Driverless Metro Train to Shuttle Between Stations

Roopa B S[1], Bhanupriya K R[2], Bindu K[3], Kruthika K[4], Pooja S G[5]
[1]*Assistant Professor, Dept. of Electronics and Communication Engineering, JNN College of Engineering, Shivamogga, Karnataka, India*
[2,3,4,5]*Dept. of Electronics and Communication Engineering, JNN College of Engineering, Shivamogga, Karnataka, India*

***Abstract*—This paper presents a low-cost prototype of a driverless metro train that shuttles between programmed stations without a human driver. An ESP32 microcontroller acts as the central controller and integrates an EM-18 RFID reader, an HC-SR04 ultrasonic sensor, an L298N motor driver with DC motors, a servo-operated door, a 16×2 LCD, and Wi-Fi-based Telegram messaging. The prototype models a five-station route that can be travelled in forward and reverse directions. The first RFID scan of a card records the entry station and increments the passenger count; the next scan records the exit station, computes the fare from the number of stations travelled, and decrements the count. The ultrasonic sensor stops the motors when an obstacle comes within 50 cm, and Telegram notifications report start-up, station arrival, passenger entry and exit, fare, and obstacle events. The work demonstrates the integration of embedded control, automated fare collection, passenger counting, obstacle sensing, automatic doors and IoT monitoring on one platform. It is an educational prototype; a deployable system would need certified train control, accurate localisation, and redundant safety systems.**

***Index Terms*—Automatic door control, Driverless metro train, IoT monitoring, Passenger counting, RFID smart ticketing, Telegram bot, Ultrasonic obstacle detection**

I. INTRODUCTION

Urban rail transport is moving steadily towards automated operation because of the need for reliable service, efficient passenger handling and better monitoring. Driverless metro systems combine automatic train control, station management, passenger information, communication networks and safety systems. Commercial systems use certified, expensive technology, but the underlying ideas can be reproduced at prototype level with microcontrollers and low-cost sensors.

Student prototypes usually demonstrate only one function, such as motor movement, RFID ticketing, or obstacle sensing. Such isolated demonstrations do not show how passenger events, vehicle movement, safety conditions, and remote monitoring work together. This paper describes a prototype in which an ESP32 controls a small vehicle travelling through five stations while also handling RFID-based ticketing, passenger counting, fare calculation, an automatic door, obstacle detection, a local LCD, and Telegram notifications over Wi-Fi.

The main objectives are: (1) to build a motorised metro train prototype capable of automatic movement; (2) to detect obstacles with an ultrasonic sensor for collision avoidance; (3) to implement station-based stopping and bidirectional movement; (4) to integrate RFID smart ticketing with passenger counting; and (5) to implement automatic door control and real-time IoT monitoring.

II. LITERATURE SURVEY

Deepthi and Reddy examined RFID and IoT for passenger tracking and online payment in railways, using RFID tags as transit cards and DBSCAN-based crowd-demand analysis [1]. Wu and Liu mined smart-card data to estimate transfer passenger flow in a metro network, showing that fare-collection records can be turned into operational information [2]. Sun et al. proposed a state-transition model that uses automatic fare collection data for real-time crowd counting in a metro system [3].

Vision-based counting has also been studied. Radovan et al. built a single-board-computer passenger counter using YOLO and DeepSORT [4] and reviewed image-processing and machine-learning approaches to passenger counting in public transport [5]. Video-

based counting for on-bus and off-bus deployment is compared in [6]. These works rely on cloud analytics or heavy computation. The present prototype instead derives occupancy directly from RFID entry and exit events on a low-cost ESP32, which is computationally light but depends on passengers presenting their cards correctly.

III. PROPOSED SYSTEM

A. System Architecture

The architecture is built around the ESP32 as the central decision-making unit, following a closed-loop concept: sensors provide observations, the ESP32 evaluates the system state, and actuators respond (Fig. 1). The inputs are the EM-18 RFID reader, the ultrasonic sensor, the LCD interface, and the Wi-Fi link. The outputs are the motor driver with DC motors, the servo door, the passenger count and fare logic, and the five-station route simulation.

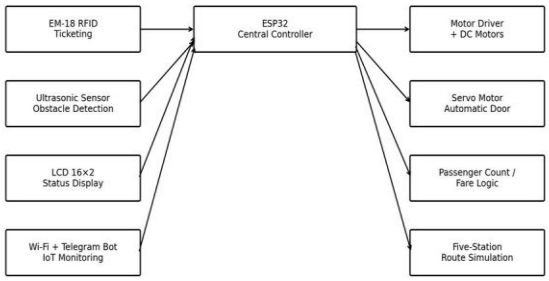


Fig. 1. System architecture of the driverless metro prototype

B. Hardware Components

The components and their roles are summarised in Table I. The ESP32 reads the RFID and ultrasonic inputs, drives the motor driver and servo, updates the LCD, and sends messages over Wi-Fi. Because GPIO pins cannot supply motor current, an L298N driver switches the two DC motors that represent the traction system. A servo with predefined open (90°) and closed (0°) angles acts as the door. Stable supply rails with a common ground are used because motor start-up and servo movement cause voltage dips.

Table I. Hardware Components and Their Roles

Component	Role in the prototype
ESP32 board	Central controller, PWM, Wi-Fi
EM-18 RFID reader, cards	Contactless passenger ID (serial)
HC-SR04 ultrasonic sensor	Forward obstacle distance
L298N driver, 2 DC motors	Forward, reverse, and stop
Servo motor	Automatic door actuation
16×2 LCD (I2C)	Station, count, fare, alerts
Regulated power supply	Logic, motor and servo rails

C. Software and Control Flow

The firmware is written for the Arduino/ESP32 environment using the ESP32Servo, Liquid Crystal, WiFi and Universal TelegramBot libraries. After initialising all modules, the door is closed, the motors are stopped, and the passenger count is zero. The main loop checks the obstacle sensor first, then route progress, then the RFID reader, and updates the LCD and Telegram only when a meaningful event occurs. Safety handling therefore has priority over normal movement (Fig. 2).

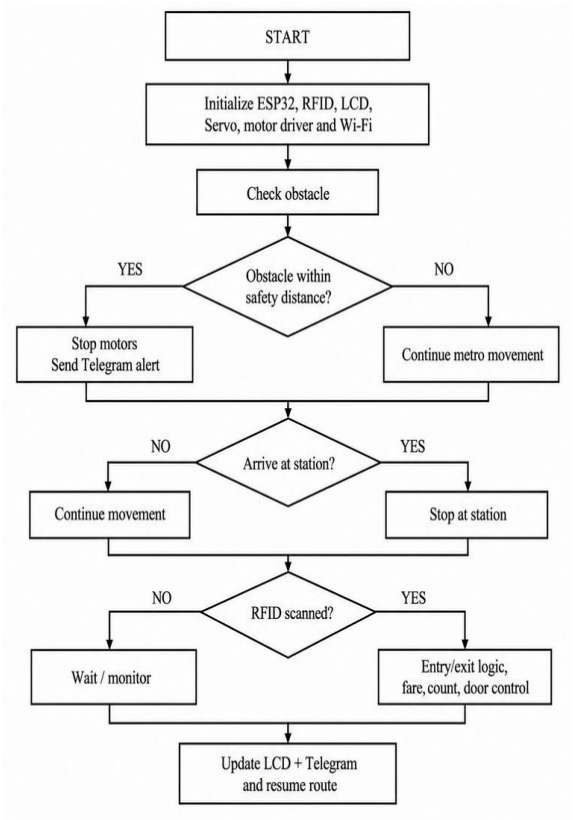


Fig. 2. Flowchart of the driverless metro prototype

IV. IMPLEMENTATION

A. Route and Movement Control

The route has five stations. Moving forward, the station index increases from 1 to 5; at Station 5, the direction is reversed, and the metro returns to Station 1. Between stations, the controller drives the motors forward or in reverse, and on arrival it stops, shows the station on the LCD, sends an arrival message, opens the door, and accepts RFID scans for a 5 s dwell period before closing the door and moving on. In the prototype, inter-station travel is simulated by a programmed distance counter rather than by physical position feedback.

B. RFID Ticketing and Passenger Counting

Each card is a passenger token. The controller searches its table of active passengers (up to 20 records, each holding card ID, entry station, and active flag). If the card is not active, the scan is an entry: the current station is stored, and the passenger count is incremented. If it is active, the scan is an exit: the current station is the destination, the fare is computed, the record is cleared, and the count is decremented, never below zero. A 3 s cooldown ignores repeated reads of the same card.

C. Fare Calculation

The fare depends on the number of stations travelled. With entry station E and exit station X, the distance D and the fare F are

$$D = |X - E| \quad (1) \qquad F = D \times R \quad (2)$$

where R is the rate per station (Rs. 10 in the implemented firmware). For a journey from Station 1 to Station 4, D = 3 and F = Rs. 30. Because the absolute difference is used, the fare is independent of travel direction.

D. Obstacle Detection and Safety

The HC-SR04 is riggered with a 10 µs pulse and the echo time is converted to distance using $d = (t \times 0.0343) / 2$ (3) where t is the echo time in microseconds and d is the distance in cm. If d is below 50 cm the motors are stopped, the LCD shows OBSTACLE and a single Telegram alert is sent. The metro resumes only after the path clears, and a “path clear” message is sent. A timed-out echo is treated as 999 cm (no obstacle) in the current firmware; a stricter fail-safe would stop the metro on invalid readings.

E. Automatic Door and IoT Monitoring

The servo opens the door for boarding or exits and closes it after 5 s. The ESP32 joins the configured Wi-Fi network (15 s timeout) and uses the Telegram Bot API to report system start-up, station arrival, passenger entry and exit with fare, and obstacle events. Telegram is used only for monitoring; movement and obstacle response run locally, so they continue if the network is unavailable.

F. Cost Estimate

The prototype was built from low-cost, readily available modules. Its estimated cost is Rs. 5,430, itemised in Table III. Components account for Rs. 2,880 (about 53%) and fabrication of the chassis, track, station models and door frame for Rs. 1,200; all software tools used are free.

V. RESULTS AND DISCUSSION

A. Motorised Prototype

The prototype consists of a wooden platform on four wheels carrying the ESP32, L298N driver, LCD and ultrasonic sensor (Fig. 3). During testing it demonstrated starting, moving and stopping under ESP32 control. Automatic movement between stations on a fixed track is identified as further development.

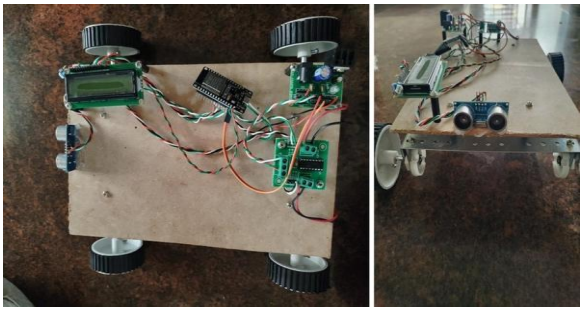


Fig. 3. (a) Motorised metro train prototype and (b) front-mounted ultrasonic sensor

B. Obstacle Detection and Station Display

The ultrasonic sensor is mounted at the front of the platform (Fig. 3b); the controller stops the motors when an object is within the 50 cm threshold. The LCD displays the current station during operation, as shown for Stations 1 and 2 in Fig. 4. Station stopping and

direction reversal are implemented as programmed route logic; their integration with a physical track remains to be completed.

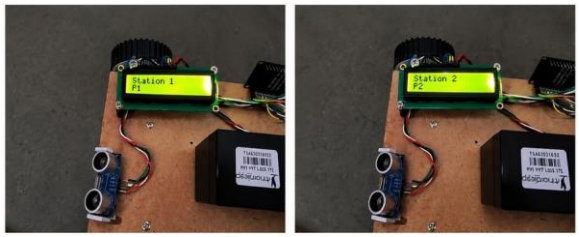


Fig. 4. LCD showing Station 1 and Station 2

C. RFID Ticketing and Passenger Counting

Presenting an RFID card to the EM-18 reader (Fig. 5) registers entry or exit, updates the passenger count and, on exit, computes the fare. The Telegram log in Fig. 7 shows two passenger entries, an obstacle alert at Station 2 and subsequent exit records with a fare of Rs. 10 per station travelled.

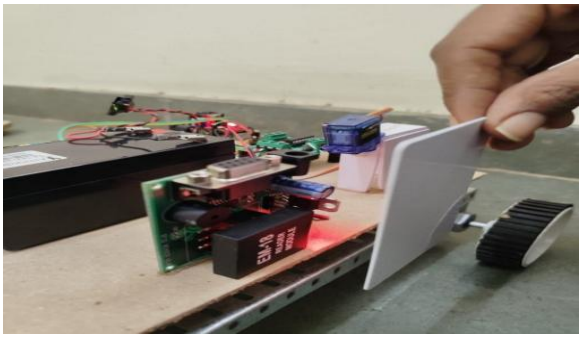


Fig. 5. RFID card being scanned for ticketing and passenger counting

D. Automatic Door and Telegram Monitoring The servo-driven door opens at stations during passenger events and closes after the dwell interval (Fig. 6). Telegram notifications give remote visibility of train status and passenger activity (Fig. 7).

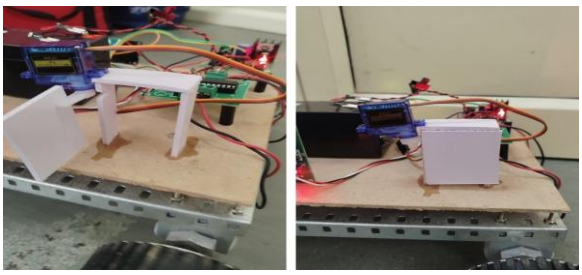


Fig. 6. Servo-operated automatic door (closed and open)



Fig. 7. Telegram bot notifications for entry, obstacle and exit events

E. Test Cases

The behaviours defined for system testing are listed in Table II.

Table II. Test Cases and Expected Behaviour

Test	Expected behaviour
System start-up	System enters ready state
RFID entry	Entry station stored, count +1
RFID exit	Fare calculated, count −1
Invalid card	Rejected, count unchanged
Repeated scan	Duplicate within cooldown ignored
Obstacle detected	Motors stop, status displayed
Obstacle cleared	Movement resumes
Door control	Servo opens, then closes
Station arrival	Motor stops, station shown
Wi-Fi unavailable	Local control continues
Terminal station	Direction reverses

F. Discussion

The main strength of the prototype is functional integration: RFID, counting, fare calculation, door, obstacle response, route control, LCD and IoT monitoring share one controller and one system state, giving a clearer picture of an intelligent transport workflow than separate demonstrations. Event-driven handling also avoids repeated Telegram messages.

The prototype has clear limits. Timing-based localisation is affected by wheel slip, battery voltage, and friction, so it is a route simulation rather than precise navigation. A single ultrasonic sensor cannot guarantee full obstacle coverage. The passenger count

is valid only if passengers use their cards correctly, and a bare RFID ID is not secure authentication.

VI. CONCLUSION AND FUTURE SCOPE

An ESP32-based driverless metro prototype was developed that combines five-station route logic, RFID ticketing with automatic passenger counting and fare calculation, a servo-operated door, ultrasonic obstacle stopping, an LCD, and Telegram-based remote monitoring. The work shows how embedded control and IoT communication can be combined in a compact intelligent transport model, while making clear the gap between a laboratory prototype and a certified railway system.

Future work includes wheel encoders or RFID station markers for accurate localisation; LiDAR, radar or multiple ultrasonic sensors for wider obstacle coverage; camera-based passenger counting; door position feedback and interlocks; emergency stop and redundant sensors; secure RFID credentials with a cloud database and web dashboard; capacity-based boarding control; and battery and motor health monitoring.

REFERENCES

[1] V. Deepthi and G. S. Reddy, “An improvised analysis of smart data for IoT-based railway system using RFID,” 2024.

[2] Y. Wu and T. Liu, “Mining smart card data to estimate transfer passenger flow in a metro network,” *IET Intelligent Transport Systems*, 2024.

[3] Sun *et al*., “STRmt: A state transition-based model for real-time crowd counting in a metro system,” *Concurrency and Computation: Practice and Experience*, 2024.

[4] Radovan, L. Mrsic, G. Dambic, and B. Mihaljevic, “A review of passenger counting in public transport concepts with solution proposal based on image processing and machine learning,” *Eng*, 2024.

[5] Radovan *et al*., “Creation of a high-precision, single-board computer-based intelligent passenger counting system,” *Procedia Computer Science*, 2024.

[6] “Video-based automatic people counting for public transport: On-bus versus off-bus deployment,” *Computers in Industry*, 2025.