



Design and Experimental Evaluation of a Three-Floor Model Elevator Using Arduino Mega: Floor-Request Logic and Safety Interlocks

**Mohamed Yusuf Mohamud*, MohamedNor Ibrahim Abdirahman, Mohamud A. Salam
Hersi, Abdirashid Abdullahi Osman**

Department of Mechanical Engineering, Faculty of Engineering, Hormuud University,
Mogadishu, Somalia

*Corresponding author's email: moyuusuf15@gmail.com

Abstract

Reliable elevator operation depends on coordinated request handling, position sensing, motion control, door sequencing, and safety interlocks. This study presents the design, implementation, and experimental evaluation of a three-floor educational elevator prototype built around an Arduino Mega controller. The system integrates floor-request push buttons, mechanical floor-position switches, a DC motor and driver, a servo-operated door, an HX711-based load-cell arrangement for overload monitoring, visual and audible indicators, and a DFPlayer Mini for floor announcements. Control is organized as a state-based process with a request queue and directional servicing of pending calls. The prototype was evaluated at subsystem and integrated-system levels. Floor detection achieved 100% accuracy over 40 transitions, button registration reached 100% over 100 recorded presses, overload detection reached 98% over 50 tests, and voice announcements were correct in 39 of 40 arrivals (97.5%). Stopping remained within ± 2 mm of the target floor against a ± 5 mm design tolerance. Across 50 consecutive random floor-request cycles, 48 were completed successfully, giving an overall success rate of 96%; the two unsuccessful cycles were associated with temporary DFPlayer communication faults rather than failures of motion, position sensing, or door control. The results show that a low-cost microcontroller platform can integrate structured request processing with physical safety interlocks and user feedback in a coherent model-scale system. The prototype is intended for mechatronics and embedded-control education rather than passenger service, and it provides a practical basis for future work on encoder-based closed-loop motion control, non-contact sensing, stronger fault recovery, and remote monitoring.

Keywords: Elevator control system, Arduino Mega, floor-request logic, Directional priority, Safety interlocks, Finite-state control, Embedded systems, Educational prototype

1. Introduction

Elevators are integrated electromechanical systems in which motion is only one part of the control problem. A controller must accept passenger requests, determine a service sequence, identify car position, command the drive, coordinate door operation, communicate status, and inhibit motion whenever a safety condition is not satisfied. These functions have long been central to vertical-transportation engineering, where traffic handling and control quality are treated together with mechanical reliability and passenger safety [1], [2].

The transition from relay-based logic to programmable controllers and embedded processors has made elevator behavior easier to modify, test, and extend. Programmable logic provides deterministic coordination of inputs and outputs, while state-based software offers a compact way to represent mutually exclusive operating modes and event-driven transitions. Finite-state and cyber-physical design principles are particularly suitable for small elevator prototypes because they connect software decisions directly to sensed physical conditions [3]–[6].

Safety remains a separate design concern from scheduling. A request for movement is valid only when the door state, load condition, emergency-stop state, and position information permit motion. Commercial elevator practice formalizes these dependencies through layered requirements for doors, braking, inspection, operation, and protective functions. Although a model-scale prototype cannot claim compliance with passenger-elevator standards, the underlying design principle is transferable: unsafe or uncertain conditions should prevent or terminate motion rather than merely generate a warning [7], [8].

Elevator models are also useful teaching systems because they combine mechanical structures, electrical drives, sensors, actuators, embedded software, and human-machine interfaces in a single observable platform. Recent educational implementations have used Arduino-based controllers, state-oriented logic, motion-control laboratories, PLC/HMI trainers, and IoT-enabled elevator models to connect classroom theory with physical behavior [9]–[14]. These studies show that the instructional value of an elevator prototype increases when control decisions, feedback signals, and safety conditions can be inspected and tested rather than demonstrated only through cabin movement.

Two technical issues motivate the present work. The first is floor-request handling. Even with only three floors, multiple calls can coexist, and the controller must decide whether to continue in the current direction, stop at an intermediate requested floor, reverse direction, or remain idle. The second is safety integration. Door closure, overload status, floor detection, and emergency stopping should constrain the movement state through physical inputs rather than exist only as software labels. In practice, these two issues are coupled: a scheduling decision has no operational meaning unless the car can execute it safely, and a safety system that is detached from request logic cannot guarantee coherent behavior. The design therefore treats request state, car position, door state, and load state as inputs to one supervisory decision process.

Accordingly, this study develops a three-floor model elevator for controlled educational testing at Hormuud University, Mogadishu, Somalia. The prototype combines an Arduino Mega with floor-request buttons, mechanical position switches, a motor-drive stage, servo door actuation, an HX711 load-cell subsystem, LEDs, a buzzer, an LCD, and a DFPlayer Mini. Its request logic uses a finite-state structure and a queue that preserves pending calls while giving practical priority to floors lying in the current direction of travel. The choice of a three-floor configuration was deliberate. It is sufficiently complex to exercise upward and downward travel, intermediate stops, queued requests, and repeated door cycles, yet small enough for students to trace each transition directly from input to actuator response. This balance between observable behavior and nontrivial logic is important in laboratory-scale mechatronics.

The objectives were to implement reliable floor selection, enforce motion-related safety interlocks, and quantify the performance of the integrated system. Evaluation therefore considered floor detection, button registration, overload classification, voice-announcement reliability, stop-position tolerance, and overall cycle completion. The contribution is an experimentally characterized educational prototype in which request logic, physical safety inputs, sensing, actuation, and user feedback are assessed together rather than as isolated demonstrations. Rather than judging the prototype only by whether the cabin moved, the evaluation asks whether the system repeatedly reached the correct floor, recognized commands,

identified unsafe loading, stopped within tolerance, and completed whole operating cycles. These measures provide a more defensible basis for assessing an educational control system.

2. Related Work

2.1 Evolution of Elevator Control Systems

Elevator control evolved from operator-driven and relay-based systems toward programmable controllers and embedded processors. The change shifted much of the sequencing logic from fixed wiring into software, enabling easier modification of operating rules, diagnostics, and interaction among sensors and actuators. For an educational prototype, this evolution justifies a programmable architecture in which the same controller can coordinate motion, doors, floor sensing, alarms, and user interfaces. The key change was architectural. Relay systems encode sequencing through physical interconnections, whereas programmable systems separate the control policy from the wiring that carries sensor and actuator signals. This separation supports rapid modification of timing, request rules, diagnostics, and fault responses without rebuilding the hardware.

At model scale, the objective is not to reproduce the redundancy and certification structure of a commercial installation. The useful abstraction is instead the control chain: requests are acquired, safety permissions are checked, motion is commanded, position feedback terminates travel, and the door sequence follows only after a valid arrival. This structure provides sufficient complexity for testing embedded control without obscuring the logic behind industrial-scale hardware. A three-floor model retains the essential event structure of a larger installation: a request is created, a target is selected, movement begins only under valid conditions, position feedback changes the motion state, and a service action occurs at arrival. Because these events can be observed individually, the model is useful for exposing race conditions, missed inputs, improper transitions, and unsafe actuator commands during development.

2.2 Floor-Selection Algorithms

Floor selection is a scheduling problem in which pending requests must be mapped to an ordered sequence of stops. Strict first-come-first-served processing is easy to implement but ignores car position and direction; nearest-request methods reduce immediate travel but can create unfair waiting; directional strategies group compatible calls and reduce unnecessary reversals. Contemporary elevator-control research extends these ideas through heuristic, metaheuristic, occupancy-aware, and direction-optimized scheduling, especially for multi-car systems [15]–[18]. The objective function also changes with system scale. A high-rise group controller may optimize waiting time, journey time, energy use, car occupancy, or service fairness, whereas a single-car teaching model primarily requires deterministic handling and avoidance of contradictory commands. For this reason, simple directional logic is often more appropriate pedagogically than a computationally sophisticated optimizer.

For a three-floor single-car prototype, such advanced optimization is unnecessary, but the same principles remain useful. A queue must preserve outstanding requests, and the controller should avoid reversing direction when a pending call can be served naturally along the current route. Smart-elevator research has likewise shown that floor selection can incorporate user history, destination prediction, and personalized service, illustrating how simple request queues can be extended when more information is available [19], [20]. Directional servicing also makes the queue easier to interpret. If the car is moving upward, requests above the current position can be grouped naturally, while lower-floor requests remain pending until reversal. The resulting behavior reduces needless changes of direction and illustrates the relationship between request ordering and physical travel cost without introducing an opaque optimization routine.

The present controller therefore uses queued directional request handling rather than a claimed new dispatching algorithm. The current floor and requested floor determine travel direction; pending calls are retained; and compatible intermediate requests can be served before reversal. This rule is intentionally transparent so that the relationship between request state and motor action remains visible to students. This distinction is important for reproducibility. The algorithm can be described directly from observable rules and implemented with a small number of state variables, making it possible to compare the intended service policy with the actual sequence executed by the model.

2.3 Safety Interlocks

Safety interlocks convert physical conditions into permissions for action. In the present context, motion is permitted only when the door is closed, the overload condition is inactive, and the emergency-stop path is intact. This is stronger than treating safety as a message displayed after a fault because the unsafe condition directly constrains the movement command. Interlocks are most effective when they are evaluated before an actuator command is issued and continue to be monitored during operation. A door-open condition should prevent the transition into a moving state; an emergency stop should remove motor power regardless of the software state; and an overload condition should keep the car stationary until the measured value returns to an acceptable range.

Recent elevator monitoring research increasingly combines distributed sensors, networked supervision, fault detection, and remote diagnostics. IoT-based elevator systems have been proposed for real-time status monitoring, safety supervision, fault identification, and condition-based maintenance, while multisensor nodes extend the observable state beyond simple floor switches [21]–[24]. These systems are more sophisticated than the present prototype, but they reinforce the importance of treating safety and condition sensing as integral control inputs. At full scale, safety monitoring extends to speed, braking, door locks, landing zones, communication health, and fault history. The present design narrows that scope to conditions that can be implemented and tested visibly in a teaching prototype. This reduction preserves the principle of layered protection without implying equivalence to certified lift safety systems.

At prototype scale, three physical mechanisms were selected: a door-frame micro switch, an emergency stop wired in series with motor power, and a load-cell-based overload detector. Together they provide independent examples of permissive interlocking, direct power interruption, and threshold-based safety sensing. The combination is also pedagogically useful because each mechanism illustrates a different protection philosophy. The door switch is a permissive interlock, the emergency stop is a direct interruption path, and the load sensor is a measured variable interpreted against a threshold. Their different behaviors can therefore be examined under normal and fault conditions.

2.4 Microcontroller Platforms in Automation and Education

Arduino-class microcontrollers are well suited to instructional automation because they expose digital and analog I/O, timers, interrupts, PWM, and serial communication through an accessible programming environment. These features allow a single board to coordinate buttons, switches, motor drivers, servos, displays, load sensors, and audio modules. The motor-drive element also gives students a practical link between embedded commands and electromechanical behavior, including the limits of open-loop speed control [25]. The Mega platform is particularly convenient where many discrete I/O channels are required simultaneously. Elevator models typically need separate inputs for floor calls, position sensors, emergency conditions, and door states, while outputs are needed for motor direction, indicators, servo commands, alarms, and display or audio interfaces. Consolidating these functions on one controller reduces wiring complexity and makes signal flow easier to trace.

Hands-on and project-based engineering education has consistently emphasized the value of linking theory to physical implementation. Low-cost embedded platforms can support this approach by making programming, debugging, sensing, and actuation visible within the same exercise, while broader engineering-education frameworks stress iterative design, testing, and system integration as core learning activities [26]–[29]. A physical elevator model also creates natural integration tasks. Mechanical misalignment can appear as a sensor problem; poor wiring can resemble a software fault; a timing error can cause a door or audio failure even when individual devices operate correctly. Students therefore encounter the cross-domain interactions that define mechatronic design rather than working with isolated electronic or mechanical exercises.

The Arduino Mega 2560 was selected because its I/O capacity and multiple serial interfaces accommodate the prototype without auxiliary controllers. Its use here is educational rather than industrial: the platform does not provide the certified redundancy, diagnostics, environmental qualification, or safety integrity expected of passenger-elevator controllers. The choice also reflects cost and maintainability. Replacement boards, sensors, and interface modules are widely available, and the programming environment permits rapid revision of control logic. These advantages are appropriate for a teaching laboratory where transparency and iteration are priorities, provided that the boundary between educational prototyping and industrial control is stated clearly.

2.5 Existing Model Elevator Systems and Research Gap

Existing model elevators commonly demonstrate cabin movement, push-button control, and floor sensing, but their technical depth varies considerably. Some are designed primarily as visual demonstrations, whereas stronger educational systems expose state transitions, motion control, safety conditions, and measurable performance. The distinction matters because a moving cabin alone does not show whether requests are scheduled consistently or unsafe states are physically inhibited. For publication-quality evaluation, the prototype must also be described at a level that permits another laboratory to reconstruct its architecture. Controller type, sensing method, actuator arrangement, safety logic, and test criteria therefore matter as much as photographs of successful operation.

A second gap concerns experimental characterization. Educational prototypes are often described functionally without reporting floor-detection accuracy, overload classification, stop-position tolerance, communication reliability, or integrated-cycle success. Quantitative testing makes the design more useful as an engineering artifact because weaknesses can be traced to particular subsystems rather than summarized as general success or failure. Repeated measurements are especially valuable in this context because they reveal subsystem-specific failure modes. A model that succeeds in most complete cycles may still contain a weak audio link, noisy overload threshold, inconsistent stop position, or unreliable button interface. Reporting these components separately prevents one aggregate success percentage from obscuring the engineering cause of failure.

The present study addresses these gaps by integrating queued request handling, physical safety interlocks, overload monitoring, user feedback, and repeated performance testing in one three-floor model. Its novelty is therefore implementation-oriented: the contribution lies in the coordinated architecture and measured behavior of a low-cost educational system. The resulting design is positioned between a simple demonstrator and a research-scale elevator simulator. It remains intentionally low cost and physically transparent, but its behavior is defined through states, testable interlocks, repeatable trials, and explicit performance targets.

2.6 Theoretical Framework

The control framework combines finite-state reasoning, event-driven input processing, and fail-safe permissions. The elevator occupies one dominant operating state at a time—IDLE,

MOVING_UP, MOVING_DOWN, DOOR_OPEN, or FAULT—and changes state when a validated event occurs. This structure limits ambiguous actuator commands because each state defines which outputs and transitions are permissible. State separation is particularly useful for preventing incompatible commands. For example, the motor cannot be commanded upward and downward simultaneously, and the door-open state excludes movement. Fault conditions can likewise override normal request processing, giving the controller a clear priority structure. User buttons and sensor inputs are sampled continuously, while floor-position events are handled through interrupt-capable inputs. Debouncing prevents a single physical button press from being interpreted as several requests. The request queue, current-floor variable, target floor, and safety flags together determine the next action. The queue acts as memory across state changes. A new request does not need to be served immediately; it remains stored while the current motion sequence completes. Once an arrival event updates the current-floor state, the remaining queue can be re-evaluated against the present direction and safety conditions. Fail-safe logic constrains those actions. An open door, overload condition, or emergency-stop activation inhibits motion; a valid floor signal terminates travel; and door actuation follows only after the car has stopped. The theoretical framework therefore links software states to physical conditions instead of treating the two as separate layers. This architecture supports modular testing. Input handling can be verified independently of motion, overload logic independently of floor scheduling, and door sequencing independently of request generation. Integrated testing can then determine whether the modules interact correctly under repeated complete cycles.

3. Materials and Methods

3.1 Research and Development Approach

The study followed a design-build-test approach comprising system definition, component selection, mechanical fabrication, controller programming, subsystem integration, and experimental evaluation. The prototype was constructed for controlled laboratory and teaching use at Hormuud University in Mogadishu, Somalia; commercial deployment and certification testing were outside the study scope. The development sequence was iterative rather than strictly linear. Subsystems were first checked independently, then combined in stages so that faults in sensing, actuation, or communication could be isolated before full-cycle testing.

Components were selected for availability, compatibility, and suitability at model scale. The final architecture combined an Arduino Mega 2560 (Figure 1), DC motor with L298N driver, mechanical floor switches, floor-request buttons, SG90 door servo, HX711 load-cell interface, display and indicator devices, buzzer, DFPlayer Mini, power supplies, and the mechanical guidance structure summarized in Table 1. The use of commonly available components was intentional because the study emphasizes a platform that can be maintained and reproduced in a university workshop without specialized industrial hardware.

Table 1. Principal hardware components and their functions.

Component	Role in the prototype
Arduino Mega 2560	Central controller for input processing, control logic, communication, and output control
Motor and L298/L298N driver	Vertical movement and direction/speed control
Zing Ear G905 micro switches	Floor-position and safety-state sensing
Push buttons	User floor-request inputs
SG90 servo motor	Door opening and closing
HX711 load-cell arrangement	Weight measurement and overload detection
DFPlayer Mini + speaker	Floor voice announcements
LCD/display and LEDs	Floor/status indication

Buzzer	Audible safety and status alert
12 V adapter / battery holder	Power supply arrangements
Bearings, shafts, wood/aluminum/polymeric materials	Mechanical guidance and structural support

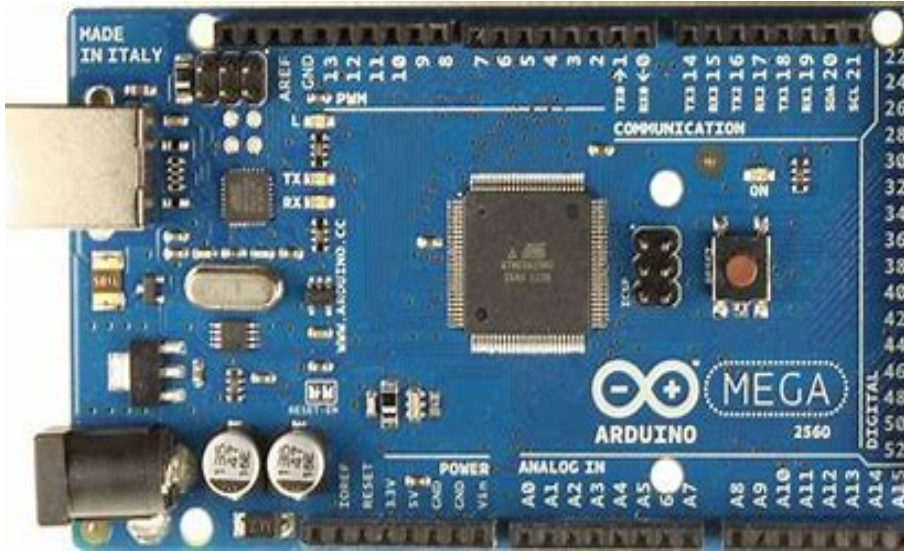


Figure 1. Arduino Mega used as the central controller in the prototype.

3.2 Mechanical Prototype

The elevator model (Figure 2) contains ground, first, and second floors within a structure approximately 120 cm high and 30 cm wide, with floor levels spaced at about 30 cm. The car is approximately 26 cm wide, 26 cm deep, and 30 cm high. Two iron shafts guide vertical movement, while a rear counterweight assists mechanical balance. Wood, aluminum, and polymeric elements were used to keep the structure light, inexpensive, and accessible for workshop fabrication.



Figure 2. Completed three-floor elevator prototype after demonstration.

3.3 System Architecture

The control architecture (Figure 3) consists of input devices, a central controller, actuation devices, feedback devices, and safety-related sensing. User floor requests are entered through push buttons. Floor-position micro switches provide discrete position information. The load-cell system provides a weight-related safety input. Door condition is monitored through a physical switch, while the emergency stop provides a direct power interruption path. The Arduino Mega processes these signals and controls the motor driver, servo door, display, LEDs, buzzer, and DFPlayer Mini. The motor driver receives direction and PWM commands from the controller, while the floor switches return discrete arrival information. The load-cell channel and door switch act as supervisory safety inputs; the display, LEDs, buzzer, and audio module provide feedback without participating directly in the motion decision.

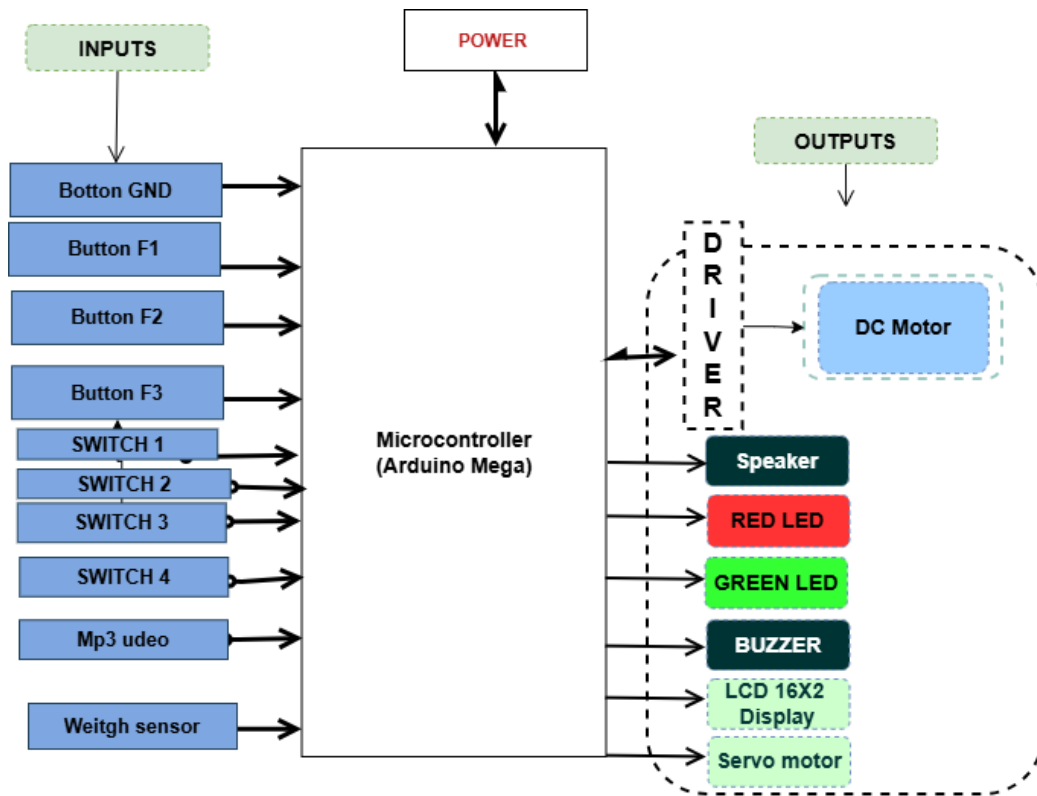


Figure 3. Functional block diagram of the elevator control system showing inputs, controller, and outputs.

The architecture was designed so that the controller could combine request processing with safety checks. A movement command is not treated as sufficient by itself; the controller must consider the relevant safety conditions. Floor arrival is detected by the position switches, which then cause the motor to stop and the door sequence to begin. The display and indicators provide visible feedback, while the DFPlayer provides an audible announcement of the reached floor. This separation of functions simplifies fault interpretation. A failure of an announcement does not necessarily imply a failure of the arrival decision, while an invalid door or overload state blocks the motion command before user-feedback devices are considered. The control circuit is shown in Figure 4.

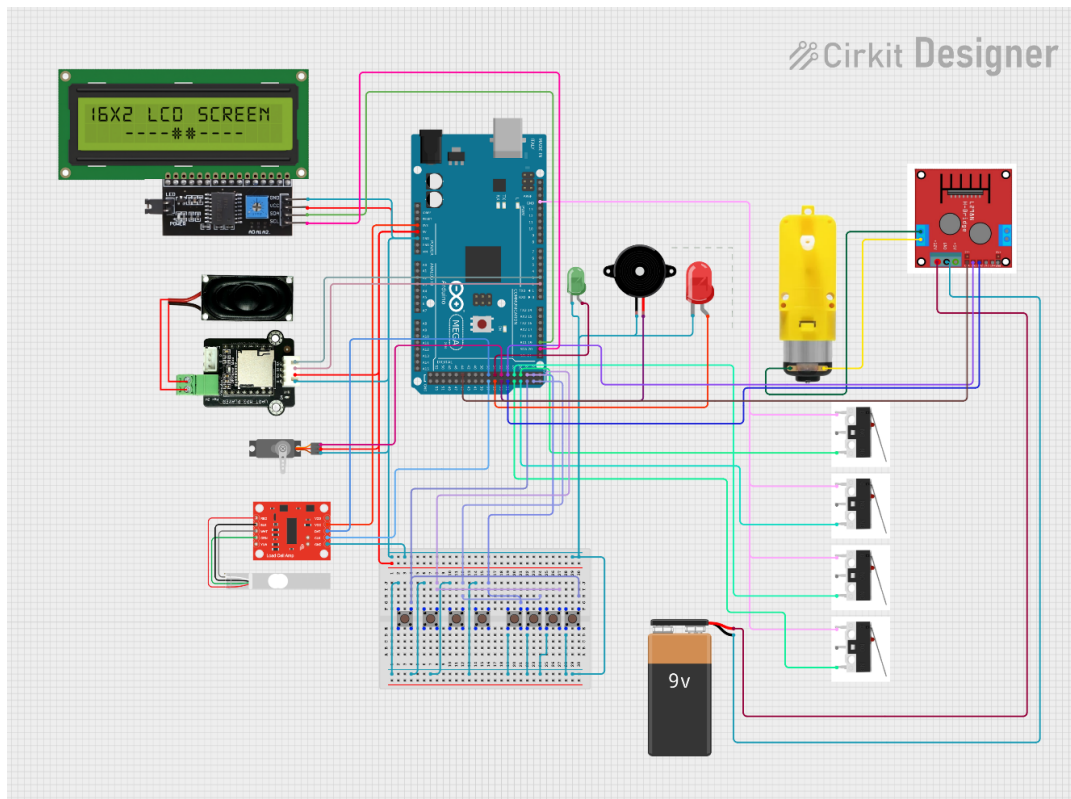


Figure 4. Control circuit used to connect the principal prototype components.

3.4 Floor-Request and Control Logic

The control program is state oriented (Figure 5), with operating conditions corresponding to idle, upward travel, downward travel, door operation, and fault handling. When a request is registered, its floor is stored in the queue. The controller compares the target with the current floor, checks safety permissions, commands the appropriate motor direction, and terminates travel when the target-floor switch is activated. The movement state is therefore event terminated rather than time terminated: the motor continues until the appropriate physical floor switch confirms arrival. This avoids relying on a fixed travel-time estimate, which could drift with supply voltage, load, friction, or mechanical alignment.

A circular buffer stores up to 10 pending floor requests. The tested multi-request behavior is directional: from the ground floor, simultaneous requests for floors 1 and 2 are served by stopping at floor 1 before continuing to floor 2. The implementation is therefore described as queued directional request handling rather than strict FCFS. Duplicate requests are ignored when a floor is already pending; when the queue is empty, the controller returns to IDLE.

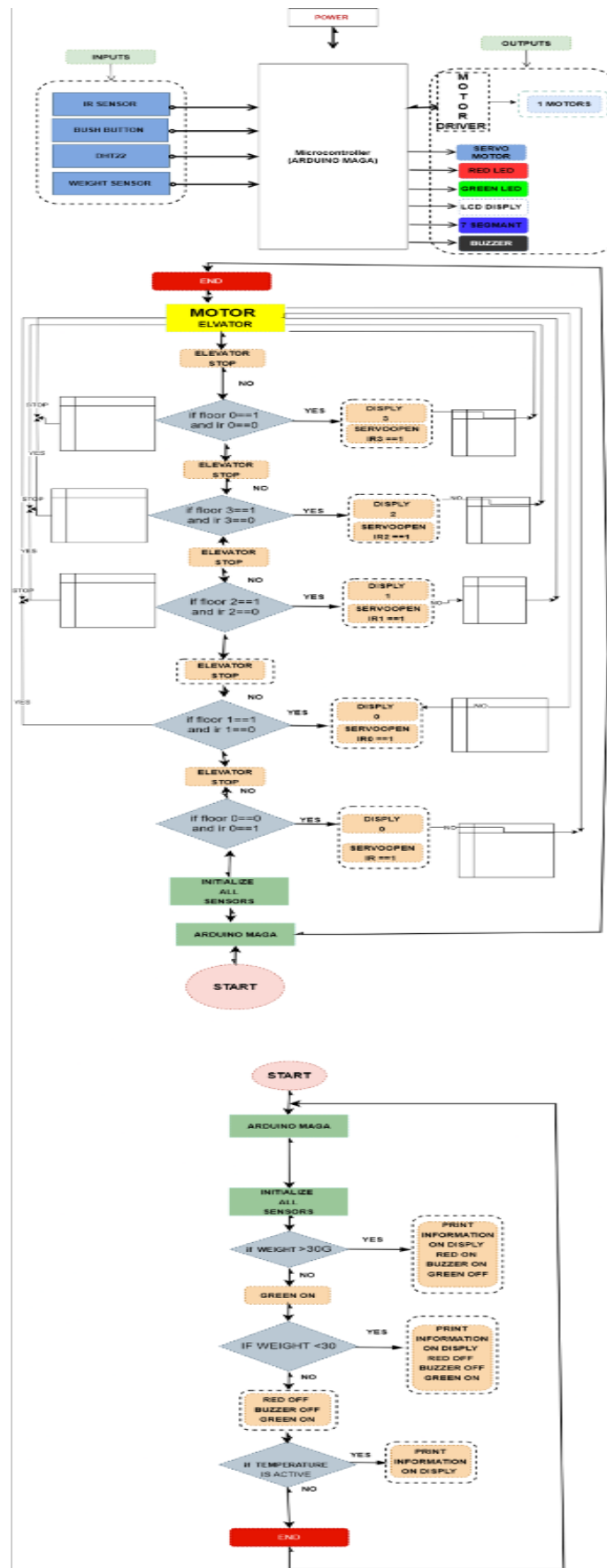


Figure 5. Flowchart representing the documented elevator control sequence.

3.5 Safety Interlock Implementation

Three principal safety mechanisms were implemented. A door-frame micro switch inhibits movement unless the door is closed; an emergency-stop button is connected in series with motor power to provide a hardware cut-off independent of normal software execution; and the HX711 load-cell subsystem inhibits normal operation when the measured load exceeds the 1 kg prototype threshold. A buzzer and red/green indicators communicate overload and fault conditions. The emergency-stop circuit was deliberately implemented outside the normal logic path so that a software fault could not prevent power interruption. This feature distinguishes a physical safety action from a software-generated alarm.

The overload routine samples the weight (Figure 6) at approximately 500 ms intervals. Above the 1 kg threshold, the red LED and buzzer are activated and the green indicator is disabled; below the threshold, the green indicator remains active and the buzzer is silent. The threshold is specific to the model and is not intended to represent passenger-elevator capacity.

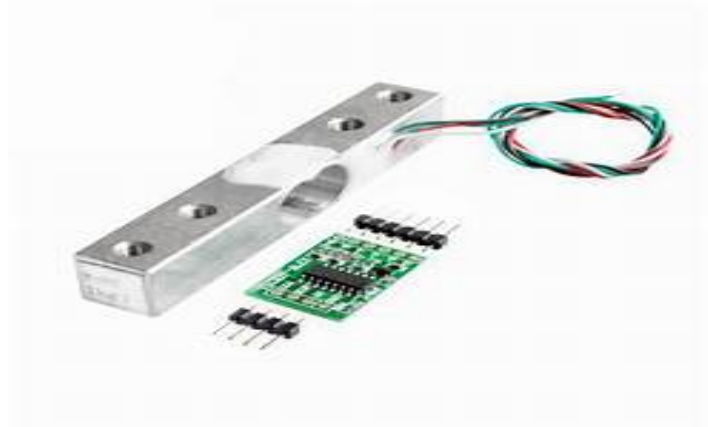


Figure 6. Weight-sensing arrangement used for overload monitoring.

3.6 Door Operation and User Feedback

The door is actuated by an SG90 servo motor. The control sequence commands approximately 90° for opening, maintains the open position for about three seconds, and then closes the door. A short buzzer indication accompanies closing. The display and LEDs communicate floor and safety status, while the DFPlayer Mini (Figure 7) provides a prerecorded floor announcement after arrival.

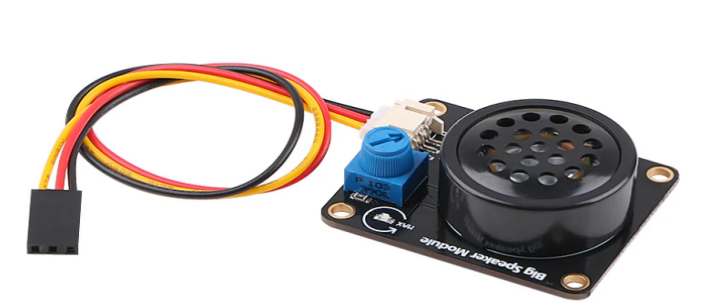


Figure 7. DFPlayer Mini audio module and speaker arrangement.

3.7 Software and Programming

The controller was programmed in C++ using the Arduino IDE (Figure 8). The implementation uses HX711.h for load-cell acquisition, DFRobot_DFPlayerMini.h for audio control, Servo.h for the door actuator, and timing functions for buzzer and weight-sampling tasks.

Software functions were separated into input acquisition, request handling, motion control, floor detection, door operation, safety monitoring, and user feedback. Push-button debouncing

uses a 50 ms delay. Floor-switch signals use interrupt-capable pins, and the motor routine selects direction by comparing the current floor with the active target floor.

Legacy IDE (1.8.X)



Figure 8. Arduino development environment used for programming the controller.

3.8 Experimental Testing and Evaluation Metrics

Testing was conducted at subsystem and integrated-system levels. Floor detection was evaluated over 40 transitions, button registration over 100 recorded presses, overload detection over 50 calibrated tests using loads around the 1 kg threshold, voice-announcement reliability over 40 arrivals, stop position over 40 runs, and complete system operation over 50 consecutive random floor-request cycles. Tests were selected to separate sensing accuracy from integrated behavior. A subsystem could therefore meet its local target even if an unrelated communication peripheral later caused a complete-cycle failure.

Accuracy was calculated as the proportion of successful events within each test set. Overall system success was defined as completed integrated cycles divided by the 50 attempted cycles. Stop-position performance was evaluated against the specified ± 5 mm tolerance and reported as the observed maximum deviation. The chosen metrics are simple proportions because the test sets are discrete pass/fail events. No inferential statistics were applied; the purpose was engineering verification of the prototype against predefined functional targets. All tests were conducted in a controlled prototype environment. The results therefore characterize model-scale functional performance and do not establish long-term durability, environmental robustness, or compliance with passenger-elevator certification requirements.

4. Results

4.1 Prototype Operation

The completed prototype (Figure 9) operated as an integrated unit using the 12 V supply arrangement. Floor calling, vertical motion, door actuation, position indication, overload monitoring, audible alerts, and voice announcements operated within the same control sequence. Floor requests were accepted and stored, and each successful arrival triggered motor stop, door operation, floor indication, and user feedback. The controller also handled multiple requests; in the tested upward sequence from ground, floor 1 was served before floor 2.



Figure 9. Completed prototype during practical demonstration.

4.2 Quantitative Performance

Table 2. Performance summary of the integrated elevator prototype.

Performance parameter	Measured result	Target	Outcome
Floor detection accuracy	100% (40/40 runs)	$\geq 98\%$	Pass
Button press accuracy	100% (100/100 presses)	100%	Pass
Overload detection accuracy	98% (49/50 tests)	$\geq 95\%$	Pass
Voice announcement reliability	97.5% (39/40 arrivals)	$\geq 95\%$	Pass
Stop position tolerance	± 2 mm	≤ 5 mm	Pass
Overall system success rate	96% (48/50 cycles)	$\geq 90\%$	Pass

As Table 2 shows, all measured performance indicators met their predefined prototype targets. Floor detection and button registration both reached 100%, overload detection reached 98%, voice announcements reached 97.5%, and stop position remained within ± 2 mm. The integrated prototype completed 48 of 50 random request cycles, corresponding to a 96% success rate.

4.3 Floor Detection

The mechanical floor-switch system produced 40 correct detections in 40 transitions. Response from switch activation to motor stop was approximately 50 ms, and no floor-detection failures were observed during the test set. This result is significant because floor detection terminates the motion state. Within the model-scale environment, the mechanical switches provided repeatable, unambiguous arrival signals without dependence on ambient optical conditions.

4.4 Button Registration and Debouncing

A 50 ms software debounce produced 100% registration across the 100 recorded button presses, with no false triggering. The approach was adequate for the low-noise prototype environment, although it should not be assumed to provide equivalent robustness in electrically noisy industrial settings.

4.5 Overload Detection

The HX711-based overload subsystem correctly classified 49 of 50 tests, giving 98% accuracy (Figure 10). The single error occurred close to the threshold, where an approximately 1.02 kg condition was read near 0.99 kg. Response time was below one second, indicating adequate prototype performance but also showing the sensitivity of threshold decisions to measurement noise.



Figure 10. Overload monitoring: normal condition below the threshold and overload condition above the threshold.

4.6 Voice Announcement Reliability

The DFPlayer Mini produced the correct floor announcement in 39 of 40 arrival tests, corresponding to 97.5% reliability. The isolated missed announcement was associated with a power-connection fault; normal floor sensing and motion control were unaffected.

4.7 Stop Position

The car stopped within ± 2 mm of the desired level in all 40 runs, remaining inside the ± 5 mm design tolerance. This accuracy was achieved without encoder feedback over the short model-scale travel distance, although occasional downward overshoot was observed and is discussed as a limitation of open-loop control.

4.8 Integrated-System Testing

Across 50 consecutive random request cycles, 48 were completed successfully, giving a 96% integrated-system success rate. The two unsuccessful cycles were linked to temporary DFPlayer serial synchronization problems. No integrated-cycle failures were attributed to floor sensing, motor direction, or door actuation.

Average travel time between adjacent floors was approximately 4.2 s, including the door sequence. Multiple pending requests were stored and processed, confirming that the controller could coordinate request handling with sensing, motion, and feedback rather than executing isolated single-floor demonstrations.

5. Discussion

5.1 Meaning of the Integrated Performance

The 96% integrated-cycle success rate is the principal system-level result because it reflects the complete chain of request acquisition, safety checking, movement, arrival detection, door operation, and user feedback. The failures were confined to the auxiliary audio subsystem, while the movement-critical functions remained operational. This separation suggests that the core control path was more robust than the communication-dependent announcement path. In systems terms, the integrated result is more informative than the individual percentages because it captures dependency among modules. A request cycle succeeds only when every required transition occurs in the correct order. Consequently, a 96% cycle success rate with two known audio-related failures provides a clearer diagnosis than an undifferentiated statement that the prototype was reliable.

Perfect floor detection over 40 transitions provided the discrete feedback needed to stop the motor at the intended floor. Mechanical switches are inexpensive and deterministic, which suits an educational model, but they introduce contact wear and alignment requirements. Non-contact sensing would be preferable for extended operation or higher-cycle testing. The 50 ms stop response after switch activation is also relevant because it links sensing to actuation latency. At the low speeds used in the model, that latency was small enough to maintain the observed positional tolerance, but the same delay would translate into greater travel distance at higher car speeds.

5.2 Request Handling and Control Logic

Queued directional handling gave the prototype behavior closer to a real elevator than a direct button-to-motor mapping. Requests remained stored while the car moved, and compatible upward calls could be served without an unnecessary reversal. The three-floor scale keeps the rule deliberately simple, allowing students to inspect how current floor, target floor, direction, and pending calls influence the next state. The logic also avoids an important teaching weakness of many small prototypes: hidden decision making. Because the active target and direction can be related directly to the queue, students can explain why a particular stop occurred and predict the next action before observing it.

The implementation should therefore be interpreted as a transparent heuristic for a single-car teaching model, not as an optimized elevator-group dispatch algorithm. Its purpose is deterministic and understandable behavior under a small request set, where correctness and observability are more important than minimizing large-building waiting-time metrics.

5.3 Safety Interlocks and Engineering Significance

The hardware interlocks are a central engineering feature. Door state, emergency stopping, and overload measurement influence whether motion is permitted, so safety is represented as a constraint on actuation rather than only as a software message. The near-threshold overload error also shows why calibration margin matters: when a decision boundary lies close to sensor noise, filtering and hysteresis become necessary even if average sensor performance is good. The overload test is particularly instructive because the only error occurred near the threshold rather than under clearly normal or clearly overloaded conditions. This pattern is consistent with a classification boundary affected by measurement noise, emphasizing the need for calibration, filtering, and hysteresis when a continuous sensor is converted into a binary safety decision.

At prototype scale, 98% overload accuracy met the design objective. Higher-consequence applications would require wider test ranges, repeated calibration, fault detection, redundant sensing or independent protective circuits, and verification against relevant safety requirements.

5.4 User Feedback and Accessibility

Visual and audible feedback broadened the prototype beyond basic motion control. The display identified the current floor, LEDs distinguished normal and overload states, the buzzer signaled safety conditions, and the DFPlayer provided floor announcements. Because the only integrated failures occurred in the audio path, the tests also illustrate an important systems lesson: auxiliary interfaces can limit overall reliability even when the primary control loop remains sound. From a human-machine-interface perspective, redundant channels are beneficial for teaching because they separate state information from safety notification. A floor display communicates location, LEDs show condition, the buzzer attracts attention, and speech provides an additional arrival cue. Improved connector retention, regulated power distribution, serial timeouts, and watchdog recovery would reduce this failure mode without changing the motion-control architecture.

5.5 Positioning Performance and Open-Loop Control

The ± 2 mm stopping tolerance shows that fixed-PWM open-loop control can be adequate for a short, lightly loaded teaching model. The same approach would become less reliable as travel distance, load range, speed, or mechanical variability increased because gravity, friction, and motor characteristics alter stopping distance. Open-loop control remains acceptable only because the mechanical and operating envelope is narrow. If load, speed, or travel distance increases, a fixed PWM duty cycle no longer guarantees the same deceleration distance. The observed downward overshoot is an early indication of this limitation. A rotary encoder and feedback controller would provide continuous position information and allow speed and stopping behavior to be corrected in real time. Such an extension would move the platform from discrete floor-position control toward closed-loop motion control and provide a stronger teaching case for servo-system design [30].

5.6 Comparison with Existing Work

Relative to the educational systems reviewed in Section 2, the present prototype combines several functions that are often demonstrated separately: request queuing, discrete floor feedback, door control, overload detection, physical emergency interruption, and multimodal user feedback. Its measured 96% integrated success rate is encouraging, but direct numerical ranking against other prototypes would be misleading because floor count, sensors, actuators, travel distance, test protocols, and success definitions differ. This integrated emphasis is the main point of comparison with recent educational elevator platforms. More advanced trainers may offer precision actuators, PLC/HMI environments, or cloud connectivity, but the present system demonstrates that meaningful control and safety experiments can be conducted with inexpensive components when the test protocol is explicit. The more useful comparison is architectural. The prototype extends a basic movement model by making safety conditions observable and testable and by reporting subsystem-level outcomes. This creates a clearer link between design requirements, implementation choices, and measured behavior.

5.7 Practical Implications

For engineering education, the platform supports practical work in finite-state control, interrupts, debouncing, threshold sensing, PWM motor drive, serial communication, servo actuation, and safety interlocking. For low-cost prototyping, it demonstrates that a single accessible controller can coordinate heterogeneous peripherals while preserving an

understandable control structure. The prototype can therefore support staged laboratory exercises: students may first verify individual inputs and outputs, then implement state transitions, then add safety permissions, and finally assess complete cycles. The same hardware can be reused to introduce fault injection, response-time measurement, and controller redesign. The prototype is not intended for passenger service. Its value lies in making elevator control architecture visible at model scale so that students can connect software states to physical motion, faults, and sensor feedback. The measured limitations also provide concrete starting points for advanced laboratory work on closed-loop control, sensor fusion, fault recovery, and remote monitoring. In that role, the measured imperfections are useful rather than incidental. Near-threshold load noise, occasional overshoot, and serial-audio faults provide realistic examples of why integrated systems require calibration, feedback, and error recovery even when individual modules appear simple.

5.8 Research Contribution

The study contributes an integrated and experimentally evaluated three-floor elevator teaching platform in which request handling, floor sensing, motor control, door actuation, overload monitoring, emergency stopping, visual indication, and voice feedback operate within one controller-based architecture. The system architecture is sufficiently explicit to support reconstruction and extension by other teaching laboratories. A second contribution is quantitative characterization. Instead of treating successful motion as sufficient evidence, the evaluation reports six performance dimensions: 100% floor detection, 100% button registration, 98% overload detection, 97.5% voice reliability, ± 2 mm stopping tolerance, and 96% integrated-cycle success. Reporting both subsystem and complete-cycle behavior also distinguishes local accuracy from system-level reliability.

The third contribution is pedagogical. The prototype exposes the relationship between software states and physical permissions, showing that safety depends on verified conditions rather than on command logic alone. This makes the platform suitable for laboratory work in mechatronics, embedded systems, automation, and introductory control engineering. The model thus links concepts that are often taught separately and makes their interactions experimentally visible. The contribution is intentionally implementation-oriented. It does not propose a new elevator scheduling theory or claim industrial certification; its significance lies in coordinated low-cost integration and transparent experimental evaluation at model scale (Table 3).

Table 3. Positioning of the study contribution relative to the reviewed project context.

Previously established / reviewed	Added by the present project
Elevator control uses automated decision logic and safety interlocks.	A physical three-floor model combining these principles in one educational prototype.
Microcontrollers can integrate sensors and actuators.	Arduino Mega implementation coordinating multiple elevator subsystems.
Floor requests can be managed through queue or directional logic.	Documented request queue with tested multi-request behavior.
Safety can use door and emergency interlocks.	Physical door, emergency-stop, and overload mechanisms integrated in the prototype.
Prototype operation can be demonstrated qualitatively.	Quantitative subsystem and integrated-cycle performance evaluation.

5.9 Limitations

The first limitation is scale. The three-floor structure, short travel distance, low load threshold, and lightweight mechanical assembly are representative of an academic prototype rather than a passenger elevator. The results should therefore be generalized only to comparable model-scale systems. Mechanical scaling also means that inertia, braking energy, door forces, and structural deflection differ substantially from full-scale systems. Second, position control is open loop between floor switches. Although stopping remained within ± 2 mm, occasional downward overshoot indicates sensitivity to gravity, friction, and load. Encoder feedback and closed-loop speed control would be required for higher precision or larger-scale operation. Third, overload detection uses a 1 kg prototype threshold and produced one near-threshold misclassification. The result is adequate for the present model but does not establish performance over a realistic passenger-load range or under long-term calibration drift.

Fourth, the DFPlayer Mini accounted for the only integrated-cycle failures. The test program was also limited in duration: 40 floor transitions, 100 button presses, 50 overload tests, 40 voice arrivals, 40 stop-position runs, and 50 integrated cycles are sufficient for functional characterization but not for durability or reliability-life estimation. Confidence intervals and life-cycle failure rates cannot be inferred from these sample sizes.

Finally, testing was performed in a controlled indoor environment and did not examine electrical interference, thermal variation, mechanical wear, switch fatigue, communication faults over extended operation, or environmental contamination. These factors would need separate evaluation before the architecture could be considered for more demanding experimental use. The current results should therefore be read as controlled functional validation, not as a reliability qualification.

6. Conclusion

A three-floor model elevator was designed, fabricated, programmed, and experimentally evaluated as a low-cost platform for embedded control and safety-oriented mechatronics education. The Arduino Mega coordinated queued floor requests, DC-motor motion, mechanical floor sensing, servo door operation, overload monitoring, emergency stopping, visual indicators, a buzzer, and floor announcements.

Subsystem performance was strong within the test conditions: floor detection and button registration reached 100%, overload classification reached 98%, voice announcements reached 97.5%, and stop position remained within ± 2 mm. The prototype completed 48 of 50 random integrated cycles, corresponding to 96% overall success; the two unsuccessful cycles were associated with the DFPlayer communication path rather than the movement-critical control functions. These results show that a modest microcontroller platform can represent the essential logic of a small elevator when sensing, state control, and physical interlocks are designed as one system. The platform is not a substitute for a certified elevator controller, but it provides a useful experimental environment for studying request scheduling, real-time inputs, safety permissions, fault behavior, and user feedback.

Future development should prioritize encoder-based closed-loop motion, non-contact position sensing, improved overload filtering and calibration, more robust audio fault recovery, and wireless monitoring. Longer-duration cycling and broader load testing would then allow the prototype to progress from functional demonstration toward a more rigorous laboratory testbed.

Overall, the project demonstrates that a carefully structured, low-cost prototype can move beyond simple elevator motion and provide a meaningful physical representation of control logic and safety principles. The results are relevant to educational and experimental settings and should not be interpreted as evidence of industrial elevator certification or readiness for passenger service.

Conflict of Interest

The authors declare no conflict of interests.

References

- [1] G. R. Strakosch and R. S. Caporale, Eds., *The Vertical Transportation Handbook*, 4th ed. Hoboken, NJ, USA: Wiley, 2010. doi: 10.1002/9780470949818.
- [2] G. Barney and L. Al-Sharif, *Elevator Traffic Handbook: Theory and Practice*. London, U.K.: Routledge, 2003.
- [3] W. Bolton, *Programmable Logic Controllers*, 6th ed. Oxford, U.K.: Elsevier, 2015.
- [4] D. Harel, "Statecharts: A visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, 1987. doi: 10.1016/0167-6423(87)90035-9.
- [5] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2nd ed. Cambridge, MA, USA: MIT Press, 2017.
- [6] M. Barr and A. Massa, *Programming Embedded Systems*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2006.
- [7] ASME, *ASME A17.1/CSA B44-2025: Safety Code for Elevators and Escalators*. New York, NY, USA: American Society of Mechanical Engineers, 2025.
- [8] British Standards Institution, *BS EN 81-20:2020, Safety Rules for the Construction and Installation of Lifts—Lifts for the Transport of Persons and Goods—Part 20: Passenger and Goods Passenger Lifts*. London, U.K.: BSI, 2020.
- [9] C. Egan, J. Lague, and M. K. Jouaneh, "Design and implementation of a model elevator system for mechatronics education," *Machines*, vol. 14, no. 5, p. 578, 2026. doi: 10.3390/machines14050578.
- [10] F. Sanfilippo, M. Økter, T. Eie, and M. Ottestad, "Teaching motion control in mechatronics education using an open framework based on the elevator model," *Machines*, vol. 10, no. 10, p. 945, 2022. doi: 10.3390/machines10100945.
- [11] M. Kelemen, M. Fabian, and T. Kelemenová, "Design and development of lift didactic model within subjects of mechatronics," *Procedia Engineering*, vol. 48, pp. 280–286, 2012. doi: 10.1016/j.proeng.2012.09.515.
- [12] L. Al-Sharif, T. A. Tutunji, D. M. Ragab, and R. Kayfi, "Using elevator system modelling and simulation for integrated learning in mechatronics engineering," in *Proc. 15th Int. Workshop Research and Education in Mechatronics (REM)*, 2014, pp. 1–8. doi: 10.1109/REM.2014.6920225.
- [13] R. A. Dimaculangan, M. L. C. Ruivivar, J. M. B. Gran, T. H. Fainsan, K. J. C. Cayetano, M. Natividad, R. J. Meimban, and D. J. O. Aquino, "Development of a four-storey elevator trainer for enhanced PLC and HMI programming skills of electronics engineering students," in *Proc. IEEE HNICEM*, 2025, pp. 1–5. doi: 10.1109/HNICEM64917.2024.11258831.
- [14] A. A. Titinchi, A. Abusalim, S. M. Yousuf, and A. Parameswaran, "An educational laboratory elevator with IoT/cloud utilizing," *International Journal of Computing*, vol. 24, no. 4, pp. 695–704, 2026. doi: 10.47839/ijc.24.4.4334.
- [15] A. Gharbi, "Exploring heuristic and optimization approaches for elevator group control systems," *Applied Sciences*, vol. 14, no. 3, p. 995, 2024. doi: 10.3390/app14030995.
- [16] M. Hanif and N. Mohammad, "Metaheuristic algorithms for elevator group control system: A holistic review," *Soft Computing*, vol. 27, no. 21, pp. 15905–15936, 2023. doi: 10.1007/s00500-023-08843-0.
- [17] Y. Wu and J. Yang, "Directional optimization of elevator scheduling algorithms in complex traffic patterns," *Applied Soft Computing*, vol. 158, p. 111567, 2024. doi: 10.1016/j.asoc.2024.111567.

- [18] S. Wang, X. Gong, M. Song, C. Y. Fei, S. Quaadgras, J. Peng, P. Zou, J. Chen, W. Zhang, and R. J. Jiao, "Smart dispatching and optimal elevator group control through real-time occupancy-aware deep learning of usage patterns," *Advanced Engineering Informatics*, vol. 48, p. 101286, 2021. doi: 10.1016/j.aei.2021.101286.
- [19] U. Reinsalu, T. Robal, and M. Leier, "Floor selection proposal for automated travel with smart elevator," in *Databases and Information Systems, Communications in Computer and Information Science*, vol. 1243. Cham, Switzerland: Springer, 2020, pp. 38–51. doi: 10.1007/978-3-030-57672-1_4.
- [20] T. Robal, U. Reinsalu, and M. Leier, "Towards personalized elevator travel with smart elevator system," *Baltic Journal of Modern Computing*, vol. 8, no. 4, pp. 675–697, 2020. doi: 10.22364/bjmc.2020.8.4.12.
- [21] Y. Zhou, K. Wang, and H. Liu, "An elevator monitoring system based on the Internet of Things," *Procedia Computer Science*, vol. 131, pp. 541–544, 2018. doi: 10.1016/j.procs.2018.04.262.
- [22] M. Zihan, H. Shaoyi, Z. Zhanbin, and X. Shuang, "Elevator safety monitoring system based on Internet of Things," *International Journal of Online and Biomedical Engineering*, vol. 14, no. 8, pp. 121–133, 2018. doi: 10.3991/ijoe.v14i08.9179.
- [23] W. Yao, V. Jagota, R. Kumar, D. Ather, V. Jain, S. J. Quraishi, and J. Osei-Owusu, "Study and application of an elevator failure monitoring system based on the Internet of Things technology," *Scientific Programming*, vol. 2022, p. 2517077, 2022. doi: 10.1155/2022/2517077.
- [24] M. B. Marinov, D. N. Nikolov, B. T. Ganey, and T. S. Djamiykov, "Smart multisensor node for remote elevator condition monitoring," in *Proc. 21st Int. Symp. Electrical Apparatus & Technologies (SIELA)*, 2020, pp. 1–4. doi: 10.1109/SIELA49118.2020.9167049.
- [25] A. Hughes and B. Drury, *Electric Motors and Drives: Fundamentals, Types and Applications*, 5th ed. Oxford, U.K.: Newnes, 2019. doi: 10.1016/C2017-0-03226-3.
- [26] F. Sanfilippo and K. Austreng, "Sustainable approach to teaching embedded systems with hands-on project-based visible learning," *International Journal of Engineering Education*, vol. 37, no. 3, pp. 814–829, 2021.
- [27] F. Sanfilippo and K. Austreng, "Enhancing teaching methods on embedded systems with project-based learning," in *Proc. IEEE Int. Conf. Teaching, Assessment, and Learning for Engineering (TALE)*, Wollongong, Australia, 2018, pp. 169–176.
- [28] M. Prince, "Does active learning work? A review of the research," *Journal of Engineering Education*, vol. 93, no. 3, pp. 223–231, 2004. doi: 10.1002/j.2168-9830.2004.tb00809.x.
- [29] E. F. Crawley, J. Malmqvist, S. Östlund, D. R. Brodeur, and K. Edström, *Rethinking Engineering Education: The CDIO Approach*, 2nd ed. Cham, Switzerland: Springer, 2014. doi: 10.1007/978-3-319-05561-9.
- [30] J. Ma, X. Li, and K. K. Tan, *Advanced Optimization for Motion Control Systems*. Boca Raton, FL, USA: CRC Press, 2020.