JEECS (Journal of Electrical Engineering and Computer Sciences)
Vol. 11, No. 2, December 2026, pp. 172-182
e-ISSN: 2579-5392 p-ISSN: 2528-0260

Design and Implementation of a Wireless Surveillance Robot with Motion Detection and Live Video Transmission

Prudence Wegba Efregede[1], Fortune Omamuzo Efregede[1], Abdullateef Mustapha[2], Charles Somtochukwu Nwafor[3], Zaid Musa Imam[2]

[1] Department of Electrical and Electronics Engineering, College of Technology and Engineering, Federal University of Petroleum Resources, Effurun, Nigeria.

[2] Department of Mechatronics and Systems Engineering, Abubakar Tafawa Balewa University, Bauchi, Nigeria.

[3] Department of Mechatronics Engineering, Federal University of Technology, Owerri, Nigeria.

Article Info

Article history:
Received: 11 July 2026
Revised: 2 November 2026
Accepted: 13 November 2026

Keyword:
Embedded systems
IoT
Motion detection
Real-time video streaming
Wireless surveillance robot

Abstract

Conventional surveillance systems provide reliable monitoring of fixed areas; however, they are constrained by fixed camera coverage, limited communication range, and dependence on human operators, making them inadequate for monitoring hazardous, restricted, or remote environments. This study addresses these limitations through the design and implementation of a low-cost IoT-based wireless surveillance robot integrating real-time motion detection and live video transmission. The system hardware comprises an Arduino Uno microcontroller, an ESP8266 NodeMCU Wi-Fi module, an ESP32-CAM 2MP camera module, a passive infrared (PIR) motion sensor, an L298N dual H-bridge motor driver, and a SIM800L GSM module. The robot supports two operational modes: a remote-controlled navigation mode and an autonomous standby mode for motion-triggered surveillance. System testing confirmed internet-based controllability from any networked device, continuous MJPEG video streaming via a browser-accessible dashboard, and automated SMS alert dispatch upon PIR-triggered motion detection. Voltage tests verified stable operation across all subsystems within specified supply tolerances, confirming reliable hardware integration. Command-response latency measurements yielded a mean delay of 3.1 ± 0.2 s under strong Wi-Fi conditions, attributable to the multi-hop internet routing architecture, increasing to 6.8 ± 1.3 s under network congestion. The proposed system demonstrates that multi-modal surveillance functions (live streaming, motion detection, and GSM alerting) can be consolidated into a single low-cost embedded platform, offering a scalable framework for smart security monitoring in environments where human presence is impractical.

Corresponding author:
Prudence Wegba Efregede, efregedeprudence@gmail.com

DOI: https://doi.org/10.54732/jeecs.v11i2.7

This is an open access article under the CC–BY license.

1. Introduction

Surveillance and security monitoring systems have become increasingly important in residential, industrial, and hazardous environments due to rising security concerns and the need for continuous remote monitoring. Conventional surveillance systems are typically static, limited by fixed camera coverage, and highly dependent on human operators for effective monitoring. These limitations reduce operational flexibility, especially in environments that are difficult or dangerous for humans to access. Mobile robotic

surveillance systems provide a promising alternative by enabling dynamic monitoring, autonomous navigation, and remote accessibility [1].

Recent advancements in the Internet of Things (IoT), embedded systems, wireless communication, and low-power microcontrollers have accelerated the development of intelligent surveillance robots capable of real-time monitoring and remote operation [2], [3]. Technologies such as Wi-Fi-enabled microcontrollers, onboard cameras, motion sensors, and web-based interfaces have made it possible to design low-cost and efficient robotic systems for security applications deployable in hazardous environments, restricted areas, and industrial facilities [1], [4].

Recent studies (2023 to 2026) show that surveillance research has moved toward low-cost ESP32-based nodes, edge processing, and messaging-based alerts. Okokpujie et al. [2] built an affordable real-time surveillance system around an ESP32 with an OV2640 camera and a PIR sensor and used the Twilio API for notification, targeting low-resource settings. Abordo et al. [4] combined an ESP32 with camera-based motion detection and instant-messaging notification, and Zhaxalikov et al. [3] reported a surveillance camera built on a Wi-Fi connection. Alsaffar and Erçelebi [5] developed a Raspberry Pi 3 Model B surveillance system in which a PIR sensor triggers video recording, with the footage delivered by email and a Telegram notification. Adilzhanova et al. [6] proposed an event-based architecture in which a PIR sensor triggers an ESP32-CAM and a Mojo FPGA performs preprocessing and detection, and they reported median PIR-to-acknowledgment latencies of 480 to 620 ms over Wi-Fi using MQTT or HTTP. On the mobile side, Ben Chaabane et al. [1] presented a Raspberry Pi 4 surveillance robot with a PIR sensor, a camera, face recognition (98.63% accuracy on 400 test images of 40 individuals), web-based remote control over Wi-Fi, and notifications and emails carrying the captured picture. Energy efficiency is also receiving attention, for example in IoT hazard detection with adaptive monitoring [7] and in low-cost camera trap networks for remote surveillance [8].

Earlier wireless surveillance robots relied on simpler links. Priyanka and Karthik [9] developed a wireless surveillance robot using Radio Frequency (RF) communication and motion detection sensors. Singh et al. [10] designed a Wi-Fi surveillance robot for real-time audio and video transmission via an Android application. Patoliya et al. [11] developed an Arduino-controlled surveillance robot for military applications using Bluetooth communication and a night vision camera. However, a clear gap remains. The earlier robots [9], [10], [11] depend on RF, Bluetooth, or LAN-bound links, which limits range and remote accessibility, and most of them provide no motion-triggered alert. The recent systems [2], [4]–[6] are mostly fixed nodes that alert users through cloud messaging services, and the one mobile platform with comparable features [1] is built on a Raspberry Pi 4 rather than on low-cost microcontrollers. As a result, few of the works reviewed here combine, on microcontroller-class hardware, (i) internet-based remote navigation, (ii) live video streaming, (iii) PIR-triggered image capture, and (iv) an SMS alert that does not depend on a data connection at the user's end, and only one of them reports measured latency, for event notification rather than remote control [6]. This is the problem addressed in this paper.

To address this gap, this study presents the design and implementation of an IoT-based wireless surveillance robot with real-time motion detection and live video transmission. The proposed system integrates an Arduino Uno microcontroller, NodeMCU Wi-Fi module, ESP32-CAM module, PIR motion sensor, GSM module, and a web-based control interface to enable remote monitoring and robotic navigation over the internet. The major contribution of this work lies in the consolidation of real-time video streaming, motion-triggered monitoring, wireless robotic navigation, and web-based control into a single low-cost embedded platform.

The platform is implemented as two sections. In the robot vehicle section, an Arduino Uno coordinates the motors, the servo, and the GSM module, a NodeMCU relays user commands, and an ESP32-CAM streams video; the monitoring section is a PHP web server, with a database managed through phpMyAdmin, and a browser dashboard exposed through an Ngrok tunnel. Control commands travel as HTTP GET requests and video is streamed as MJPEG. In standby mode, a PIR event triggers image capture and upload, and an SMS containing a link to the image is sent through the SIM800L. The contributions of this paper are therefore (i) the integrated design and its implementation, (ii) measured voltage and command-response latency results under four Wi-Fi conditions, and (iii) a comparison with recent and earlier systems. The rest of the paper is organized as follows: Section 2 describes the methodology, Section 3 presents and discusses the results, and Section 4 gives the conclusions and future work.

2. Research Methodology

The proposed surveillance system consists of two major sections: the robotic vehicle section and the monitoring section. The robotic vehicle section comprises the Arduino Uno microcontroller, NodeMCU Wi-Fi module, ESP32-CAM module, PIR motion sensor, GSM module, servo motor, DC motors, and L298N motor driver. The monitoring section consists of a web-based control interface through which users remotely monitor video streams and control robot movement over the internet.

The Arduino Uno serves as the central processing unit responsible for coordinating communication between the sensors, motor driver, GSM module, and wireless communication unit. The NodeMCU module establishes internet connectivity and enables wireless communication between the robotic platform and the web server. The ESP32-CAM module captures and transmits real-time video streams, while the PIR sensor detects motion within the monitored environment. Upon motion detection, the GSM module sends notification messages containing surveillance alerts to authorized users.

2.1 System Architecture and Hardware Design

The block diagram of the system illustrating the interaction between components and the microcontroller is shown in Figure 1. Figure 1 can be read as four functional paths that meet at the Arduino Uno (the user link, the vision path, the sensing and alert path, and the actuation path) together with two power blocks and the monitoring section.

User link. The ESP8266 Wi-Fi Module is a self-contained SoC with an integrated TCP/IP protocol stack capable of providing any microcontroller access to a Wi-Fi network. The ESP8266 can create its own Wi-Fi network (access point mode) or join existing networks (station mode) to access the internet. In Figure 1 the NodeMCU module is drawn above the Arduino Uno and joined to it by a two-way arrow, so user commands received from the network pass to the Arduino Uno and status information can pass back. The NodeMCU also connects to the internet block, which links the web server with the user's mobile device or computer.

Control core. A detachable, dual-inline-package (DIP) ATmega328 microcontroller forms the foundation of the Arduino Uno board. The board provides 14 digital input/output (I/O) pins, of which 6 serve as PWM outputs, and 6 analog inputs. Additional features include a 16 MHz crystal oscillator, a USB connection, a power jack, and a reset button. At the centre of Figure 1 it coordinates every peripheral except the camera.

Vision path. The ESP32-CAM offers a compact camera module measuring 27 x 40.5 x 4.5 mm with a deep sleep current of up to 6 mA. It incorporates a 2MP OV2640 camera on the ESP32-S chip and includes 4 MB PSRAM for buffering camera images for video streaming. The module provides 10 open GPIOs, power pins, and microSD card support. In Figure 1 the ESP32-CAM has its own power supply block and sends its output straight to the web server instead of through the Arduino Uno, so video traffic does not compete with command traffic.

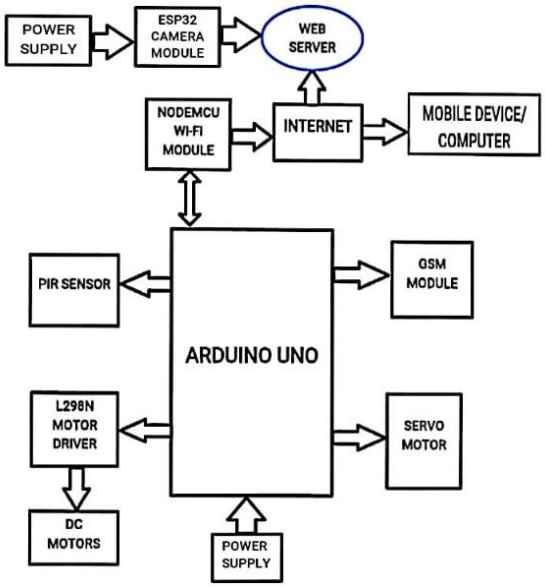

Figure 1. System Block Diagram

Sensing and alert path. The PIR sensor detects movement in the monitored area and provides the motion event on which standby-mode surveillance is based. The GSM module is the output used to send SMS alerts. The SIM800L is a compact cellular module that enables SMS transmission, voice calls, and GPRS data transfer. Its low cost, small size, and quad-band frequency support make it suitable for long-distance connectivity applications, interfacing directly with Arduino and other microcontroller units.

Actuation path. The L298N is a dual H-bridge controller enabling simultaneous speed and direction control of two DC motors. The module incorporates a 78M05 5V regulator and an L298 motor driver IC, capable of managing up to 4 DC motors or 2 DC motors with directional and speed control. In Figure 1 the L298N drives the DC motors for forward, reverse, left, and right motion, while the servo motor adjusts the viewing direction of the camera.

Power. Figure 1 shows two power supply blocks, one beside the ESP32-CAM and one beneath the Arduino Uno. In the implementation they are realised as follows. The power supply unit comprises an external power bank, lithium-ion batteries, and a voltage regulator. Two Li-ion batteries (3.7 V, 2000 mAh each) connected in parallel power the motor driver (3.7 V, 4000 mAh combined). A single 3.7 V, 2000 mAh battery powers the GSM module, while a 10,000 mAh power bank powers the Arduino and ESP32-CAM. A voltage regulator steps down 5 V to 3.3 V for the ESP8266 NodeMCU.

Monitoring section. The web server and the mobile device or computer in Figure 1 form the monitoring section. The user's browser both issues the control commands and displays the ESP32-CAM stream.

2.2 Software Design

Firmware for the Arduino Uno, NodeMCU, and ESP32-CAM are developed using the Arduino IDE and written in embedded C. The web-based front-end is designed with HTML, CSS, and JavaScript, while server-side applications are developed using PHP with a phpMyAdmin database backend. Data transfer employs HTTP GET requests for data retrieval and POST requests for secure login authentication and image upload. Access from outside the local network is provided by an Ngrok tunnel that forwards the web server and the camera stream to a public address.

2.3 System Flowchart

The program flowchart of the system illustrating the operational logic and decision pathways is presented in Figure 2. Figure 2 describes the logic in three coordinated parts: the client and web server (top and centre), the ESP32 camera node (left), and the ESP8266 and Arduino control path (right).

Client and web server. A user first reaches the login handler. If the login is not successful the user is returned to the login step; if it succeeds, the user is saved and the Ngrok information is saved, and the user then selects a mode (stream or capture). The selected mode is sent to the web server, which reads from and updates the database and serves the corresponding page back to the client.

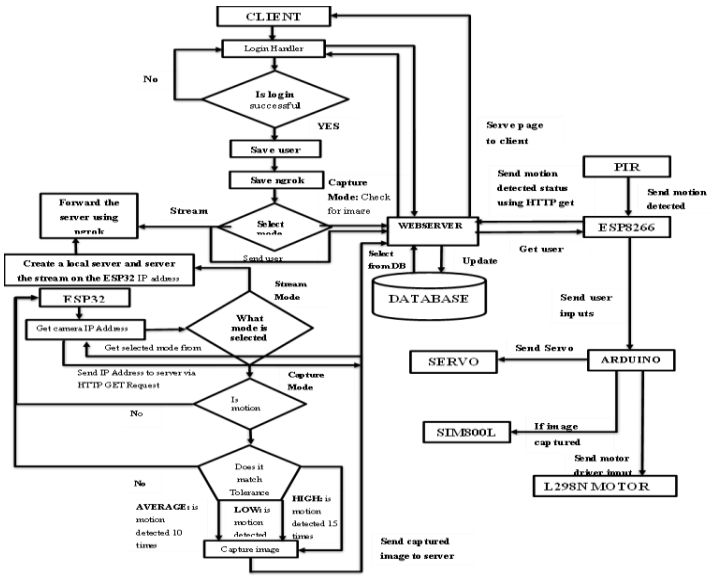

Figure 2. System Flowchart

ESP32 camera node. The ESP32 obtains its camera IP address and sends it to the web server in an HTTP GET request. It then retrieves the selected mode from the server. In stream mode it creates a local server and serves the stream on the ESP32 IP address, and the web server forwards this stream to the client through Ngrok. In capture mode the node checks whether motion is present and returns to the start of the loop if it is not. When motion is present, the detection count is compared with the selected tolerance. As labelled in Figure 2, the average level requires motion to be detected 10 times and the high level requires 15 times. Once the tolerance is met, the node captures an image and sends it to the web server.

ESP8266 and Arduino control path. The PIR sensor sends a motion-detected signal to the ESP8266, which reports the motion status to the web server with an HTTP GET request and also gets the current user inputs from the server. The ESP8266 passes these user inputs to the Arduino Uno, which sends commands to the servo motor and to the L298N motor driver and activates the SIM800L when an image has been captured.

2.4 Mode of Operation

The system operates through two sections: the robot vehicle section and the monitoring section. The monitoring section operates in two modes:

(a) Video Streaming Mode: Any internet-enabled device can control the robot by transmitting control signals to the Arduino Uno over the internet. The ESP32-CAM simultaneously streams live video feedback to the remote user via a web dashboard integrating both control and streaming interfaces [10].

(b) Standby Mode: The robot remains stationary at a designated position and detects motion using the PIR sensor. Upon detection, snapshots are captured, uploaded to the web server, and an SMS notification containing an image access link is dispatched via the GSM module [2], [6].

2.5 Circuit Design

The complete circuit diagram of the proposed system is shown in Figure 3. Figure 3 is the schematic captured in Proteus, which was used for virtual circuit verification. The ATmega328P (Arduino 328) sits at the centre, with the NodeMCU V3 above it to the right and the PIR motion sensor connector at the far right. Two motor-driver stages (U1 and U3), each with its motor outputs, are placed above the microcontroller, and their input pins are connected to digital outputs of the ATmega328P. The simulation uses generic H-bridge driver models in place of the L298N module used in the hardware. A terminal block labelled CAM provides the camera connection, a terminal block labelled SIM MODULE connects the SIM800L, and two battery sources (BAT1, 12 V, and BAT2, 5 V) are shown. Two 10 kΩ resistor pairs (R5/R6 on the SIM module line and R7/R8 on the PIR line at the NodeMCU) form voltage dividers for logic-level matching between 5 V and 3.3 V devices. A servo motor symbol sits to the right of the microcontroller, and the reset and LED circuit is at the bottom of the schematic.

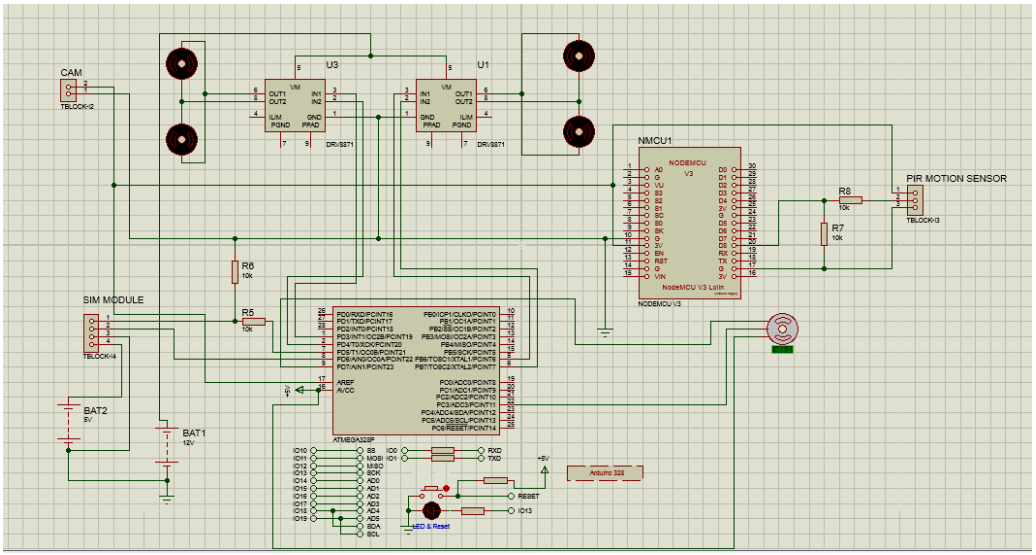

Figure 3. Complete Circuit Diagram of the System

2.6 System Implementation

The project was implemented in stages to simplify the construction process. Each hardware unit was implemented and tested independently before final integration. The complete system was assembled on a Vero board. All firmware for the Arduino Uno, NodeMCU, and ESP32-CAM was written in C and compiled using the Arduino IDE, with Proteus software used for virtual circuit verification.

2.6.1 Drive Unit Construction

The drive unit is responsible for the robot's mobility and navigation. A lightweight plastic robot chassis with predefined mounting holes was selected for the four-wheel configuration. Two DC gear motors serve as power wheels, while two free wheels provide directional guidance. A 16-pin IC socket was soldered to the Vero board to host the L298N motor driver IC, facilitating easy replacement and troubleshooting.

2.6.2 Control Unit Construction

The control unit manages robot mobility and navigation. The motor driver and the NodeMCU Wi-Fi module are mounted on the Vero board beside the Arduino Uno, and the PIR sensor input is connected to the NodeMCU. The physical connection of the motor driver and NodeMCU is shown in Figure 4.

In Figure 4 the Arduino Uno is at the top left of the Vero board, the red L298N module is at the centre, and the NodeMCU is at the right, with the USB cable connected to it marked by the white arrow. The green arrows mark two USB sockets mounted on the board, and the cyan arrow marks the PIR sensor on the chassis. Three switches are visible on the board.

2.6.3 Software Implementation

The firmware uploaded to the microcontroller includes source code for the motor driver, Wi-Fi module, GSM module, servo motor, and PIR sensor (Figure 5). The code enables serial communication between the ESP8266 Wi-Fi module and the Arduino Uno, continuously transmitting and receiving data. Upon motion detection, the firmware dispatches an SMS notification containing an image access link via the GSM module.

3. Results and Discussions

This section reports the electrical and communication tests and discusses them together with the observed behaviour of each subsystem.

3.1 Voltage Test

Voltage measurements were performed on the power supply unit and connected subsystems using a digital multimeter under DC operating conditions to verify that regulated supply voltages were within the acceptable operating ranges of all components. The results are presented in Table 1.

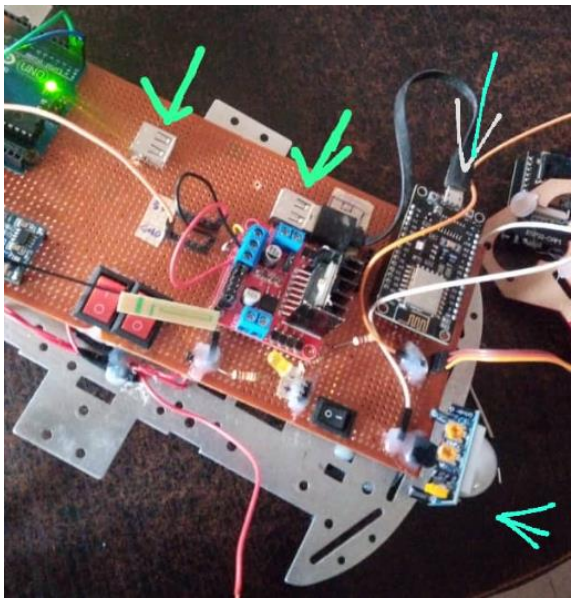

Figure 4. Connection of the Motor Driver and NodeMCU

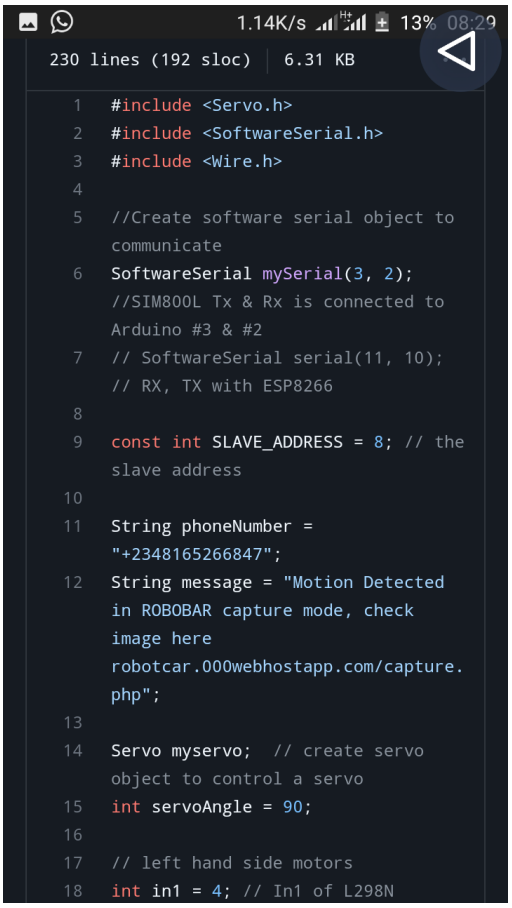

Figure 5. Program Code of the Microcontroller

Table 1. Voltage Test Results for System Subsystems

Component	Expected (V)	Measured (V)	Status
Arduino Uno	5.00	4.97	Pass
ESP8266 NodeMCU	3.30	3.28	Pass
ESP32-CAM Module	3.30	3.31	Pass
L298N Motor Driver	5.00	4.95	Pass
DC Gear Motors	3.70	3.68	Pass
SIM800L GSM Module	3.70	3.72	Pass
Servo Motor	5.00	4.96	Pass

Table 2. Command-Response Latency Under Varying Wi-Fi Conditions

Test Condition	Min Latency (s)	Max Latency (s)	Mean ± SD (s)
Strong Wi-Fi (< 5 m)	2.8	3.4	3.1 ± 0.2
Moderate Wi-Fi (10–20 m)	3.5	4.8	4.0 ± 0.4
Weak Wi-Fi (> 25 m)	4.5	7.2	5.6 ± 0.9
Network Congestion	5.0	9.1	6.8 ± 1.3

All subsystems operated within specified supply tolerances, with measured voltages deviating by no more than 0.03 V from expected values, confirming stable voltage regulation and proper electrical integration across all hardware components.

3.2 Command-Response Latency

Command-response latency was measured under four distinct Wi-Fi conditions to characterize the communication performance of the web-based control interface. Latency was taken as the delay between issuing a command on the web dashboard and the response of the robot, and each condition was tested repeatedly to obtain the minimum, maximum, and mean ± SD values. The results are summarized in Table 2.

A mean command-response delay of 3.1 ± 0.2 s was recorded under strong Wi-Fi conditions (distance under 5 m), increasing to 6.8 ± 1.3 s under network congestion. The mean delay also rose from 3.1 s to 4.0 s and 5.6 s as signal conditions weakened from strong to moderate and weak, and the standard deviation grew from 0.2 s to 1.3 s, so responsiveness became both slower and less predictable. The delay is attributed to multi-hop internet routing through the Ngrok tunnelling service (Section 3.4). For context, a PIR-triggered ESP32-CAM node using MQTT or HTTP over Wi-Fi was reported to reach median event-to-acknowledgment latencies of 480 to 620 ms [6]. That metric is not identical to ours, because it does not include a browser-to-robot round trip through a tunnel, but the several-fold gap suggests that relay routing, rather than the hardware, dominates our delay. In the current design, control commands are therefore best suited to non-time-critical patrol tasks.

3.3 Hardware Performance

The assembled hardware demonstrated stable operation across all subsystems. The Arduino Uno successfully coordinated communication between the motor driver, Wi-Fi module, PIR sensor, and GSM module without observed timing conflicts. The L298N motor driver provided reliable bidirectional DC motor control, enabling forward, reverse, left, and right navigation. The servo motor responded accurately to directional control inputs from the web interface, extending the surveillance field of view beyond a fixed forward perspective.

The rear view of the assembled robot is presented in Figure 6. The power configuration, with lithium-ion batteries in parallel (3.7 V, 4000 mAh) for the drive unit and a 10,000 mAh power bank for the Arduino and ESP32-CAM, provided adequate operational endurance during testing. Runtime was not measured quantitatively, and energy-efficient designs such as adaptive monitoring [7] and low-power camera networks [8] indicate how it could be extended.

3.4 Wireless Communication and Latency

The NodeMCU (ESP8266) successfully established TCP/IP connectivity with the web server over a 2.4 GHz Wi-Fi network. The robot was controllable from smartphones, tablets, and desktop computers, confirming platform-independent web accessibility. The observed command latency (Table 2) arises from multi-hop internet routing through the Ngrok tunneling service. Publish/subscribe messaging (MQTT) and persistent WebSocket connections are candidate replacements for HTTP GET requests and are listed as future work in Section 4.

3.5 Live Video Transmission

The ESP32-CAM successfully streamed live MJPEG video to the web dashboard via the Ngrok tunneling interface. Stream quality degraded under poor signal strength, consistent with the findings of Singh et al. [10]. MJPEG sends every frame as an independent JPEG image, which keeps the ESP32-CAM firmware simple but uses more bandwidth than inter-frame video codecs, so stream quality is sensitive to signal strength. Separating the video stream IP from the command data channel improved streaming stability and reduced packet collision. Night vision capability was confirmed under low-light conditions.

3.6 Motion Detection and Alert System

The PIR motion sensor demonstrated functional detection capability in standby mode, triggering the image capture and SMS alert pipeline upon detection of human or animal movement. Inconsistencies were observed under environmental variations including ambient temperature fluctuations, direct sunlight exposure, and proximity to reflective surfaces. These are well-documented limitations of passive infrared sensing technology, highlighting the need for multi-modal sensor fusion in future iterations. Related PIR-based systems show a similar sensitivity to background conditions: false-alarm rates of 0.03 to 0.17 were reported for scenes without target movement, depending on background and lighting [6], and fusing the PIR sensor with ultrasonic or vision-based motion detection has been suggested to improve reliability [1]. Sensor-fusion approaches are reviewed in [12]. In our prototype, the tolerance levels in Figure 2 provide a software filter that requires repeated detections before an image is captured, but detection accuracy and false-alarm rate were not quantified (Section 3.8).

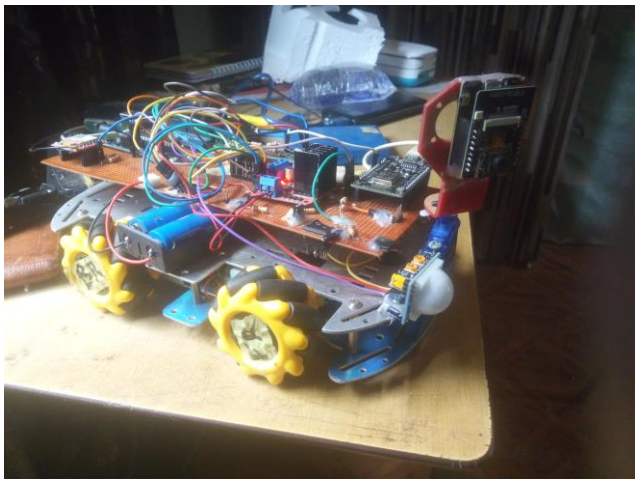

Figure 6. Rear View of the Assembled Robot

Table 3. Comparative Analysis with Recent and Earlier Works

Reference	Platform	Link / range	Live video	Motion detection	Mobile robot	Alert channel
Ben Chaabane et al. [1]	Raspberry Pi 4	Wi-Fi	Yes	Yes (PIR)	Yes	Notification and email
Adilzhanova et al. [6]	ESP32-CAM + FPGA	Wi-Fi; MQTT or HTTP to cloud	No (detection results)	Yes (PIR)	No	Detection results to cloud
Alsaffar and Erçelebi [5]	Raspberry Pi 3 B	Wi-Fi	Recorded clip	Yes (PIR)	No	Email and Telegram
Okokpujie et al. [2]	ESP32-CAM	Wi-Fi	Real-time video	Yes (PIR)	No	Twilio API
Priyanka and Karthik [9]	Microcontroller	RF; limited line of sight	No	Yes	Yes	No GSM alert
Singh et al. [10]	Arduino + app	Wi-Fi; LAN only	Yes (audio and video)	No	Yes	No GSM alert
Patoliya et al. [11]	Arduino	Bluetooth; about 30 m	Yes (IR camera)	No	Yes	No GSM alert
Proposed system	Arduino + NodeMCU + ESP32-CAM	Wi-Fi + GSM; internet via tunnel	Yes (MJPEG)	Yes (PIR)	Yes	SMS with image link

3.7 Comparative Analysis

Table 3 compares the proposed system with recent and earlier works from the literature. The recent systems [2], [5], [6] pair a PIR trigger with a camera and send alerts through messaging or cloud services, but they are fixed nodes rather than robots. The closest mobile platform [1] uses a Raspberry Pi 4 and adds face recognition, which the proposed system does not offer. The earlier robots [9], [10], [11] are limited to RF, LAN, or Bluetooth links. The proposed system offers internet-scale control range, live MJPEG streaming, PIR-based motion detection, and GSM alert notification within a single low-cost platform. While RF and Bluetooth systems offer lower latency, their range is constrained to line-of-sight or short-range coverage. The ESP32-CAM provides a cost-effective vision alternative with onboard image buffering via 4 MB PSRAM. The trade-offs are higher control latency than event-driven nodes [6] and no on-board recognition.

3.8 Limitations and Unresolved Challenges

Notwithstanding the successful implementation of the proposed system, several limitations were identified: (a) Control Latency: The mean command-response delay ranged from 3.1 s under strong Wi-Fi to 6.8 s under network congestion, with a maximum of 9.1 s (Table 2), and is attributable to multi-hop internet routing through the Ngrok tunneling service. This restricts applicability in time-critical surveillance scenarios where prompt robotic response to detected threats is necessary; (b) PIR Sensor Environmental Sensitivity: The PIR sensor exhibited inconsistent readings under direct sunlight, ambient temperatures above 25°C, and proximity to reflective surfaces, introducing false positive and missed detection events. Reliance on a single sensing modality limits detection reliability; (c) Network Dependency: The system is entirely dependent on an active internet connection for control and video transmission.

Interruptions render the platform non-operational. Offline fallback mechanisms were not implemented in this prototype; (d) Terrain Limitations: The standard plastic robot chassis is optimized for flat indoor surfaces. Reduced traction and navigation control were observed on uneven or outdoor terrain, limiting deployment to controlled indoor environments; (e) Unquantified Detection Performance and Security: Detection accuracy and false-alarm rate were not measured, and the dashboard is exposed through a public tunnel protected by a login page. Security and privacy hardening of IoT links is an active concern [13], and encrypted transport and stronger authentication should be added before wider deployment.

4. Conclusion

This study presented the design and implementation of an IoT-based wireless surveillance robot integrating real-time motion detection, live MJPEG video streaming, internet-based robotic navigation, and automated GSM alert notification within a unified low-cost embedded platform. The system successfully demonstrated remote controllability over the internet, live video transmission via the ESP32-CAM module, functional PIR-based motion detection, and automated SMS alert dispatch through the SIM800L GSM module. Voltage tests confirmed that all subsystems operated within specified supply ranges, and a mean command-response delay of 3.1 ± 0.2 s was measured under strong Wi-Fi conditions. The proposed system offers a practical and scalable framework for smart security surveillance in residential, industrial, and hazardous environment monitoring applications. The main limitations are control latency of several seconds through the tunnel and single-sensor motion detection, and they are the focus of the future work below.

Future work should address the limitations identified in Section 3.8: (a) Low-Latency Communication Protocols: Future implementations should replace HTTP GET-based commands with WebSocket or MQTT protocol. A PIR-triggered ESP32-CAM node has been reported to reach median event-to-acknowledgment latencies below 1 s [6]. Deployment of a local MQTT broker would eliminate dependency on cloud relay services such as Ngrok; (b) Multi-Modal Sensor Fusion: Future systems should integrate PIR detection with ultrasonic ranging, microwave Doppler sensing, and computer vision-based motion analysis using the onboard ESP32-CAM for background subtraction or frame-differencing algorithms to improve detection accuracy and reduce false alarm rates. Sensor fusion [12]s and on-device intelligence [14], [15] are active research directions; (c) GPS Integration and Autonomous Navigation: A GPS module (e.g., NEO-6M) would enable geolocation tracking and support waypoint-based patrol routines. Integration of path-planning algorithms such as A* or Dijkstra's algorithm, combined with ultrasonic obstacle avoidance, would provide a basis for autonomous patrol operation. ESP32-based motion control with PID has been demonstrated on a cleaning robot [16]; (d) Enhanced Power Management: A dedicated battery management system (BMS) with a real-time voltage indicator on the web dashboard is recommended. Implementing deep-sleep modes on ESP8266 and ESP32 modules during inactivity, as demonstrated in low-power IoT node designs, [7], [8], would extend battery life. Ethical Considerations: The prototype described in this study was operated exclusively within controlled laboratory and campus environments with the full knowledge and consent of all individuals present during testing. No personally identifiable information was captured, stored, or transmitted beyond the immediate session. Future deployments should adhere to applicable data protection regulations and ethical guidelines governing autonomous surveillance technologies.

References

[1] S. Ben Chaabane, R. Harrabi, A. Bushnag, and H. Seddik, “Development of Surveillance Robots Based on Face Recognition Using High-Order Statistical Features and Evidence Theory,” *Journal of Imaging*, vol. 12, no. 3, p. 107, 2026, doi: 10.3390/JIMAGING12030107.

[2] K. Okokpujie, I. P. Okokpujie, F. T. Young, and R. E. Subair, “Development of an Affordable Real-Time IoT-Based Surveillance System Using ESP32 and TWILIO API,” *International Journal of Safety and Security Engineering*, vol. 13, no. 6, pp. 1069–1075, 2023, doi: 10.18280/IJSSE.130609.

[3] A. Zhaxalikov, A. Mombekov, and Z. Sotsial, “Surveillance Camera Using Wi-Fi Connection,” *Procedia Computer Science*, vol. 231, pp. 721–726, 2024, doi: 10.1016/J.PROCS.2023.12.147.

[4] P. R. C. Abordo *et al.*, “Smart surveillance system using ESP32 and camera-based motion detection with IM technology,” *International Journal of Research Studies in Educational Technology*, vol. 8, no. 2, 2024, doi: 10.5861/IJRSET.2024.8012.

[5] H. G. Alsaffar and E. Erçelebi, “Design and Implementation of the IoT Surveillance System using Electronic Appliances with Raspberry Pi,” *Al-Iraqia Journal for Scientific Engineering Research*, vol. 3, no. 3, pp. 71–79, 2024, doi: 10.58564/IJSER.3.3.2024.225.

[6] S. Adilzhanova, G. Amirkhanova, M. Kunelbayev, and A. Rakhysh, “A Hardware Platform for Smart

Video Monitoring Based on ESP32-CAM and Mojo FPGA (Spartan-6) with Event Activation Triggered by a PIR Sensor," *Engineering, Technology & Applied Science Research*, vol. 16, no. 3, pp. 36089–36096, 2026, doi: 10.48084/ETASR.18359.

[7] C. L. Kok, J. B. Heng, Y. Y. Koh, and T. H. Teo, "Energy-, Cost-, and Resource-Efficient IoT Hazard Detection System with Adaptive Monitoring," *Sensors 2025, Vol. 25, Page 1761*, vol. 25, no. 6, p. 1761, 2025, doi: 10.3390/S25061761.

[8] P. Lech, B. Marciniak, and K. Okarma, "A Low-Cost Energy-Efficient IoT Camera Trap Network for Remote Forest Surveillance," *Electronics 2025, Vol. 14, Page 4266*, vol. 14, no. 21, p. 4266, 2025, doi: 10.3390/ELECTRONICS14214266.

[9] D. Priyanka and V. Karthik, "Wireless Surveillance Robot with Motion Detection and Live Video Transmission and Gas Detection," *International Journal of Scientific Engineering and Technology Research*, vol. 04, no. 17, pp. 3099–3106, 2015.

[10] D. Singh, P. Zaware, and A. Nandgaonkar, "Wi-Fi surveillance bot with real time audio & video streaming through Android mobile," *RTEICT 2017 - 2nd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, Proceedings*, vol. 2018-Janua, pp. 746–750, 2017, doi: 10.1109/RTEICT.2017.8256696.

[11] J. Patoliya, H. Mehta, and H. Patel, "Arduino controlled war field spy robot using night vision wireless camera and Android application," *NUiCONE 2015 - 5th Nirma University International Conference on Engineering*, 2016, doi: 10.1109/NUICONE.2015.7449624.

[12] Y. Liu, S. Wang, Y. Xie, T. Xiong, and M. Wu, "A Review of Sensing Technologies for Indoor Autonomous Mobile Robots," *Sensors 2024, Vol. 24, Page 1222*, vol. 24, no. 4, p. 1222, 2024, doi: 10.3390/S24041222.

[13] T. Zhukabayeva, L. Zholshiyeva, N. Karabayev, S. Khan, and N. Alnazzawi, "Cybersecurity Solutions for Industrial Internet of Things–Edge Computing Integration: Challenges, Threats, and Future Directions," *Sensors 2025, Vol. 25, Page 213*, vol. 25, no. 1, p. 213, 2025, doi: 10.3390/S25010213.

[14] A. Trigkas, D. Piromalis, and P. Papageorgas, "Edge Intelligence in Urban Landscapes: Reviewing TinyML Applications for Connected and Sustainable Smart Cities," *Electronics 2025, Vol. 14, Page 2890*, vol. 14, no. 14, p. 2890, 2025, doi: 10.3390/ELECTRONICS14142890.

[15] K. S. Velaga, Y. Guo, and W. Yu, "Edge AI for Smart Cities: Foundations, Challenges, and Opportunities," *Smart Cities 2025, Vol. 8, Page 211*, vol. 8, no. 6, p. 211, 2025, doi: 10.3390/SMARTCITIES8060211.

[16] L. O. M. Ali, A. A. Mochtar, and F. Djamaluddin, "Design and Development of Motion Control for a Metal Waste Cleaning-24 Robot Using ESP32 and PID Control," *Engineering, Technology & Applied Science Research*, vol. 16, no. 1, pp. 31770–31778, 2026, doi: 10.48084/ETASR.15660.