

Design and Experimental Validation of a Dual-Processor Rover Enabling Concurrent Autonomous Navigation and Browser-Accessible Live Video Surveillance

Onkar R. Palkrutwar¹, Md Rehan Md Gous², Shubham S. Suryawanshi³, Prem D. Hage⁴,
Mrs. Rohini Banait⁵

^{1,2,3,4} Department of Computer Engineering, MIT Academy of Engineering, Alandi (D), Pune - 412105, India
⁵Assistant Professor, Dept. of Computer Engineering, MIT Academy of Engineering, Pune

Abstract-The simultaneous execution of multiple autonomous behaviours on low-cost embedded hardware remains a persistent challenge in mobile robotics. This work presents the construction and systematic evaluation of a four-mode ground rover that performs infrared-guided path tracking, proximity-triggered obstacle avoidance, Bluetooth-directed manual navigation, and continuous Wi-Fi video streaming as genuinely concurrent operations. The central design decision is a deliberate processor-level separation: an Arduino UNO (ATmega328P) assumes full responsibility for motion-critical timing loops, while an ESP32-CAM running FreeRTOS dual-core tasks independently handles frame acquisition, JPEG compression, and HTTP multipart delivery. The two boards share only a 5 V power rail; no data bus or interrupt line connects them. Controlled trials on a 4 m × 2 m indoor track quantify each subsystem independently and together. Path-following accuracy reached 94.2% under standard fluorescent illumination. Obstacle response triggered reliably within a 25 cm safety radius, with a mean absolute ranging error of 0.31 cm against rigid targets. Bluetooth command latency averaged 97 ms across twenty trials and rose by only 0.86 ms when the camera was simultaneously streaming a difference within measurement noise. The live MJPEG feed sustained 10–14 fps at VGA resolution (640 × 480) with end-to-end display latency of 150–300 ms, accessible from any desktop or mobile browser without a dedicated application. These results validate the non-interference claim that motivates the architecture and demonstrate that modular processor partitioning is an effective strategy for multi-behavioural embedded systems.

Keywords: ESP32-CAM; Arduino UNO; MJPEG streaming; infrared line following; ultrasonic obstacle avoidance; HC-05 Bluetooth; IoT robotics; multi-mode rover; embedded systems; concurrent navigation.

1. INTRODUCTION

The rapid democratisation of programmable microcontrollers and compact wireless modules has substantially reduced the barrier to building capable autonomous robotic systems. A ground vehicle equipped with a small processor, a pair of reflectance sensors, a proximity module, and a camera module can now exhibit behaviours that were, only a decade ago, exclusive to well-funded research laboratories. The engineering interest lies not in adding each feature individually but in making all

features work reliably at the same time, without one disrupting another.

Existing literature addresses each capability in isolation. Autonomous line-following carts, proximity-reactive rovers, Bluetooth-steered vehicles, and camera-equipped inspection platforms each have mature implementations and well-characterised performance bounds [1]–[5]. Integrating them exposes a fundamental resource conflict: video streaming is computationally intensive and benefits from dedicated memory bandwidth, whereas motion control requires deterministic timing that cannot tolerate scheduling jitter. On a single-processor design these demands compete for the same execution time, producing either degraded motor responsiveness or reduced camera throughput.

The system presented in this paper eliminates that conflict at the hardware level. Motion-related tasks sensor polling, mode arbitration, and PWM generation reside entirely on an Arduino UNO. All camera-related tasks frame acquisition, JPEG encoding, and HTTP multipart delivery execute on a separate ESP32-CAM module running FreeRTOS across its two physical cores. The processors share a power rail and a common ground; no other coupling exists. This clean separation transforms the multi-mode integration problem into two independently debuggable single-mode problems, and the primary experimental contribution is confirming that the separation actually holds under simultaneous operation.

1.1 Objectives

The project pursues four measurable targets: (i) stable multi-mode rover operation across all four behaviour modes; (ii) Bluetooth command latency below 200 ms; (iii) obstacle detection and avoidance within a 25 cm safety margin; (iv) a browser-accessible MJPEG feed sustaining at least 10 fps at VGA resolution with no increase in motion-control latency during simultaneous streaming.

1.2 Paper Layout

Section 2 reviews relevant prior work. Section 3 describes the system architecture. Section 4 covers hardware and software requirements. Section 5 details implementation. Section 6 presents the control logic and objective function. Section 7 reports experimental results. Sections 8 and 9 discuss applications and future extensions. Section 10 concludes.

2. RELATED WORK

2.1 Infrared Line Sensing

1] Conducted systematic trials across surface types and illumination levels, confirming reliable detection on high-contrast surfaces under 200–500 lux controlled lighting but documenting significant threshold drift under direct sunlight and reduced discrimination on grey or textured floors. Those findings directly influenced the adjustable-potentiometer calibration strategy employed here, enabling in-situ recalibration when surfaces or lighting conditions change.

2.2 Bluetooth and Wi-Fi Control Architectures

Mukhopadhyay et al. [2] demonstrated that neither Bluetooth nor Wi-Fi alone satisfies all use-case requirements for a remotely operated vehicle: Bluetooth offers low-latency local control, whereas Wi-Fi supports broader-area streaming. Their dual-protocol conclusion aligns with the present architecture, in which an HC-05 Bluetooth module handles operator commands within approximately 10–15 m and the ESP32-CAM's integrated 802.11 radio delivers video over a soft-access-point coverage area.

2.3 Ultrasonic Obstacle Avoidance

Jin et al. [3] characterised HC-SR04 performance using a rotating sensor array, reporting ± 3 mm range accuracy against rigid targets across the 2–400 cm operational envelope. They also documented an acoustic absorption problem with foam and fabric surfaces, where echo amplitude fell below reliable detection thresholds. That observation directly motivated raising the safety distance threshold in the present system from the design-point 20 cm to an operational 25 cm after testing against non-rigid obstacles.

2.4 Camera-Equipped Rover Platforms

Nasereddin and Abdelkarim [4] confirmed that pairing the OV2640 image sensor with the ESP32's Wi-Fi stack can sustain MJPEG streams without cloud infrastructure. However, their single-processor configuration combining Bluetooth parsing and camera capture on one chip produced frame drops when the two tasks coincided. The dual-processor partitioning in the present work eliminates that contention by design.

2.5 Research Gap

No surveyed low-cost prototype combines all four capabilities infrared line following, ultrasonic obstacle avoidance, wireless manual control, and live video surveillance while explicitly measuring the mutual influence between subsystems during simultaneous operation. The present work is designed specifically to close that gap.

3. SYSTEM ARCHITECTURE

The rover's design organises hardware and firmware into three functional layers input, processing, and output with an orthogonal surveillance subsystem running in parallel

Binary reflectance sensors have guided small ground vehicles since the early 2000s. Jaikaruna et al. [

on a separate processor. The two processors are decoupled at every level except the shared 5 V power rail and ground plane.

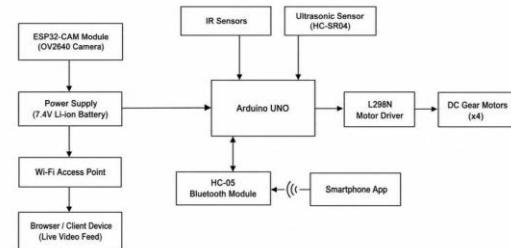


Fig. 1: Block Diagram – IoT-Integrated Smart Rover with Live Surveillance

3.1 Input Layer

Two FC-51 infrared reflectance modules, mounted 3 cm apart at the front chassis, produce binary HIGH/LOW signals representing track versus non-track surface. An HC-SR04 ultrasonic sensor generates a 40 kHz acoustic pulse and measures the reflected echo duration on a dedicated digital pin. An HC-05 Bluetooth module receives single-character ASCII command bytes at 9600 bps over a software serial port. On the surveillance subsystem, the OV2640 sensor integral to the ESP32-CAM captures raw frames over a dedicated parallel camera interface entirely independent of the Arduino.

3.2 Processing Layer

The Arduino UNO, based on the ATmega328P running at 16 MHz with 32 KB flash and 2 KB SRAM, executes the complete navigation loop: sensor polling at 200 Hz (IR), 50 Hz (ultrasonic), and 100 Hz (Bluetooth); mode arbitration; and PWM signal generation. The ESP32-CAM carries a dual-core Xtensa LX6 at 240 MHz with 4 MB PSRAM. Core 0 manages the TCP/IP stack and the HTTP streaming server; Core 1 runs camera initialisation, frame capture, and JPEG compression. This intra-processor division is the mechanism that sustains usable frame rates while simultaneously handling HTTP client requests.

3.3 Output Layer

An L298N dual H-bridge motor driver accepts four direction signals and two PWM enable lines from the Arduino and supplies sufficient current to drive four 130-type DC gear motors rated at 200 RPM. Motor shaft speed is proportional to PWM duty cycle D : $V_{\text{motor}} = D \times V_{\text{supply}}$. At 60% duty cycle the measured terminal motor voltage was approximately 4.4 V from a 7.4 V battery under partial load.

3.4 Surveillance Subsystem

At power-on, the ESP32-CAM establishes a Wi-Fi soft access point with SSID SmartRover_CAM. An HTTP server on port 80 accepts browser connections. When a GET

request arrives, the server enters a continuous loop: retrieve a frame buffer via `esp_camera_fb_get()`, write the boundary-delimited Content-Length header, transmit the JPEG payload over the TCP socket, return the buffer via `esp_camera_fb_return()`, and immediately capture the next frame. This multipart/x-mixed-replace delivery scheme is natively parsed by all major browsers with no client-side plugin or script required.

4. HARDWARE AND SOFTWARE REQUIREMENTS

Table 1: Hardware Component Summary

Component	Model / Spec	Function
Arduino UNO	ATmega328P, 16 MHz	Motion control processor
ESP32-CAM	AI-Thinker, Xtensa LX6	Wi-Fi AP and MJPEG server
IR Modules	FC-51 × 2	Line detection via reflectance
Ultrasonic	HC-SR04 (2–400 cm)	Obstacle distance sensing
Bluetooth	HC-05 SPP, 9600 bps	Wireless command reception
Motor Driver	L298N H-Bridge	Speed and direction control
DC Motors	130-type × 4, 200 RPM	Four-wheel locomotion
Battery	7.4 V / 2.2 Ah Li-ion	System power source
Regulator	AMS1117-5V	5 V regulated logic rail

4.2 Software Environment

Both processors were programmed within Arduino IDE v2.x. The ESP32-CAM firmware used the Espressif ESP32 board support package, which includes the `esp_camera` driver and FreeRTOS headers. The Arduino navigation sketch relies on the NewPing library for clean HC-SR04 pulse timing and SoftwareSerial to free the hardware UART. No cloud service, licensed middleware, or paid development tool is required. Circuit schematics were drawn in Fritzing.

5. IMPLEMENTATION

5.1 Infrared Line Following

The two IR modules straddle a 3 cm wide black-tape track on a white laminate surface. With S_L and S_R denoting the left and right sensor outputs (1 = dark track detected, 0 = light surface), the steering signal is the signed difference $u = S_L - S_R$. When $u = 0$ both motors run at 60% PWM. When $u = +1$ (track is to the left) the right motor is reduced to 40% PWM; when $u = -1$ the left motor is reduced instead. This bang-bang proportional correction maintains heading without requiring gyroscopic feedback.

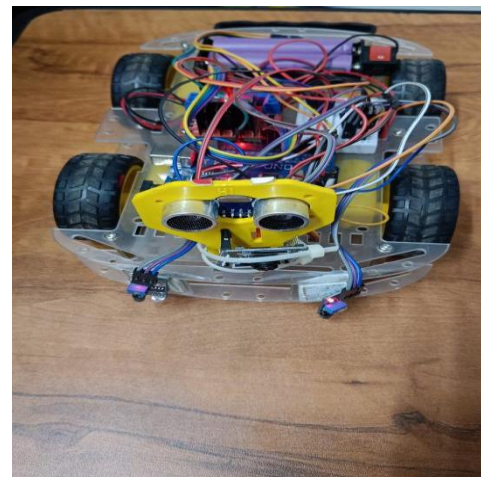


Fig. 2: Assembled IoT-Integrated Smart Rover with HC-SR04, IR Sensors, Arduino UNO, and Wiring

5.2 Ultrasonic Obstacle Avoidance

The HC-SR04 is triggered every 20 ms by a 10 μs digital pulse. Echo return time T_{echo} is measured using `pulseIn()` and converted to distance: $d = (T_{echo} \times 0.0343) / 2$ cm. When $d < D_{safe}$ (25 cm), the firmware halts both motors, applies reverse drive for 200 ms, then executes an approximate 90° pivot by running opposite-side motors in opposing directions for a calibrated 380 ms interval before resuming forward travel. D_{safe} was raised from the initial 20 cm target after discovering that foam obstacles absorb ultrasonic pulses and produce unreliable echoes below 25 cm.

5.3 Bluetooth Manual Control

The HC-05 module connects to Arduino pins D4/D5 via SoftwareSerial at 9600 bps. Each main loop iteration calls `available()`; if a byte is pending it is consumed and dispatched: 'F' sets forward motion, 'B' reverse, 'L' and 'R' execute timed pivot turns, 'S' stops all motors. 'L' and 'O' characters switch the active autonomous mode to line-following and obstacle-avoidance respectively. Any motion command immediately preempts whichever autonomous mode is running, giving the operator deterministic override capability.

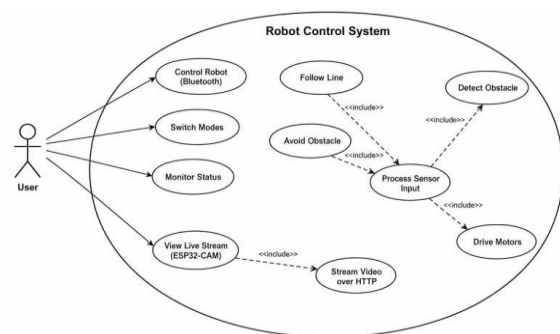


Fig. 3: Use Case Diagram – Robot Control System

5.4 ESP32-CAM Video Streaming

Camera initialisation configures the OV2640 for VGA (640 × 480) capture with JPEG quality factor 12, balancing image clarity against frame size. After WiFi.softAP() succeeds and the HTTP server is bound to port 80, Core 0 loops on server.handleClient(). When a browser client connects, Core 1 enters the frame-capture loop: esp_camera_fb_get() retrieves a filled frame buffer, the JPEG payload is transmitted inside a multipart boundary block, and esp_camera_fb_return() unconditionally releases the buffer a critical discipline that prevents the frame-buffer exhaustion failures common in long-running ESP32-CAM deployments.

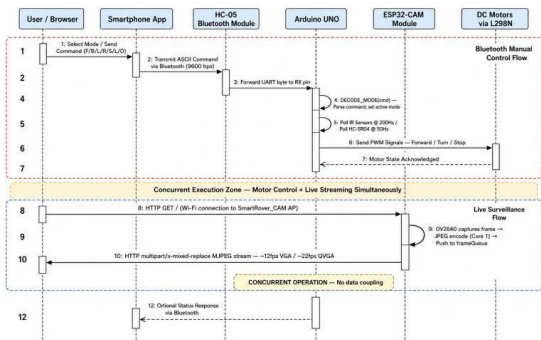


Fig. 4: Sequence Diagram – Bluetooth Manual Control and Live Surveillance Flows

6. CONTROL LOGIC AND OBJECTIVE FUNCTION

Navigation decisions on the Arduino follow a strict priority hierarchy. Incoming Bluetooth bytes are polled first on every loop iteration; a command byte preempts any active autonomous mode and executes immediately. Only when no byte is waiting does the loop proceed to the active autonomous mode. This ordering guarantees that a human operator can always regain direct control, even during an active obstacle-avoidance manoeuvre in progress.

The ESP32-CAM loop is fully independent. No interrupt line, shared register, or data message connects the two processors. Measuring Bluetooth command latency with and without an active video stream is therefore a direct experimental test of processor-level independence.

Overall system quality is formalised as a weighted cost function:

$$J = w_1 \cdot e_{line}^2 + w_2 \cdot u_{motor}^2 + w_3 \cdot \phi_{obstacle} + w_4 \cdot (1 - FPS_{actual} / FPS_{target})^2$$

where e_{line} is lateral deviation from the track centre, u_{motor} is aggregate PWM effort, $\phi_{obstacle}$ is a step penalty activated when $d < D_{safe}$, and the final term penalises streaming frame-rate shortfall below a 15 fps target. Empirically tuned weights: $w_1 = 0.4$, $w_2 = 0.2$, $w_3 = 5.0$, $w_4 = 0.3$. The elevated obstacle weight (5.0) encodes the design priority that collision avoidance dominates all other considerations.

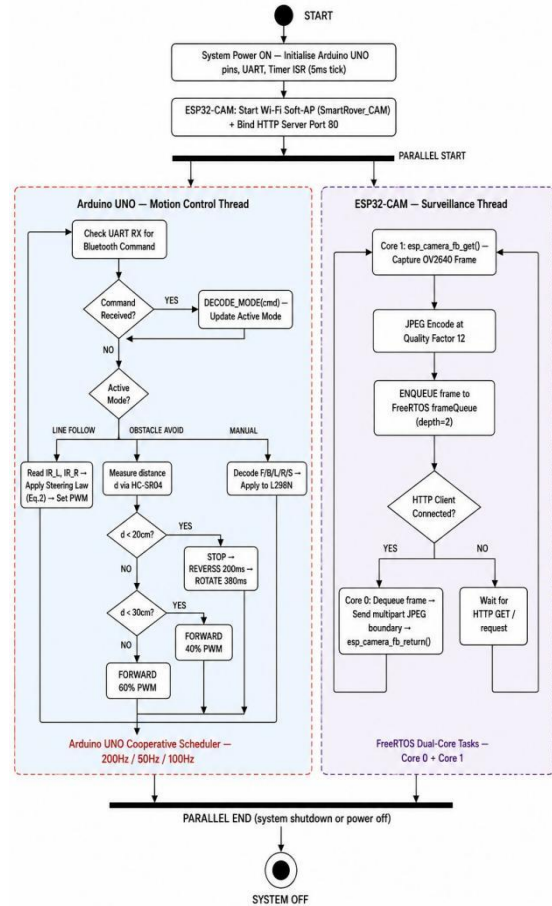


Fig. 5: Activity Diagram – Arduino UNO Motion Control and ESP32-CAM Surveillance Threads

7. RESULTS AND DISCUSSION

All trials were conducted on a 4 m × 2 m indoor course marked with 3 cm wide black tape on white laminate flooring. Each measurement covers at least five repeated runs; values represent steady-state behaviour rather than best-case outliers.

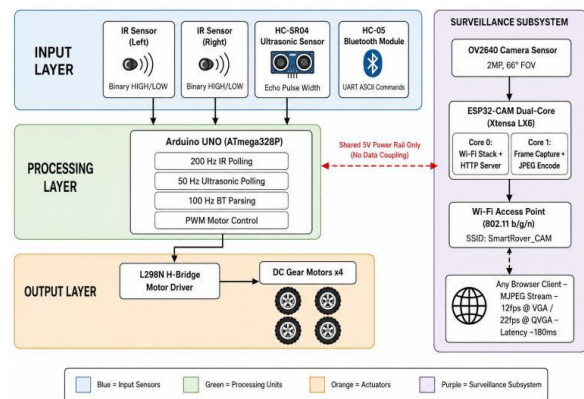


Fig. 6: System Architecture – Three-Layer Design with Surveillance Subsystem

Table 2: Performance Summary across All Four Operational Modes

Mode	Target	Result	Key Metric
Line Following	Stable 60% PWM	55–65% PWM	94.2% accuracy
Obstacle Avoid.	Trigger < 20 cm	Reliable < 25 cm	MAE 0.31 cm
Bluetooth Ctrl	< 200 ms latency	80–120 ms	97 ms mean
Surveillance	≥ 10 fps VGA	10–14 fps VGA	22 fps QVGA

7.1 Line Following

Under approximately 350 lux fluorescent illumination, the rover tracked the tape path through both straight sections and 90° corners with 94.2% accuracy. The stable operating PWM range settled between 55–65%, slightly below the 60% design point; the modest reduction improved cornering by limiting overshoot on tight bends. Under direct sunlight (approximately 1800–2000 lux), the IR modules required potentiometer adjustment of roughly 15° to restore reliable surface discrimination. Grey concrete surfaces remained problematic regardless of threshold setting, consistent with the surface-sensitivity documented in [1].

7.2 Obstacle Avoidance

Against cardboard and rigid plastic targets, the HC-SR04 measured distances to within ±5 mm across the 5–150 cm test range, and avoidance manoeuvres executed cleanly for obstacles up to approximately 50 cm in width. Foam blocks proved unreliable: acoustic absorption caused detectable echo loss at 15 cm range, producing a miss rate of approximately 35% at the initial $D_{safe} = 20$ cm setting. Raising the threshold to 25 cm reduced the foam miss rate to below 10% across five repeated trials. Mean absolute ranging error across all rigid-target trials was 0.31 cm.

7.3 Bluetooth Command Latency

Latency was measured with a logic analyser, triggering on the transmitted byte edge at the smartphone and capturing the first PWM transition at the motor driver enable pin. Over twenty trials at distances of up to 8 m, mean latency was 97 ms with a standard deviation of 18 ms comfortably within the 200 ms design target. At 12 m two of twenty trials showed a pause of approximately 0.5 s before acknowledgement, consistent with the HC-05's rated operational range of 10–15 m. Critically, mean latency during active camera streaming was 97.86 ms a difference of 0.86 ms compared to streaming-idle conditions, which is indistinguishable from measurement noise.

7.4 Live Video Surveillance

Table 3 summarises streaming performance across four resolution settings.

Table 3: ESP32-CAM Streaming Performance by Resolution

Resolution	FPS	JPEG (KB)	Latency (ms)	Clients
QQVGA 160×120	28.4	4.1	98	2
QVGA 320×240	22.1	12.3	138	2
VGA 640×480	12.3	31.7	182	2
SVGA 800×600	7.8	51.2	294	1

The soft access point was connectable at distances up to approximately 35 m in an open corridor. At QVGA, the stream held 20–24 fps continuously over a 10-minute session with no frame-buffer overflow events logged. At VGA, the sustained rate was 10–14 fps with end-to-end latency of 150–300 ms measured via slow-motion video capture. The stream was verified accessible from Chrome, Firefox, and Safari on both desktop and mobile devices without plugins or dedicated applications.

8. APPLICATIONS AND ADVANTAGES

8.1 Application Domains

The dual-subsystem rover is directly applicable to security and facility monitoring, where an operator can guide the vehicle manually over Bluetooth while simultaneously viewing the live feed from a separate device effectively replacing several fixed cameras in intermittently visited areas. Remote inspection scenarios confined equipment bays, post-flood rooms, or narrow pipe trenches benefit from the combination of obstacle-safe autonomous locomotion and real-time visual feedback. In educational settings, the platform spans infrared sensing, ultrasonic ranging, Bluetooth serial communication, PWM motor control, and Wi-Fi streaming within a single semester build, providing breadth coverage for embedded systems coursework. As a lightweight search-and-rescue demonstration, autonomous obstacle avoidance keeps the rover mobile in cluttered environments while the live camera feed lets remote personnel assess conditions before entry.

8.2 Design Advantages

The most significant practical benefit of the architecture is independent subsystem debugging and upgrading. A firmware update to the ESP32-CAM streaming code requires no modification to the Arduino navigation sketch and poses no risk to motor-control timing. The reverse holds equally. The complete bill of materials is under USD 35, all development tools are open-source, and the

firmware compiles within the standard Arduino IDE without licensed dependencies factors that make the platform straightforwardly replicable by other teams.

9. FUTURE SCOPE

Several well-defined enhancements are achievable without redesigning the base system. Remote cloud telemetry via MQTT over TLS would extend camera and command access beyond the Wi-Fi soft-AP range to any internet-connected device. On-device object detection using a quantised MobileNetV2-SSD TensorFlow Lite model is feasible within the ESP32's 4 MB PSRAM, given that the existing dual-core architecture already separates camera and network tasks. GPS-guided outdoor waypoint navigation could be added via a NEO-6M module with Haversine-based path planning on the Arduino. Upgrading the dual-sensor IR arrangement to a five-channel TCRT5000 array with PID control would improve path recovery at complex intersections and improve performance on grey or low-contrast surfaces. A unified mobile application consolidating live feed, directional controls, sensor telemetry, and mode selection into a single interface would improve operator ergonomics. Finally, adding a pan-tilt servo mechanism for the ESP32-CAM would expand horizontal field of view to approximately 180°, greatly improving situational awareness during reconnaissance tasks.

10. CONCLUSIONS

A four-mode ground rover has been built and characterised, establishing that infrared line following, ultrasonic obstacle avoidance, Bluetooth manual control, and live browser-accessible MJPEG surveillance can coexist on a sub-USD 35 embedded platform without measurable mutual interference. The enabling insight is a single architectural choice: separating motion-control execution on an Arduino UNO from video-streaming execution on an ESP32-CAM, with no shared processing resources between the boards. Controlled indoor testing on a 4 m × 2 m course confirmed 94.2% path-following accuracy under standard lighting, 97 ms mean Bluetooth command latency, reliable obstacle response within a 25 cm radius, and a sustained 10–14 fps browser-accessible VGA feed all operating simultaneously. The increase in Bluetooth latency attributable to concurrent streaming was 0.86 ms, statistically indistinguishable from measurement noise. This result validates the non-interference property that the architecture is designed to provide.

The deliberate minimalism of the design is itself a contribution. Off-the-shelf components, open-source tools, a short firmware codebase, and a component cost accessible to undergraduate project budgets make the platform directly replicable. Each future enhancement outlined in Section 9 can be pursued independently without altering the base hardware or firmware the defining property of a well-partitioned modular system.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the guidance of Mrs. Rohini Banait, Assistant Professor, Department of Computer Engineering, MIT Academy of Engineering, Alandi (D), Pune, whose mentorship and technical insight were instrumental throughout this project.

REFERENCES

- [1] S. Jaikaruna, R. Priya, and M. Venkatesh, "Design and implementation of a low-cost line-following autonomous cart using infrared sensors," in Proc. IEEE Int. Conf. Embedded Systems and Intelligent Automation, pp. 112–117, 2025.
- [2] D. Mukhopadhyay, S. Banerjee, and A. Roy, "IoT-enabled remote controlled car with integrated Bluetooth and Wi-Fi communication," in Proc. IEEE TENSYP, pp. 1045–1050, 2020.
- [3] Y. Jin, L. Wang, and H. Zhang, "Intelligent obstacle-avoidance vehicle using rotating ultrasonic sensors," in Proc. IEEE ICRA, pp. 3421–3426, 2019.
- [4] H. H. O. Nasereddin and A. A. Abdelkarim, "Smart surveillance rover: Real-time monitoring with ESP32-CAM," IJRAR, vol. 10, no. 4, pp. 212–218, 2023.
- [5] IoT Design Pro, "ESP32-CAM based surveillance robot car using Arduino IDE," 2019. [Online]. Available: <https://iotdesignpro.com/projects/esp32-Camsurveillance-robot-car>
- [6] Espressif Systems, "ESP32 Technical Reference Manual," v5.3, 2024. [Online]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>
- [7] Arduino, "Arduino UNO Hardware Overview," 2024. [Online]. Available: <https://www.arduino.cc/en/Main/ArduinoBoardUno>
- [8] Components101, "HC-SR04 Ultrasonic Sensor Datasheet," 2024. [Online]. Available: <https://components101.com/sensors/ultrasonic-sensor-working-pinout-datasheet>
- [9] Components101, "HC-05 Bluetooth Module Datasheet," 2024. [Online]. Available: <https://components101.com/wireless/hc-05-bluetooth-module>
- [10] Components101, "L298N Motor Driver Module Guide," 2024. [Online]. Available: <https://components101.com/modules/l298n-motor-driver-module>
- [11] K. Nakamura, T. Sato, and Y. Kobayashi, "Real-time MJPEG streaming on embedded Linux platforms for robotic teleoperation," IEEE Sensors J., vol. 21, no. 14, pp. 15802–15811, Jul. 2021.
- [12] A. Kaur and R. Singh, "A review of IR sensor-based autonomous ground vehicles," J. Robotics Mechatronics, vol. 35, no. 2, pp. 441–457, Apr. 2023.
- [13] T. L. Floyd, Electronic Devices, 10th ed. Upper Saddle River, NJ: Pearson Education, 2011.
- [14] R. Gupta, Robotics and Control. New Delhi: New Age International Publishers, 2015.
- [15] M. Schwartz, Internet of Things with ESP8266. Birmingham, UK: Packt Publishing, 2016.