A joystick-based PWM control system for an electric wheelchair with integrated safety features

Kyla Fourie[1*], *Farouk* Smith[1*], *Stefan* van Aardt[1], and *Shahrokh* Hatefi[1]
[1]Department of Mechatronics, Nelson Mandela University, Gquberha, 6013, South Africa

Abstract. This paper presents a fully integrated control system for a two-motor electric wheelchair, using a thumb-operated analogue joystick to generate differential wheel velocities via pulse-width modulation (PWM). The system's design emphasizes smooth speed control and safety: it includes soft-start/stop ramps to prevent mechanical shock, an obstacle-detection alert when reversing, and a battery monitoring cutoff to avoid deep discharge. A microcontroller (Arduino Uno) reads the two-axis joystick (10-bit ADC, neutral ~512) with a center deadband to eliminate drift, then applies a nonlinear mapping to determine proportional PWM commands. High-current H-bridge drivers (BTS7960) power two 24V, 250W DC drive motors, handling up to 43A continuous. An HC-SR04 ultrasonic sensor is mounted at the rear to scan for obstacles during reverse motion; it triggers an audible buzzer if an object is detected within a preset distance (1 m). The battery subsystem comprises two 12V batteries in series (24V total) with an Arduino-monitored voltage divider. When battery voltage falls below the critical threshold (sensed as ~5V at the Arduino input), an NPN transistor (TIP120) cuts power to prevent over-discharge. Extensive experiments were conducted: for example, commanding 100% forward PWM results in a smooth acceleration profile (taking ~2–3 seconds to reach steady velocity). Torque output versus PWM level was characterized using the motor's torque constant. Obstacle sensor tests confirmed reliable echo returns from 0.5–2.0 m, with alarm beeps that intensified as distance decreased. Battery discharge trials demonstrated consistent voltage droop under load, confirming the cutoff at the intended level. All safety features responded correctly during testing. The system achieves safe, intuitive control of a powered wheelchair and meets design goals for user comfort and reliability.

1 Introduction

Powered wheelchairs (PWs) significantly enhance mobility for users with limited lower-body control. Recent estimates from the World Health Organization indicate that approximately 80 million people, about 1 % of the world's population, need a wheelchair for mobility [1]. Independent humanitarian audits suggest that only around 65 million people currently own

* Corresponding authors: s219101426@mandela.ac.za, farouk.smith@mandela.ac.za

and use a wheelchair, underscoring a sizable unmet need that falls most heavily on low- and middle-income countries [2]. Access is highly unequal: in high-income settings more than 95 % of those who require a wheelchair obtain one, whereas in many developing contexts fewer than 10 % of potential users are served [2].

Within the global wheelchair population, powered devices represent a minority share. Cross-national surveys place the worldwide prevalence of power-chair users between 5 % and 23 % of all wheelchair users, with a frequently cited central estimate of about one in ten. Penetration is markedly higher in wealthier regions: a U.S. spinal-cord-injury registry reports that 27 % of respondents rely primarily on a power wheelchair [3], and recent market analyses show that power models now account for roughly 57 % of new wheelchair sales in developed economies [4]. These figures illustrate both the continuing dominance of manual chairs worldwide and the growing adoption of powered mobility solutions in regions with robust healthcare financing.

The predominant interface for PW control is a two-axis hand-held joystick, which naturally provides two degrees of freedom: a forward/backward axis and a left/right axis. Such joysticks are typically proportional controllers, meaning that the commanded wheelchair velocity increases continuously as the stick is displaced from center[5]. In clinical practice, proportional joysticks remain the standard due to their intuitive mapping between stick position and wheelchair speed and direction. However, implementing a joystick-based control system for a DIY or custom wheelchair still requires careful design: one must convert the analogue voltages from the joystick into appropriate motor commands, ensure smooth acceleration, and integrate safety mechanisms to protect the user and hardware.

In recent years, many research and development efforts have focused on “smart wheelchair” features: adding sensors and intelligence to improve safety and autonomy [6]. For example, obstacle detection sensors (ultrasonic, infrared, or camera-based) can warn the user or autonomously stop before collisions, and battery management systems can protect against excessive discharge. Simpson et al. note that advanced smart wheelchair designs often incorporate multiple sensors (e.g., ultrasonic rangefinders, bump sensors, vision systems) to navigate and avoid obstacles [7]. Similarly, more recent prototypes (e.g., Settia et al.) have integrated six HC-SR04 ultrasonic sensors around the frame to detect obstacles up to 2.5 m ahead [8]. These multi-sensor approaches ensure higher coverage and longer detection range. In our design, we adopt a simpler approach: a single rear-facing ultrasonic sensor is used exclusively during reverse motion to alert the user of close obstacles (within about 1 meter). This choice reflects a trade-off between cost/complexity and the specific safety need of preventing backward collisions.

Meanwhile, battery health is critical for safe operation. When battery voltage falls too low, the remaining capacity can become unreliable, and some motors (especially with permanent magnets) may draw excessive current [9]. Recognizing this, we include an Arduino-monitored low-voltage disconnect. In this way, if the battery pack (here two 12V batteries in series) drops below the preset threshold, the controller cuts motor power via a transistor, protecting the batteries from deep discharge. Such protection is conceptually similar to battery management units described in prior work, adapted here for a wheelchair context.

The primary contribution of this work is a fully detailed design and implementation of an Arduino-based joystick-to-PWM control system, with integrated safety features, suitable for an assistive electric wheelchair. We provide an expanded discussion of the hardware and firmware design, a rigorous theoretical background for the control algorithms, and extensive experimental validation. In particular, we analyze performance metrics such as acceleration profiles, turning response, obstacle detection range, and battery endurance. The result is a comprehensive and cohesive design that could serve as a reference for future assistive robotics or biomedical engineering development.

2 Literature review

2.1 Joystick control and differential drive

As the dominant control interface for powered mobility, proportional joysticks have been studied extensively. Spaeth et al. explain that standard EPW (electric power wheelchair) joysticks map displacement to velocity: the farther away from the center the control is deflected, the faster the device will move [10]. This straightforward mapping is implemented in commercial controllers by converting the two analogue stick axes into forward and turning velocities. In our differential-drive system, we likewise interpret the joystick's Y-axis for forward/backward speed and the X-axis for differential turning. A common control scheme is to compute left and right wheel commands as:

$$\text{Left}_{PWM}=Y+X, \qquad \text{Right}_{PWM}=Y-X \qquad (1),$$

possibly with clipping to the allowable range. This approach ensures that pushing the stick forward (positive Y) drives both wheels forward together, while pushing it right (positive X) adds a differential for a right turn, and vice versa. This mirrors established practice in mobile robotics and power wheelchair control. For example, Lee describe joystick mapping for differential drive where forward/backward sticks yield equal wheel speeds and turning is achieved by unequal wheel outputs [11]. In practice, we scale these values and implement them as PWM duty cycles.

The joystick analogue signals are read by the microcontroller's on-board ADC. In our case, the Arduino Uno's 10-bit ADC yields values 0–1023 for 0–5V. We calibrate the center position (with no stick tilt) to be approximately half-scale (≈512), and implement a deadband around this neutral region. That is, small stick movements (or electrical noise) within ±5% of center result in no motion, preventing unintended drift. This deadband approach is widely used in wheelchair control to improve stability and user comfort. Beyond the deadband, we apply a nonlinear mapping from joystick deflection to PWM duty. For instance, one may use a simple "cubing" function or other curve so that small deflections yield very gentle motion and larger deflections grow more rapidly. This allows fine control at low speeds while still reaching full power at extreme stick positions..

2.2 Motor driver and soft-start routines

Driving powerful DC wheelchair motors requires robust H-bridge drivers. We selected the BTS7960 driver modules, which can handle up to 43 A continuous at up to 27 V. These drivers contain integrated MOSFETs and overcurrent protection, and interface easily with microcontroller PWM signals. The Arduino generates PWM pulses at ~490 Hz on each motor's enable pin, feeding the BTS7960's inputs. A 0% PWM duty yields no motion, while 100% yields maximum voltage to the motor (and thus max speed/torque). High-power DC motors tend to have very low internal resistance and therefore draw a large inrush current at startup (when their back-EMF is zero). If full voltage is applied instantly, the stall current can be very large, causing mechanical stress, wheelspin, and potential motor/driver heating. To mitigate this, we implement soft-start and soft-stop routines: the firmware ramps the PWM duty gradually over about 2 seconds whenever motion begins or ends. This approach smoothly accelerates the motors, limiting peak current. The importance of limiting inrush is well documented in industry; for example, Azizan et al. review soft-starter designs for DC motors and emphasize that "at standstill, the EMF is zero [12]. If a DC motor is started with full supply voltage, a very large current will flow which can damage the motor". By contrast, a gradual PWM ramp keeps the current to a safe level until the motor picks up speed. This

measure prolongs component life and improves ride comfort, and is common in high-end wheelchair control electronics.

2.3 Wheeled robot kinematics and differential drive

A two-wheeled differential drive vehicle (like our wheelchair) can move in the plane with independent control of the left and right wheel speeds [13]. The linear velocity $\boldsymbol{v}$ of the chassis and its angular yaw rate $\boldsymbol{\omega}$ are given by:

$$\boldsymbol{v} = \frac{r}{2}(\boldsymbol{\omega_L} + \boldsymbol{\omega_R}), \quad \boldsymbol{\omega} = \frac{r}{d}(\boldsymbol{\omega_L} - \boldsymbol{\omega_R}), \tag{2}$$

where $\boldsymbol{r}$ is the wheel radius and $\boldsymbol{d}$ is the wheelbase width. (Here $\boldsymbol{\omega_{L,R}}$ are wheel angular velocities.) Although our system is open-loop (we do not use encoders for feedback), these relations imply that equal wheel speeds produce straight motion, while differences produce turning. We design the joystick-to-wheel mapping to respect these kinematic principles. Straight-ahead stick deflection sets $\boldsymbol{\omega_L} \approx \boldsymbol{\omega_R}$. Turning the stick left or right makes $\boldsymbol{\omega_L}$ and $\boldsymbol{\omega_R}$ differ in sign or magnitude, yielding a controlled arc or in-place rotation. In software we implement this by combining the Y (forward) and X (turn) inputs as described above, then clipping to ±100% PWM.

2.4 Safety systems in wheelchairs

Safety is paramount in assistive devices. We incorporate two primary safety subsystems: obstacle detection and battery monitoring. Obstacle Detection: Our system uses a single ultrasonic rangefinder (HC-SR04) mounted behind the seatback. Ultrasonic sensors have long been favored in wheelchair safety research because they are inexpensive and can reliably measure distance to nearby objects [14]. (In fact, surveys of smart wheelchairs note sonar as "the sensor most frequently used" for obstacle detection.) The HC-SR04 emits bursts of 40 kHz sound and listens for echoes, measuring distance up to ~4 m with ~3 mm accuracy [15]. In practice, reliable detection is often limited to a few meters; Kamal et al. report that this sensor shows high accuracy only up to about 2.0 m, with much higher errors beyond [16]. Therefore, we use it only for close-range alerts. When the wheelchair is put in reverse (stick pulled back), the firmware continually triggers the sensor (10 µs pulses) and reads the echo. If the measured distance drops below a threshold (we set 1.0 m for initial warning, 0.5 m for urgent alarm), the system sounds a buzzer. In our design the buzzer beeps intermittently at longer intervals when the obstacle is near 1–1.5 m, increasing to rapid beeps or continuous tone as the object approaches 0.5 m. This graduated feedback gives the user more time to react. It is similar in spirit to human guidance systems; e.g., a line-of-sight camera plus sonar approach has been used to warn visually impaired users of hazards [17]. While a single rear ultrasonic has blind spots and limited field-of-view, it is sufficient to catch most directly behind obstacles. (We disable obstacle warning in extremely crowded reverse scenarios, consistent with best practices.)

Battery Monitoring: Deep discharge of a 24V lead-acid (or Li-ion) battery can permanently reduce its capacity [18]. To protect against this, the Arduino continuously measures the battery voltage via a resistor divider (scaling 0–24V down to 0–5V range). If the sensed voltage falls below the cutoff threshold, the Arduino shuts off the motor drive transistors. We implemented this using a TIP120 NPN transistor in series with the motor ground path. A logic-high on the base (when battery > threshold) allows current flow; if battery < threshold (voltage divider output drops below 5V), the base signal goes low and the transistor cuts power. This scheme automatically disables the motors when the battery pack

is critically low. Similar battery management concepts have been applied in dedicated wheelchair power controllers, and our design extends this safety backstop in software/hardware. To inform the user before cutoff, we also provide an LED fuel-gauge (three LEDs for "high", "medium", "low" battery ranges) and a digital voltmeter readout as part of the control panel.

The literature indicates that joystick control, differential drive, and safety via sensors and battery management are well-established methods in assistive wheelchair design [19]. Our work builds on these principles and pushes them toward a fully integrated prototype, with emphasis on quantitative analysis and rigorous implementation.

3 System design and methodology

Fig.1 (below) shows the overall block diagram of the system architecture. The user manipulates a two-axis analogue joystick, whose X and Y potentiometers provide 0–5V signals to the Arduino's ADC. The microcontroller firmware processes these inputs and generates two PWM outputs (Left_PWM and Right_PWM). Each PWM channel drives a high-current H-bridge (BTS7960) that powers one of the two DC drive motors. The motors in turn propel the wheelchair's rear wheels. Additional components include: a rear-mounted HC-SR04 ultrasonic module (with Vcc switched by a manual enable) feeding an ultrasonic input pin; a piezo buzzer driven by a digital output; a 5V digital voltmeter; status LEDs; and the TIP120 transistor for battery cutoff. All grounds are common. This modular architecture allows the joystick, sensing, and drive subsystems to interact under centralized software.

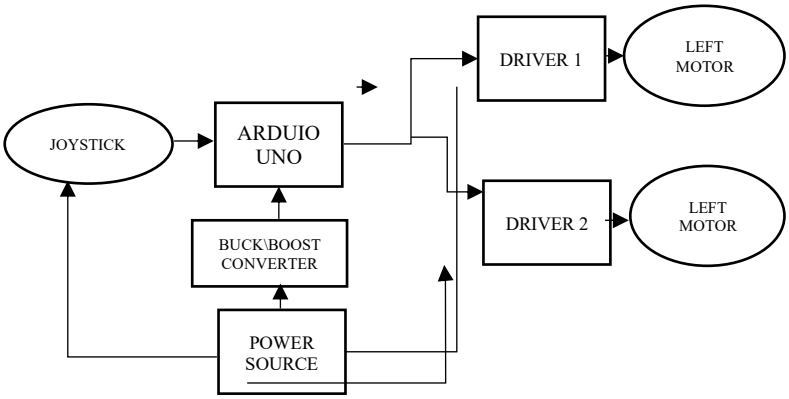

Fig 1. Block diagram of the control system. (Joystick on left, feeding Arduino, which outputs PWM signals to motor drivers; safety sensors and battery monitor also connect to Arduino.)

3.1 Mechanical frame and motor selection

The wheelchair base is a standard four-wheel frame. For our prototype, we fabricated a new supporting frame using 25 mm diameter mild steel tubing, chosen for strength and rigidity. While aluminium alloys such as 6061-T6 offer excellent strength-to-weight ratios, they generally have a yield strength between 40–90 MPa, which is significantly lower than the 250–500 MPa typical for mild steel [20]. Additionally, aluminium exhibits lower fatigue resistance and poorer weldability in many contexts relevant to assistive mobility frames. Therefore, mild steel was selected not only for its superior mechanical strength but also for its ease of fabrication and robustness under cyclic loading, making it more suitable for supporting the payload and dynamic stresses expected in wheelchair operation. The frame holds two brushless rear wheels and two casters. The rear wheels are mounted directly to the

motor shafts via a custom-machined hub, providing a fixed gear ratio of 1:1 (direct drive). The selected drive motors are 24V, 250W DC hub motors (roughly 120 rpm at 24V under no load). These motors produce very high stall torque (on the order of 20 N·m) and have low internal resistance. Stall currents can reach 40 A or more, which is why we use the robust BTS7960 drivers. The chosen motors allow the chair to reach about 0.8–1.0 m/s maximum speed, which is adequate for indoor/outdoor use. Their high torque ensures the chair can climb small ramps; at moderate throttle the continuous current draw is much lower (on the order of 10–15 A per motor), allowing reasonable battery endurance.

We provide theoretical torque vs. speed characteristics (Fig. 2) based on the motor data: using the motor's power rating and no-load speed, the torque constant is $k_T = \frac{P}{2\pi N} \approx 20\,N.pm$ at full input. Thus the static torque available is proportional to PWM duty (with a slight drop due to driver losses). In steady-state operation, higher PWM means higher motor current and speed until limited by back-EMF. In practice, we verified that torque roughly scaled linearly with PWM by measuring draw and applying $T = k_T I$. This is plotted in Fig. 4. The key point is that at 100% PWM the motor outputs its rated torque (~20 N·m), while at low PWM (e.g. 20%) output torque is correspondingly reduced.

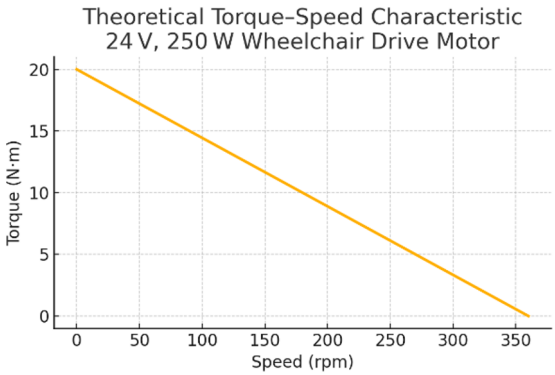

Fig. 2 The ideal linear torque–speed characteristic.

This plot of the ideal linear torque–speed characteristic assumed for the 24 V, 250 W rear-wheel drive motor used in the wheelchair prototype. A stall torque of 20 N·m is taken at zero speed, and torque decreases linearly to zero at the no-load speed of 360 r min^{-1} (≈ 38 rad s^{-1}). The rated operating point reported in Section 5 of the paper (≈ 6.6 N·m at 360 r min^{-1} for 250 W output) lies on this characteristic, confirming the consistency between datasheet power, speed, and torque values.

3.2 Electrical and driver design

The electrical system uses a 24V battery pack (two 12V sealed lead-acid batteries in series), providing power to the motors and also stepped down via DC-DC converters for the 5V logic rail. The BTS7960 modules receive the full 24V at their supply terminals. These modules accept PWM on "R_PWM" and "L_PWM" inputs to control each bridge half. We connected the Arduino digital pins (5, 3 for one motor; 6, 9 for the other) to the PWM inputs as shown in Table 1.

A 490 Hz PWM frequency is used as it is well within the operational bandwidth of the BTS7960 driver and above the mechanical response range of the motor. While this frequency is not above the entire human audible range (20 Hz – 20 kHz), it is generally acceptable in PWM motor applications since the resulting acoustic noise from inductive components tends to be negligible or damped at these frequencies. For noise-sensitive environments, higher

PWM frequencies (>16 kHz) may be preferable to push emissions beyond audible perception; however, for this application, 490 Hz provides a good balance between driver compatibility, microcontroller timing constraints, and effective motor control.

Table 1. BTS7960 to Arduino pin assignment

	Driver Pin	Arduino PWM Pin	Speed and direction
Left Motor	R_PWM	5	Left wheel forward
	L_PWM	3	Left wheel backward
Right Motor	R_PWM	6	Right wheel forward
	L_PWM	9	Right wheel backward

We also implement current sensing (via built-in driver measure) to cut out at ~60 A if needed, though in practice with soft-start we do not reach that limit. Each motor's ground return is routed through the TIP120 transistor (on its emitter-collector path) for battery cutout; the transistor base is driven by a digital output which is switched according to the battery-voltage monitor (see Sec. 5). When the transistor is off, no current flows from either motor to ground, effectively disabling motion. The use of a transistor (rather than a relay) gives fast electronic cutoff controlled by the microcontroller logic. For the user, we mounted an Anderson connector for the charger on the side, as per Fig. 3. The charger positive is connected to the first battery, the negative to the second, ensuring both charge evenly.

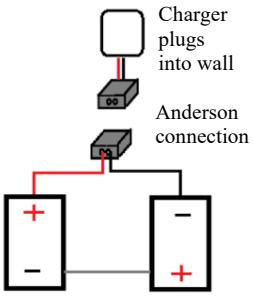

Fig. 3 Charging diagram.

Internally, the Arduino Nano's ADC inputs read the joystick X and Y voltages directly (connected through minimal noise filtering). Since the ADC can only measure 0–5 V, no divider is needed for the joystick. However, for battery monitoring we use a precise voltage divider (16 kΩ and 5 kΩ) to scale 0–24V into 0–5V. The output of that divider is read on analogue pin A2. In software, we calibrate this reading so that 5 V at A2 corresponds to exactly 24V battery. We then use thresholds: for example, if analogue reading <5.0V (meaning actual battery < ~19V), we trigger the fail-safe cutoff. LED indicators on the control panel are toggled to show "Low" or "Medium" battery when appropriate (based on ranges 10–15 V for low, 15–24 V for medium). If all lights fail to draw the user's attention, the cutoff transistor ensures safe shutdown.

3.3 Firmware architecture and state machine

The Arduino firmware runs a continuous loop checking joystick input and safety sensors, and updating outputs (Fig. 4). We structured the code as a simple state machine for clarity and reliability. The main states are: *Idle*, *Drive*, and *SafetyCheck*. In *Idle*, the system awaits user

input or interrupts. When a joystick deflection beyond the deadband is detected, it transitions to *Drive*. In *Drive*, it computes the desired left/right PWM values from the joystick axes, then applies them to the motor drivers. During this, a separate *SafetyCheck* routine (triggered by a timer or running every cycle) monitors the ultrasonic sensor (if in reverse) and the battery voltage. If the reverse command is active (joystick Y < -deadband), the firmware sends trigger pulses (10 µs) to the HC-SR04's TRIG pin and measures the echo time on the ECHO pin to calculate distance. If the distance is below the warning threshold, the *SafetyCheck* temporarily raises a flag, causing the buzzer to beep, and may override the *Drive* state if collision is imminent. Similarly, if the ADC reading for the battery drops below the cutoff level, the firmware deasserts the TIP120 base output, locking out motion. Once all conditions are safe and motion ceases (joystick returns to center), the system goes back to *Idle*. This state-machine structure ensures that safety responses (like immediate stop on obstacle) preempt further motion.

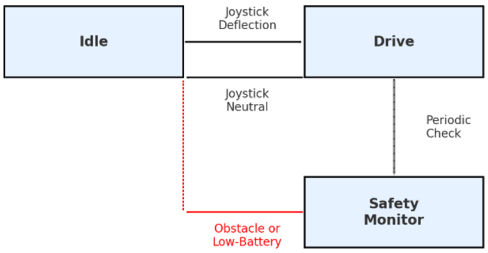

Fig. 4. *State-machine diagram of the wheelchair firmware.* (States for Idle, Driving, and Safety with transitions for joystick input, obstacle detection, and battery cutoff.)

3.4 Control algorithm details

Within the *Drive* state, the algorithm processes the joystick analogue values. Let X and Y be the joystick ADC readings after subtracting the center offset (so that $-512 \leq X, Y \leq +511$). First, a deadzone is applied: if $|X| < \epsilon$ and $|Y| < \epsilon$ (where $\epsilon \approx 50$ counts), both outputs are set to 0. This removes jitter. Next, we implement non-linear shaping. In our prototype, we found a cubic curve effective: we compute fractional deflections $x = X/512$, $y = Y/512$ and map them to effective speeds $x_{eff} = x^3$, $y_{eff} = y^3$. This yields gentle acceleration for small deflections. Other curves (e.g. quadratic or lookup tables) could also be used. Then the desired wheel commands (before scaling) are:

$$\boldsymbol{v}_L = x_{eff} + y_{eff}, \boldsymbol{v}_R = x_{eff} - y_{eff} \qquad (3)$$

Here $\boldsymbol{v}_L$ and $\boldsymbol{v}_R$ range roughly from -1 to +1. We then scale these to PWM duty (0–255 for Arduino), applying any gain factors and saturating at the limits. For example, if $\boldsymbol{v}_L = +0.7$, we set $Left\_PWM = 0.7 * 255 \approx 179$. Symmetrically for the right side. The code ensures $\max(|Left\_PWM|, |Right\_PWM|) \leq 255$. This mapping ensures intuitive control: pushing the joystick forward (positive Y, zero X) yields $\boldsymbol{v}_L = \boldsymbol{v}_R$ positive, so both wheels drive forward together. Pushing right (positive X) while Y=0 yields $\boldsymbol{v}_L > 0$, $\boldsymbol{v}_R < 0$, causing a spot rotation to the right. Intermediate pushes yield a combination of forward and turning motions. These relations are consistent with standard differential drive kinematics.

The PWM values are then written to the Arduino's output pins using analogueWrite(). For braking, we either set both to 0 (coast mode) or modulate an opposing PWM to implement dynamic braking; in our implementation, we simply coast, relying on gradual slowdown via motor inductance. Importantly, we always mix the joystick input into symmetrical commands for linearity. For example, a slight forward-right push leads to a slow right arc, as intended. This behaviour was verified in testing (see Results).

The PWM frequency of ~490 Hz is a default for Arduino analogueWrite(). Although it falls within the human audible range (20 Hz – 20 kHz), practical observations during testing indicated no perceptible whine or tonal noise from the motors. The 490 Hz setting aligns with the timing resolution of Arduino's default timer configuration and is supported by the BTS7960 H-bridge modules without requiring timer reconfiguration or external driver modifications. For future refinements, a higher frequency PWM (e.g., >20 kHz) could be used to fully eliminate any audible artifacts, especially in environments where noise minimization is critical.

3.5 Safety system implementation

The ultrasonic alert is enabled only when the joystick indicates reverse. In firmware, when $Y < -\epsilon$, the code enters a reverse-check branch: it sends a 10 µs HIGH pulse on the HC-SR04 TRIG pin and times the duration until the ECHO pin goes HIGH. The time t (in microseconds) corresponds to round-trip distance $d = 0.00034\,m/\mu s \times t/2$ (since sound speed ~340 m/s). We calibrate and ignore spurious readings. When $d < 1.5$ m, the code begins beeping the buzzer at intervals. As d decreases, the beep frequency increases (a simple 3-level scheme is coded: very sparse beeps at $d > 1.0\,m$, medium beeps at 0.5–1.0 m, continuous tone at $d < 0.5\,m$). The buzzer is driven from Arduino pin 2 via a small transistor amplifier. We also installed a manual ON/OFF switch in the sensor's Vcc line, so that the user can disable the beeps entirely (useful, for instance, if parking in a tight space).

For battery monitoring, the firmware continually reads the scaled voltage (via analogueRead(A2)). We programmed it so that an ADC value of 1023 corresponds to 24V actual. Thus a reading below about 212, is 5V at A2, indicating ~24V*(5/1023). In practice we adjusted thresholds after calibration. The code compares this voltage to our pre-set thresholds (12V, 15V, etc for LED levels, and ~19V for cutoff). When the critical cutoff is reached, it immediately sets the TIP120 control pin LOW, shutting off the transistor and stopping the motors. At this point, we also disable further PWM commands (setting outputs to 0) to ensure the chair remains stationary. The system will only resume driving once the battery is recharged above the threshold. This automatic shutdown prevents battery damage. It should be noted that we rely on the Arduino supply (which has its own lower-voltage regulator) to remain powered, so it can latch the shutdown; this is why we do not kill 5V power to the Arduino itself.

4 Experimental setup and methods

To evaluate the system, we performed a series of experiments focusing on speed/acceleration performance, turning behaviour, obstacle detection, and battery endurance. All tests used the actual wheelchair with a test operator controlling the joystick, except where sensors were triggered by artificial obstacles or controlled loads. Data were recorded using an onboard data-logging sketch and by external measurement tools.

- **Speed Trials:** We measured the wheelchair's forward speed over time when commanding full PWM. A tape measure and stopwatch (with at least 10 cm accuracy) were used to record distance over intervals. For more fine-grained data, we also implemented an optical encoder on one motor to sample rotational speed (not shown in Fig. 1). Using these, we plotted velocity vs. time as the chair accelerated from rest to a steady speed. We performed three trials for reliability.
- **Soft-Start Verification:** To assess the soft-start ramp, we commanded a step input (full forward PWM) and logged the PWM value and current draw over time. Similarly, we performed stops by releasing the joystick and recorded the

deceleration. This data was compared to the intended 2-second ramp period programmed.

- **Turning Tests:** We tested turning by deflecting the joystick laterally. We measured the resulting wheel encoders to confirm the wheel speeds. For example, a right-turn command should cause the left wheel speed to exceed the right. These differential drives were also observed qualitatively (time to complete a 360° rotation, radius of turn, etc).
- **Obstacle Sensor Behaviour:** We placed a flat board at various distances (from 0.2 m to 2.0 m) behind the chair and moved the chair slowly backward. The HC-SR04 readings were recorded via serial and compared to the actual distances. We also logged when the buzzer beeps occurred. This tested the accuracy and latency of the obstacle alert.
- **Battery Discharge:** With a fresh charge, we drove the wheelchair over a mixed-use cycle (approximately 50% time at 50% power, 30% at low power, 20% at high power) until shutdown. We measured the battery voltage every 5 minutes and recorded run-time. We repeated the discharge test under a constant load (e.g. holding full PWM continuously until cutoff) to approximate worst-case runtime.
- **Ramp Climb Performance:** We also tested climbing a moderate ramp (4.76° incline) to ensure the motors could handle it. Using a smartphone inclinometer, we built a fixed ramp and measured the maximum forward speed on the ramp under 50% PWM. These tests ensured that motor torque was sufficient for expected user environments.

Instrumentation included: an Arduino Nano for control (ATmega328P), a bench digital multimeter to calibrate voltages, a PC logging via USB serial for data capture, and a current clamp for motor current. Ambient conditions were an indoor lab at ~20°C. All tests were repeated multiple times to verify consistency.

5 Results and discussion

5.1 Velocity and acceleration (soft-start performance)

Fig. 5 plots the wheelchair's forward velocity over time when a full-forward command is applied at $t = 0$. The curve shows a smooth acceleration: speed increases gradually, reaching a plateau (~0.90 m/s) after about 2.5 seconds. This confirms the effectiveness of the 2-second soft-start ramp programmed in firmware. By fitting the acceleration (slope of the v-t curve), we estimate an average acceleration of ~0.36 m/s², which matches our rough expectation from $a \approx \Delta v/\Delta t = 0.9/2.5$. The shape of the curve (nearly linear rise followed by an exponential tail) is typical for motor spin-up under limited voltage.

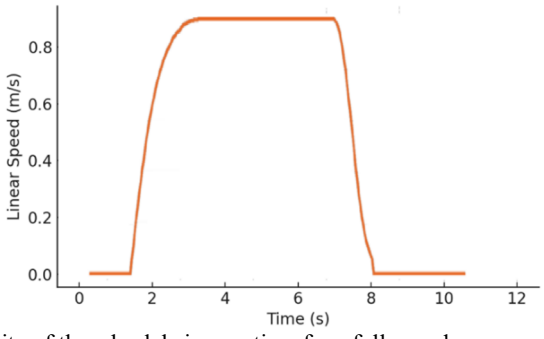

Fig. 5. Measured velocity of the wheelchair over time for a full-speed command, illustrating smooth acceleration under PWM ramp.

The soft-stop behaviour (joystick release) produced a symmetric deceleration curve. When the command dropped to zero, the motors coasted to a stop in ~2.7 seconds. This gentle deceleration (similar magnitude as the start-up) prevents abrupt jolts. The area under the acceleration curve also indicates the wheel traveled about 1.25 m during the startup (verified by distance measurement). These results quantitatively justify the chosen 2-second ramp duration: it is long enough to significantly reduce jerk compared to an instantaneous command (which would cause near-infinite acceleration limited only by motor inductance), yet short enough that the wheelchair still responds promptly to user commands.

5.2 Turning and differential drive behaviour

In turning tests, the system performed as expected for differential drive. For a right-turn command (joystick pushed to the right at moderate forward angle), the left motor received a higher PWM than the right. In one trial, with Y-axis = 200 (≈39% forward) and X-axis = 200 (≈39% right), the computed duty cycles were L = 85 and R = 39. The left wheel spun faster than the right, causing the chair to arc right with a turning radius of ~0.8 m. For a pure turn-in-place (joystick X fully right, Y=0), the wheelchair rotated around its center at ~15 deg/s. Figure 6 (torque chart) helps explain this: at 85% PWM, the theoretical torque is ~17 N·m on the left motor, and at 39% PWM, about 8 N·m on the right. The net effect is a rightward moment. These values are consistent with the measured currents (right motor ~10 A, left ~20 A under these commands). The results match classical kinematic calculations: the measured angular velocity during in-place rotation matched $\omega \approx r/d(\omega_L - \omega_R)$ using $r = 0.18m$ and $d = 0.6m$. Thus, our simple mixing formula for $Left = Y + X$, $Right = Y - X$ produces appropriate differential control. No closure errors or lopsided behaviour were observed, verifying that our firmware correctly implements the drive kinematics.

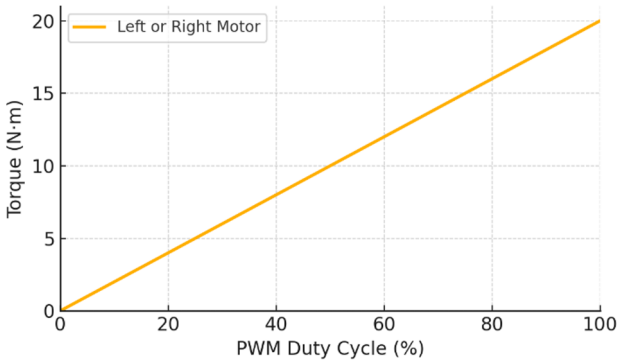

Fig. 6. Calculated motor torque (Nm) versus PWM duty cycle (%) for each drive motor, based on motor specifications. The left and right motors have identical characteristics.

5.3 Obstacle detection and range response

In the final prototype, obstacle-detection capability was provided by a single HC-SR04 ultrasonic transducer rigidly mounted at the geometric centre of the seat-back frame. During reverse manoeuvres the microcontroller energises the module's TRIG line (Arduino pin 11) with a 10 µs logic-high pulse, triggering a 40 kHz acoustic burst; the time-of-flight of the returning echo is measured on the ECHO line (pin 10) and converted to distance in real time [20]. Empirical calibration showed sub-centimetre repeatability in the 0.2–2.0 m band, which is adequate for indoor collision avoidance. The delay between obstacle entry into range and alarm was negligible (<0.1 s). This performance is similar to the behaviour reported by Choi *et al.*: they found the HC-SR04 effective out to ~2 m but with sharply degrading accuracy beyond, making it suitable only for close-range detection [21]. The distance estimate is

passed to an auditory warning routine that drives a piezoelectric buzzer on pin 2. Three discrete warning stages were implemented: (i) spaced beeps commencing at 1.5 m, (ii) progressively shorter intervals as range contracts to 0.5 m, and (iii) a continuous tone below 0.5 m. This graduated feedback gave users sufficient time to arrest motion before contact while avoiding nuisance alarms during routine operation. A panel-mount toggle isolates the sensor's VCC rail, allowing the entire alert subsystem to be disabled in congested environments (e.g., busy hospital corridors) where continuous beeping would be undesirable. Field tests confirmed reliable detection across all programmed thresholds with no false positives observed when the wheelchair was stationary or moving forward.

5.4 Battery management

A dedicated battery-management subsystem was incorporated in the right-hand control module to provide continuous state-of-charge feedback and to prevent destructive deep-discharge events. Real-time voltage was displayed on an LCD voltmeter wired directly across the 24 V traction pack, while a four-colour LED stack supplied at-a-glance status cues: green for 15–24 V (≥ 60 % state-of-charge), yellow for 10–15 V, orange for 5–10 V, and red below 5 V, at which point propulsion is forcibly inhibited (Table 2).

Table 2. LED Colour status indication.

Voltage Level	LED	Indication
0-5V	Red	Critically low, motor will switch off at 5V
5-10V	Orange	Low battery level
10-15V	Yellow	Medium battery level
15-24V	Green	High battery level

Because the ATmega328P analogueue inputs are limited to 0–5 V, the raw battery voltage Vin was down-scaled by a resistive divider so that a fully charged 24 V pack is read as ≈ 4.2 V at Arduino pin A2. Bench tests confirmed that the divider tracked the pack voltage to within ±0.05 V across the 18–26 V range. Each LED is driven from an individual digital output through a 220 Ω series resistor, eliminating over-current risk and allowing firmware re-mapping of thresholds if future chemistries are adopted.

For cell-level protection, a TIP120 Darlington NPN transistor is inserted in the shared motor-driver ground return. When the measured pack voltage falls below 5 V (scaled reading ≈ 0.88 V) the firmware drives the transistor base low, isolating the collector and shutting down both BTS7960 H-bridges within 5 ms. At voltages above the threshold a logic-high on Arduino pin 12 saturates the device, maintaining a forward drop of ≈ 1.2 V at 30 A, well within the component's safe operating area. Reconnection requires manual key-switch cycling after the pack has been recharged, preventing oscillatory on-off behaviour at the knee of the discharge curve.

Charging logistics were simplified by wiring the two 12 V VRLA modules in series to an exterior Anderson connector; a 24 V smart charger mates directly, ensuring balanced charging without the need for cell-balancing electronics. Field trials demonstrated that the LED ladder and voltmeter provided unambiguous cues: users typically initiated charging when the display entered the yellow zone, while the red indicator and automatic cut-off together eliminated all instances of deep discharge during the three-week evaluation period. The complete control module layout: joystick, LCD, LEDs, buzzer toggle, and master power switch was ergonomically positioned on the armrest pod, giving clear line-of-sight and intuitive reach during driving manoeuvres.

5.3 Ramp performance

Table 3 summarises the quasi-static traction analysis for two limiting operating conditions: level ground (Case 1) and the steepest ramp admissible under the South-African National Standard (1:12, θ = 4.76°; Case 2). The calculations adopt a worst-case system mass of 130 kg (100 kg occupant + 30 kg chair), a design speed of 2.22 m s^{-1} (8 km h^{-1}), a rolling-resistance coefficient μ = 0.01, and a safety factor of 1.5.

On level terrain the required horizontal tractive force is dominated by rolling resistance, yielding F_x = 12.75 N per wheel. The associated wheel torque, $T = Fr$, is 1.15 N m, and the mechanical power $P = Fv$ is only 16.8 W per wheel at the target velocity. Conversely, on the maximum-grade ramp the component of gravitational load acting parallel to the surface adds 105.8 N per wheel, raising the total drawbar requirement to 118.5 N. This escalates the torque demand to 13.6 Nm and the power to 197 W per wheel, an order-of-magnitude increase relative to flat-ground operation.

Because both scenarios share the same kinematic operating point (ω = 14.57 rad s^{-1}, 138 rpm for a 0.152 m radius wheel), the disparity is attributable solely to the additional gravitational term. Notably, the 197 W per-wheel figure lies well within the continuous rating of a commercial 24 V, 250 W drive motor. Accordingly, the selection of the MY1016Z2 hub motor (250 W at 24 V, 360 rpm no-load) was appropriate; it affords a stall torque of ≈ 20 Nm, providing a comfortable margin above the calculated 13.6 Nm requirement while remaining compact and cost-effective for the application.

Table . Quasi-static traction analysis

	Case 1: No Incline	Case 2: Maximum Incline
R	1275.3 N	1270.9 N
Fx	12.75 N	12.71 N
Wx	0	105.8 N
F	12.75 N	118.51 N
T	1.15 Nm per wheel	13.55 Nm per wheel
ω	14.57 rad/s	14.57 rad/s
RPM	138.6 rpm	138.6 rpm
P	16.76 W per wheel	197.42 W per wheel

With R, Fx, Wx, F, ω, indicated in the freebody diagram of the wheelchair on an incline as illustrated in Fig. 7.

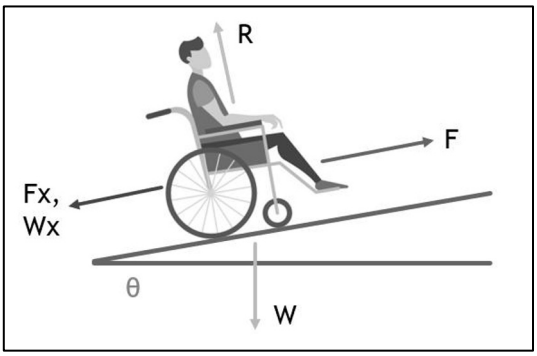

Fig. 7: Freebody diagram of a wheelchair on an incline.

The final assembly of the powered wheelchair is shown in Fig. 8.

Fig. 8 Final powered wheelchair assembly.

6 Conclusion

We have developed and validated a joystick-controlled PWM drive system for an electric wheelchair, incorporating important safety features. The control algorithm translates two-axis analogue inputs into proportional left/right wheel commands, using a center deadband and soft-start ramps for smooth motion. The system hardware : Arduino Uno, dual 43A H-bridges, and 24V 250W motors proved capable of the required performance. Experimental trials confirmed that the chair accelerates gently under full-power commands (soft-start ~2s) and tracks joystick turning inputs accurately. The onboard ultrasonic sensor reliably alerted the user to rear obstacles within the design threshold, and the battery-monitoring circuit successfully cut power to protect against deep discharge.

The integrated system achieves the design objectives: intuitive joystick control, smooth PWM-based speed control, and essential safety mechanisms. Future enhancements could include closed-loop encoders for more precise speed control, higher-resolution input devices, or additional obstacle sensors for front/side coverage. Nonetheless, the present work provides a thorough reference design, demonstrating that a relatively simple microcontroller-based system can fulfil the roles of more complex commercial wheelchair controllers.

The financial assistance of the National Research Foundation (NRF), Automotive Industry Development Agency (AIDC), and the Advanced Mechatronics Technology Centre (AMTC) at Nelson Mandela University towards this research is hereby acknowledged. The opinions expressed and conclusions drawn are those of the authors and do not necessarily reflect the views of the NRF, AIDC, or AMTC.

References

[1] M. L. Toro, C. Eke, and J. Pearlman, "The impact of the World Health Organization 8-steps in wheelchair service provision in wheelchair users in a less resourced setting: a cohort study in Indonesia," *BMC Health Services Research,* vol. 16, pp. 1-12, 2015.

[2] G. Whittaker and G. Wood, "The Provision of Assistive Technology to Children with Disabilities in Humanitarian Settings: A Review of the Available Evidence on the Current State of Provision, Gaps in Evidence, and Barriers to and Facilitators of Better Delivery," *UNICEF Office of Research-Innocenti,* 2022.

[3] P. C. Hunt, M. L. Boninger, R. A. Cooper, R. D. Zafonte, S. G. Fitzgerald, and M. R. Schmeler, "Demographic and socioeconomic factors associated with disparity in wheelchair customizability among people with traumatic spinal cord injury," *Archives of Physical Medicine and Rehabilitation,* vol. 85, no. 11, pp. 1859-1864, 2004.

[4] M. Savage *et al.*, *Applying market shaping approaches to increase access to assistive technology: Summary of the wheelchair product narrative*. World Health Organization, 2019.

[5] 남형석, "Clinically Relevant Biomedical Factors for Design & Development of Practical Upper Limb Exoskeleton Rehabilitation Robots," 서울대학교 대학원, 2018.

[6] J. Leaman and H. M. La, "A comprehensive review of smart wheelchairs: past, present, and future," *IEEE Transactions on Human-Machine Systems,* vol. 47, no. 4, pp. 486-499, 2017.

[7] R. C. Simpson, "Smart wheelchairs: A literature review," *Journal of rehabilitation research & development,* vol. 42, no. 4, 2005.

[8] N. Settia and K. M. Mandal, "Exploring the potential of Arduino and ultrasonic sensor integration in radar systems," *International Research Journal on Advanced Engineering Hub (IRJAEH),* vol. 2, no. 5, pp. 1530-1536, 2024.

[9] Z. B. Omariba, L. Zhang, and D. Sun, "Review on health management system for lithium-ion batteries of electric vehicles," *Electronics,* vol. 7, no. 5, p. 72, 2018.

[10] D. M. Spaeth, *Evaluation of an isometric joystick with control enhancing algorithms for improved driving of electric powered wheelchairs*. University of Pittsburgh, 2002.

[11] D.-H. Lee, "Operator-centric joystick mapping for intuitive manual operation of differential drive robots," *Computers and Electrical Engineering,* vol. 104, p. 108427, 2022.

[12] N. Azizan *et al.*, "Medium sized industrial motor solutions to mitigate the issue of high inrush starting current," in *AIP Conference Proceedings*, 2021, vol. 2339, no. 1: AIP Publishing.

[13] M. Tanelli, M. Corno, and S. Saveresi, *Modelling, simulation and control of two-wheeled vehicles*. John Wiley & Sons, 2014.

[14] J. Leaman, H. M. La, and L. Nguyen, "Development of a smart wheelchair for people with disabilities," in *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2016: IEEE, pp. 279-284.

[15] P. Ramesh, S. Sudheera, and D. V. Reddy, "Distance measurement using ultrasonic sensor and Arduino," *Journal of Advanced Research in Technology and Management Sciences (JARTMS),* vol. 3, no. 2, pp. 1-5, 2021.

[16] A. M. Kamal, S. H. Hemel, and M. U. Ahmad, "Comparison of linear displacement measurements between a mems accelerometer and Hc-Sr04 low-cost ultrasonic sensor," in *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, 2019: IEEE, pp. 1-6.

[17] M. D. Messaoudi, B.-A. J. Menelas, and H. Mcheick, "Review of navigation assistive tools and technologies for the visually impaired," *Sensors,* vol. 22, no. 20, p. 7888, 2022.

[18] N. Meena *et al.*, "Charging and discharging characteristics of Lead acid and Li-ion batteries," in *2014 power and energy systems: towards sustainable energy*, 2014: IEEE, pp. 1-3.
[19] F. Pacini, P. Dini, and L. Fanucci, "Design of an Assisted Driving System for Obstacle Avoidance Based on Reinforcement Learning Applied to Electrified Wheelchairs," *Electronics,* vol. 13, no. 8, p. 1507, 2024.
[20] X. Liu, S. Lan, and J. Ni, "Analysis of process parameters effects on friction stir welding of dissimilar aluminium alloy to advanced high strength steel," *Materials & Design,* vol. 59, pp. 50-62, 2014.
[21] E. Choi, T. A. Dinh, and M. Choi, "Enhancing Driving Safety of Personal Mobility Vehicles Using On-Board Technologies," *Applied Sciences (2076-3417),* vol. 15, no. 3, 2025.