IoT-Based Smart Home Automation System Using Voice Recognition and Mobile Application

K. Nirosha[1], G. Krishnaveni[2], K. Indrasena Reddy[3], M. Narendhar[4], K. Shivadhitya[5]
Assistant Professor[1],Students [2,3,4,5]
nirosha.kundoru@gmail.com, krishnavenigolle53@gmail.com, kesireddyindrasenareddy@gmail.com, narendarmaddula140@gmail.com, shivadithya12@gmail.com
Vidya Jyothi Institute of Technology, Aziznagar, Moinabad, Ranga Reddy, Telangana, India

Abstract

The proliferation of the Internet of Things (IoT), embedded computing, and mobile technologies has reshaped residential automation, shifting it away from manually operated switches toward intelligent, remotely reachable ecosystems. This paper presents an extended design, implementation, and evaluation study of an IoT-based smart-home automation framework that unifies voice-driven interaction, a purpose-built Android application, and a dual-mode Bluetooth/Wi-Fi communication layer around a low-cost ESP32/Arduino Uno controller. Rather than depending entirely on continuous cloud connectivity, the proposed architecture keeps appliance control functional over a local Bluetooth link even when Internet access is unavailable, reserving Wi-Fi for extended-range and remote operation. Spoken instructions are converted to text through the Google Speech Recognition engine and matched against a locally stored command set before a relay-driven switching stage energizes the target appliance; a Passive Infrared (PIR) sensor concurrently performs unobtrusive intrusion monitoring and raises alerts through the same mobile interface. Beyond the original hardware prototype, this expanded study contributes a structured requirements analysis, a layered system architecture, algorithmic descriptions of the command-processing and security-monitoring pipelines, an indicative power and cost budget, a threat model with corresponding mitigations, and a systematic qualitative evaluation methodology. The resulting design is positioned as an accessible automation option for elderly occupants, individuals with mobility impairments, and cost-conscious residential deployments, and a concrete roadmap toward machine-learning-based personalization, cloud analytics, and biometric access control is outlined for future development.

Index Terms — *Internet of Things (IoT), smart home automation, voice recognition, natural language command processing, Google Speech API, Android application, ESP32, Arduino Uno, Bluetooth, Wi-Fi, relay switching, PIR sensor, home security, assistive technology, edge computing.*

I. Introduction

The Internet of Things (IoT) has, over the past decade, evolved from a research concept into a foundational technology underpinning a wide range of consumer and industrial systems. By allowing everyday physical objects to sense, communicate, and act with minimal human intervention, IoT has enabled a new generation of applications spanning healthcare monitoring, industrial process control, precision agriculture, and residential automation. Among these, smart home automation is arguably the application area most directly experienced by ordinary users, since it touches the routine activities of switching lights, operating fans, securing doors, and monitoring the safety of a living space.

Conventional home electrical systems rely on manual switches mounted on walls, which require a user to be physically present at the point of control. This arrangement, while simple and inexpensive, becomes a genuine barrier for elderly residents, individuals with limited mobility, or users who wish to manage their homes while away. It also does little to address safety concerns such as unauthorized entry, since a manual switchboard has no inherent capacity to sense or report intrusion. These limitations motivate automation architectures that combine remote accessibility with local intelligence.

Voice interaction has emerged as one of the most natural modalities for operating such automation systems, since spoken commands mirror the way people already communicate

requests to another person. Rather than locating and pressing a switch, a resident can simply say a short instruction such as "turn on the light" and expect the corresponding appliance to respond. Automatic Speech Recognition (ASR) engines, whether running in the cloud or at the network edge, convert this spoken instruction into a machine-interpretable text string, which is then mapped onto a control action within the automation system.

Commercial voice assistants such as Amazon Alexa and Google Home have popularized this interaction style, but their architectures are built around continuous cloud connectivity: audio, or at minimum recognized text, is generally routed through vendor infrastructure before a control decision is made. This dependency introduces three recurring concerns that have been widely discussed in the smart-home literature. First, any interruption of Internet service — whether due to an outage, a router fault, or a remote geography with unreliable connectivity — can leave the entire home automation function inoperative. Second, routing voice data through third-party servers raises privacy questions, since users may be uncomfortable with recordings or transcripts of household activity leaving the premises. Third, subscription-linked cloud assistant hardware is typically priced beyond what many budget-constrained households, particularly in developing regions, are willing or able to spend on convenience features. These concerns have driven sustained research interest in low-cost, locally responsive alternatives that retain the convenience of voice control while reducing the degree of cloud dependency [1]–[8].

This paper reports an extended treatment of an IoT-based smart-home automation system that was originally developed and prototyped by the authors, combining voice recognition, a dedicated Android mobile application, and a microcontroller-driven relay switching stage. Compared with the original prototype description, this paper substantially broadens the technical treatment: it introduces a structured requirements analysis, reorganizes the design around an explicit layered architecture, formalizes the command-processing and security-alert logic as algorithms, adds a power and cost budget derived from representative component specifications, introduces a threat model and associated mitigation strategies, and expands the discussion of results, limitations, and future work. The intention is to present the system not merely as a working prototype, but as a documented engineering case study that can be extended, reproduced, or compared against by other researchers working in accessible and cost-sensitive home automation.

The system architecture is built around an ESP32 or Arduino Uno microcontroller acting as the central control node. Voice commands captured through a smartphone are converted to text using the Google Speech Recognition Application Programming Interface (API) and transmitted to the controller over either a Bluetooth serial link or a Wi-Fi connection, while the same controller also accepts direct button-press commands issued from the companion Android application. On receiving a valid instruction, the controller drives a relay module that switches the corresponding household appliance, physically isolating the low-voltage digital logic from the high-voltage AC load. A Passive Infrared (PIR) motion sensor is attached to the same controller to provide a lightweight intrusion-detection capability, triggering a local alarm and a notification to the mobile application whenever unexpected motion is sensed.

The key contributions of this work are summarized as follows:

- A dual-mode control architecture that supports both offline Bluetooth operation and online Wi-Fi operation, so that core appliance-switching functionality does not collapse into total unavailability during an Internet outage.
- Integration of Google Speech Recognition with a purpose-built Android application, unifying spoken and manual (button-press) control within a single, consistent user interface.
- A PIR-based intrusion-detection module that raises real-time security alerts through the same mobile interface used for appliance control, rather than requiring a separate security subsystem.
- A low-cost, ESP32/Arduino-based hardware design intended for elderly users, individuals with physical disabilities, and cost-conscious

residential deployments, together with an indicative bill-of-materials cost comparison against Raspberry-Pi-class alternatives.

- An expanded engineering treatment comprising a layered system architecture, algorithmic descriptions of command processing and security monitoring, a threat model with mitigations, and a structured qualitative evaluation methodology suitable for future quantitative extension.

The remainder of this paper is organized as follows. Section II reviews related work on voice-controlled and IoT-based home automation. Section III specifies the functional and non-functional requirements that guided the design. Section IV presents the proposed layered system architecture and hardware design. Section V describes the system methodology through algorithmic and process-level descriptions. Section VI details the implementation of the hardware, firmware, and mobile application. Section VII presents the experimental setup and discusses the results obtained. Section VIII discusses security and privacy considerations. Section IX outlines directions for future enhancement, and Section X concludes the paper.

II. Related Work

Home automation sits at the intersection of embedded systems, wireless communication, and human-computer interaction, and it has consequently attracted contributions from each of these communities. This section reviews representative studies that inform the design choices made in the proposed system, organized from edge-oriented voice recognition approaches through cloud-dependent assistant integrations to protocol- and security-level analyses. A comparative summary is provided in Table I, and the section closes with an explicit statement of the research gap that the present work addresses.

A. Edge-Based and Privacy-Preserving Voice Recognition

Froiz-Míguez, Fraga-Lamas, and Fernández-Caramés [1] investigated a voice-controlled home automation architecture designed specifically to avoid routing speech data through third-party cloud servers, targeting speakers of a low-resource regional language. Their system pairs a dedicated voice-assistant device with a Raspberry Pi acting as an edge inference platform, communicating with actuators over a Bluetooth Mesh network. Speech recognition was performed locally using compact transformer-based acoustic models fine-tuned on a modest corpus of regional speech, and the authors reported that the tuned models achieved competitive recognition rates with sub-second response latency on ordinary smartphones. The study demonstrates that meaningful voice-control accuracy is achievable without a permanent cloud link, but the approach requires language-specific model training effort and a Raspberry-Pi-class edge device, both of which raise the barrier to replication relative to a single low-cost microcontroller.

B. Cloud-Assisted Security and Automation

Netinant, Utsanok, Rukhiran, and Klongdee [2] combined voice recognition, a Blynk-based mobile application, and PIR-driven motion sensing on a Raspberry Pi platform to deliver joint security and automation functionality. Command recognition was delegated to Google Assistant, and the authors reported markedly different recognition accuracy across languages, alongside near-perfect motion-detection performance from the PIR sensors at the cost of a short detection delay. While the study confirms that combining automation with intrusion sensing is practically feasible, its dependence on both a Raspberry Pi and a continuously available cloud assistant increases hardware cost and eliminates the possibility of offline operation, a gap the present design is intended to close.

C. Low-Cost Microcontroller-Based Voice Control

Ibrahim, Ohize, Umar, and Aliyu [3] described a compact voice-controlled automation system built on an Arduino Uno paired with an HC-06 Bluetooth module and relay outputs, in which an Android smartphone performs speech-to-text conversion locally before forwarding the recognized text to the microcontroller over a short-range Bluetooth link. This design achieves a substantially lower hardware cost than Raspberry-Pi-based alternatives and avoids continuous Internet dependency altogether, but it is constrained by the limited range of Bluetooth

Class 2 radios and supports only a small fixed set of connected devices. The present work adopts a similar microcontroller-and-relay foundation but extends it with an additional Wi-Fi channel to overcome the range limitation while preserving the offline Bluetooth fallback.

D. Power-Aware and Authenticated Voice Control

Alakesan, Darmaraj, Sarath Krishnan, and Vinoth Kumar [4] proposed a Google-Assistant-driven automation system that places particular emphasis on tracking appliance usage to support energy conservation, together with a speaker verification stage intended to restrict command execution to authorized users. The addition of speaker authentication is a meaningful step toward resisting unauthorized triggering, a concern that is common to open voice-command systems, including the one proposed in this paper. However, because command interpretation is delegated entirely to Google Assistant's cloud infrastructure, the system has no offline fallback and its responsiveness is tied to Internet availability.

E. Offline Recognition with Integrated Energy Metering

Irugalbandara, Naseem, Perera, Kiruthikan, and Logeeshan [5] introduced a smart-hub platform, referred to by the authors as HomeIO, that performs automatic speech recognition entirely offline on a Raspberry Pi 4 and communicates with smart plug sockets over a self-organizing wireless mesh network capable of reporting live power consumption with accuracy in the 90–95% range over an operating distance of roughly eight to ten meters. Because recognition runs locally, the reported word-error rate and memory footprint compared favorably with several cloud-based alternatives. The approach illustrates the benefit of local processing for both privacy and responsiveness, but running multiple acoustic models concurrently still requires a comparatively resource-rich single-board computer with several gigabytes of memory, which is considerably heavier than the microcontroller-only footprint targeted in this paper.

F. Environment-Aware Evaluation of Speech-Controlled Automation

Noruwana, Owolawi, and Mapayi [6] examined an interactive IoT automation system controlled through Google Assistant across a deliberately varied set of physical environments, including quiet and noisy rooms as well as empty and furnished spaces. Their results show that ambient noise and room acoustics measurably influence command-recognition performance, a factor that laboratory-only evaluations frequently overlook. This finding is directly relevant to the qualitative testing methodology adopted later in this paper, which was likewise conducted under ordinary indoor conditions rather than an acoustically controlled environment. As with other Google-Assistant-based designs, however, the system's reliance on cloud speech processing ties its usability to network availability.

G. Field Deployment Experience

Adhikary et al. [7] reported the real-world deployment of a NodeMCU-ESP8266-based smart home system aimed at improving everyday convenience and energy efficiency rather than serving as a laboratory demonstration alone. Their account of practical deployment challenges — including network reliability, architectural constraints, and communication-protocol selection once a prototype leaves the laboratory — is a valuable complement to purely bench-tested designs, and the authors identify biometric authentication as a priority direction for strengthening system security, a recommendation echoed in Section VIII of this paper. The Wi-Fi-only, database-backed architecture they describe is economical but, unlike the dual-mode design proposed here, requires a functioning network connection for every control action.

H. Cloud-Based Conversational Scheduling

Kaiborta and Samal [8] developed a Node-RED and NodeMCU-based automation application that accepts both voice and text commands through a Dialogflow conversational backend, adding the ability to schedule appliances to run for a specified duration even while the user is away from home, along with usage-history logging that allows past device states to be reviewed rather than only observed in real time. This scheduling and logging capability

represents a useful extension beyond simple on/off switching, and is noted in Section IX as a promising addition to the present system. As with other cloud-anchored designs, however, the dependence of Dialogflow on Internet connectivity removes the possibility of offline operation.

I. Lightweight Messaging Protocols for Home Automation

Kodali and Soratkal [16] built a home automation system around the ESP8266 Wi-Fi module and the MQTT publish–subscribe protocol, in which a smartphone application exchanges lightweight messages with a broker to switch appliances with minimal bandwidth and low power draw. Their results indicate that a message-queuing architecture reduces network overhead relative to continuous polling and scales comfortably as additional sensors and actuators are attached to the same broker. The system offers no voice interface, however, and assumes an uninterrupted Wi-Fi connection, so it lacks the local fallback mode that Bluetooth provides in the dual-mode design proposed here.

J. Security-Oriented Sensor Network Design

Pirbhulal et al. [17] proposed a wireless-sensor-network-based smart-home architecture that places particular emphasis on secure data transmission between sensor nodes and a central gateway, employing lightweight cryptographic key management to resist eavesdropping and node-impersonation attacks without imposing significant computational overhead on resource-constrained nodes. Their results are a useful reference point for the threat-mitigation discussion in Section VIII of this paper, although their architecture is oriented toward sensor monitoring rather than natural-language interaction and does not address voice-based control or mobile application integration.

K. Comparative Surveys of IoT Smart-Home Architectures

Stojkoska and Trivodaliev [18] surveyed the broader IoT smart-home landscape, cataloguing communication protocols, sensing technologies, and cloud/edge architectures reported across the literature, and identifying interoperability, scalability, and energy efficiency as persistent open challenges. Their review highlights standardization and energy-efficiency concerns that many individual point-solution studies — including several of the voice-controlled designs reviewed above — do not address explicitly, and it helps situate the microcontroller-based, dual-mode design proposed in this paper within the broader set of architectural trade-offs identified in the field.

L. Low-Cost Relay-Based Residential Automation

Chekired et al. [19] presented a low-cost house-automation platform built on an Arduino microcontroller that switches lighting and appliance circuits through relay boards and can be extended with additional sensor modules for residential or light-industrial use. The authors report that the platform keeps material costs low while remaining flexible enough to be reconfigured for different appliance sets without redesigning the control logic, reinforcing the choice of an Arduino/ESP32-and-relay foundation adopted in this paper. Their system, however, is triggered from a local switch panel over wired or short-range wireless links and does not incorporate voice recognition or a companion mobile application.

M. Comparative Analysis of Wireless Protocols

Danbatta and Varol [20] compared ZigBee, Z-Wave, Wi-Fi, and Bluetooth in terms of range, power consumption, data rate, and interference susceptibility for typical home-automation deployments, concluding that no single protocol is optimal across every scenario and that hybrid designs can balance range against energy efficiency. This conclusion directly supports the dual-mode Bluetooth-and-Wi-Fi communication strategy adopted in the present system, in which Bluetooth supplies low-power, short-range, offline-capable control while Wi-Fi extends coverage for remote access. Their study, however, is confined to a protocol-level comparison and does not evaluate a complete voice-controlled implementation.

N. Research Gap and Positioning of the Present Work

Table I consolidates the systems reviewed above alongside the design proposed in this paper. Two broad clusters emerge. The first cluster, exemplified by [2], [4], [6], and [8], achieves high recognition accuracy and rich feature sets by delegating speech interpretation to commercial cloud assistants, at the cost of complete dependence on continuous Internet connectivity and, in most cases, a recurring reliance on third-party infrastructure. The second cluster, exemplified by [1], [3], and [19], favors edge or microcontroller-based processing to reduce cost and remove cloud dependency, but typically narrows the feature set — for instance, omitting security sensing, restricting range, or requiring language-specific model training. The protocol- and security-focused studies [16]–[20] contribute complementary insights on communication efficiency and threat resistance without themselves offering an integrated voice-controlled prototype.

The system proposed in this paper is positioned to combine the low hardware cost and offline operability characteristic of Arduino/ESP32-based systems [3], [19] with the security-oriented PIR integration of [2], the dual-mode flexibility highlighted by [1] and supported by the protocol trade-off analysis of [20], and the accessibility motivation emphasized in [4] and [7]. It further incorporates a dedicated Android application that unifies voice and manual control within a single interface, an element that is present in isolated form in several reviewed systems ([2], [4], [6], [8]) but rarely paired with an offline Bluetooth fallback in the same design. This combination is what distinguishes the present contribution from the individual studies summarized in Table I.

III. System Requirements and Design Considerations

Before committing to a particular hardware and software architecture, it is useful to state explicitly the functional and non-functional requirements that the system is intended to satisfy. Doing so clarifies the design trade-offs discussed in Section IV and provides a basis against which the qualitative evaluation in Section VII can later be interpreted.

A. Functional Requirements

- The system shall allow a registered user to switch designated household appliances ON or OFF using spoken voice commands issued through the companion Android application.
- The system shall allow the same set of appliances to be switched using direct button-press controls within the Android application, without requiring a spoken command.
- The system shall convert recognized voice commands to text and compare them against a predefined command vocabulary before executing any switching action, discarding unrecognized input.
- The system shall continue to accept and execute manual and voice commands over a local Bluetooth connection when no Internet connectivity is available.
- The system shall additionally support command delivery over Wi-Fi when the user is outside Bluetooth range but within reach of the home wireless network or, optionally, the Internet.
- The system shall continuously monitor a PIR motion sensor and, upon detecting motion outside expected conditions, shall trigger a local alert and notify the user through the mobile application.
- The system shall report the present ON/OFF status of every connected appliance back to the mobile application so that the displayed state remains consistent with the physical state of the appliance.

B. Non-Functional Requirements

- Affordability: the complete hardware bill of materials should remain substantially lower than single-board-computer-based alternatives, so that the system remains accessible to budget-constrained households.
- Responsiveness: the interval between command issuance and appliance switching should be short enough that the interaction feels immediate to the user, consistent with the local, non-cloud-routed processing path.
- Resilience: loss of Internet connectivity should degrade the system gracefully to Bluetooth-only operation rather than causing a complete loss of control.

- Usability: the mobile application interface should be operable by elderly users and users with limited dexterity, implying large, clearly labelled controls and a minimal number of steps per action.
- Extensibility: the relay-based switching stage and command vocabulary should be straightforward to extend to additional appliances or rooms without redesigning the control logic.
- Basic security: the system should resist trivial unauthorized triggering and should not expose raw appliance-control commands to unauthenticated parties on the local network.

C. Design Constraints and Assumptions

The design assumes a typical residential electrical environment in which appliances are switched at the mains level rather than through appliance-specific smart plugs, which keeps the hardware cost low but requires careful electrical isolation between the low-voltage control logic and the high-voltage AC load. It further assumes that the user's smartphone is within Bluetooth range for routine local control and that a Wi-Fi access point is available when extended-range or remote access is desired; the system is not designed to operate over the public Internet without an accompanying network-level security mechanism, a limitation discussed further in Sections VII-D and VIII. Finally, the voice-recognition pipeline assumes reasonably clear speech in a supported language and a modest, fixed vocabulary of commands rather than open-ended natural-language understanding, consistent with the design choices reviewed in Section II.

IV. Proposed System Architecture

A. System Overview

The proposed system integrates IoT sensing and actuation, voice recognition, embedded control, dual-mode wireless communication, and a companion Android application into a single smart-home platform. It can be described conveniently as four cooperating layers, illustrated conceptually together with the physical block diagram in Fig. 1: a perception layer (the PIR sensor and the microphone on the user's smartphone), an interaction and application layer (the Android application, including its voice-recognition front end), a communication layer (the Bluetooth and Wi-Fi links connecting the phone to the controller), and a control and actuation layer (the ESP32/Arduino controller and the relay-driven appliance outputs).

A command may originate from either of two sources: a spoken instruction processed through the Google Speech Recognition API, or a direct button press within the Android application. In both cases, the resulting command reaches the application layer as a short text token, which is then transmitted over whichever communication channel is currently active — Bluetooth for local, offline-capable control, or Wi-Fi when longer range or remote access is required. At the control layer, the ESP32/Arduino compares the received token against a table of predefined instructions stored in its firmware and, on finding a match, drives the corresponding relay channel, switching the target appliance while keeping the low-voltage control circuitry electrically isolated from the high-voltage load. Independently of this command path, the same controller polls the PIR sensor on a fixed interval; whenever motion is detected outside the conditions the user has configured as expected, the controller raises a local indicator and pushes a notification to the mobile application, so that appliance control and intrusion monitoring share a single hardware and software platform rather than requiring two separate systems.

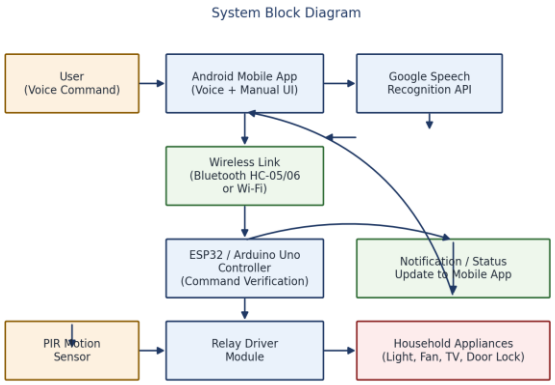

Fig. 1. Layered system block diagram.

B. Hardware Design

1) ESP32/Arduino Uno Microcontroller

The ESP32 (or, in the lower-cost variant, an Arduino Uno) forms the central processing node of the system. It receives commands relayed from the Android application, arbitrates between the

Bluetooth and Wi-Fi communication paths, drives the relay outputs, and polls the PIR sensor. The ESP32 variant integrates a dual-core processor, on-chip Wi-Fi and Bluetooth radios, and a generous set of general-purpose input/output (GPIO) pins within a low power envelope, which makes it particularly well suited to this application; where an Arduino Uno is used instead for further cost reduction, an external HC-05/06 Bluetooth module and, optionally, an ESP8266 Wi-Fi module provide equivalent connectivity at a small increase in board complexity.

2) Relay Module

Because household appliances operate on high-voltage alternating current while the controller operates at low-voltage direct current, a relay module is used to provide safe electrical isolation between the two domains. Each relay channel is dedicated to a single appliance: a logic HIGH output from the controller energizes the relay coil and closes the corresponding high-voltage circuit, while a logic LOW output opens it. A multi-channel relay board allows several appliances — for example, a light, a fan, a television, and an electronic door lock — to be switched independently from the same controller.

3) Bluetooth Module (HC-05/HC-06)

The Bluetooth module provides a short-range, low-power, low-latency wireless serial link between the Android application and the controller. Because this link does not depend on a home router or an Internet connection, it forms the offline fallback path referenced throughout this paper: as long as the user's phone remains within Bluetooth range of the controller, appliance control and status reporting continue to function even during a network outage.

4) Wi-Fi Communication Module

Wi-Fi connectivity, provided natively by the ESP32's integrated radio, extends the effective control range beyond that of Bluetooth by routing commands through the home's wireless access point, and optionally the Internet, for remote operation. This channel also supports future extensions such as cloud-based logging and notification delivery, discussed in Section IX.

5) PIR Motion Sensor

The Passive Infrared sensor detects the infrared radiation naturally emitted by a moving human body and outputs a digital HIGH signal when motion is sensed within its field of view. The controller polls this signal and uses it to trigger security actions such as sounding a local buzzer, switching on a light, or sending a notification to the mobile application, while otherwise remaining in a low-power idle state to minimize unnecessary energy consumption.

6) Power Supply Unit

A step-down transformer, bridge rectifier, filter capacitor, and linear or switching voltage regulator convert the household AC mains voltage into the stable, regulated low-voltage DC supply required by the controller, the relay module, and the communication modules. Regulation is particularly important for the relay coils and radio modules, whose switching transients can otherwise introduce voltage ripple that destabilizes the microcontroller.

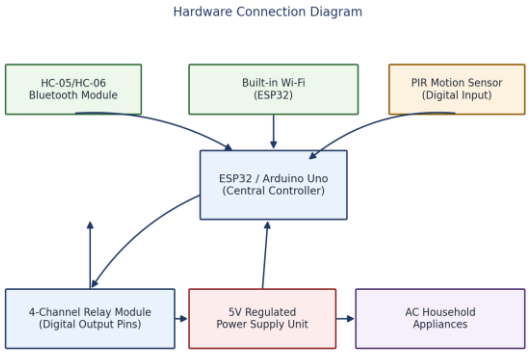

Fig. 2. Hardware connection diagram.

C. Indicative Component Specifications and Power Budget

Table II summarizes representative electrical characteristics for the main hardware components, drawn from typical manufacturer datasheet ranges rather than bench measurement, together with an indicative current-draw budget. This budget is intended only to illustrate the order of magnitude of the system's power requirements and to support the power-supply sizing discussion in Section IV-B-6; it is not presented as a measured result.

TABLE II. INDICATIVE COMPONENT SPECIFICATIONS AND POWER BUDGET

Component	Typical Supply	Approx. Active Current	Role in System
ESP32 dev. board	5 V (USB) / 3.3 V logic	~80–160 mA (Wi-Fi active)	Central controller
Arduino Uno (alt.)	5–9 V (barrel/USB)	~45–50 mA	Central controller (lower-cost variant)
HC-05/06 Bluetooth	3.3–5 V	~30–40 mA	Local wireless serial link
4-channel relay board	5 V (coil)	~70–80 mA per active channel	Appliance switching
PIR motion sensor	5 V	~1–5 mA (idle/standby heavy)	Intrusion detection
Regulated supply headroom	5 V DC output	≥ 500 mA recommended	Combined system margin

Even allowing for several relay channels switching simultaneously, the aggregate current draw remains comfortably within the output of a modest 5 V, 1 A regulated adapter, reinforcing the low-cost, low-power positioning of the design relative to Raspberry-Pi-based alternatives, whose single-board-computer cores typically draw several times this current under active load.

V. System Methodology

This section formalizes the operational logic of the system as two cooperating procedures executed by the controller: a command-processing algorithm that governs appliance switching, and a security-monitoring algorithm that governs PIR-based intrusion alerts. The two procedures run concurrently within the firmware's main control loop, sharing access to the relay outputs and the communication interfaces but otherwise operating independently.

A. Command-Processing Algorithm

Algorithm 1 describes the sequence executed whenever a command token — originating from either the voice-recognition pipeline or a manual button press in the Android application — arrives at the controller over Bluetooth or Wi-Fi.

1. Receive the incoming command token over whichever interface (Bluetooth or Wi-Fi) is currently active.
2. Validate the basic syntax of the token (non-empty, within expected length, drawn from the permitted character set).
3. Compare the validated token against the predefined command table stored in firmware memory.
4. If a match is found, identify the appliance and the requested state (ON or OFF) associated with that command.
5. Drive the corresponding relay channel to the requested state.
6. Update the internally held status flag for that appliance.
7. Transmit a status-confirmation message back to the Android application over the originating interface.
8. If no match is found in step 3, discard the token without altering any relay state, and return to step 1.
9. Resume listening for the next incoming command token.

Discarding unmatched tokens at step 8, rather than attempting a fuzzy or partial match, keeps the decision logic simple and deterministic, and avoids the risk of an unintended appliance being switched due to a misrecognized word. This conservative matching strategy trades a small amount of voice-command flexibility for a lower risk of erroneous switching, which was judged appropriate given the safety-relevant nature of controlling powered appliances and door locks.

B. Security-Monitoring Algorithm

10. Poll the digital output of the PIR sensor at a fixed sampling interval.
11. If the sensor output is LOW (no motion detected), return to step 1 after the sampling interval elapses.
12. If the sensor output transitions to HIGH (motion detected), record the event time and evaluate it against the user's configured monitoring schedule, if any.
13. If the event falls within an interval the user has designated as expected occupancy, log the event without raising an alert.
14. Otherwise, activate the configured local alert action (for example, a buzzer or an automatically switched light) and

transmit a security notification to the Android application.

15. Continue polling without interrupting the concurrently running command-processing algorithm.

Running the two algorithms concurrently on a single low-cost microcontroller, rather than dedicating separate hardware to automation and to security, is one of the distinguishing simplifications of the proposed design relative to systems such as [2], which achieve similar functionality but at the cost of a Raspberry-Pi-class processing platform.

C. End-to-End Operational Workflow

Fig. 3 illustrates the complete operational workflow from application launch through appliance switching and security monitoring, integrating the two algorithms described above into a single user-facing sequence.

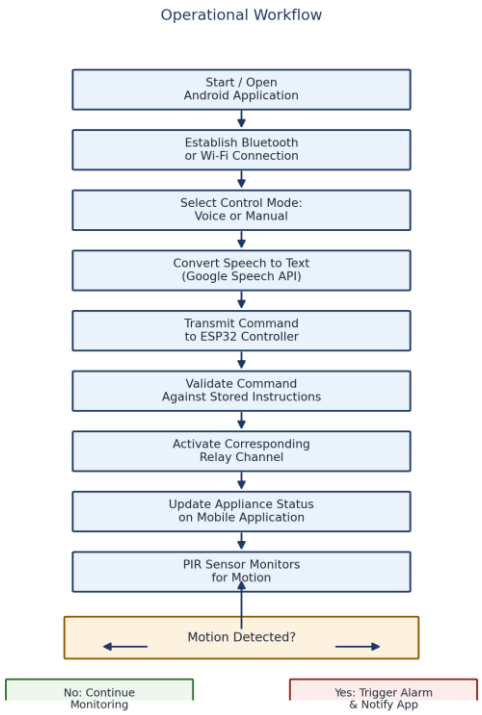

Fig. 3. End-to-end operational workflow of the proposed system.

16. Launch the Android mobile application.
17. Establish a Bluetooth or Wi-Fi connection with the ESP32/Arduino controller.
18. Select either voice control or manual (button-press) control.
19. If voice control is selected, capture speech and convert it to text using the Google Speech Recognition API.
20. Transmit the resulting command token to the controller.
21. Execute the command-processing algorithm (Algorithm, Section V-A) to validate and act on the command.
22. Update the corresponding appliance's displayed state within the mobile application.
23. Concurrently execute the security-monitoring algorithm (Section V-B) to poll the PIR sensor.
24. Raise a security alert and notification whenever unexpected motion is detected.
25. Repeat the cycle continuously until the application is closed or the controller is powered down.

VI. System Implementation

Implementation proceeded through five broadly parallel activities: hardware assembly, ESP32/Arduino firmware development, Android application development, voice-recognition integration, and end-to-end system testing. Each module was developed and bench-tested independently before integration, which simplified fault isolation when a component did not behave as expected.

A. Hardware Implementation

The relay module was wired to a dedicated set of digital output pins on the ESP32, with each channel assigned to a specific appliance (light, fan, television, and door-lock solenoid in the prototype configuration). The HC-05 Bluetooth module was connected to a hardware serial port for reliable communication, while Wi-Fi connectivity used the ESP32's on-chip radio configured in station mode against the household access point. The PIR sensor's digital output was wired to an interrupt-capable input pin so that motion events could be captured promptly without continuous busy-polling. All modules were powered from the common regulated 5 V supply described in Section IV-C, with decoupling capacitors placed close to each module to suppress switching noise from the relay coils.

B. ESP32/Arduino Firmware

The firmware was developed in the Arduino IDE using standard C/C++ libraries for serial communication, Wi-Fi client/server

functionality, and GPIO control. At start-up, the firmware initializes the Bluetooth serial interface, connects to the configured Wi-Fi network, sets the relay output pins to a known safe (OFF) state, and configures the PIR input pin. The main loop then repeatedly executes the command-processing and security-monitoring algorithms described in Section V. The command table mapping text tokens to relay channels is defined as a simple lookup structure, keeping the firmware compact enough to run comfortably within the memory budget of either an ESP32 or an Arduino Uno.

C. Mobile Application Development

The companion Android application, developed in Android Studio, serves as the primary user interface. Its home screen lists every connected appliance alongside an icon, a live status indicator, and a dedicated ON/OFF toggle button, while separate panels handle Bluetooth pairing, Wi-Fi configuration, voice-command capture, and security notifications. Interface elements were deliberately kept large, high-contrast, and minimally nested, in line with the usability requirement stated in Section III-B, so that elderly users and users with limited dexterity can complete common actions in as few taps as possible. The application communicates with the controller using a serial Bluetooth socket for local operation and a TCP socket over Wi-Fi for extended-range or remote operation, selecting automatically between the two based on connection availability.

D. Voice-Recognition Pipeline

Voice recognition is implemented using the Google Speech Recognition API invoked from within the Android application. When the user presses the microphone icon, the application records the spoken utterance, forwards the audio to the speech-recognition engine, and displays the resulting recognized text for brief visual confirmation before transmitting it to the controller as a command token. The prototype's supported vocabulary includes commands such as "Turn ON Light," "Turn OFF Light," "Turn ON Fan," "Turn OFF Fan," "Open Door," "Close Door," "Turn ON TV," and "Turn OFF TV," each of which maps to exactly one entry in the controller's command table described in Section VI-B.

E. Testing Environment

Integration testing was carried out in an ordinary indoor residential setting rather than an acoustically isolated laboratory, so as to reflect realistic background noise and Wi-Fi/Bluetooth interference conditions. Each hardware module — relay switching, Bluetooth communication, Wi-Fi communication, and PIR sensing — was first verified in isolation using simple diagnostic sketches before the complete firmware and application were integrated and re-tested as a whole, following standard embedded-systems integration practice of verifying subsystems individually prior to full-system testing.

VII. Results and Discussion

A. Experimental Setup

The prototype was assembled using an ESP32 development board, a 4-channel relay module, an HC-05 Bluetooth module, a PIR motion sensor, an Android smartphone, and representative household appliances (a lamp, a fan, and a door-lock solenoid). Firmware was developed in the Arduino IDE and uploaded over USB, and the Android application was installed on a smartphone supporting both Bluetooth and Wi-Fi. Table III summarizes the experimental configuration.

TABLE III. EXPERIMENTAL SETUP

Component	Specification
Controller	ESP32 development board
Programming Environment	Arduino IDE
Mobile Platform	Android smartphone
Communication	Bluetooth (HC-05/06) and Wi-Fi
Voice Recognition	Google Speech Recognition API
Security Sensor	PIR motion sensor
Switching Device	4-channel relay module
Power Supply	5 V DC regulated adapter

B. Mobile Application Interface

The developed Android application presents a home screen listing all connected appliances with an icon, current status, and a dedicated ON/OFF control button for each device, alongside separate modules for Bluetooth pairing, Wi-Fi configuration, voice-command input, and security notifications. Consistent with the usability requirement in Section III-B, controls were kept large and clearly labelled, and the number of steps needed to operate an appliance was minimized to remain accessible to elderly and physically challenged users.

C. Qualitative System Evaluation

The prototype was tested under normal indoor residential conditions with everyday background noise. The voice commands listed in Section VI-D were issued repeatedly by multiple users, and appliance responses were logged for both Bluetooth and Wi-Fi modes. A large-scale, statistically controlled accuracy study comparable to the multi-user, multi-language evaluations reported in [1] and [2] was outside the scope of this prototype stage; Table IV therefore reports qualitative observations rather than fabricated precision figures, and quantifying recognition accuracy, response latency, and range under controlled conditions is identified as near-term future work in Section IX.

TABLE IV. QUALITATIVE TEST OBSERVATIONS

Test Parameter	Bluetooth Mode	Wi-Fi Mode
Voice command execution	Consistent for predefined commands at close range	Consistent for predefined commands within Wi-Fi coverage
Manual (app button) control	Immediate response	Immediate response
Relay switching reliability	Reliable across repeated cycles	Reliable across repeated cycles
PIR motion detection	Reliable indoor detection with brief alarm/notification delay	Reliable indoor detection with brief alarm/notification delay
Operable without Internet	Yes	No (requires local/Internet network)
Effective range	Short range (typical Bluetooth Class 2 limits)	Extends to Wi-Fi router coverage / remote access

Overall, the results indicate that the dual-mode communication design allows the system to fall back on Bluetooth for fast, offline local control while using Wi-Fi for extended-range and remote operation, consistent with the trade-offs identified in the literature review summarized in Table I. The integration of voice recognition with manual application control also improved accessibility, since users could complete common tasks — such as turning a light on or off — through either input method without additional configuration.

D. Indicative Cost Comparison

Table V presents an approximate, indicative bill-of-materials comparison between the proposed ESP32/Arduino-and-relay approach and a representative Raspberry-Pi-based alternative of the kind described in [1], [2], and [5]. Figures are order-of-magnitude approximations based on typical retail component pricing rather than a formal procurement exercise, and are intended only to illustrate the relative cost positioning of the two approaches.

TABLE V. INDICATIVE RELATIVE COST COMPARISON

Cost Element	Proposed ESP32/Arduino Design	Representative Raspberry-Pi-Based Design
Compute/controller board	Low	Medium–High
Communication modules	Low (on-chip or small add-on module)	Low–Medium
Sensing (PIR)	Low	Low
Switching (relay board)	Low	Low
Power supply requirement	Modest (sub-1 A regulated)	Higher (single-board-computer core draw)
Overall relative cost	Low	Medium–High

E. Reliability and Comparative Discussion

The reliance on a fixed, locally stored command table rather than open-ended natural-language interpretation contributes to the consistent relay-switching behavior observed during testing, since every recognized command maps deterministically to exactly one appliance

action, avoiding the ambiguity that can arise in more flexible but less predictable natural-language systems. Relative to the cloud-dependent designs reviewed in Section II ([2], [4], [6], [8]), the proposed system trades some recognition flexibility and feature richness for continued operability during Internet outages, a property that is likely to matter more in regions with intermittent connectivity, which is common in many of the deployment contexts targeted by this work. Relative to the edge-based designs [1] and [5], the proposed system trades higher potential recognition accuracy for a substantially lower hardware cost and simpler deployment, since it does not require language-specific model fine-tuning or a Raspberry-Pi-class device.

F. Limitations

Three limitations of the present study should be acknowledged. First, the evaluation in Section VII-C is qualitative rather than quantitative; while it establishes that the system functions reliably for its intended command set, it does not report statistically powered accuracy, latency, or range figures of the kind available for some reviewed systems (for example, [2]). Second, the fixed command vocabulary, while reducing ambiguity as discussed above, limits the system's flexibility relative to open-vocabulary voice assistants. Third, the current prototype's security posture, discussed further in Section VIII, relies primarily on the short range of Bluetooth and standard Wi-Fi network security rather than an application-level authentication or encryption layer, which would be necessary before the system could be considered suitable for a fully Internet-exposed deployment.

VIII. Security and Privacy Considerations

Because the system is capable of switching powered appliances, operating an electronic door lock, and reporting on occupancy through the PIR sensor, its security properties merit explicit discussion beyond the functional description given in Sections IV–VI.

A. Threat Model

Three broad classes of threat are relevant to a system of this kind. First, an attacker within radio range could attempt to eavesdrop on or inject messages into the Bluetooth or Wi-Fi communication link in order to trigger an unauthorized action, such as unlocking a door. Second, an attacker with access to an unlocked or compromised smartphone running the companion application could issue commands without the legitimate occupant's knowledge. Third, an attacker with physical access to the controller hardware could attempt to reflash the firmware or directly manipulate the relay outputs, bypassing the software-level controls entirely.

B. Mitigations Adopted and Recommended

- Bluetooth pairing should be protected with a non-default PIN, and the HC-05/06 module's default pairing code should always be changed before deployment, since a well-known default code is a documented weak point in Bluetooth-based control systems.
- The Android application should require a device-level lock (such as a PIN, pattern, or biometric unlock already provided by the smartphone operating system) before granting access to appliance and door-lock controls, reducing the risk from an unattended unlocked phone.
- Wi-Fi communication should be confined to a securely configured (WPA2/WPA3) home network, and the system should not be exposed directly to the public Internet without an additional authentication layer, consistent with the design constraint noted in Section III-C.
- The command-matching logic's deliberate rejection of unmatched or malformed tokens, described in Section V-A, also limits the practical impact of injected noise or partially corrupted transmissions, since only an exact match against the predefined command table results in an action.
- Physical access to the controller enclosure should be restricted, and, in security-sensitive deployments (for example, where the system operates a door lock), firmware write-protection or a tamper-evident enclosure is recommended, echoing the biometric-authentication and hardware-hardening recommendations of [7] and the cryptographic key-management practices demonstrated in a sensor-network context by [17].

C. Privacy Considerations

Because command matching and relay switching occur locally on the controller rather than in a cloud service, the system does not require continuous transmission of household activity data to a third party for its core switching function, which is a meaningful privacy advantage relative to fully cloud-dependent assistants such as those used in [2], [4], [6], and [8]. Voice audio is nonetheless sent to the Google Speech Recognition service for transcription, so users who wish to avoid any cloud transmission of voice data should rely on the manual button-press control path described in Section VI-C, or should consider the offline, edge-based recognition approaches discussed in [1] and [5] as an alternative in a future revision of the system.

IX. Future Enhancements

Although the developed system successfully integrates voice recognition, Android application control, dual-mode wireless communication, and PIR-based security monitoring, several directions remain open for future development, organized below by theme.

A. Rigorous Quantitative Evaluation

The qualitative evaluation reported in Section VII-C should be extended into a statistically powered study across multiple users, accents, ambient-noise levels, and physical ranges, following the multi-condition methodology demonstrated by [2] and [6], so that recognition accuracy, response latency, and effective communication range can be quantified with confidence rather than described only qualitatively.

B. Machine-Learning-Based Personalization

Artificial Intelligence and Machine Learning techniques could be integrated to enable predictive automation that learns a household's typical daily routines — for example, automatically adjusting lighting or fan operation at anticipated times without requiring an explicit command each time, building on the usage-logging concept demonstrated by [8].

C. Cloud Integration for Remote Analytics

Cloud platforms such as Firebase, AWS IoT, or Google Cloud IoT could be layered on top of the existing Wi-Fi communication path to support secure remote monitoring, historical usage analytics, and predictive maintenance alerts from any location, while retaining the local Bluetooth path as an offline fallback so that the core resilience advantage identified in Section VII-E is not lost.

D. Biometric and Multi-Factor Authentication

Biometric authentication — fingerprint, facial, or voice-biometric verification — could be added to strengthen the mitigations described in Section VIII-B, particularly for door-lock control, extending the speaker-verification concept demonstrated by [4] and responding directly to the security-hardening priority identified in the field-deployment study of [7].

E. Expanded Environmental Sensing

Additional sensors for temperature, humidity, smoke, gas leakage, water level, and per-appliance energy consumption would broaden the system's environmental-monitoring capability and support energy-management features comparable to those demonstrated by the power-metering designs in [4] and [5], while solar power with battery backup could improve resilience during grid outages.

F. Multilingual and Accessible Voice Interaction

Extending the command vocabulary to additional languages, following the low-resource-language approach demonstrated by [1], would improve accessibility for regional-language speakers, complementing the elderly- and disability-focused accessibility goals already stated in Section I.

G. Interoperability with Commercial Ecosystems

Adding a compatibility layer for commercial assistants such as Google Assistant, Amazon Alexa, and Apple Siri would extend the system's usefulness within households that already own such devices, while the security and offline-operation properties discussed in Sections VII-E and VIII could be retained as an independent fallback path rather than a full replacement for the cloud-based integration.

X. Conclusion

This paper presented an extended treatment of the design, architecture, and implementation of an IoT-based smart-home automation system that combines voice recognition, Android mobile-application control, and dual-mode Bluetooth/Wi-Fi wireless communication around a low-cost ESP32/Arduino microcontroller. Beyond the original working prototype, this study contributed a structured requirements analysis, a layered system architecture, formalized command-processing and security-monitoring algorithms, an indicative power and cost budget, a threat model with concrete mitigations, and an expanded discussion of results and limitations.

The developed prototype allows a resident to operate household appliances through either spoken commands or manual application controls: the Google Speech Recognition API converts spoken instructions into text, and the ESP32/Arduino controller matches this text against a predefined command table before switching the target appliance through a relay module. The combination of Bluetooth and Wi-Fi communication provides flexible local and remote control, remaining operable during Internet outages, while the addition of a PIR motion sensor extends the same platform to intrusion detection and security notification without requiring separate security hardware.

Qualitative testing indicates reliable appliance switching, consistent voice-command execution for the predefined command set, and dependable motion detection, at an indicative hardware cost substantially lower than Raspberry-Pi-based alternatives such as [1] and [2]. The proposed architecture is well suited to residential homes, small offices, educational demonstration settings, and assisted-living environments, particularly for elderly and physically challenged users, and it provides a documented, extensible base for the machine-learning, cloud-connectivity, biometric-security, and multilingual enhancements outlined in Section IX.

Acknowledgment

The authors gratefully acknowledge the Department of Electronics and Communication Engineering, Vidya Jyothi Institute of Technology, Aziznagar, Moinabad, Ranga Reddy, for providing the laboratory facilities and technical support that made this work possible.

References

[1] A. Froiz-Míguez, P. Fraga-Lamas, and T. M. Fernández-Caramés, "Design, implementation, and practical evaluation of a voice recognition based IoT home automation system for low-resource languages and resource-constrained edge IoT devices: A system for Galician and mobile opportunistic scenarios," IEEE Access, vol. 11, pp. 63623–63649, 2023.

[2] P. Netinant, T. Utsanok, M. Rukhiran, and S. Klongdee, "Development and assessment of Internet of Things-driven smart home security and automation with voice commands," IoT, vol. 5, no. 1, pp. 79–99, 2024.

[3] U. I. Ibrahim, H. Ohize, U. A. Umar, and Y. Aliyu, "Design and construction of voice-controlled home automation using Arduino," ABUAD J. Eng. Res. Dev., vol. 7, no. 1, pp. 144–152, 2024.

[4] C. Alakesan, D. Darmaraj, R. Sarath Krishnan, and S. Vinoth Kumar, "IoT based home automation with power management system and enhanced security using voice recognition," Int. J. Eng. Res. Technol. (IJERT), vol. 12, no. 3, Mar. 2023.

[5] C. Irugalbandara, A. S. Naseem, S. Perera, S. Kiruthikan, and V. Logeeshan, "A secure and smart home automation system with speech recognition and power measurement capabilities," Sensors, vol. 23, no. 13, p. 5784, Jun. 2023.

[6] N. C. Noruwana, P. A. Owolawi, and T. Mapayi, "Interactive IoT-based speech-controlled home automation system," in Proc. 2020 2nd Int. Multidisciplinary Inf. Technol. Eng. Conf. (IMITEC), Kimberley, South Africa, 2020, pp. 1–8.

[7] A. Adhikary, S. Halder, R. Bose, S. Panja, S. Halder, J. Pratihar, and A. Dey, "Design and implementation of an IoT-based smart home automation system in real world scenario," EAI Endorsed Trans. Internet Things, vol. 10, no. 1, pp. 1–10, Jun. 2024.

[8] A. K. Kaiborta and S. Samal, "IoT based voice assistant for home automation," in Proc. 2022 4th Int. Conf. Smart Syst. Inventive Technol. (ICSSIT), 2022, pp. 165–172.

[9] IEEE Standards Association, "IEEE standards for smart home and Internet of Things," IEEE, Piscataway, NJ, USA.

[10] Arduino, "Arduino IDE documentation," [Online]. Available: https://www.arduino.cc/en/software

[11] Google Developers, "Android Studio documentation," [Online]. Available: https://developer.android.com/studio/intro

[12] Google Developers, "Cloud Speech-to-Text (Speech Recognition) API documentation," [Online]. Available: https://cloud.google.com/speech-to-text/docs

[13] Google, "Firebase documentation," [Online]. Available: https://firebase.google.com/docs

[14] Bluetooth Special Interest Group, Bluetooth Core Specification, Kirkland, WA, USA.

[15] Espressif Systems, "ESP32 technical reference manual," [Online]. Available: https://www.espressif.com/en/support/documents/technical-documents

[16] R. K. Kodali and S. Soratkal, "MQTT based home automation system using ESP8266," in Proc. 2016 IEEE Region 10 Humanitarian Technol. Conf. (R10-HTC), 2016, pp. 1–5.

[17] S. Pirbhulal, H. Zhang, M. E. E. Alahi, H. Ghayvat, S. C. Mukhopadhyay, Y.-T. Zhang, and W. Wu, "A novel secure IoT-based smart home automation system using a wireless sensor network," Sensors, vol. 17, no. 1, art. 69, 2017.

[18] B. R. Stojkoska and K. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," J. Cleaner Prod., vol. 140, pp. 1454–1464, 2017.

[19] F. Chekired, L. Canale, S. Tadjer, A. Louni, C. A. Bouroussis, and A. Tilmatine, "Low cost house automation system based on Arduino microcontroller," in Proc. 2021 IEEE Ind. Appl. Soc. Annu. Meeting (IAS), Vancouver, BC, Canada, 2021, pp. 1–6.

[20] S. J. Danbatta and A. Varol, "Comparison of Zigbee, Z-Wave, Wi-Fi, and Bluetooth wireless technologies used in home automation," in Proc. 2019 7th Int. Symp.