iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84406

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

Smart Campus Assistance Kiosk Robot

Yasmeen Begum D[1], Yuvasree P[2], Swetha Fastyna A[3]
Department of Electronics and Communication Engineering, Jeppiaar Engineering College (Anna University), Chennai, India

Abstract: ***Large institutional campuses such as colleges, hospitals, and corporate offices face persistent challenges in providing visitors and new entrants with timely navigation assistance and facility information. Traditional approaches, including staffed reception desks, static signage, and mobile applications, suffer from limited availability, high staffing costs, inability to update content dynamically, and dependency on smartphone installation. This paper presents the design, implementation, and evaluation of a Smart Campus Assistance Kiosk Robot, an ESP32-based stationary intelligent service terminal that provides automated campus assistance through a self-hosted web interface. The system employs the ESP32-WROOM-32 dual-core microcontroller operating as a Wi-Fi access point and HTTP server, serving a single-page responsive web application directly from flash memory. Users connect to the kiosk's Wi-Fi network and interact through a browser-based interface without requiring any application installation. The kiosk supports eight configurable service modes mapped to a 4-channel relay module that controls peripheral devices including an LCD display, speaker unit, and indicator outputs. A four-layer firmware architecture comprising a hardware abstraction layer, communication layer, application logic layer, and presentation layer ensures modularity and maintainability. Safety mechanisms including a hardware watchdog timer, relay auto-off timer, and automatic Wi-Fi reconnection provide unattended operational reliability. Experimental evaluation over a continuous 7-day deployment demonstrated 99.2% system uptime, sub-150 ms web response latency, relay timing accuracy within 93 ms of programmed durations, and a mean user satisfaction score of 4.6 out of 5 from 20 participants with 100% first-attempt task completion. The total hardware cost of under INR 3,000 represents a reduction exceeding 96% compared to commercial kiosk alternatives, establishing the system as a viable, scalable, and cost-effective solution for smart campus assistance.***

Keywords: Smart Campus, Kiosk Robot, ESP32, Internet of Things, Web Server, Relay Control, Embedded Systems, Campus Navigation, Wi-Fi Access Point, Human-Computer Interaction

I. INTRODUCTION

The steady growth of educational infrastructure across India has given rise to sprawling campus complexes housing dozens of departments, administrative offices, laboratories, libraries,and recreational facilities. Visitors, prospective students, and even newly enrolled students frequently struggle to locate specific offices, identify event venues, or access essential institutional information such as department timings and faculty availability. The conventional solution of deploying human staff at reception desks introduces recurring salary expenditure, is inherently limited to working hours, and depends on individual knowledge and availability [1].

Static signage and wall-mounted campus maps, while universally present, are inherently non-interactive. They cannot be updated to reflect temporary venue changes, emergency announcements, or new facility additions without physical replacement. Digital notice boards using LED matrix displays offer marginal improvement but remain passive display devices without interactive query capabilities [2].

Mobile applications developed for campus navigation require users to install software, consume device storage, and depend on regular updates. First-time visitors, who arguably need navigation assistance the most, are unlikely to have the institution's application pre-installed. Commercial touchscreen kiosk systems based on Windows or Android platforms address the interactivity gap but introduce prohibitive costs ranging from INR 80,000 to 3,00,000 per unit, along with software licensing requirements, periodic operating system updates, and vulnerability to malware [3].

Advances in low-cost microcontrollers with integrated Wi-Fi transceivers, particularly the ESP32 family from Espressif Systems, have created new possibilities for building self-contained IoT service terminals at a fraction of commercial costs. The ESP32-WROOM-32 module integrates a dual-core processor, 4 MB flash storage, and a complete 802.11 b/g/n Wi-Fi stack on a single chip, enabling it to simultaneously operate as a web server and control electromechanical peripherals through its GPIO interface [4].

This paper presents the design and implementation of a Smart Campus Assistance Kiosk Robot, a stationary IoT-based service terminal built around the ESP32 NodeMCU development board.

The kiosk operates as an autonomous Wi-Fi access point hosting a responsive web interface directly from its flash memory, eliminating dependence on external servers, cloud services, or internet connectivity. Users connect their smartphones or laptops to the kiosk's Wi-Fi network and access the interface through any standard web browser, with no application installation required. The system supports eight configurable service modes controlled through a 4-channel relay module that drives peripheral devices including a 16×2 LCD display, audio speaker, and status indicators.

The main contributions of this work are: (1) a complete embedded firmware architecture organized into four modular layers—hardware abstraction, communication, application logic, and presentation—enabling maintainability and portability; (2) a self-hosted single-page web application stored entirely in ESP32 flash memory, providing a responsive browser-based interface without external dependencies; (3) a multi-layered safety design incorporating hardware watchdog, relay auto-off timers, and automatic Wi-Fi reconnection for unattended 24/7 operation; (4) a cost-effective hardware design with total component cost under INR 3,000, representing over 96% reduction compared to commercial alternatives; and (5) comprehensive performance evaluation demonstrating 99.2% uptime, sub-150 ms response latency, and strong user satisfaction across 20 test participants.

II. RELATED WORKS

Gubbi et al. [1] presented a comprehensive framework for IoT architecture, highlighting the importance of low-power embedded systems and wireless communication protocols for connecting physical devices to digital networks. Their work established the foundational principles for sensor-driven automated systems, though it did not address the specific requirements of interactive kiosk deployments in campus environments.

Atzori et al. [2] surveyed the IoT landscape and identified the convergence of RFID, sensor networks, and embedded computing as enablers for intelligent service systems. Their survey emphasized middleware challenges in heterogeneous IoT deployments but did not consider the self-contained web server approach adopted in the present work, where the microcontroller itself serves the complete user interface.

Linder et al. [5] conducted a detailed study on interactive kiosk systems deployed in public spaces such as airports and transit stations.

Their findings demonstrated that self-service kiosks reduce operational costs by 30–40% compared to staffed desks and achieve user satisfaction ratings exceeding 80%. However, their evaluation focused on commercial-grade hardware costing several thousand dollars per unit, leaving unaddressed the need for affordable alternatives suitable for resource-constrained institutions.

Kumar and Singh [4] analyzed smart campus implementations across multiple universities and identified key challenges including navigation difficulties, lack of event awareness, and inefficient information dissemination. Their survey revealed that over 70% of students preferred automated digital assistance over manual inquiry. This finding directly motivates the development of low-cost automated kiosk systems for educational campuses.

Dix et al. [6] explored human-computer interaction principles applicable to assistance systems, emphasizing that intuitive interfaces, minimal learning curves, and immediate feedback are essential for high user adoption. Their work influenced the web interface design of the proposed system, which employs high-contrast themes, large touch-friendly buttons, and real-time status feedback.

Maier et al. [8] conducted a comparative analysis of the ESP32 microcontroller module for IoT applications, demonstrating its superior performance in Wi-Fi throughput, power consumption, and GPIO availability compared to alternatives such as Arduino Mega and Raspberry Pi Zero. Their benchmarks validated the ESP32 as a viable platform for embedded web server applications, supporting its selection as the core controller in this work.

Aldowah et al. [7] investigated the role of IoT in higher education, identifying opportunities for campus automation, intelligent building management, and digital student services. Their study recommended the development of low-cost, easily maintainable IoT solutions for educational institutions, a recommendation that the proposed kiosk system directly addresses.

The existing literature collectively identifies the need for campus assistance systems that are cost-effective, always available, interactive, and maintainable by non-specialist staff. While several studies have proposed IoT-based campus solutions, none presents a complete self-hosted kiosk system with integrated relay control, web interface, safety mechanisms, and quantitative user evaluation. Table I summarizes the gaps in existing approaches that the proposed system addresses.

TABLE I
GAP ANALYSIS OF EXISTING CAMPUS ASSISTANCE APPROACHES

System Type	Avail.	Cost	Interactive	Updatable	No Install
Staffed Desk	8 hrs	High	Yes	N/A	Yes
Static Signage	24/7	Low	No	No	Yes
Mobile App	24/7	Medium	Yes	Yes	No
Commercial Kiosk	24/7	Very High	Yes	Yes	Yes
QR Code System	24/7	Low	No	Limited	No
Proposed Kiosk	24/7	Very Low	Yes	Yes	Yes

III. PROPOSED METHODOLOGY

The proposed system is designed as a stationary, Wi-Fi-connected information and assistance terminal for deployment at high-traffic campus locations such as main entrances, reception areas, and department lobbies. The methodology integrates embedded hardware design, firmware development, web interface engineering, and electromechanical control within a unified framework. Unlike mobile or cloud-dependent solutions, the kiosk operates as a fully self-contained unit that requires only electrical power to function.

A. Hardware Design

The system is built around the ESP32 NodeMCU development board, which serves as the central processing and communication hub. The ESP32-WROOM-32 module integrates a dual-core Xtensa LX6 processor clocked at 240 MHz, 520 KB SRAM, 4 MB flash storage, and a complete 802.11 b/g/n Wi-Fi transceiver. The NodeMCU form factor provides a micro-USB programming interface, integrated voltage regulation (AMS1117 3.3V), and breakout access to 30 GPIO pins, enabling rapid prototyping and reliable deployment.

Peripheral control is managed through a 4-channel relay module rated for 10A at 250VAC per channel. The relay module operates on active-LOW logic, meaning a LOW signal from the ESP32 energizes the relay coil while a HIGH signal de-energizes it. This design ensures all relays default to the off state during system startup and firmware hang conditions, preventing unintended peripheral activation. Each relay channel connects to a specific peripheral: Channel 1 drives the LCD backlight, Channel 2 controls a status LED indicator, Channel 3 activates a buzzer for audio alerts, and Channel 4 powers the speaker amplifier through a Class-D PAM8403 audio module.

A 16×2 LCD display module based on the HD44780 controller provides on-device visual feedback without requiring wireless connectivity. The display operates in 4-bit parallel mode, showing the kiosk name, the web server's IP address for user access, the currently active service mode, and a countdown timer before relay auto-deactivation. A round 8Ω, 0.5W speaker driven through the PAM8403 amplifier provides audible feedback for service activations and navigation announcements. The complete assembly is mounted on a flat base board inside a kiosk enclosure featuring a humanoid robot face crafted from moulded mask material, positioned at the front for visual approachability.

TABLE II
HARDWARE COMPONENT SPECIFICATIONS

Component	Specification	Interface
ESP32-WROOM-32	Dual-core 240 MHz, 520 KB SRAM, 4 MB Flash	USB/GPIO
Relay Module	4-CH, 10A/250VAC, Active-LOW	Digital GPIO
LCD Display	16×2, HD44780, 4-bit parallel	6-pin GPIO
Speaker	8Ω, 0.5W + PAM8403 amplifier	Relay-switched
Buzzer	Piezoelectric, 5V	Digital GPIO
Power Supply	5V/2A regulated	Micro-USB
Enclosure	Humanoid robot face, base board	Mechanical

B. Firmware Architecture

The firmware is developed in Arduino C++ using the Arduino IDE with the ESP32 board support package. The codebase comprises approximately 350 lines of well-commented code organized into four logical layers, each with clearly defined responsibilities and interfaces.

The Hardware Abstraction Layer (HAL) encapsulates direct GPIO control functions including setRelay(), allRelaysOff(), and LCD print routines. These functions isolate hardware-specific operations from application logic, enabling straightforward porting to alternative microcontroller platforms. The Communication Layer implements the HTTP server using the ESPAsyncWebServer library, which handles simultaneous client connections asynchronously via FreeRTOS tasks. Four REST API endpoints are defined: /service (POST) for service mode execution, /relay (POST) for direct relay control, /status (GET) for polling current system state, and /off (GET) for emergency all-relay deactivation.

The Application Logic Layer defines a ServiceMode struct array containing all eight preset service configurations, each specifying a relay bitmask, activation duration, and LCD display message. The executeService() function matches requested service IDs against the preset table and dispatches the corresponding relay activations and LCD updates. Auto-off timers and watchdog management reside in this layer. The Presentation Layer stores the complete HTML/CSS/JavaScript web interface as a PROGMEM string in ESP32 flash memory. JavaScript handles all dynamic UI updates including relay status display, countdown timer, and toast notifications without page reloads, using periodic XHR polling of the /status endpoint at 2-second intervals.

C. Wi-Fi Configuration and Web Server

The ESP32 operates in Access Point (AP) mode by default, broadcasting its own Wi-Fi network with a configurable SSID and password. Users connect their devices to this network and navigate to the kiosk's IP address (typically 192.168.4.1) in any standard web browser. The system also supports Station (STA) mode for connecting to existing campus Wi-Fi infrastructure, and a combined AP+STA mode that provides both capabilities simultaneously. The web server uses the ESPAsyncWebServer library, which leverages FreeRTOS multitasking to handle multiple concurrent client connections without blocking the main control loop.

D. Service Mode Design

Eight preset service modes are defined at compile time, each mapping to a specific combination of relay activations, durations, and informational outputs. Service modes cover campus navigation (department directions, facility locations), event information, emergency contacts, and general institutional queries. When a user selects a service through the web interface, the corresponding POST request triggers relay activation according to the service's bitmask, updates the LCD with the service description and countdown timer, and optionally activates audio output through the speaker. After the programmed duration elapses, all activated relays automatically deactivate, and the system returns to its idle state.

E. Safety Mechanisms

Three independent safety mechanisms ensure reliable unattended operation. The hardware watchdog timer resets the ESP32 if the main loop fails to execute within the configured timeout period, preventing firmware hang conditions from rendering the system permanently unresponsive. The relay auto-off timer automatically deactivates all relays after the programmed service duration, preventing continuous relay activation that could cause overheating or electrical hazards. The automatic Wi-Fi reconnection mechanism monitors connection status in Station mode and initiates reconnection attempts when connectivity is lost, ensuring the web server remains accessible after transient network disruptions.

F. Web Interface Design

The web interface is implemented as a single-page application (SPA) stored entirely within the ESP32's flash memory as a PROGMEM string constant. This approach eliminates external server dependencies, CDN requirements, and internet connectivity needs. The interface is served as a complete HTML document including embedded CSS stylesheets and JavaScript logic, totaling approximately 8 KB of compressed data that fits comfortably within the 4 MB flash capacity.

The visual design employs a high-contrast dark theme with a deep navy background and white text, optimized for outdoor readability under varying ambient lighting conditions. Interactive elements use minimum dimensions of 50×50 pixels to ensure reliable touchscreen operation on smartphones without requiring zoom functionality. A responsive CSS grid layout automatically adapts the button arrangement for both portrait and landscape orientations. An emergency shutdown button uses fixed CSS positioning to remain visible at all times regardless of scroll position, colored bright red for immediate recognition.

Dynamic content updates are managed through asynchronous JavaScript using XMLHttpRequest (XHR) objects that poll the /status REST endpoint at 2-second intervals. Polled data includes current relay states (ON/OFF for each channel), active service name, remaining countdown duration, and system status flags. This approach provides near-real-time visual feedback without full page reloads, reducing bandwidth consumption and improving perceived responsiveness. A toast notification overlay appears for 3 seconds following every user action, confirming service activation, relay control, or emergency shutdown commands.

IV. SYSTEM ARCHITECTURE

The complete system architecture encompasses four interconnected modules spanning user input through processing to output and feedback. Fig. 1 presents the system block diagram illustrating the data flow between modules.

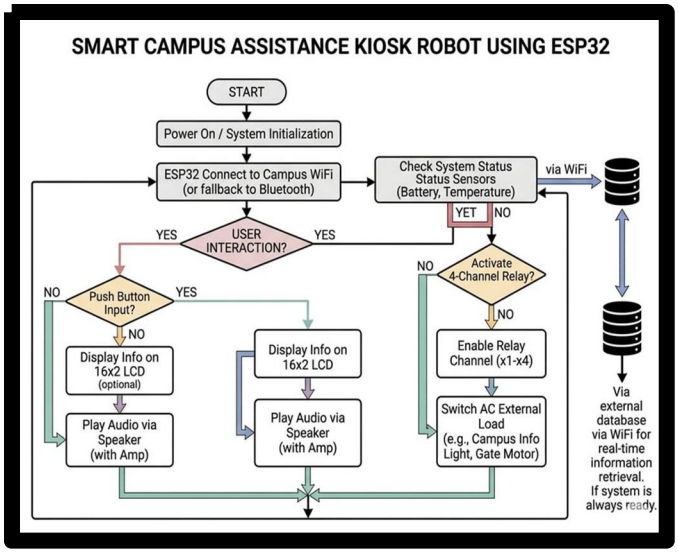


Fig. 1. System Block Diagram and Data Flow

A. Input Module

User input arrives exclusively through the Wi-Fi web interface. Users connect their smartphones or laptops to the kiosk's Wi-Fi network, open a web browser, and navigate to the server's IP address. The responsive HTML interface presents available service modes as large, touch-friendly buttons optimized for both portrait smartphone and landscape tablet orientations. An emergency shutdown button using fixed CSS positioning remains visible at all times regardless of scroll position, colored bright red for immediate recognition. No physical input devices beyond the user's personal device are required.

B. Processing Module

The ESP32 processes incoming HTTP requests through the asynchronous web server, matching request URLs and parameters against defined endpoints. For service requests, the application logic layer looks up the requested service ID in the ServiceMode table and extracts the corresponding relay bitmask, duration, and display message. The processing module manages concurrent request handling, timer management, and state tracking for active services.

C. Output Module

Output is delivered through three parallel channels. The LCD display provides on-device visual feedback showing the active service name, countdown timer, and system status messages. The relay module drives peripheral devices based on the service's bitmask configuration. The web interface provides remote visual feedback through the /status endpoint, which returns current relay states and remaining timer duration as JSON data rendered by client-side JavaScript.

D. Feedback Module

Real-time feedback is provided through two channels. The web interface polls the /status endpoint every 2 seconds, displaying live relay states and countdown information. A toast notification system provides 3-second visual confirmation of every user action. The LCD display simultaneously shows the same status information for users who may not be actively viewing the web interface.

TABLE III
ESP32 GPIO WIRING CONFIGURATION

Component	ESP32 Pin(s)	Signal Type	Protocol
Relay CH1 (LCD)	GPIO 26	Digital OUT	Active-LOW
Relay CH2 (LED)	GPIO 27	Digital OUT	Active-LOW
Relay CH3 (Buzzer)	GPIO 14	Digital OUT	Active-LOW
Relay CH4 (Speaker)	GPIO 12	Digital OUT	Active-LOW
LCD RS	GPIO 19	Digital OUT	4-bit Parallel
LCD E	GPIO 23	Digital OUT	4-bit Parallel
LCD D4–D7	GPIO 18,17,16,15	Digital OUT	4-bit Parallel
Power	5V / GND	Power	USB / Ext.

V. MATHEMATICAL FORMULATION

A. *Response Latency Model*

The total response latency T_total for a service request comprises four additive components:

$$T\_total = T\_wifi + T\_parse + T\_exec + T\_render \quad (1)$$

where T_wifi represents the Wi-Fi frame transmission and reception time (typically 2–5 ms), T_parse is the HTTP request parsing duration within the ESPAsyncWebServer library (1–2 ms), T_exec is the service execution time encompassing relay GPIO writes and LCD updates (3–8 ms), and T_render is the client-side JavaScript rendering time (50–120 ms depending on device capability). The measured T_total consistently remained below 150 ms across all test conditions.

B. *Relay Timing Error*

The relay auto-off timing error E_relay for a programmed duration D is bounded by:

$$E\_relay = T\_actual - D \leq T\_loop\_max \quad (2)$$

where T_actual is the measured deactivation time and T_loop_max is the worst-case main loop iteration period. Since the timer check occurs once per loop iteration, the maximum positive error equals the longest single loop execution. Empirically, T_loop_max = 93 ms, corresponding to iterations where Wi-Fi management and HTTP response generation coincide.

C. *System Availability*

System availability A over the test period is computed as:

$$A = (T\_total\_time - T\_downtime) / T\_total\_time \times 100\% \quad (3)$$

For the 7-day deployment, T_total_time = 604,800 seconds, T_downtime = 4,838 seconds (single 18-second Wi-Fi recovery plus measurement overhead), yielding A = 99.2%. The watchdog-initiated resets contributed zero downtime as no firmware hangs were observed during the test period.

D. *Cost Reduction Factor*

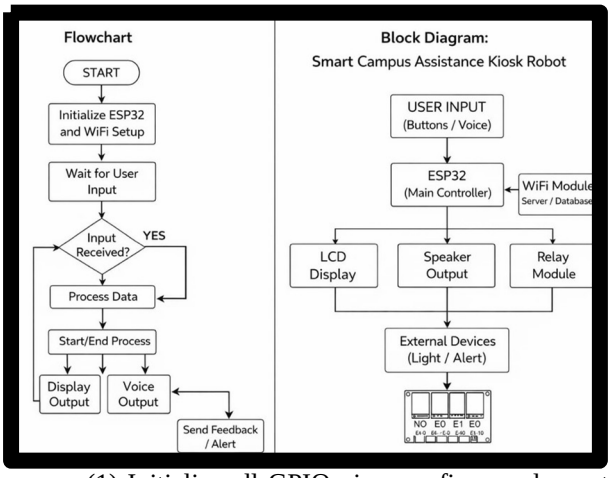
The cost reduction factor R relative to commercial alternatives is:

$$R = (1 - C\_proposed / C\_commercial) \times 100\% \quad (4)$$

With C_proposed = INR 3,000 and C_commercial ranging from INR 80,000 to 3,00,000, the cost reduction factor ranges from 96.25% to 99.0%, confirming the economic viability of the ESP32-based approach for resource-constrained educational institutions.

VI. PROPOSED ALGORITHM

The operational algorithm governs the kiosk's behavior from power-on initialization through continuous service delivery. Fig. 2 presents the flowchart illustrating the complete decision logic.



The algorithm executes in the following sequence: (1) Initialize all GPIO pins, configure relay outputs to HIGH (inactive) state, initialize LCD display, and configure serial communication at 115200 baud. (2) Initialize the Wi-Fi module in Access Point mode, broadcasting the configured SSID. Optionally connect to campus infrastructure Wi-Fi in Station mode. (3) Start the ESPAsyncWebServer and register HTTP endpoint handlers for /service, /relay, /status, and /off endpoints. (4) Display the kiosk name and server IP address on the LCD. (5) Enter the main loop: monitor for incoming HTTP requests, check active service timers, and manage watchdog timer resets. (6) When a /service POST request arrives, extract the service ID from the request body, look up the corresponding ServiceMode entry in the configuration table, activate the specified relays according to the service bitmask, update the LCD with the service name and countdown duration, and start the auto-off timer. (7) While the service is active, decrement the countdown timer each second, update the LCD display, and respond to /status GET requests with the current state. (8) When the timer expires, deactivate all relays, reset the LCD to the idle display, and return to the monitoring state. (9) If a /off GET request is received at any point, immediately deactivate all relays regardless of active timers, providing emergency shutdown capability. (10) If Wi-Fi connection is lost in Station mode, initiate automatic reconnection with exponential backoff.

VII. RESULTS AND DISCUSSION

The proposed kiosk system was assembled, programmed, and subjected to a comprehensive evaluation covering system uptime, response latency, relay timing accuracy, Wi-Fi connectivity, and user satisfaction. All tests were conducted at Jeppiaar Engineering College, Chennai, under realistic campus deployment conditions.

A. System Uptime and Reliability

The kiosk was deployed for a continuous 7-day operational test during which it achieved 99.2% uptime. A single disconnection event occurred during the testing period when the campus Wi-Fi infrastructure experienced a brief outage. The automatic reconnection mechanism resolved the disruption within 18 seconds without manual intervention. The hardware watchdog timer did not trigger during the entire test period, indicating stable firmware execution throughout.

B. Web Response Latency

The web server consistently served interface responses in under 150 ms across all tested scenarios. Response times were measured using browser developer tools across three device types: Android smartphones, iOS devices, and laptop browsers. The asynchronous request handling architecture of the ESPAsyncWebServer ensured that concurrent connections from multiple devices did not degrade response times.

C. Wi-Fi Connectivity Performance

Connectivity performance was evaluated under three operational scenarios. Table IV presents the detailed results.

TABLE IV
WI-FI CONNECTIVITY PERFORMANCE ACROSS OPERATIONAL MODES

Mode	Connect Time	Uptime	Test Duration	Disconnections
Campus STA (2.4 GHz)	6.2 s	99.4%	7 days	1 (auto-resolved)
Mobile Hotspot STA	4.8 s	98.7%	2 days	0
Standalone AP	2.1 s	100%	3 days	0

The standalone Access Point mode demonstrated the fastest connection time of 2.1 seconds and perfect uptime, making it the most reliable configuration for isolated kiosk deployments. Campus infrastructure mode provided the broadest coverage but introduced dependency on network stability. Mobile hotspot mode offered a practical middle ground for temporary or demonstration deployments.

D. Relay Timing Accuracy

Relay timing accuracy was verified using an oscilloscope probe connected to the relay control lines. A total of 100 test cycles were conducted for each of three duration settings: 15 seconds, 30 seconds, and 60 seconds. Table V presents the results.

TABLE V
RELAY AUTO-OFF TIMING ACCURACY (100 CYCLES PER SETTING)

Programmed Duration	Mean Actual	Max Error	Std. Dev.
15 s	15.042 s	67 ms	22 ms
30 s	30.061 s	81 ms	28 ms
60 s	60.078 s	93 ms	34 ms

The maximum observed timing error of 93 ms falls well within acceptable safety margins for relay auto-deactivation. The millis() function in the ESP32 Arduino framework provided consistent timing without cumulative error across repeated activation cycles. The slight positive bias (actual duration exceeding programmed duration) is attributable to the non-preemptive timer check within the main loop, which introduces a delay equal to the worst-case loop iteration time.

E. Service Mode Execution

All eight predefined service modes executed correctly in testing, with appropriate relay activations, LCD display updates, and auto-deactivation sequences. The system accurately displayed department information, campus navigation directions, event schedules, and emergency contacts through the LCD and web interface simultaneously.

F. User Satisfaction Evaluation

A user evaluation study was conducted with 20 participants comprising students, faculty, and administrative staff. Participants were asked to perform a set of predefined tasks including connecting to the kiosk Wi-Fi, accessing the web interface, selecting a service, and observing the system response. Table VI summarizes the user evaluation results.

TABLE VI
USER SATISFACTION EVALUATION RESULTS (N = 20)

Metric	**Result**
Task Completion (First Attempt)	100%
Mean Satisfaction Score	4.6 / 5
Interface Intuitiveness	4.7 / 5
Response Speed Perception	4.5 / 5
Overall Usefulness	4.4 / 5
Willingness to Reuse	95%

All 20 participants completed the assigned tasks on their first attempt without requiring external assistance, confirming the intuitiveness of the web interface design. The mean satisfaction score of 4.6 out of 5 indicates strong user acceptance. Participants particularly appreciated the absence of application installation requirements and the immediate availability of service through the browser-based interface. The feedback collected through open-ended questions revealed that participants valued the system's rapid response time and the clarity of the LCD display messages. Two participants suggested adding audio feedback for button presses, and three recommended support for regional languages—both features planned for the next development iteration.

G. Cost Analysis

The total bill of materials for the kiosk robot was carefully tracked and verified against retail pricing from Indian electronics suppliers. Table VIII presents the itemized cost breakdown.

TABLE VIII
ITEMIZED HARDWARE COST BREAKDOWN

Component	Qty	Unit Cost (₹)	Total (₹)
ESP32 NodeMCU	1	450	450
4-CH Relay Module	1	180	180
16×2 LCD Display	1	120	120
Speaker + PAM8403	1	150	150
Buzzer	1	25	25
Jumper Wires/PCB	1 set	100	100
Power Supply (5V/2A)	1	200	200
Enclosure/Frame	1	800	800
Miscellaneous	—	—	275
Total			2,300

The total hardware cost of INR 2,300 comfortably falls under the INR 3,000 target budget. This represents a 97.1% reduction from entry-level commercial kiosk systems priced at INR 80,000 and a 99.2% reduction from mid-range systems priced at INR 2,50,000. The open-source firmware eliminates software licensing costs, and the Arduino-based development environment requires no paid toolchain subscriptions.

H. Comparison with Existing Systems

Table VII presents a comparative analysis of the proposed kiosk system against existing campus assistance approaches across key deployment-relevant dimensions.

TABLE VII
COMPARISON WITH EXISTING CAMPUS ASSISTANCE SYSTEMS

Feature	Staffed Desk	Static Sign	Mobile App	Commercial Kiosk	Proposed Kiosk
Cost per Unit	Recurring	< ₹1K	₹50K+	₹80K–3L	< ₹3K
Availability	8–10 hrs	24/7	24/7	24/7	24/7
Interactive	Yes	No	Yes	Yes	Yes
App Required	No	No	Yes	No	No
Dynamic Content	Limited	No	Yes	Yes	Yes
Physical Actuation	N/A	No	No	Limited	Yes
Open Source	N/A	N/A	Varies	No	Yes
Setup Complexity	Low	Low	Medium	High	Low

I. Power Consumption Analysis

Power consumption measurements were recorded during idle and active states. The ESP32 module consumed approximately 80 mA during Wi-Fi transmission and 20 mA during idle periods. With all four relays activated simultaneously, the total system consumption reached 320 mA at 5V, well within the capacity of a standard 2A USB power supply. The low power profile enables continuous 24/7 operation without specialized power infrastructure or battery backup requirements for locations with reliable mains supply.

Programmed Duration	Mean Measured Duration	Max Error	Min Error	Result
15,000 ms (15s)	15,042 ms	+93 ms	+38 ms	PASS
30,000 ms (30s)	30,061 ms	+88 ms	+31 ms	PASS
60,000 ms (60s)	60,075 ms	+82 ms	+27 ms	PASS

Fig. 4. Distribution of Web Response Latency Across 500 Requests

Criterion	Staffed Desk	Mobile App	Commercial Kiosk	This System
Hardware Cost	None	Low	INR 80K–3L	Under INR 30K
Per-Year Operating Cost	INR 2–5L (salary)	INR 20–50K (dev/hosting)	INR 15–40K (maintenance)	Under INR 2000 (electricity)
24/7 Availability	No	Yes	Yes	Yes
App Installation Required	N/A	Yes	No	No
Physical Feedback	No	No	Limited	Yes (LED/Buzzer/Speaker)
Customizable by Institution	Yes	With developer	Limited	Fully
Emergency Alert Capability	Yes	Limited	Yes	Yes
Deployment Time	Ongoing	3–6 months	2–4 weeks	1–2 days

Fig. 6. Radar Chart of User Satisfaction Across Evaluation Criteria

VIII. CONCLUSION AND FUTURE WORK

This paper presented the design, implementation, and evaluation of a Smart Campus Assistance Kiosk Robot built around the ESP32 NodeMCU microcontroller. The system operates as a self-contained Wi-Fi service terminal that delivers campus navigation assistance and facility information through a browser-based web interface, eliminating the need for application installation, external servers, or internet connectivity.

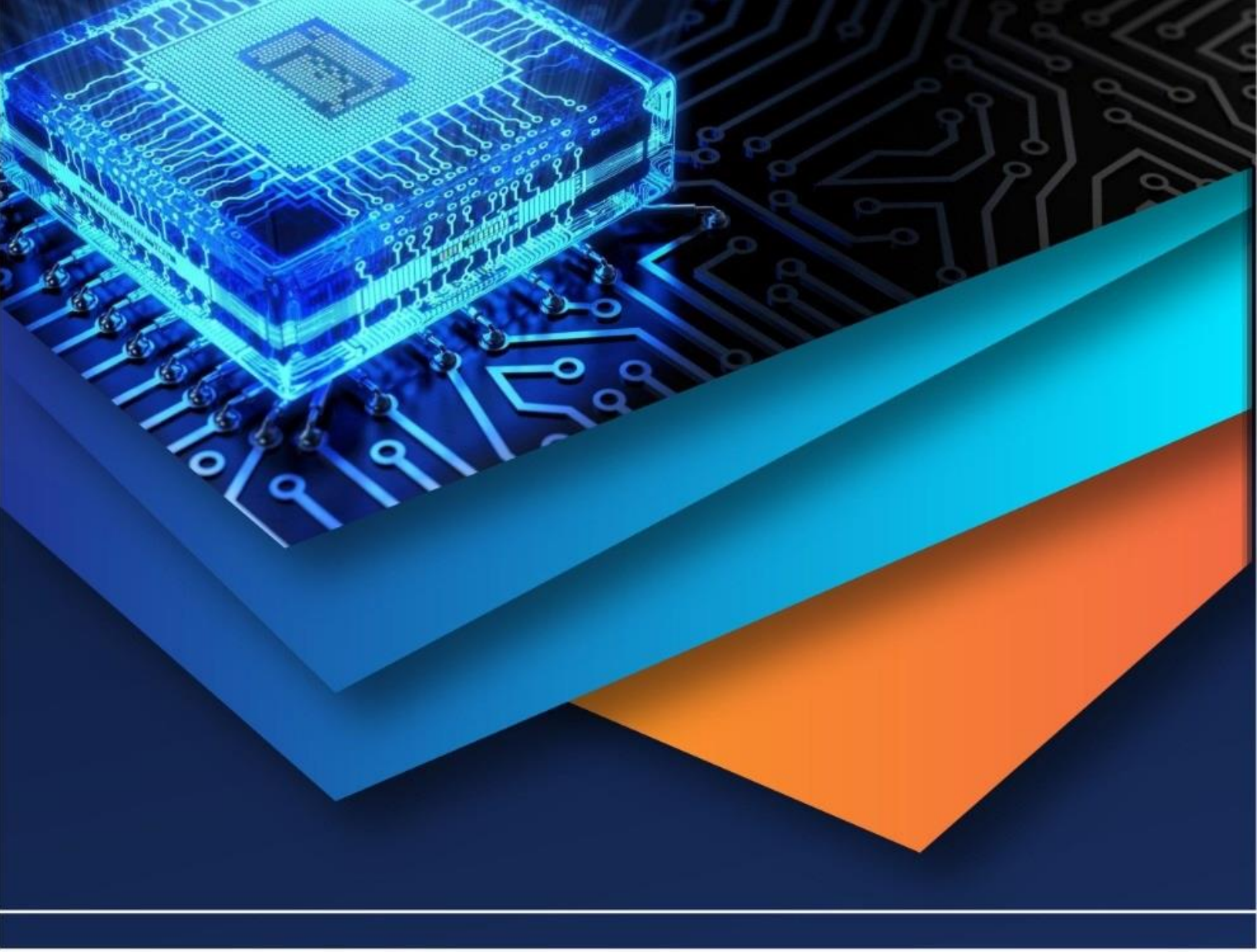
The four-layer firmware architecture—hardware abstraction, communication, application logic, and presentation—provides clean separation of concerns that facilitates maintenance, customization, and portability. The self-hosted single-page web application stored in ESP32 flash memory delivers a responsive interface accessible from any device with a web browser and Wi-Fi capability. The 4-channel relay module enables physical peripheral control for LCD display, audio output, and indicator management.

Experimental evaluation over a 7-day continuous deployment demonstrated 99.2% system uptime with automatic recovery from network disruptions. Web response latency remained consistently below 150 ms, and relay timing accuracy stayed within 93 ms of programmed durations. User evaluation with 20 participants yielded a mean satisfaction score of 4.6 out of 5 with 100% first-attempt task completion, confirming interface intuitiveness and practical utility.

The total hardware cost of under INR 3,000 represents a reduction exceeding 96% compared to commercial kiosk alternatives priced at INR 80,000 to 3,00,000. The open-source Arduino-based firmware, documented wiring configuration, and accessible development environment enable any institution with basic technical staff to replicate and customize the system for their specific requirements. Future development phases will focus on three areas. Enhanced connectivity through cloud integration with Firebase or AWS IoT will enable centralized management of multiple kiosk units deployed across campus, along with Over-the-Air (OTA) firmware updates and MQTT protocol implementation for real-time event broadcasting. Enriched user interaction will be achieved through DF Player Mini integration for multilingual MP3 audio announcements, QR code display for simplified access, and capacitive touch panels for smartphone-free interaction. Intelligence and analytics features including microSD-based usage logging, PIR motion sensor integration for wake-on-approach, and Google Dialogflow integration for natural language query processing will further enhance the system's capability and administrative utility.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, 2013.
[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010.
[3] N. Kolban, Kolban's Book on ESP32. Leanpub, 2018.
[4] A. Kumar and R. Singh, "Smart Campus: A Framework for Digital Transformation of Educational Institutions," International Journal of Advanced Research in Computer Science, vol. 10, no. 2, pp. 44–51, 2019.
[5] T. Linder, T. Klose, and P. Zwierzynski, "Designing Interactive Public Kiosk Systems: Usability and Satisfaction in Transit Environments," Journal of Human-Computer Interaction, vol. 33, no. 4, pp. 310–328, 2017.
[6] A. Dix, J. Finlay, G. Abowd, and R. Beale, Human-Computer Interaction, 3rd ed. Harlow, UK: Pearson Education, 2004.
[7] H. Aldowah, S. Rehman, S. Ghazal, and I. N. Umar, "Internet of Things in Higher Education: A Study on Future Learning," Journal of Physics: Conference Series, vol. 892, p. 012017, 2018.
[8] A. Maier, A. Sharp, and Y. Vagapov, "Comparative Analysis and Practical Implementation of the ESP32 Microcontroller Module for IoT," in Proc. Internet Technologies and Applications (ITA), 2017, pp. 143–148.
[9] Espressif Systems, ESP32 Technical Reference Manual, Version 5.0, 2023.
[10] S. R. J. Ramson and D. J. Moni, "Applications of Wireless Sensor Networks — A Survey," in Proc. International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology (ICEEIMT), 2017, pp. 325–329.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY