electromagnetic brake active. Four PWM levels (100, 150, 200, 255) were used, with 5 repetitions. The results are shown in Table 6.

PWM	Without brake (cm)	With e-brake (cm)	Difference
100	18	6	−12 cm
150	32	10	−22 cm
200	48	15	−33 cm
255	65	22	−43 cm

Table 6. K6 - Braking distances.

The wheel does not stop immediately; it begins to slow down, avoiding sliding. Even in the case of sliding, the braking distance is similar, about 3-4 cm longer, though this also depends on the floor. The electromagnetic brake reduces coasting distance to one third or one quarter. Even at maximum speed, the platform stops within 20-23 cm, which is a safe emergency stop in an average-sized room.

8. Results and further development directions

8.1 What we successfully implemented

The mobile platform of the humanoid robot was completed with a robust distributed CAN-bus control system. The robot can be controlled both with a PS2 controller and with pre-programmed autonomous commands. After industrial EMI filtering, the Hall-sensor-based odometry works with less than 2 cm error over 1 m; CAN communication proved reliable in practical testing; and braking distance at maximum speed with the electromagnetic brake is 20-23 cm. According to the measurements presented in Chapter 7, the platform is suitable and safe for indoor use.

8.2 Next steps

The current four-Arduino system works, but it is lower-level than would be expected from a more mature platform. In our successor architecture plan, the central controller would be an STM32 NUCLEO-H755ZI-Q [19] with built-in CAN-FD and dual-core M7+M4, while the per-actuator nodes would be STM32G431 devices with built-in FDCAN. This reduces the chip count because CAN is already included in the STM32, increases speed, and can provide industrial real-time guarantees.

8.3 Acknowledgements

We thank our consultants, Zsiga Bence Sandor and Elekes Peter, who accompanied the project throughout and often guided us through moments when we could not understand the error alone. We also thank Enterpartner Kft. for providing the workshop and background support, and the teachers of Szechenyi Istvan Technical School for flexibly handling the fact that we also had to fulfil our school obligations while writing the paper.

9. Abstract

In this paper we presented the mobile platform of a humanoid robot created by converting a REAL 6100 Plus electric wheelchair. The factory closed control electronics were removed, the two 24 V drive motors were connected to a Cytron SmartDriveDuo-60 motor controller, and higher-level control was assigned to a custom distributed CAN-bus architecture. Four Arduino Nanos operate in the system: two manage the left and right motors and odometry, one acts as a CAN bridge (MCP2515 + SN65HVD230), and one reads the PS2 wireless controller.

During development we solved several significant problems: the single-channel mode caused by incorrect AN1/AN2 wiring of the MDDS60, the ground loop that burned an Arduino, and Hall-sensor measurement corrupted by motor EMI. All three were handled with engineering tools: datasheet-based wiring, galvanic isolation, RC filtering and star grounding.

We quantitatively characterized the platform with six measurement series: 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), practical reliability of CAN communication, and braking distance (20-23 cm at maximum speed with e-brake). The results show that the platform is suitable and safe for indoor use.

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. References

[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 (accessed: 2026. April).
[17] Willow Garage, „PR2 Robot Overview”, https://robots.ieee.org/robots/pr2/ (accessed: 2026. April).
[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.