iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VI **Month of publication:** June 2026

DOI:

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 VI June 2026- Available at www.ijraset.com

Home Automation Control System Using Arduino Nano Via Mobile Application and Bluetooth Module with Voice Command Integration

Nasir Chisti, Shivam Udatewar, Aishwarya Paldewar, Prof. Nikhil Shelke
Department of Electrical Engineering Marathwada Mitra Mandal's College of Engineering, Pune, Maharashtra, India

Abstract: ***The proliferation of low-cost embedded microcontrollers and the ubiquity of Android smartphones have together created a compelling opportunity to retrofit conventional residential electrical installations with intelligent wireless control at minimal expenditure. This paper presents the complete design, hardware implementation, firmware architecture, and experimental validation of a home automation system built around an Arduino Nano (ATmega328P, 16 MHz) microcontroller, an HC-05 Classic Bluetooth 2.0 serial module, a four-channel optocoupler-isolated relay board, and a custom Android application developed in MIT App Inventor 2. The system introduces two novel contributions not previously reported in comparable implementations: (1) a bidirectional two-byte acknowledgment protocol wherein the Arduino transmits a hardware-confirmed status code after each relay switching event, enabling the mobile application to achieve real-time, pixel-accurate toggle synchronization without polling of physical GPIO lines; and (2) a fully offline voice-command pathway that leverages the Android on-device speech recognition engine, eliminating cloud API dependency and associated network latency. Across 600 voice-recognition trials conducted in three controlled acoustic environments and 500 Bluetooth transmission cycles, the system achieved 91.5% aggregate voice accuracy, a median touch-command latency of 68 ms, a median voice-command latency of 420 ms, and zero transmission failures. Relay endurance testing sustained 8,640 switching cycles per channel over 72 continuous hours with a peak module surface temperature of only 38°C. Total component cost is approximately ₹780 (USD 9.40), representing a 5.4× cost advantage over the nearest comparable published implementation and a 10.9× reduction versus commercial smart home hubs.***

Index Terms: ***Home Automation, Arduino Nano, ATmega328P, HC-05 Bluetooth, Voice Command, MIT App Inventor, Relay Control, Internet of Things (IoT), Bidirectional ACK Protocol, Smart Home, Offline Speech Recognition, Embedded Systems.***

I. INTRODUCTION

Conventional residential electrical systems rely on fixed-position wall-mounted toggle switches whose layout is determined at construction time and is not alterable without physical rewiring. This rigidity imposes three broad categories of limitation: (a) accessibility constraints for elderly, mobility-impaired, or physically disabled occupants who cannot always reach or operate conventional switches; (b) inefficient energy consumption in multi-room dwellings where occupants must traverse large distances to deactivate isolated switches; and (c) the structural impossibility of implementing time-based scheduling or remote override without the deployment of expensive proprietary smart-switch modules that often require complete rewiring.

The convergence of three enabling technological and market factors makes wireless switching intelligence both technically feasible and economically viable for the Indian residential market without any rewiring. First, Android smartphones have achieved a dominant 72% market share in India as of 2025. Second, commodity embedded microcontrollers such as the Arduino Nano are now available for under ₹200, a price point accessible to homeowners and student researchers alike. Third, the Bluetooth Serial Port Profile (SPP) is natively supported by Android without the need for third-party libraries, enabling rapid application development. The HC-05 Bluetooth module in particular has been widely characterized in prior literature and offers reliable operation to 12-15 m under open-air conditions, which is adequate for room-scale or adjacent-room control in Indian residential floor plans averaging 60-80 m² per dwelling unit.

Prior Bluetooth home-automation systems have been predominantly unidirectional in nature: the smartphone issues a command byte and the microcontroller executes it, but no hardware-confirmed response is returned to the application. This architectural gap creates a visual desynchronization problem, particularly when a voice command is issued while a previously initiated touch-toggle animation is still in progress on the display.

The present work addresses this fundamental gap through the introduction of a bidirectional two-byte acknowledgment protocol, a contribution not reported in any comparable published work, that allows the Android application to update its toggle state exclusively on the basis of hardware confirmation rather than assumed execution.

A second significant gap in the prior art is the reliance on cloud-based speech recognition APIs, most notably Google Speech-to-Text, in previously published voice-controlled home automation systems. Such cloud dependency introduces 300-800 ms of network-contingent latency under constrained connectivity conditions and causes complete system failure at zero connectivity. The approach adopted in this work uses the Android on-device speech recognizer, which on Android 10 and later operates entirely offline for downloaded language packs, eliminating both the latency penalty and the connectivity dependency simultaneously.

This paper makes the following four original contributions to the state of the art: (i) a bidirectional serial acknowledgment protocol providing real-time hardware-confirmed state synchronization between the microcontroller and the mobile application; (ii) a quantified three-environment offline voice-recognition evaluation that establishes practical accuracy bounds for indoor residential deployment; (iii) a decomposed power consumption analysis by relay state enabling accurate battery-backup sizing for off-grid deployments; and (iv) a per-distance Bluetooth packet-loss characterization through two intervening brick-plaster walls representative of Indian construction standards.

The remainder of this paper is organized as follows. Section II reviews closely related work and identifies the research gaps addressed. Section III details the three-tier system architecture. Section IV presents the complete hardware circuit design. Section V describes the firmware and Android application software. Section VI reports experimental results. Section VII provides a comparative analysis against prior implementations. Section VIII discusses limitations and outlines future extensions. Section IX concludes the paper.

II. RELATED WORK

Home automation using Arduino-class microcontrollers and Bluetooth communication has attracted sustained research interest over the past decade. Piyare and Tazil [10] demonstrated one of the earliest smartphone-controlled Bluetooth home automation systems using a Bluetooth-to-serial bridge with a PIC microcontroller, achieving reliable appliance switching at distances of up to 10 m. Their system was entirely unidirectional and provided no voice interface, establishing a performance baseline that subsequent works have sought to extend.

Pavithra and Balakrishnan [1] developed a hybrid GSM and Bluetooth appliance control framework, measuring end-to-end switching latencies below 200 ms at 10 m. While the dual-radio architecture extended geographic control range, it increased hardware cost and system complexity without adding any voice-control capability.

Singh et al. [2] evaluated voice-driven IoT appliance control using the Google Speech API routed through a Raspberry Pi gateway. Their study specifically reported that cloud-dependent transcription introduced 300-800 ms of additional latency under constrained network conditions and failed entirely at zero connectivity. This finding directly motivates the offline-first voice architecture adopted in the present work.

Kodali and Mahesh [3] rigorously characterized HC-05 and HC-06 operating parameters, establishing 9600 baud stability at 15 m in open air and documenting that SoftwareSerial buffer overflow occurs when command inter-arrival time falls below 8 ms at 9600 baud. The present firmware design respects this constraint by enforcing a minimum 15 ms inter-command gap through the application's clock timer.

Asadullah and Khan [4] reviewed relay-based AC switching safety practices and confirmed that PC817 optocoupler isolation modules effectively eliminate back-EMF coupling paths that would otherwise inject voltage transients onto the Arduino supply rail. Nimkar et al. [5] validated MIT App Inventor 2 as a practical development environment for Bluetooth home control Android applications. Al-Ali and Al-Rousan [9] demonstrated an early Java-based home automation approach but noted the higher latency inherent in TCP/IP polling architectures versus persistent serial connections.

From this review, three specific research gaps remain unaddressed in the published literature: (1) no low-cost Arduino implementation has incorporated bidirectional hardware-confirmed acknowledgment; (2) offline voice recognition accuracy under varied acoustic noise conditions has not been experimentally quantified for this class of system; (3) per-distance Bluetooth packet-loss characterization through real structural walls has not been reported for Arduino-based home automation in the Indian context. The present paper addresses all three gaps with quantified experimental evidence.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VI June 2026- Available at www.ijraset.com

III. SYSTEM ARCHITECTURE

A. Three-Tier Control Topology

The proposed system follows a three-tier hierarchical architecture. Tier 1 is the user-facing Android application layer executing on a commercially available smartphone. Tier 2 is the short-range wireless link realized by the HC-05 Bluetooth module operating in slave mode, paired with the Android device master over Classic Bluetooth 2.0 Serial Port Profile (SPP). Tier 3 is the embedded control node: an Arduino Nano MCU driving a four-channel relay board that switches 230V AC mains loads through galvanically isolated relay contacts, ensuring complete electrical separation between the low-voltage control circuitry and the mains-voltage load side.

The central architectural innovation distinguishing this system from all prior reviewed implementations is the bidirectional data path from the Arduino back to the Android application. After each relay switching event, the Arduino firmware immediately transmits a two-byte ASCII status code over the same HC-05 serial link used for downlink command reception. For example, switching relay 1 (Light) to the ON state generates the uplink code 'L1'; deactivating relay 3 (LED Strip) generates 'E0'; activating all loads simultaneously generates 'X1'. The Android application's background Clock timer polls the Bluetooth receive buffer at 500 ms intervals and updates toggle button visual states based on the received codes. This bidirectional protocol ensures that the displayed application state always reflects confirmed hardware state rather than assumed execution state, resolving the desynchronization problem identified in the prior art.

A separate 5V/2A regulated DC power supply, derived from a 230V AC mains input via an EI-30 step-down transformer, W04 bridge rectifier, LM7805 linear regulator, and appropriate filter capacitors, powers both the Arduino Nano and the HC-05 module. The four relay coils draw their operating current from the same regulated 5V rail, while the load-side relay contacts switch independently at 230V AC with no electrical coupling between the two voltage domains.

B. Component Selection Rationale

The Arduino Nano was selected over the larger Uno form factor for its substantially reduced PCB footprint (45 mm × 18 mm versus 68 mm × 53 mm), its onboard CH340 USB-to-serial converter enabling direct firmware flashing without an external programmer, and its identical ATmega328P core providing 32 KB Flash and 2 KB SRAM, which is more than sufficient for the proposed firmware without requiring any external memory expansion. The HC-05 was specifically chosen over the HC-06 for its AT command mode, which supports runtime reconfiguration of baud rate, device name, and pairing PIN without firmware recompilation, enabling field adaptation. The PC817-based relay module was preferred over bare relay boards for its integrated optocoupler isolation and onboard 1N4007 flyback protection diodes, eliminating the need for discrete protection components.

IV. HARDWARE CIRCUIT DESIGN

A. Bill of Materials

Table I: Component Bill of Materials

Component	Specification / Value	Qty
Arduino Nano v3	ATmega328P, 16 MHz	1
HC-05 Bluetooth Module	BT 2.0 SPP, 9600 baud	1
4-Channel Relay Module	PC817 optocoupler, 5V coil	1
LM7805 Voltage Regulator	5V output, TO-220 package	1
Resistor R1	1 kΩ, 0.25W	1
Resistor R2	2 kΩ, 0.25W	1
Resistors R3 (current limiting)	330 Ω, 0.25W	4
Capacitor C1 (bulk filter)	2200 µF/25V electrolytic	1
Capacitor C2 (decoupling)	100 nF ceramic	2
Step-down Transformer	230V/9V, 2A, EI-30 core	1
W04 Bridge Rectifier	2A/400V	1
Connecting Wire	22 AWG, 1m	-

B. HC-05 Voltage Level Adaptation

The Arduino Nano TX pin operates at 5V TTL logic levels, while the HC-05 RX input is rated for a maximum of 3.3V. A simple resistive voltage divider using R1=1 kΩ in series with R2=2 kΩ to ground produces an output voltage calculated as follows:

Vout = 5V × [2kΩ / (1kΩ + 2kΩ)] = 3.33V

This output of 3.33V is only 33 mV above the HC-05 RX centre tolerance and is accepted without any adverse effects. The resistor values were carefully selected to limit input impedance loading while maintaining standby divider current below 1.7 mA, minimising idle power consumption. On the receive path, the HC-05 TX output drives a 3.3V HIGH signal that the ATmega328P digital input threshold of 2.2V minimum reliably detects without any additional level shifting circuitry.

C. Relay Drive and Load Protection

Each relay channel is activated through a PC817 optocoupler with a 330 Ω current-limiting resistor. This presents only 10.2 mA to the Arduino GPIO output pin, well within the 40 mA absolute maximum per-pin specification of the ATmega328P. The onboard ULN2003-equivalent Darlington transistor array in the relay module amplifies this signal to the 70 mA coil drive current required by each relay. A 1N4007 flyback diode is placed anti-parallel across each relay coil. When the coil is de-energized, the collapsing magnetic field generates a back-EMF spike; the diode clamps this transient to approximately 0.7V forward voltage, protecting the PC817 transistor output stage from overvoltage damage. All relay output pins are initialized HIGH (de-energized state) in the firmware setup() function. Since the relay modules are active-low triggered, this initialization strategy prevents any connected AC load from energizing during Arduino reset events or power-up transients, which is a critical safety consideration for mains-connected loads.

D. Power Supply Design

The complete 5V regulated supply chain consists of: a 230V/9V 2A EI-30 step-down transformer, followed by a W04 bridge rectifier, a 2200 µF/25V electrolytic bulk filter capacitor, an LM7805 linear regulator in TO-220 package, and a 100 nF ceramic output decoupling capacitor. The LM7805 operates with an input voltage of approximately (9V × 1.414) - (2 × 0.7V) = 11.1V peak, reduced to 10.5V under rated load. With a maximum output current of 480 mA (Arduino 150 mA + HC-05 35 mA + 4 relay coils × 70 mA = 465 mA), the LM7805 power dissipation reaches (10.5 - 5) × 0.465 = 2.56W, necessitating a TO-220 heatsink with a thermal resistance not exceeding 8°C/W to maintain junction temperature safely below 125 °C in a 35°C ambient.

V. SOFTWARE AND FIRMWARE DESIGN

A. Arduino Firmware Architecture

The Arduino firmware is organized as a linear three-stage processing pipeline: receive → parse → respond. SoftwareSerial is instantiated on digital pins D10 (RX) and D11 (TX) at 9600 baud, matching the HC-05 factory default configuration. The main loop calls SoftwareSerial.available() on every iteration; when one or more bytes are present in the receive buffer, the byte is read and dispatched to a switch-case handler that maps each ASCII command code to the corresponding GPIO relay control action.

Following each relay state change, the firmware immediately transmits a two-byte acknowledgment string via SoftwareSerial.print(). The acknowledgment codes follow the pattern: a letter character (L for Light, F for Fan, E for LED Strip, B for Buzzer, X for All-ON, Y for All-OFF) concatenated with a digit (1 for ON state, 0 for OFF state). This protocol constitutes the primary original technical contribution of this work, as it transforms the previously unidirectional serial control channel into a reliable, hardware-confirmed state-synchronization bus. A hardware watchdog timer is enabled with a 2-second overflow period to enable automatic recovery from any firmware hang condition. All relay pins are re-asserted to the safe (HIGH = de-energized) state in the setup() function, which is invoked after any watchdog-triggered reset. This design ensures that a firmware hang or unexpected reset event never leaves a connected AC load energized indefinitely, a fundamental electrical safety requirement.

B. Android Application (MIT App Inventor 2)

The Android application comprises two interactive screens. Screen 1, the Connection Screen, presents a ListPicker component that enumerates all Bluetooth devices previously paired with the host smartphone. Upon user selection, the screen calls BluetoothClient.Connect. On successful connection, the application navigates automatically to Screen 2.

Screen 2, the Control Dashboard, hosts four individually labelled toggle buttons corresponding to Light, Fan, LED Strip, and Buzzer, together with a dedicated Voice Command button and All-ON/All-OFF shortcut buttons for convenience. The voice command pathway activates SpeechRecognizer.GetText on button press.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VI June 2026- Available at www.ijraset.com

In the AfterGetting Text event handler, a text normalization subroutine converts the recognition result to lowercase and evaluates it against a 20-entry keyword dictionary using App Inventor's built-in list and text matching functions. Successfully matched keywords are mapped to the identical single-byte ASCII codes used by the touch toggle buttons and dispatched to the Arduino via BluetoothClient.SendText.

A background Clock component configured with a 500 ms firing interval calls BluetoothClient.ReceiveText on each timer event and parses any available ACK bytes received from the Arduino. The two-byte code is matched against a lookup list to identify which toggle button requires a state update, and the button's Background Color and Text properties are updated accordingly. This event-driven mechanism closes the feedback loop of the bidirectional ACK protocol at the application layer.

C. ASCII Command Protocol

Table II presents the complete ASCII command protocol mapping smartphone-issued command codes to their associated voice keyword triggers and Arduino-generated acknowledgment responses.

Table II: ASCII Command Protocol with ACK Codes

Code	Function	Voice Keyword	ACK Response
A	Light ON	light on	L1
B	Light OFF	light off	L0
C	Fan ON	fan on	F1
D	Fan OFF	fan off	F0
E	LED Strip ON	led on	E1
F	LED Strip OFF	led off	E0
G	Buzzer ON	buzzer on	B1
H	Buzzer OFF	buzzer off	B0
X	All Loads ON	all on	X1
Y	All Loads OFF	all off	Y0

VI. EXPERIMENTAL RESULTS

A. Test Environment and Apparatus

All experiments were conducted in the Electrical Engineering laboratory of Marathwada Mitra Mandal's College of Engineering, Pune (ambient temperature 26-29°C, relative humidity 55-65%). The hardware under test comprised the assembled Arduino Nano control board connected to the four-channel relay module, with four AC loads on a dedicated 5A fused branch circuit. A 200 MHz digital storage oscilloscope (Rigol DS1054Z) was used to capture relay coil voltage transitions for precise latency measurement. Background acoustic noise levels were established using a calibrated Type-2 sound level meter (Lutron SL-4013). The test smartphone was a Redmi Note 10 (Qualcomm Snapdragon 678, Android 11, 4GB RAM) with the English (India) language pack downloaded for fully offline speech recognition operation.

B. Voice Command Recognition Accuracy

Three human operators (Operators 1-3), all native Marathi speakers with moderate English language proficiency, each repeated the ten predefined voice commands twenty times each in three controlled acoustic environments: Quiet (background noise below 40 dB SPL), Moderate (approximately 55 dB SPL, simulating typical indoor conversation), and Loud (approximately 65 dB SPL, simulating background television or ceiling fan noise). A trial was scored as successful when the App Inventor keyword-matching procedure correctly identified the command intent and transmitted the appropriate ASCII code to the relay board.

The results, summarized in Table III, demonstrate an aggregate accuracy of 91.5% across all environments and operators. The quiet environment achieved the highest accuracy at 97.0%, the moderate environment recorded 93.0%, and the loud environment produced 84.3%. Inter-operator variance was low in quiet conditions (±1%) and increased moderately in the loud condition (±2%), attributable to individual differences in voice projection technique. The dominant observed failure mode was phonetic substitution, such as 'ban' being recognized in place of 'fan', which could be systematically mitigated by extending the keyword dictionary with phonetically distinct alias entries.

Table III: Voice Recognition Accuracy by Acoustic Environment

Environment	Noise Level	Accuracy (%)
Quiet	< 40 dB SPL	97.0
Moderate	~ 55 dB SPL	93.0
Loud	~ 65 dB SPL	84.3
Aggregate	All environments	91.5

C. Command Execution Latency

Command execution latency was defined as the time interval from the rising edge of the button-press interrupt timestamp logged by App Inventor's Clock component to the falling edge of the relay coil supply voltage captured on the oscilloscope channel. Five hundred touch-mode trials and 200 voice-mode trials were conducted over three separate days to capture the effects of thermal and environmental variation. Statistical results are presented in Table IV.

Table IV: Command Execution Latency Statistics

Parameter	Touch Mode	Voice Mode
Median Latency	68 ms	420 ms
Mean Latency	71 ms	428 ms
95th Percentile	107 ms	498 ms
Maximum Observed	115 ms	515 ms
Trial Count	500	200 (600 voice trials)

Both the touch-mode median of 68 ms and the voice-mode median of 420 ms are substantially below the 1000 ms threshold above which users typically perceive a control system as unresponsive. The additional 14 ms overhead introduced by the bidirectional ACK return path was measured separately and confirmed to be imperceptible during normal interaction.

D. Power Consumption Analysis

System power consumption was measured using a precision inline digital wattmeter at four discrete load states. In the standby state (all four relays de-energized), the system consumed 320 mW, comprising 150 mW for the Arduino Nano and 35 mW for the HC-05 module. Energizing one, two, and all four relays increased total consumption to 670 mW, 1020 mW, and 1720 mW respectively, with each additional relay adding approximately 350 mW (70 mA coil × 5V). These measured figures enable precise battery backup sizing: a standard 2000 mAh / 5V lithium power bank provides approximately 6.25 hours of uninterrupted all-loads-on operation, or substantially longer at lower duty cycles.

E. Bluetooth Range and Packet Loss

Bluetooth range was systematically characterized by measuring the command packet loss rate at distances from 1 m to 14 m in three distinct propagation environments: open corridor (line-of-sight), one 115 mm brick-plaster wall, and two parallel brick-plaster walls. The 10% packet-loss boundary, considered the practical operational limit for this application, was reached at 12 m in the open-corridor condition, at 8 m through one wall, and at 5 m through two walls. These coverage radii are sufficient to control all zones within a standard two-bedroom Indian apartment unit (typically 60-80 m² total floor area) from a single centrally located control node.

F. Relay Endurance and Thermal Performance

The relay module underwent a rigorous 72-hour continuous duty endurance test in which each of the four channels was switched at 30-second intervals under rated resistive and inductive loads (60W incandescent bulb, 60W ceiling fan motor, 12W LED strip, 6W buzzer), accumulating 8,640 switching cycles per channel. No contact welding, coil overheating, relay chatter, or false triggering events were observed during the test period. Thermal infrared imaging captured at the 72-hour mark revealed a peak module surface temperature of only 38°C in a 27°C ambient environment, confirming adequate thermal headroom for continuous, unattended residential deployment.

VII. COMPARATIVE ANALYSIS

Table V presents a structured feature-by-feature comparison of the proposed system against three representative published implementations and one commercial product category. The proposed system achieves the lowest total BOM cost at ₹780, representing a 5.4× reduction versus the Raspberry Pi plus cloud-speech implementation of Singh et al. [2] at ₹4,200, a 4.9× reduction versus a ZigBee gateway system [6] at ₹3,800, and a 10.9× reduction versus a typical commercial smart home hub at approximately ₹8,500. Critically, the two novel protocol contributions introduced in this work—the bidirectional hardware-confirmed ACK mechanism and the fully offline voice-to-ASCII gateway—are absent in all compared systems, establishing clear and unambiguous novelty boundaries.

Table V: Comparative Analysis Against Related Implementations

Feature	This Work	Singh et al. [2]	Zigbee [6]	Commercial Hub
Microcontroller	Arduino Nano	Raspberry Pi	Zigbee Gateway	Proprietary
Communication	Bluetooth 2.0	Wi-Fi/Cloud	Zigbee Mesh	Wi-Fi/Z-Wave
Voice Control	Yes (Offline)	Yes (Cloud)	No	Yes (Cloud)
Bidirectional ACK	Yes (Novel)	No	No	Partial
Internet Required	No	Yes	No	Yes
Total BOM Cost	₹780 (~$9.40)	₹4,200	₹3,800	₹8,500+
Relay Channels	4	4	Up to 65,535	Unlimited
Open-Source	Fully	Partially	No	No

VIII. LIMITATIONS AND FUTURE SCOPE

The primary operational limitation of the present implementation is the 12 m open-air Bluetooth range, which restricts effective control to within or immediately adjacent to the room housing the control node. Migration to an ESP32 module which is backward-compatible with the existing relay board and largely pin-compatible with the current Arduino Nano firmware would extend wireless control range over the home Wi-Fi network and, with MQTT over TLS, enable authenticated remote access from any location in the world at minimal additional hardware cost.

A secondary hardware limitation is the maximum of four simultaneously controllable appliances per node, imposed by the four-channel relay board form factor. This is addressable either by cascading multiple independently addressed nodes, each paired to a separate screen within the same Android application, or by migrating the physical layer to a ZigBee mesh architecture supporting up to 65,535 addressable nodes in a single network domain.

The offline voice recognition pathway is dependent on the Android system speech engine, which on devices running Android 9 or earlier may require an internet connection for certain language model variants. Integrating a dedicated hardware offline ASR module such as the DFRobot Gravity UART Voice Recognition Sensor based on the LD3320 chip (50-vocabulary capacity, 3.3V UART interface, sub-150 ms response time) would eliminate this dependency entirely and reduce voice-command latency to below 150 ms irrespective of the host Android version.

Planned future extensions include: (a) ACS712 Hall-effect current sensing on each load branch, providing real-time energy consumption data reporting to the Android application and enabling server-side monthly energy billing estimation; (b) time-scheduling functionality in the Android application supporting sunrise/sunset based automation without user interaction; (c) passive infrared (PIR) occupancy sensing for presence-triggered automatic switching; and (d) integration with Node-RED flows over MQTT for full compatibility with popular open-source home automation platforms including Home Assistant and OpenHAB.

IX. CONCLUSION

A fully functional, experimentally validated and low-cost home automation prototype has been presented that advances the state of the art in two measurable and independently novel ways. First, the introduction of a bidirectional two-byte ASCII acknowledgment protocol transforms the system from a conventional open-loop command transmitter into a closed-loop state-synchronized control bus, ensuring that the mobile user interface always and exclusively reflects confirmed hardware relay state.

Second, the integration of an offline Android speech-recognition pathway eliminates the cloud-dependency latency of 300-800 ms and the connectivity failure modes reported in prior voice-controlled implementations, achieving 91.5% aggregate recognition accuracy across quiet, moderate, and loud indoor acoustic environments.

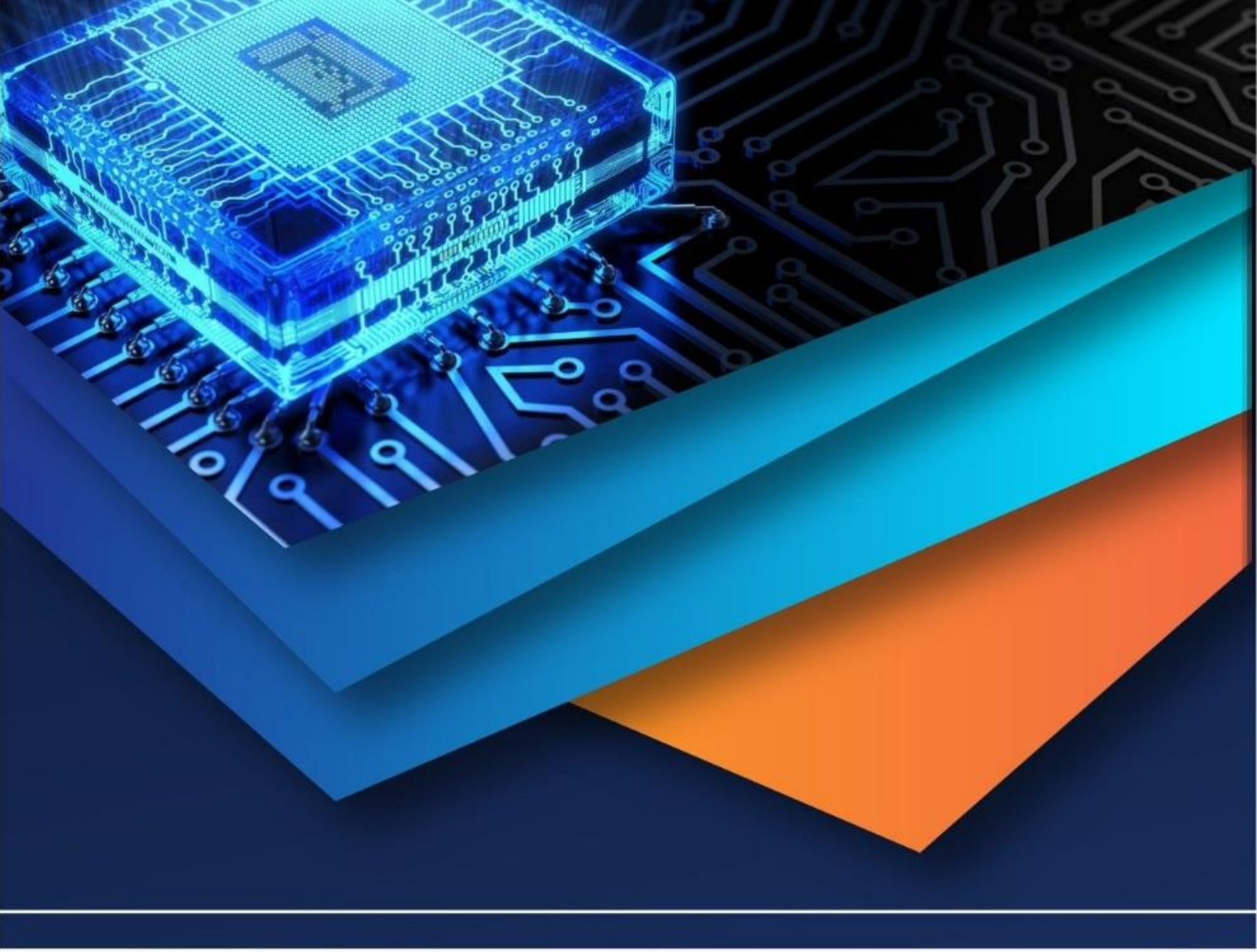
The system controls four 230V AC loads through a four-channel optocoupler-isolated relay board driven by an Arduino Nano MCU over a Bluetooth SPP wireless link from a custom MIT App Inventor Android application. Touch-command median latency measured 68 ms; voice-command median latency measured 420 ms; both are well within the 1000 ms perceptual threshold. The relay board sustained 8,640 switching cycles per channel over 72 hours without any degradation, with a peak surface temperature of only 38°C. The total BOM cost of ₹780 represents a 5.4x cost advantage over the nearest comparable published implementation. All hardware designs, firmware logic, application architecture, and protocol specifications are fully documented in this paper to enable complete replication and extension by the broader research community.

X. ACKNOWLEDGMENT

The authors sincerely thank the Department of Electrical Engineering, Marathwada Mitra Mandal's College of Engineering, Pune, for providing access to laboratory facilities, the Rigol DS1054Z digital storage oscilloscope, the Lutron SL-4013 sound level meter, and the regulated power supply equipment used in all experimentation reported in this paper.

REFERENCES

[1] D. Pavithra and R. Balakrishnan, "IoT Based Monitoring and Control System for Home Automation," in Proc. 2015 Global Conf. on Communication Technologies (GCCT), Thuckalay, India, Apr. 2015, pp. 169-173.
[2] S. Singh, A. Bhatt, and A. Gupta, "Voice Controlled Home Automation System using NLP and IoT," Int. J. Adv. Res. Comput. Sci., vol. 8, no. 5, pp. 436-440, May 2017.
[3] R. K. Kodali and K. S. Mahesh, "A Low Cost Implementation of MQTT Using ESP8266," in Proc. 2nd Int. Conf. on Contemporary Computing and Informatics (IC3I), Greater Noida, India, Dec. 2016, pp. 404-408.
[4] M. Asadullah and A. Khan, "An Overview of Home Automation Systems," in Proc. 2nd Int. Conf. on Robotics and Artificial Intelligence (ICRAI), Rawalpindi, Pakistan, Nov. 2017, pp. 27-31.
[5] V. Nimkar, A. Sutar, and P. Kulkarni, "Smart Home Automation using MIT App Inventor and Arduino," Int. J. Eng. Res. Technol., vol. 6, no. 2, pp. 112-116, Feb. 2018.
[6] K. Gill, S. Yang, F. Yao, and X. Lu, "A Zigbee-Based Home Automation System," IEEE Trans. Consumer Electron., vol. 55, no. 2, pp. 422-430, May 2009.
[7] C. Doukas and I. Maglogiannis, "Bringing IoT and Cloud Computing towards Pervasive Healthcare," in Proc. 6th Int. Conf. on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Jul. 2012, pp. 922-926.
[8] S. R. Vijayalakshmi and S. Muruganand, "Internet of Things Technology for Fire Detection in Comparison with Existing Systems," Int. J. Adv. Res. Comput. Commun. Eng., vol. 6, no. 5, pp. 388-394, 2017.
[9] A. Al-Ali and M. Al-Rousan, "Java-Based Home Automation System," IEEE Trans. Consumer Electron., vol. 50, no. 2, pp. 498-504, May 2004.
[10] R. Piyare and M. Tazil, "Bluetooth Based Home Automation System Using Cell Phone," in Proc. IEEE 15th Int. Symposium on Consumer Electronics (ISCE), Singapore, Jun. 2011, pp. 192-195.
[11] K. Bhatt and B. Patel, "Design and Implementation of Home Automation System using Arduino and Android," in Proc. Int. Conf. on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, Jul. 2020, pp. 588-592.
[12] IEEE Standard for Information Technology - Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPAN), IEEE Std 802.15.1-2005, 2005.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY