Hindawi
Scientific Programming
Volume 2022, Article ID 6886086, 15 pages
https://doi.org/10.1155/2022/6886086

Research Article

Skeleton of Implementing Voice Control for Building Automation Systems

Waleed Dweik,[1] **Musa Abdalla**,[2] **Yazan AlHroob**,[1] **Anas AlMajali**,[3] **Suad Alhaj Mustafa**,[2] **and Mohammad Abdel-Majeed**[1]

[1]*Computer Engineering Department, University of Jordan, Amman, Jordan*
[2]*Mechanical Engineering Department, University of Jordan, Amman, Jordan*
[3]*Department of Computer Engineering, Faculty of Engineering, The Hashemite University, Zarqa, Jordan*

Correspondence should be addressed to Waleed Dweik; w.dweik@ju.edu.jo

Received 29 November 2021; Revised 30 June 2022; Accepted 15 July 2022; Published 5 September 2022

Academic Editor: Shah Nazir

Copyright © 2022 Waleed Dweik et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Building automation is one of the most popular trends in the 21st century, as it provides buildings with modern and luxurious touches. Furthermore, voice-controlled automation systems are very convenient for the current fast-paced life style as people need to quickly, efficiently, and safely control everything around them. The goal of this article is to provide a skeleton for implementing voice control to a building automation system. The article also presents and evaluates a use case of the proposed skeleton integrated with a KNX-based office automation system. Through the integrated system, lights and blinds are controlled and sensors are polled using voice commands from an authorized personal. This article serves as an educational reference for engineering students interested in applying authorized voice control capability to automation systems.

1. Introduction

With the emerging Internet of things (IoT), smart building automation is gaining massive popularity [1–4]. A building automation system (BAS) is normally installed to control and monitor services such as heating, cooling, ventilation, lighting, shading, and security. Nowadays, automation systems are deployed in commercial buildings (e.g., offices, malls, factories, hotels, and medical institutions) as well as residential buildings (e.g., houses and apartments). In the commercial domain, the main goals of a BAS are to reduce energy consumption and facilitate building monitoring, maintenance, and operation. In the residential domain, a BAS helps people achieve many of their daily routines efficiently while maintaining their comfort, which is especially needed for elderly and individuals with disabilities. To achieve the aforementioned goals, a BAS allows users to control devices/appliances/services either according to the readings of a network of specialized sensors or according to their personal preferences. For example, a home automation system might allow a resident to control the electrical light intensity according to his/her mood or according to the measured natural light intensity in the room in order to save electricity.

Traditional building automation systems interact with the user using a graphical user interface [5–8]. On the other hand, recent automation systems (e.g., Alexa [9], Apple [10]) integrate voice control for extra convenience and flexibility. There are two main approaches to implement voice recognition module for a BAS: first, an offline approach that relies on trained machine learning algorithms capable of recognizing uttered control commands. While this approach provides better availability as it does not require Internet connection, it requires substantial amount of training data in order to achieve high recognition accuracy. The second approach is to leverage one of the voice recognition services that are available through cloud APIs such that the actual voice processing is performed in the cloud and the BAS is only responsible for processing the returned textual data and translating it into actions.

Integrating voice recognition with a BAS is especially important for people who suffer from paralysis. Every year, around 18,000 new spinal core injuries, which are the main cause of paralysis, are registered in the United States [11]. The main causes of these new cases are accidents, diseases, and acts of violence. The most severe types of physical paralysis are monoplegia, hemiplegia, paraplegia, and quadriplegia [12]. People with monoplegia suffer from paralysis in one upper or lower limb (i.e., one arm or one leg), while people with hemiplegia suffer from paralysis in one arm and one leg on the same side of the body. More severely, people with paraplegia cannot move both lower limbs, while people who suffer from quadriplegia lose the ability to voluntarily move any of their upper or lower limbs. Although paralyzed people can be intellectually capable, they normally cannot perform their daily activities like ordinary people and they need continuous support and monitoring. By integrating voice recognition with home/office automation systems, paralyzed people can control all devices and appliances around them; hence, live a normal life and be productive members of the society with minimal supervision.

The main contributions of this article are as follows:

(I) Deriving a skeleton for integrating voice control with any building automation system. The proposed voice control skeleton supports online and offline modes. During online mode (i.e., Internet connection is available), the skeleton relies on Google Speech Recognition API to recognize user commands and to allow other smart features (e.g., web search and run applications). During offline mode, the skeleton utilizes an adapted version of the CMUSphinx open-source speech recognizer [13]. In addition, the proposed skeleton uses a wake-up-word (WUW) model with speaker verification, so that the control commands are only initiated by the authorized user.

(II) Implementing the proposed skeleton and integrating it with an office automation system as a proof of concept use case. The office is equipped with Konnex (KNX) components (e.g., lightning, blinds/shutters, heating, and air-conditioning) [14]. The integrated system allows the user to issue voice commands in order to control the KNX components (e.g., "turn on all lights").

The rest of the article is organized as follows: Section 2 provides some background on KNX automation systems, Section 3 presents related work, and Section 4 covers the voice control skeleton design. Section 5 describes the experimental use case along with the results, and Section 6 concludes the article.

2. KNX Automation Systems

Building automation or smart building is the process of transforming traditional buildings into ones that can be automatically controlled using electronic devices (e.g., computers). Most modern buildings are smart with many smart devices [15]. One of the most widely used building automation standards is Konnex (KNX) [16]. KNX is an open standard that is based on three previous standards: the European Home Systems Protocol (EHS), BatiBUS, and the European Installation Bus (EIB or Instabus). KNX is a standardized technology, which controls the automation of integral functions of any residential, commercial, or industrial building.

KNX uses a wired bus system for building automation using a central software provided by KNX association, which is called Engineering Tool Software (ETS). ETS provides creating projects, configuring KNX devices, and a huge products' catalog and database of KNX devices from various manufacturers such as Siemens and ABB. KNX enables various areas of a building to be controlled, link them together, and adjust them to each other. This includes, for example, controlling, managing, and monitoring of lighting, roller shutters, security, energy management, heating, ventilation, air-conditioning (HVAC) systems, and many other applications.

KNX achieves its functionality by utilizing dedicated controllers that are connected with abundant and diverse sensors and actuators using an open worldwide standard with over 300 different manufacturers producing products that all interwork and operate together seamlessly. KNX operates by ensuring all components, devices, features, and functions of any building (or outdoor space) communicate via one common language (i.e., KNX standard) instantly and remotely as depicted by Figure 1. Moreover, KNX offers interfaces to many other technologies, such as Ethernet (LAN) and lighting controls with DALI and BACnet network, making it easy to exchange information and data via the KNX network. In particular, the KNXnet/IP supports connection to building control (e.g., OPC, PROFINET, SIMATIC S7).

The basic unit of the KNX communication protocol is the Telegram, which is similar to the TCP/IP data packet. A telegram contains several fields such as control field, source address, and receiver address [17]. Figure 2 shows the telegram structure. Each KNX device deployed as part of an automation system is configured with an individual address of 16 bit. The individual address is divided into three fields—4-bits for area, 4-bits for line, and 1-byte for device as shown in Figure 3. The source address field in the telegram represents the individual address of the sender.

In addition, each functionality or channel of KNX devices has a group address. The group address can be shared between two or more functionalities/channels inside the same device or different devices. Figure 4 shows how the group address can be formatted in two levels (i.e., main group and subgroup) or three levels (i.e., main group, middle group, subgroup) [17]. The group address requires only 15-bits (i.e., bit D15 is reserved). The receiver address field in the telegram can be either an individual address in the case of point-to-point connection or a group address in the case of multicast/broadcast connection.

The N-PDU field in the telegram is destined/generated by the network layer, and it consists of two parts—8-bit value and T-PDU as shown in Figure 2. The most significant bit in the 8-bit field determines whether the receiver address is an individual or group address. The next 3-bits provide routing

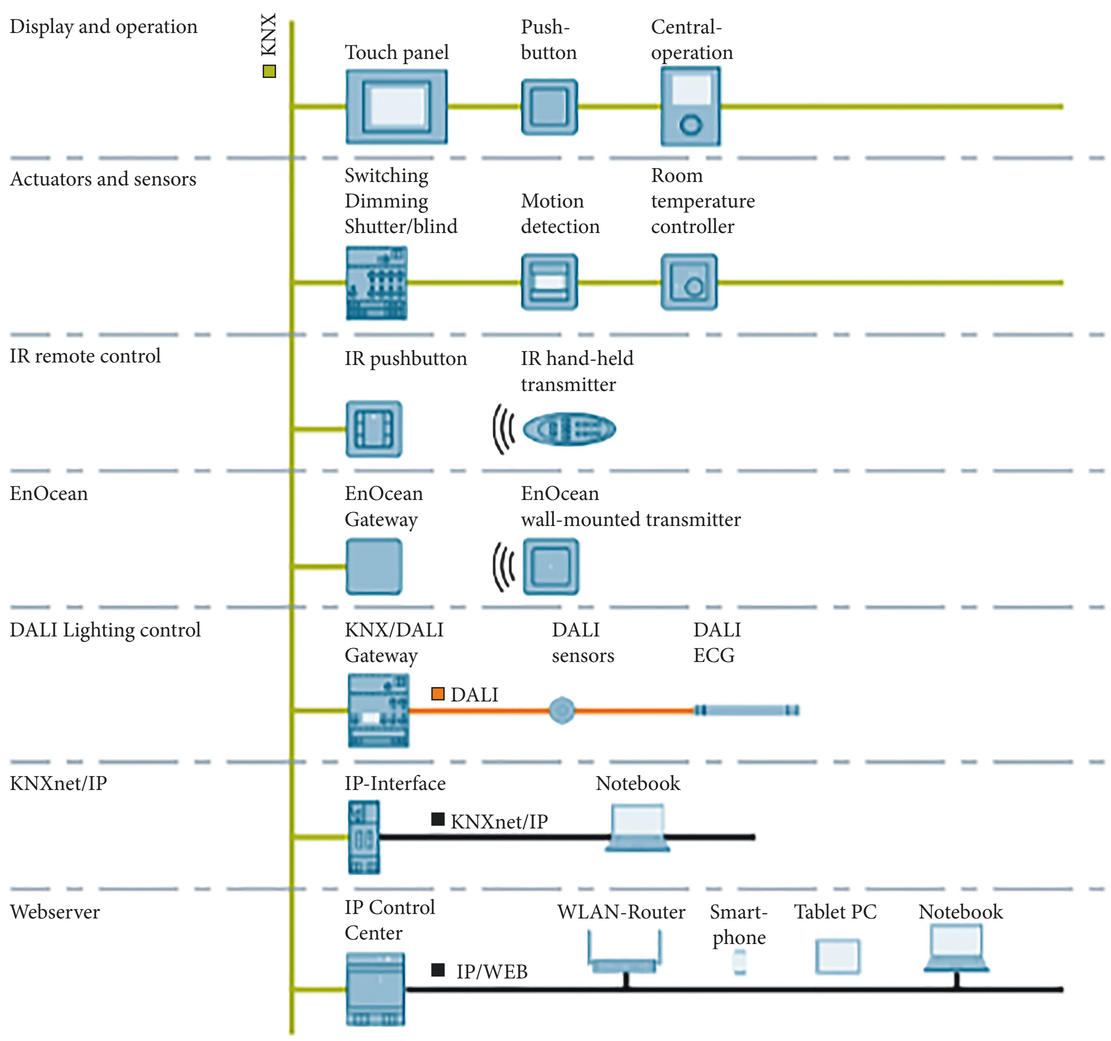

Figure 1: KNX devices and network capabilities illustration.

Control Field <8 bit>	Source Address <16 bit>	Receiver Address <16 bit>	N-PDU			Check Field <8 bit>
			<8 bit>	T-PDU		
				<6 bit>	A-PDU	

Figure 2: KNX telegram structure.

Area <4 bit>	Line <4 bit>	Device <8 bit>

Figure 3: Individual address structure.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
■	Main Group				Sub Group										
■	Main Group				Middle Group			Sub Group							

Figure 4: Group address structure.

information, and the following 4-bits provide the length of the payload. The T-PDU part is destined/generated by the transport layer, and it consists of two parts: 6-bit value and A-PDU. The most significant 2-bits in the 6-bit field determine the type of the transport layer communication, and the next 4-bits represent the sequence number for numbered communication types. The A-PDU is destined/generated by the application layer, and it represents the actual useful information (i.e., payload). More information about the telegram structure can be found in Ref. [17].

3. Related Work

In this section, we first discuss previous work related to building automation systems in general. Second, we present the most recent related work in the area of voice-controlled building automation systems. The second part of the related work is divided into three subsections according to the type of the deployed speech recognition mechanism. The author in Ref. [18] provided the basic principles and technologies for building automation. The work discusses four main issues related to building automation: network communication protocols and standards, integration and interfacing of building automation subsystems and multiple building systems, control strategies, configuration, and technologies for automation control.

Sauter et al. discussed the historical evolution of factory and building automation systems [19]. The article lists typical services provided by building automation systems: climate control, visual comfort, safety, security, transportation, one-way audio, energy management, supply and disposal, communication, and information exchange. Regarding building automation, the authors concluded that the main challenge lies in the applications due to the large amount of data collected from an increasing number of data points. To overcome this challenge, smart ways of data processing are required. In Ref. [20], the authors provided a more detailed overview by discussing the fundamental concepts, features, standards, and requirements of building automation systems. Different building automation technologies (e.g., BACnet, KNX, and ZigBee) were analyzed and compared according to the coverage of the functional aspects (i.e., grouping, event notification, alarm notification, historical data access, scheduling, and scenarios). The authors concluded that no specific technology is capable of providing all required functionalities of building automation.

An important domain of building automation is its efficiency in reducing energy and improving occupants' quality of life. The authors in Ref. [21] discussed the importance of building automation and control systems (BACSs) and technical building management (TBM) systems in optimizing building performance and gave a general framework for the analysis of the performance for a wide range of BACS. The framework focuses on the need for building performance simulation, data analytics, performance tracking, and the need for sensors and actuators to measure the quality of building environment.

The research work in Ref. [22, 23] focused on the interaction opportunities between occupants and building automation systems. In Ref. [22], the authors focused on scaling the demand management programs from the level of a single building to the level of a district microgrids. Although modern building automation systems integrate thermostatic control with improved sensors and data analytics to reduce energy consumption by regulating the thermostat set point, the demand management actions are either predetermined by a field engineer or set by the user with static rule-based options. As a result, the efficacy of these systems is not fully exploited as aspects such as thermal inertia, real-time user activity, and weather conditions are ignored. To overcome this shortage, the authors proposed a distributed demand management system with three main features: first, using feedback mechanisms to scale up the states of the different buildings (i.e., temperatures, occupancy schedule, availability of renewable energy) to district-level microgrids; second, integrating occupant behavior using multiobjective optimization; And third, leveraging feedback-based "personalized" corrections for each building to provide consistent performance against weather conditions and user behavior at the district level. The simulation results of the proposed system show improvement in the energy cost and reduction in the percentage of people dissatisfied.

In Ref. [23], Baldi et al. tackled the inefficiency associated with using fixed rules to manage thermostatic loads of buildings. These rules represent a barrier against dynamic load regulation according to rooms geometry, orientation, and human occupancy (i.e., smart zoning). The authors proposed to formulate the problem as multiple-mode feedback-based optimal control problem and solve it by specifying the parameters of the possible control strategies and adaptively finding the optimal strategy. The proposed solution reduces energy consumption by 15% and increases thermal comfort by 25% when compared to strategies with predetermined rules. On the other hand, the proposed solution reduces energy consumption by 4% and increases thermal comfort by 8% when compared to optimized strategies without smart zoning.

The main idea in Ref. [24–26] is to improve the efficiency of building energy management systems (BEMS). The authors in Ref. [24] discussed the potential of using Internet of Energy (IoE) to solve the efficiency problems of BEMS such as energy data loss and energy overloading. Analogous to data generation, storage, and transmission in IoT and smart grid, IoE facilitates the generation, storage, and transmission of energy using different types of nodes such as renewable sources, batteries, and transformers, respectively. The demand and supply information from IoE can help to reduce buildings power consumption. In Ref. [25], the authors proposed an optimization control system for indoor thermal comfort and building energy consumption using predictive mean vote and simplified thermal model. The proposed system can efficiently regulate heating, ventilation, and air-conditioning systems of buildings. Hettiarachchi et al. proposed an IoT-based energy management system to reduce energy wastage caused by shortage of monitoring and automation [26]. The proposed system utilizes a trained XGBoost machine learning model hosted on the cloud to predict consumption based on device and sensor data (e.g., electricity load, weather data, and temperature) collected and stored on the cloud as well. The predictions can be used either to perform manual adjustment of device operation by the user or to perform automatic adjustment with priority given to devices tagged as critical.

3.1. Hardware-Based Speech Recognition for Voice Control. In this subsection, we focus on the related work, which leverages specialized hardware components for the speech recognition of control commands. In Ref. [27], Guo et al. described the implementation of a voice-controlled home automation system using the embedded sound-controller, SUNPLUS SPCE061A, combined with Microchip 2.4 GHz ZigBee modules. A multistage algorithm and the speech recognition library provided by SUNPLUS are used to determine the target room, appliance, and operation. Examples of tested commands are (living room, light, ON) and (bedroom, TV, volume up). No recognition accuracy results are presented. Similarly, the authors in Ref. [28] added multiple LD3320 voice recognition modules to ZigBee-based home automation network. The LD3320 is based on speaker-independent automatic speech recognition (SI-ASR) technology with a built-in library. Although the recognition accuracy of the LD3320 module is high (i.e., 95%), only 50 keywords of specific languages can be recognized.

The authors in Ref. [29] proposed a smart system to control distributed home appliances using either voice or gesture commands. The main purpose of the proposed system is to improve the quality of life for senior citizens and people with disabilities. The system mainly consists of a transmitter module and a receiver module. In the transmitter module, a microcontroller receives input commands from an audio-capturing device and a tilt sensor. After processing the input command, an RF module is used to transmit the signals to the appliance of interest. In the receiver module, the received signals are forwarded to a microcontroller, which control the relay of the device of interest.

In Ref. [30], the authors proposed a multifunctional smart home automation system. The system leverages a dedicated hardware called Voice Recognition Module (V3) [31], which can be adapted to a particular speaker's voice. The module supports up to 80 voice commands and requires two setup steps: training and loading. The authors tested the system using 10 speakers (5 males and 5 females) to train the voice recognition module and then measure the best distance between the speaker's mouth and the microphone and the average number of attempts for the command to be correctly recognized. The experimental results showed that the best distance is between 1 cm and 15 cm and the average number of attempts is between 1 and 1.3 for various commands.

3.2. Off-the-Shelf Software-Based Speech Recognition for Voice Control. In this subsection, we focus on the related work, which leverages off-the-shelf free software packages for the speech recognition of control commands. The Wolfram Alpha, a cloud-based API, is used in Ref. [32] to recognize the spoken command and send it to the desired home appliance through a Raspberry Pi board. Unfortunately, no analysis of the recognition accuracy is provided for the system. In Ref. [33], the authors presented the design and implementation details of the wireless home automation system. The system consists of three main components: a microphone module to capture voice commands, a centralized controller to receive and process (i.e., filter and sample) commands before sending them to a personal computer for recognition, and appliances' control modules to receive the recognized commands and control the appliances' operations accordingly. The speech recognition part was accomplished using Microsoft Speech AI library. Using 1225 commands from 35 male and female subjects, the recognition accuracy was 79.8%.

A mobile application is proposed in [34] to provide voice and touch control of home automation. The mobile application leverages free applications on Google Play Store (i.e., AMR-Voice and Arduino Bluetooth Controller) to receive voice commands and provide a graphical user interface of a remote control. The Arduino Uno board acts as the main controller and is connected to the application via Bluetooth. However, the authors did not provide any recognition accuracy results. In Ref. [35], the authors proposed to use Kinect V2 to receive voice commands and then send them to a trained computer system with Microsoft Speech Recognition Engine for identification. The reported accuracy is 95% when the distance between the user and the Kinect is less than four meters and when the noise level is less than 53 dB.

The authors in [36] discussed and evaluated three approaches to implement voice control for smart home automation. The first approach relies on a cloud-based voice recognition service and a cloud-based home automation system (e.g., Google Speech API, Amazon Alexa voice services, and Bing speech API). The second presented architecture relies on a local home automation system with a gateway to a cloud-based voice recognition service (e.g., Google Speech API and Bing speech API). The third approach leverages an offline voice recognition engine integrated with the local home automation system.

In [37], the authors proposed a smart home automation system with voice control using BitVoicer software [38]. The system is built using an Arduino board, set of specialized sensors, RF receiver/transmitter, and Ethernet shield. The system has two operational modes: automatic mode, which controls the appliances according to sensor readings without user intervention, and manual mode, which controls the appliances according to user voice commands. The system was tested using light emitting diodes (LEDs) instead of actual appliances, and no accuracy results are provided.

The authors in Ref. [39] developed a voice control home security and automation system. The security aspect of the project is simply represented by an SMS message sent to the user whenever an ultrasonic sensor detects any motion. Voice recognition is achieved by capturing the voice commands and converting them to text using an Android mobile application (i.e., AMR-Voice), and then, the text is sent to the main microcontroller using Bluetooth for analysis. Only ON/OFF commands are used for testing, and LEDs are used as outputs instead of real appliances.

Putthapipat et al. implemented a speech recognition gateway [40] on top of the previously published home automation framework, PiFrame [41]. PiFrame runs full-feature Linux OS to shorten the development cycle and reduce system's limitation. Google Speech API is used to convert voice commands into text and send it back to the PiFrame, which controls the home appliances accordingly. Most recently, Vishwakarma et al. [42] presented a smart home automation system using IoT. User commands can be received as voice commands through Google Assistant or through a web-based application in cases of noisy backgrounds.

3.3. Bespoke Software-Based Speech Recognition for Voice Control. In this subsection, we focus on the related work, which implements customized software speech recognition engines to identify control commands. In Ref. [43], the authors presented a voice-controlled home automation system using natural language processing (NLP) and Internet of Things (IoT). According to the authors, the system provides better control of the home appliances than the traditional mimicking of ON/OFF switching. The central component of the system is a mobile application, which is responsible for processing the user voice commands using artificial intelligence (AI) and NLP algorithms. In addition, the system contains MKR1000 Arduino boards connected to home appliances. The MKR1000 boards receive the commands from the mobile application and collect responses and send them back. The authors neither provided any elaborations on the supported commands nor presented any voice commands recognition accuracy results.

A phoneme-based voice recognition engine is proposed in Ref. [44] and is utilized to control a home automation system. The proposed engine consists of four stages: endpoint detection, feature extraction using Mel frequency cepstral coefficients (MFCCs), phoneme recognition using support vector machine (SVM), and word recognition using SVM. The reported average recognition accuracy using a sample of 7 words and 10 utterances per word (i.e., 70 utterances) is 90.1% with an average recognition time of 0.91 second. In Ref. [45], the authors implemented a voice command interface (VCI) for smart home automation systems. The proposed VCI defines the patterns of commands that can be mapped to the functionalities and locations of home devices. The VCI is compatible with the existing voice recognition services (e.g., Alexa from Amazon) and customized ones. The authors only confirmed that the VCI is successfully operational without any recognition accuracy results.

Compared to previous literature, this article describes a general approach that can be used to integrate voice control with wake-up word and speaker verification to any existing building automation system. The complete design details of the proposed approach are presented. In addition, the proposed approach is implemented and tested using an office automation system with KNX equipment. Finally, recognition accuracy for three speakers is measured and reported.

4. Skeleton Design Details

In this section, we start by discussing the flow diagram of the proposed voice control system shown in Figure 5. Upon initializing the system, a wake-up word (WUW) engine is activated. The WUW engine recognizes and records human voice, and then, it checks if the recorded audio file contains a predetermined wake-up word (e.g., Hey Computer). Once the wake-up word is detected, the recorded audio file is used to authenticate the user via a speaker verification module. After the speaker is verified, the Internet connectivity is checked to select the most appropriate speech recognition scheme (i.e., online or offline recognition). For online speech recognition, the Google Speech recognition API and a natural language processor are used to identify the desired functionality and whether there are any defined responses to be generated and processed. For offline speech recognition, the proposed system leverages an adapted version of the famous CMUSphinx toolkit to recognize predefined automation commands. Next, we discuss the details of each stage separately.

4.1. Wake-Up Word Engine. Wake-up word (WUW) detection, also called automatic voice trigger, is one of the most prominent applications related to speech recognition. In order to attract system's attention, the user should utter a predefined WUW that the system is trained to recognize. WUW detection can be achieved using algorithms that are based either on pattern matching with predefined templates or on the automatic speech recognition (ASR) framework. The latter type is more robust and generalized.

The steps associated with a typical ASR-based WUW detection process are discussed in Ref. [46] and reproduced in Figure 6. After speech enhancement algorithm, a voice activity detection (VAD) module continuously detects human speech and acoustic events. Afterwards, the features of the detected speech are extracted and inputted to a decoder, which cooperates with an acoustic model to complete the delicate alignment of the speech phoneme boundaries. For each speech unit that is output from the decoder, a variety of confidence metrics are calculated from multiple perspectives and a classifier is used to determine whether the processed speech is the predetermined WUW or not.

There are multiple readily implemented WUW engines that can be utilized to add voice control for automation systems such as the Snowboy hotword detection engine [47] and the Porcupine wake word detection engine [48]. Both engines are lightweight (i.e., have low memory footprint), highly customizable (i.e., user can define and train his/her own single or multiple wake-up words and determine the detection sensitivity), and portable. However, when comparing the two engines in terms of detection accuracy and CPU usage, Porcupine is 5.7 times more accurate than

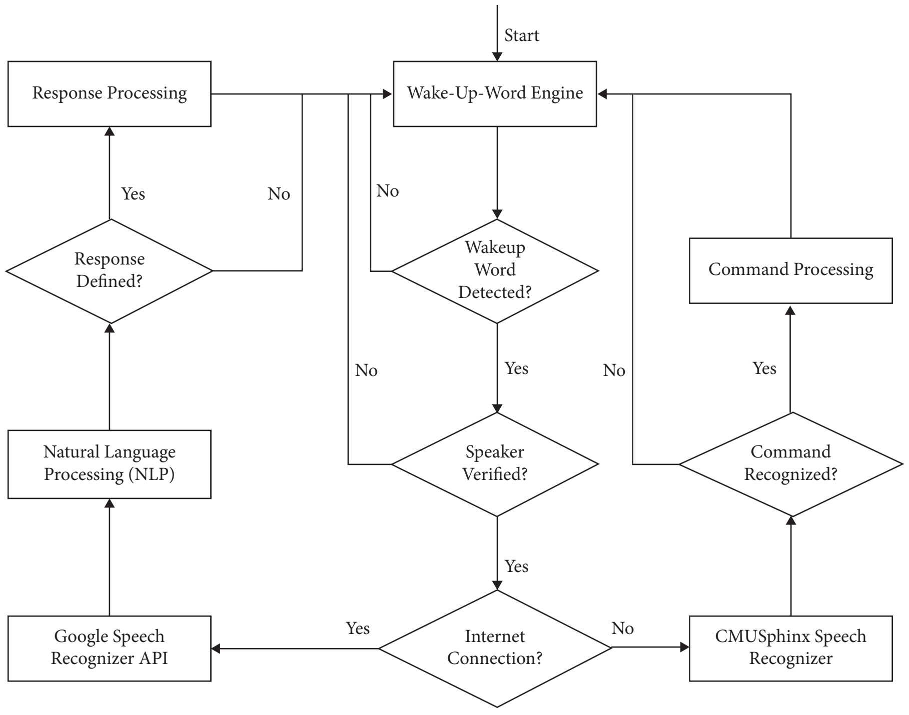

Figure 5: General flow diagram.

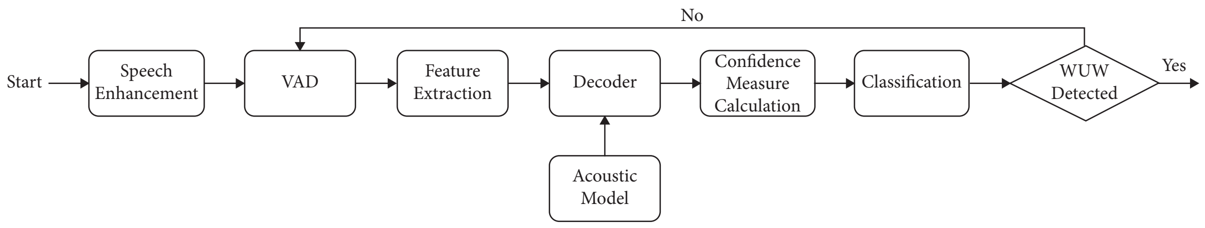

Figure 6: WUW engine diagram.

Snowboy while requiring 6.5 times less CPU usage [49]. In addition to the licensed business editions, these engines have open-source personal editions.

4.2. Speaker Verification. Speaker recognition (SR) is the process of recognizing the speaking person based on the voice bio-metrics extracted from the speech signal. SR includes speaker identification (SI), speaker diarization (SD), and speaker verification (SV). SI is the process of determining which preregistered speaker is speaking. In other words, SI answers the question (who is speaking?) [50]. On the other hand, SD receives an audio file with multiple speakers as an input and the output is a separate audio signal for each speaker. In other words, SD answers the question (who spoke what and when?) [51]. SV is the process of authenticating the identity of a speaker.

In building automation, SV is the most relevant as it is used to determine whether or not a specific speaker has permissions to use the system. SV can be classified either as text-dependent (i.e., verification is done for a predetermined phrase) or text-independent (i.e., verification is done for any spoken phrase) [52]. In this work and as shown in Figure 5, a recorded audio file with the correctly detected predefined WUW is provided by the WUW engine to the SV engine in order to authenticate the speaker. Next, we present two possible techniques to implement the more general text-independent SV despite that the proposed skeleton always validates the speaker for the same phrase (i.e., the predetermined WUW).

The first technique is based on a Gaussian mixture model (GMM) fitted to the voice features of the user with permissions to control the automation system of concern. A Gaussian mixture is a function consisting of several Gaussians, each of which is determined by $k\epsilon$ $(1, \ldots, K)$, where K is the number of clusters in the dataset. Figure 7 shows the process of generating a GMM file for a specific authorized user. The features vectors of N audio files from the targeted user are extracted and used to train the GMM and generate its output file.

The feature extraction in SV is normally based on MFCCs, filterbank energies, and delta functions, which can be applied using python_speech_features library. SciKit-learn Python machine learning can be used to define the

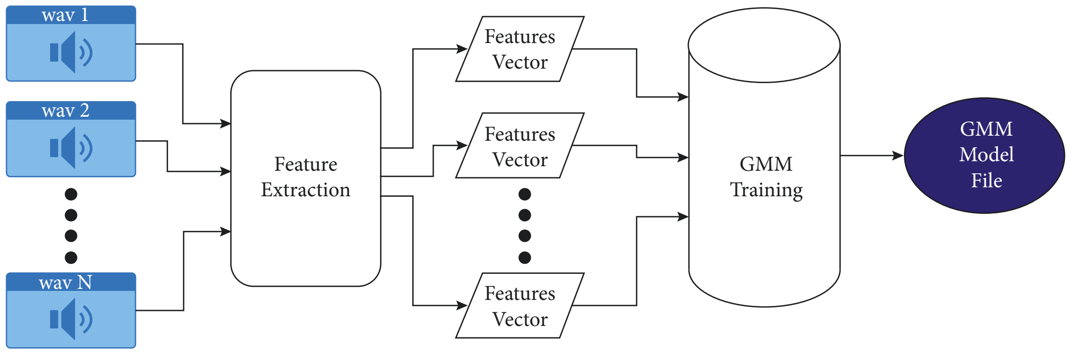

FIGURE 7: GMM for speaker verification.

GMM. The input audio files might need to be processed for noise reduction, sample rate changing, and format conversion using libraries such as Librosa and pysndfx with sound exchange (SoX) package. With every access trial to the automation system, the features vector extracted from the audio file, which contains the correctly detected WUW is inputted to the created GMM file. The GMM uses the score method to compute the cosine similarities between the input vector and the dataset used for training. The output of the score method is compared to a predetermined similarity threshold (e.g., 95%) to authenticate the user.

The second technique is based on residual neural network (ResNet), an artificial neural network (ANN) that utilizes skip connections or shortcuts to jump over some network layers in order to mimic the pyramidal cells in the cerebral cortex. Normally, ResNet models are implemented using two- or three-layer skips with in-between rectified liner units (ReLUs) and batch normalization. Similar to the first technique, feature extraction for the dataset can also be achieved using the python_speech_features library.

In order for the ResNet model to verify a specific speaker, the features extracted from the audio files of this speaker need to be included in the training set of the model. During training, enrollment embeddings are generated from the audio features of each speaker and saved. With every access trial to the automation system, the feature extracted from the audio file, which contains the correctly detected WUW, is inputted to the trained ResNet to generate its embeddings and compare them to the enrolled embeddings. The comparison score determines whether speaker is verified or not according to predetermined threshold.

4.3. Automatic Speech Recognition. Automatic speech recognition (ASR) is the field in which algorithms and procedures are developed to translate the spoken languages into text. To achieve this task, the ASR process converts the spoken sentences into audio signals and apply statistical models to extract the audio statistical properties such as Mel Frequency Cepstrum Coefficients (MFCCs), Linear Predictive Cepstral Coefficients (LPCCs), and discrete Fourier transform (DFT). Based on the collected features, the phoneme and word identification is done using pattern recognition probabilistic models such as hidden Markov model (HMM) and Gaussian mixture model (GMM). Recent ASR techniques leverage the advances in deep learning and big data fields. The latter is clear by the wide industrial adoption of deep learning techniques to perform ASR tasks. Siri, Alexa, Google Assistant, and Cortana are examples of voice assistants that use state-of-the-art speech recognition technologies to translate and understand the spoken sentences and take actions such as schedule a meeting and place purchase orders.

One way to perform ASR is by handing off the task to cloud servers on which the recognition models and data are installed. We refer to this approach as online speech recognition. On the other hand, offline speech recognition does not require Internet connection because the models are installed locally on the device. Accuracy, speed, security, and maintenance are aspects that should be taken into consideration when deciding which ASR approach to use (i.e., online vs offline). The skeleton proposed in this article deploys an online ASR model that is used whenever an Internet connection is present to benefit from the high recognition accuracy. At the same time, the skeleton deploys an offline model that is only used when no Internet connection is present to provide availability.

In a typical online speech recognition engine, a voice signal is captured in the local system through a microphone and sent over the Internet to designated servers in order to be processed by machine learning techniques and converted to text (i.e., speech to text). The text is then returned to the local system and inputted to a natural language processor for recognition. There are multiple popular cloud-based speech-to-text APIs that one can leverage such as Google Cloud Speech API, IBM Watson Speech to Text, Microsoft Azure Bing Speech API, and Amazon Transcribe.

Natural language processing (NLP) is a subfield of AI that allows computers to understand and process human spoken languages. In this skeleton, the purpose of NLP is to extract a meaningful command (e.g., light_one_on) from a spoken sentence (e.g., switch on light number one). A typical natural language processor performs the following steps [53]:

(i) Sentence segmentation: each paragraph is divided into separate sentences.
(ii) Word tokenization: each sentence is divided into separate tokens (i.e., words).

(iii) Predicting parts of speech: each token is identified as a noun, verb, adjective, etc.

(iv) Lemmatization: each token is converted to its base or dictionary form; that is, all inflections are removed (e.g., "switching" becomes "switch").

(v) Identifying stop words: each token that represents a stop word (e.g., "and," "a," "the") is filtered out.

(vi) Dependency parsing: the relationships between tokens are identified.

(vii) Named entity recognition: each noun token is labeled with real-word concept that represent it. For example, "John" is labeled as "People's name."

(viii) Conference resolution: each token that represents a pronoun (e.g., "it," "he," "she") is associated with the token to which it refers.

Depending on the nature of the targeted speech, a subset of the above steps can be used to a perform the required NLP. The proposed skeleton targets identifying user commands for automation systems, which are normally short sentences with specific desired actions; hence, word tokenization, lemmatization, and identifying stop words can be sufficient to recognize the spoken command as discussed later in Section 5.2.

In offline speech recognition, the entire recognition process is completed locally using dedicated software. The process consists of receiving an acoustic waveform, dividing it in order to isolate sounds from noise and silence, and analyzing the sounds to determine the best matching set of words. There are multiple offline speech recognition platforms that are free, open source, and customizable such as CMUSphinx and Mozilla DeepSpeech. CMUSphinx accumulates 20 years of research at Carnegie Mellon University and is continuously updated to utilize most recent technologies. DeepSpeech engine is based on open-source code algorithms and Google TensorFlow machine learning toolkit.

4.4. Security Aspects. Any automation system with authorized access might be at risk of attacks that could bypass the authentication step. For systems with speaker verification, examples of such attacks include deepfake audio attacks and replay attacks. Deepfake audio attacks are still no the rise as there are many reports of deepfake attacks that tricked human beings but there are no reported cases where machines were exploited. On the other hand, there are many reports of successful replay attacks. For example, the authors in Ref. [54] claim that when a recorded voice of an Amazon Echo device owner was replayed asking: "Alexa, who am I?", the system replied with the owner's name. Similar vulnerability was reported in "Google Home" system because it only verifies the wake-up word, which is "OK Google"; whatever the user says after that is not verified.

Similarly, the proposed skeleton only authenticates the WUW, which makes it vulnerable to replay attacks. So, if another user gives a command after the authenticated owner utters the WUW, the command will be executed. The question here becomes what are the risks of such attack? To answer that, the probability of the attack and its impact need to be analyzed. The impact of reply attacks heavily depends on the criticality of the system controlled by the skeleton. For example, the use case of the office automation system discussed in Section 5 has low criticality; hence, no high risks are posed (e.g., high electricity bill for keeping lights ON for long hours). However, if highly critical systems are controlled (e.g., automation systems in hospitals), then the probability of such attacks needs to be reduced by deploying techniques such as higher-order spectral analysis (HOSA)-based features to capture traces of replay attacks [54]. The detailed analysis of the probability and impacts of the security aspects will be addressed in future work.

5. Experimental Setup and Results

As a test case, the proposed skeleton is integrated into a KNX-based office automation system, which is described in the following subsection. Afterwards, the implementation details of the skeleton and the experimental results are presented.

5.1. KNX-Based Office Automation System. The under-study office automation system is built using the KNX devices and components described below. The drivers of devices used in the system are downloaded from ETS devices catalog or from the SIEMENS official website. The office monitored physical parameters include presence, humidity, temperature, and carbon dioxide (CO_2) levels, while the control parameters were selected to be three rows of lights and a blind shutter. These parameters were selected to demonstrate the ability to voice control the main aspects of a BAS that is energy management, control, and monitoring. Figure 8 illustrates a reduced KNX diagram of the full schematics of the tested system.

As mentioned in Section 2, the programming/configuration of the KNX devices in our office is done using the ETS. Each KNX device has a specific button that must be pressed such that it can be programmed/configured. Moreover, each device and functionality must be configured with a group address and a specific datapoint type (DPT) to determine the value set or sent to that device/functionality. Table 1 lists the KNX devices installed in our experimental office along with their configuration parameters.

5.2. KNX Switching/Dimming Actuator. This device provides three channels (i.e., A, B, and C) of switching and dimming functionalities for lights and three sensor channels. In our office, three rows of lights are connected to the three switching/dimming channels in order to control each light independently. The installed lights do not support dimming functionality; hence, it is sufficient to only use a 1-bit KNX DPT as shown in Table 1 in order to switch each row of lights ON/OFF. If the dimming functionality was available, a more complex structured DPT would be required (i.e., 3.007 DPT_Control_Dimming). In addition to lighting, a presence detector with integrated brightness sensor is also connected

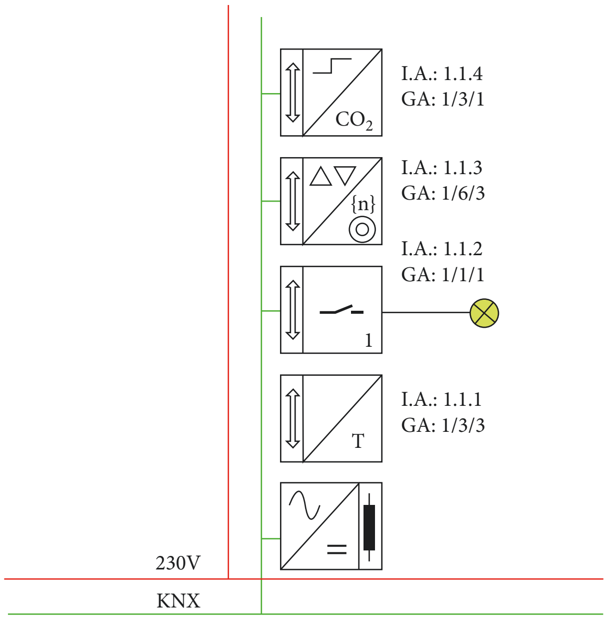

Figure 8: Office basic KNX schematics.

Table 1: KNX device configuration.

Device/functionality	Group address	KNX Datapoint id and name	Datapoint size
Switch channel A (light 1)	1/1/1	1.001 DPT_switch	1 bit
Switch channel B (light 2)	1/1/2	1.001 DPT_switch	1 bit
Switch channel C (light 3)	1/1/3	1.001 DPT_switch	1 bit
Presence detector with brightness sensor	1/1/150	9.004 DPT_value_lux	2 byte (float)
CO_2 sensor	1/3/1	9.008 DPT_value_airquality	2 byte (float)
Humidity sensor	1/3/2	9.007 DPT_value_humidity	2 byte (float)
Temperature sensor	1/3/3	9.001 DPT_value_temp	2 byte (float)
Blind shutter actuator	1/6/3	1.008 DPT_updown	1 bit

to this actuator. At the current stage of our implementation, only the brightness sensor is leveraged to report the office brightness level through a 2-byte float DPT in the illuminance unit (i.e., Lux). The measured brightness can be used to control the movement of the shutter blinds through the KNX shutter actuator described below.

5.3. KNX CO_2, Humidity, and Temperature Sensor. This sensor has three functionalities: first, it measures the CO_2 concentration level in the air (i.e., air quality measurement) with 2-byte float DPT in parts per million (ppm) unit. Typical CO_2 concentrations in occupied indoor spaces with good air exchange are in the range of 350-to-1000 ppm. Second, the sensor measures the air humidity percentage in the room using a 2-byte float DPT. Third, the room temperature is measured in Celsius degree unit using a 2-byte float DPT. The three readings can be used to control heating, ventilation, and air-conditioning (HVAC) in the office.

5.4. KNX Shutter Actuator. The shutter actuator has a switch functionality that is used to move the office blinds up or down. The configuration in Table 1 shows that 1-bit DPT (i.e., 1.008 DPT_UpDown) is used to control the movement of the office blinds (i.e., 0 for up and 1 for down). This indicates that the blinds in our testing environment will be either all the way up or all the way down. However, one could configure an extra 1-bit DPT to start/stop blinds movement (i.e., 1.010 DPT_Start) or a more complex structured DPT (i.e., 3.008 DPT_Control_Blinds) given that it is supported by the actuator.

5.5. KNX IP Router. The KNX IP router is the interface between the KNX bus and the computer system, which runs the ETS for configuration and the proposed voice command application for controlling the office automation system. Using Ethernet connections, the router forwards telegrams with required functionalities to the targeted devices. The telegrams are forwarded between different KNX lines through a LAN as a fast backbone (i.e., KNXnet/IP Routing connection).

5.6. Voice Control Application. The proposed skeleton is implemented as a voice control desktop application using python programming language version 3.6. The application imports the asynchronous python library for KNX (i.e., XKNX) that supports a huge number of devices' DPTs, which are defined and ready to use. In addition, the XKNX library leverages the asynchronous start () and stop () functions along with the async/await syntax of the Asynchronous I/O python library (i.e., asyncio) to provide gateway scanning and tunneling to the KNX IP router.

For WUW detection engine, the application utilizes the Porcupine wake word detection engine version V1.8, which was chosen for the reasons mentioned in Section 4.1 and because it supports Windows OS. Since the proposed voice control application is implemented on a desktop computer, we chose "Hey Computer" as the WUW in our experimental work. The optimizer tool within the Porcupine engine is used to train the wake word model on the chosen WUW. Notice that there is a dedicated optimizer tool for each platform supported by the Porcupine engine. Since the Porcupine engine is leveraged as is without any modification, its accuracy is the same as the one reported in [49] (i.e., miss rate of 5.9% at one false alarm per 10 hours). Hence, no testing is required for the WUW engine.

For speaker verification, our application utilizes the ResNet model developed by the Korea Advance Institute of Science and Technology (KAIST) [55]. The model is trained on a dataset that contains 240 speakers and 100 utterances per speaker, all in Korean language. In order to be able to verify a user who is authorized to control the experimental office automation system, the extracted audio features of that user need to be enrolled in the model using the process described in Section 4.2. Similar to Ref. [55], in our work, feature extraction is also achieved using the python_speech_features library.

After verifying the speaker using the audio file that contains the correctly detected WUW, the next step in our proposed skeleton is to receive voice commands from the user and identify them using ASR. The implemented control application tests Internet connection to choose between offline and online speech recognition. For online speech recognition, our application leverages the online Google Speech API supported by the Python Speech Recognition 3.8.1 library. This API supports multiple languages, and in our application, the US English language is chosen. The captured voice signals are transmitted over the Internet to Google servers where machine learning techniques are used to process and convert them to text. In order to reduce the effect of ambient noise, the microphone's energy threshold is automatically calibrated so that the voice control application can reliably recognize the speech.

Consequently, the generated text (e.g., switch on light 1) is returned to the local host where a natural language processor is used to extract a meaningful command (e.g., light_one_on). The natural language processor in our application is built using Python Natural Language Toolkit (NLTK) library and TensorFlow AI library. NLTK is used to extract features from the received text using the bag-of-words (BoW) approach. The extracted features in a form of a vector are inputted to a trained machine learning model developed using TensorFlow in order to output the expected desired control command.

The dataset used to train, validate, and test the machine learning model of the natural language processor is represented by the supported commands (e.g., light_one_on, shutter_up, temperature, open_youtube). Each command is represented by a tag, multiple patterns, and a response stored in JSON format. For example, the command "light_one_on" is represented by the tag "light_one_on," seven patterns (i.e., "can you turn on light one," "please turn on light one," "turn on light one," "switch light one on," "switch on light one," "light one on," "on light one"), and the response "light_one_on." Patterns represent the speech commands that the user might say and must lead to the same response.

The preprocessing of the dataset includes the following steps: first, extract the words from all patterns using word tokenization from NLTK. Second, perform stemming using NLTK LancasterStemmer to reduce each word to its root. Third, implement BoW approach using the unique words found in the second step. For each pattern in the dataset, a vector of bits is generated such that each bit is associated with one unique word. When the word exists in the pattern, its associated bit is set to one; otherwise, it is set to zero. For example, if the unique words are ["light," "on," "off," "one," "two," "three"], then the vector for pattern "light one on" is [1, 1, 0, 1, 0, 0]. The fourth step is to map each pattern vector to an output vector that indicates which tag (i.e., command or response) is associated with the respective pattern. Every bit in the output vector is associated with exactly one tag; hence, only one bit in an output vector is set to one, while the remaining bits are set to zero. Lastly, store the list of pattern vectors (inputs) and their respective output vectors into variables and convert them to Numpy arrays in order to be used for training, validation, and testing.

The machine learning model of the natural language processor in our application is a four-layer neural network (i.e., one input layer, two hidden layers, and one output layer). The dataset described above is divided into three parts—training, validation, and testing sets, and the results are discussed in Section 5.4. After the model is trained, validated, and tested, it is deployed in our voice control application. Once the text is returned from the online Google API, it is processed using NLTK to generate its associated vector based on the words in the text. The vector is used as an input to the neural network, which generates an output vector that indicates the expected command.

On the other hand, the offline speech recognition in our application is implemented using CMUSphinx system. The recognition accuracy results of using the CMUSphinx system as is were very low. To improve recognition accuracy, CMUSphinx provides Sphinxbase library and Sphinxtrain tools, which can be used to train a new acoustic model. However, this approach requires substantial audio for training and testing the new model. Another way to leverage the CMUSphinx system is adapting the default acoustic model, which is the approach used in our application. This process requires preparing a ".fileids" file and a ".transcription" file; the first contains the names of the adaptation ".wav" files, and the second contains the sentences uttered in those files.

In order to improve the recognition accuracy of the adapted model, a special dictionary is adopted instead of the full one. The special dictionary only contains the command

Table 2: Speaker verification results.

Enrolled speaker	Tested speaker	Test files	Accepted files	Rejected files	Verification accuracy	False positive rate	False negative rate
User1	User1	15	14	1	97.8%	0%	2.2%
	User2 and User3	30	0	30			
User2	User2	15	15	0	100%	0%	0%
	User1 and User3	30	0	30			
User3	User3	15	15	0	95.5%	4.5%	0%
	User1 and User2	30	2	28			

Table 3: Sample of NLP dataset.

Command	Patterns
Light_two_three_on	"Please turn on light two and three," "Turn on light two and three," "Switch light two and three on," "Switch on light three and two"
Shutter_down	"Move down the shutter," "Move shutter down," "Move down the blinds," "Close the blinds," "Blinds down," "Shutter down"
Get_temperature	"What is the temperature," "How cold is it," "How hot is it," "Tell me the temperature," "How is the weather," "Read the temperature sensor"

words that are used to control our experimental office automation system and their phonemes. In addition, a new grammar that is specified for our control commands is built with the Java Speech Grammar Format (JSGF) and used. The integration of the adapted CMUSphinx recognizer into our application is straightforward because the Python speech recognition library contains the PocketSphinx system with general language model for US English language. It is worth noting that higher accuracy is achieved when using the CMUSphinx recognizer with noise reduction techniques (e.g., ambient noise reduction) and noise cancelling microphones.

Once the desired command is identified either through online or offline speech recognition, the application sends a telegram to the respective KNX device/functionality requesting the command to be executed. For example, the application declares an object of class "Light" for every installed light line, gives it the group address programmed to it, and uses set_on () and set_off () functions to turn the light ON and OFF when "light on" and "light off" commands are received, respectively. For sensors, it is the same process, but the function used is sync (), which is used to synchronize the measured value.

The KNX switch, shutter actuator, and sensors are connected to the PC on which the ETS and the voice control application are running through the KNX IP router. After configuring the KNX devices, the communications between the PC and the KNX devices were successfully tested through direct commands from the PC. Next, the functionalities of the voice control application were tested, and the results are presented in the following three subsections.

5.7. Speaker Verification Results. The speaker verification ResNet model was tested with three users. For each user, 16 English language sentences were recorded in ".wav" format. For each user, one ".wav" file is used for enrollment as discussed in Section 4.2, while the remaining 15 files and the files of other speakers are used for testing. The threshold of comparison score is set to 95%. Table 2 lists the verification results of each user after being enrolled. For example, when User1 is enrolled, 14 out of his 15 ".wav" file are correctly verified and all User2 and User3 files are correctly not verified (i.e., verification accuracy = 44/45 = 97.8%). For all users, the verification accuracy is higher than 95%.

5.8. Online Speech Recognition Results. The dataset used to evaluate the accuracy of the implemented NLP consists of 167 patterns associated with 29 commands (i.e., on average six patterns per command). Table 3 lists a sample of the dataset that consists of three commands and their respective patterns. The patterns represent the text generated by the online Google Cloud Speech API after converting the recorded audio files. Once the dataset is preprocessed as described in Section 5.2, it is divided into training, validation, and testing sets according to the percentages: 70%, 15%, and 15%, respectively.

Figure 9 shows the training and validation accuracies of the proposed NLP. The training accuracy saturates at 100% after 500 steps and the validation accuracy saturates at 97.5% after 450 steps. The testing set consists of 25 patterns (i.e., 0.15 x 167), four of which were incorrectly predicted (i.e., testing accuracy = 21/25 = 84%). For example, the pattern "Close the blinds" was wrongly predicted as the command "Shutter_up" because the word "Close" was neither included in the training set nor in the validation set. The accuracy of the implemented online speech recognition is expected to increase and the size of the dataset increases.

5.9. Offline Speech Recognition Results. The implemented CMUsphinx recognizer is tested using 300 ".wav" audio files recorded from the same three users in the speaker verification testing (i.e., 100 files from each user). The files contain commands that are used to control our experimental office automation system such as turn on all lights, turn off all

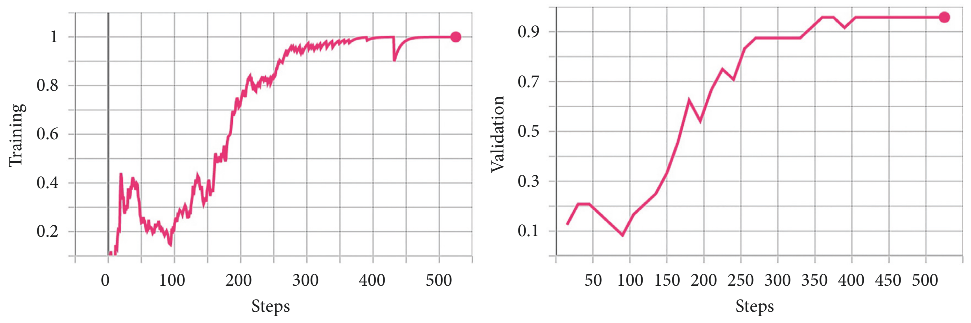

Figure 9: Training and validation accuracies of the NLP.

Table 4: Offline speech recognition results.

Speaker	Recognition accuracy (%)
User1	98
User2	100
User3	96

lights, please turn off all lights, can you turn on light 1 and 2, what is the temperature now, please tell me what is the temperature, move shutter up, and please move down the blind. The test results are shown in Table 4. For User2, all 100 commands were accurately recognized and executed. On the other hand, 98 and 96 commands were accurately recognized and executed for User1 and User3, respectively. The inaccurate recognition of the two and four commands for User1 and User3 is mainly attributed to the high noise in the recorded commands.

6. Conclusions

Advancements in building automation are evolving rapidly with the development of highly intelligent control systems through computers and personal mobile devices. Moreover, the emergence of IoT has revolutionized building automation systems by making them proactive, intuitive, and easily managed by occupants. One of the most important features in modern building automation systems is reliable voice control. In this article, we propose a generic skeleton of implementing voice control for building automation systems. The skeleton consists of three main stages: WUW detection, speaker verification, and dual-mode speech recognition (i.e., online and offline). The proposed skeleton is implemented and integrated into a KNX-based office automation system, which allows the authorized personal to successfully control lights and shutters and poll sensors' readings using voice commands.

Data Availability

The data can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. K. Kodali, V. Jain, S. Bose, and L. Boppana, "IoT based smart security and home automation system," in *Proceedings of the 2016 International Conference on Computing Communication and Automation*, pp. 1286–1289, Greater Noida, India, April 2016.

[2] P. Gupta and J. Chhabra, "IoT based Smart Home design using power and security management," in *Proceedings of the 2016 International Conference on Innovation and Challenges in Cyber Security*, pp. 6–10, Greater Noida, India, Feburary 2016.

[3] D. Pavithra and R. Balakrishnan, "IoT based monitoring and control system for home automation," in *Proceedings of the 2015 Global Conference on Communication Technologies*, pp. 169–173, Thuckalay, India, April 2015.

[4] S. Tanwar, P. Patel, K. Patel, S. Tyagi, N. Kumar, and M. S. Obaidat, "An advanced internet of thing based security alert system for smart home," in *Proceedings of the 2017 International Conference on Computer, Information and Telecommunication Systems*, pp. 25–29, Dalian, China, July 2017.

[5] P. Priyadarshni and S. Sharma, "GUI-MATLAB based home/industrial automation using MCU89S52," *International Journal of Science and Research*, vol. 3, pp. 158–161, 2014.

[6] A. Telepatil, A. N. Hambar, and P. Terwadkar, "Home automation with MATLAB and ARDUINO interface," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, pp. 5501–5507, 2017.

[7] Archiexpo, "Commercial home automation system," 2021, https://www.archiexpo.com/architecture-design-manufacturer/commercial-home-automation-system-57152.html.

[8] C. S. Homes and LIVE LIFE BRILLIANTLY, https://www.control4.com/, 2021.

[9] A. A. team, "Alexa voice service (AVS) overview," 2021, https://developer.amazon.com/en-US/docs/alexa/alexa-voice-service/get-started-with-alexa-voice-service.html.

[10] Apple, "Your home at your command," 2021, https://www.apple.com/ios/home/.

[11] N. B. Jain, G. D. Ayers, E. N. Peterson et al., "Traumatic spinal cord injury in the United States, 1993-2012," *JAMA*, vol. 313, no. 22, pp. 2236–2243, 2015 06.

[12] A. J. Spittle, C. Morgan, J. E. Olsen, I. Novak, and J. L. Y. Cheong, "Early diagnosis and treatment of cerebral palsy in children with a history of preterm birth," *Clinics in Perinatology*, vol. 45, no. 3, pp. 409–420, 2018, https://www.sciencedirect.com/science/article/pii/S0095510818313708.

[13] CMUSphinx, “OPEN. SOURCE,” *SPEECH RECOGNITION TOOLKIT*, https://cmusphinx.github.io/, 2020.

[14] KNX, “Turn every idea into a top project,” 2021, https://www.knx.org/knx-en/for-offices/smart-home-control/.

[15] M. Intelligence, “Open Source S. P. E. E. C. H. Recognition T. O. O. L. K. I. T,” 2021, https://www.mordorintelligence.com/industry-reports/smart-building-market.

[16] Knx, “Smart Building Market - Growth, T. R. E. N. D. S,” *COVID-19 IMPACT, AND FORECASTS*10 pages, 2021, https://www.knx.org/knx-en/for-professionals/newsroom/en/press/KNX-is-the-leading-communication-protocol-in-the-European-Chinese-smart-homes-light-commercial-market/.

[17] K. Association, “Knx basic course,” 2021, http://knx.com.ua/attachments/article/132/KNX-basic_course_full.pdf.

[18] S. Wang, *Intelligent Buildings and Building Automation*, Spon Press, London, United Kingdom, 2010, https://books.google.jo/books?id=BA2LuAAACAAJ.

[19] T. Sauter, S. Soucek, W. Kastner, and D. Dietrich, “The evolution of factory and building automation,” *IEEE Industrial Electronics Magazine*, vol. 5, no. 3, pp. 35–48, 2011.

[20] P. Domingues, P. Carreira, R. Vieira, and W. Kastner, “Building automation systems: concepts and technology review,” *Computer Standards & Interfaces*, vol. 45, pp. 1–12, 2016, https://www.sciencedirect.com/science/article/pii/S0920548915001361.

[21] N. Aste, M. Manfren, and G. Marenzi, “Building Automation and Control Systems and performance optimization: a framework for analysis,” *Renewable and Sustainable Energy Reviews*, vol. 75, pp. 313–330, 2017, https://www.sciencedirect.com/science/article/pii/S1364032116307365.

[22] C. D. Korkas, S. Baldi, and E. B. Kosmatopoulos, “9 - grid-connected microgrids: demand management via distributed control and human-in-the-loop optimization,” in *Advances in Renewable Energies and Power Technologies Elsevier*, pp. 315–344, 2018, https://www.sciencedirect.com/science/article/pii/B9780128131855000255.

[23] S. Baldi, C. D. Korkas, M. Lv, and E. B. Kosmatopoulos, “Automating occupant-building interaction via smart zoning of thermostatic loads: a switched self-tuning approach,” *Applied Energy*, vol. 231, pp. 1246–1258, 2018, https://www.sciencedirect.com/science/article/pii/S0306261918315095.

[24] V. T. Nguyen, T. Luan Vu, N. T. Le, and Y. Min Jang, “An overview of internet of energy (IoE) based building energy management system,” in *Proceedings of the 2018 International Conference on Information and Communication Technology Convergence*, pp. 852–855, ICTC, Jeju, Korea (South), November2018.

[25] R. Carli, G. Cavone, M. Dotoli, N. Epicoco, and P. Scarabaggio, “Model Predictive Control for thermal comfort Optimization in Building Energy Management Systems,” in *Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 2608–2613, Bari, Italy, November2019.

[26] D. G. Hettiarachchi, G. M. A. Jaward, V. P. V. Tharaka, J. M. D. S. Jeewandara, and K. T. M. U. Hemapala, “IoT based building energy management system,” in *Proceedings of the 2021 3rd International Conference on Electrical Engineering*, pp. 69–73, Colombo, Sri Lanka, November 2021.

[27] J. K. Guo, C. L. Lu, J. Y. Chang et al., “Interactive voice-controller applied to home automation,” in *Proceedings of the 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 828–831, Kyoto, Japan, November 2009.

[28] J. Zhu, X. Gao, Y. Yang, H. Li, Z. Ai, and X. Cui, “Developing a Voice Control System for ZigBee-Based home Automation Networks,” in *Procedings of the 2010 2nd IEEE InternationalConference on Network Infrastructure and Digital Content*, pp. 737–741, Beijing, China, September 2010.

[29] V. Jeet, H. S. Dhillon, and S. Bhatia, “Radio frequency home appliance control based on head tracking and voice control for disabled person,” in *Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies*, pp. 559–563, Gwalior, India, April 2015.

[30] S. Ajgaonkar, M. Chauhan, H. Humane, K. Jain, and A. Malhotra, “Voice controlled multi-functional smart device,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, pp. 7634–7638, 2020.

[31] Y. F. Kung, S. W. Liou, G. Z. Qiu, B. C. Zu, Z. H. Wang, and G. J. Jong, “Home monitoring system based internet of things,” in *Proceedings of the 2018 IEEE International Conference on Applied System Invention*, pp. 325–327, ICASI), Taiwan, China, April 2018.

[32] S. Hidayat and S. F. Firmanda, “Scheduler and voice recognition on home automation control system,” in *Proceedings of the 2015 3rd International Conference on Information and Communication Technology*, pp. 150–155, ICoICT, Nusa Dua, Bali, Indonesia, September 2015.

[33] A. Paul, M. Panja, M. Bagchi, N. Das, R. M. Mazumder, and S. Ghosh, “Voice recognition based wireless room automation system,” in *Proceedings of the 2016 International Conference on Intelligent Control Power and Instrumentation*, pp. 84–88, ICICPI, Kolkata, October 2016.

[34] S. Kumar and S. S. Solanki, “Voice and touch control home automation,” in *Proceedings of the 2016 3rd International Conference on Recent Advances in Information Technology*, pp. 495–498, RAIT, Dhanbad, India, March 2016.

[35] K. A. S. V. Rathnayake, S. I. A. P. Diddeniya, W. K. I. L. Wanniarachchi, W. H. K. P. Nanayakkara, and H. N. Gunasinghe, “Voice operated home automation system based on Kinect sensor,” in *Proceedings of the 2016 IEEE International Conference on Information and Automation for Sustainability*, pp. 1–5, ICIAfS, Galle, Sri Lanka, December 2016.

[36] T. Erić, S. Ivanović, S. Milivojša, M. Matić, and N. Smiljković, “Voice control for smart home automation: evaluation of approaches and possible architectures,” in *Proceedings of the 2017 IEEE 7th International Conference on Consumer Electronics - Berlin*, pp. 140–142, ICCE-Berlin, Sepetmber 2017, Berlin, Germany.

[37] I. Krishna and K. Lavanya, “Intelligent home automation system using BitVoicer,” in *Proceedings pf the 2017 11th International Conference on Intelligent Systems and Control*, pp. 14–20, ISCO, Dhaka, Bangladesh, December 2017.

[38] B. Tecnologia and B. V. Server, “Speech recognition and synthesis,” 2020, http://www.bitsophia.com/en-US/BitVoicer/Overview.aspx.

[39] M. Ebrahim Abidi, A. L. Asnawi, N. F. Azmin et al., “Development of voice control and home security for smart home automation,” in *Proceedings of the 2018 7th International Conference on Computer and Communication Engineering*, pp. 1–6, ICCCE, Kuala Lumpur, Malaysia, September 2018.

[40] P. Putthapipat, C. Woralert, and P. Sirinimnuankul, “Speech recognition gateway for home automation on open platform,” in *Proceedings of the 2018 International Conference on Electronics, Information, and Communication*, pp. 1–4, ICEIC, Honolulu, HI, USA, January 2018.

[41] P. Putthapipat and K. Techakittiroj, “PiFrame: a framework for home automation platform on the full feature OS,” in *Proceedings of the n2016 International Conference on Electronics, Information, and Communications*, pp. 1–4, ICEIC, Danang, Vietnam, September 2016.

[42] S. K. Vishwakarma, P. Upadhyaya, B. Kumari, and A. K. Mishra, “Smart energy efficient home automation system using IoT,” in *Procedings of the 2019 4th International Conference on Internet of Things: Smart Innovation and Usages*, pp. 1–4, IoT-SIU, Ghaziabad, India, April 2019.

[43] P. J. Rani, J. Bakthakumar, B. P. Kumaar, U. P. Kumaar, and S. Kumar, “Voice controlled home automation system using natural language processing (NLP) and internet of things (IoT),” in *Proceedings of the 2017 Third International Conference on Science Technology Engineering Management*, pp. 368–373, ICONSTEM, Chennai, India, March2017.

[44] G. B. Karan, D. Kumar, K. Pai, and J. Manikandan, “Design of a phoneme based voice controlled home automation system,” in *Proceedings of the 2017 IEEE International Conference on Consumer Electronics-Asia*, pp. 31–35, ICCE-Asia, Bengaluru, India, October 2017.

[45] S. Milivojša, S. Ivanović, T. Erić, M. Antić, and N. Smiljković, “Implementation of voice control interface for smart home automation system,” in *Proceedings of the 2017 IEEE 7th International Conference on Consumer Electronics - Berlin*, pp. 263-264, ICCE-Berlin, Berlin, Germany, September 2017.

[46] F. Ge and Y. Yan, “Deep neural network based wake-up-word speech recognition with two-stage detection,” in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2761–2765, ICASSP, New Orleans, LA, USA, June 2017.

[47] K. Ai, “Snowboy hotword detection,” 2020, https://github.com/Kitt-AI/snowboy.

[48] P. ai, “Porcupine,” 2020, https://github.com/Picovoice/Porcupine.

[49] P. ai, “Porcupine^{TM} wake word engine - V1.8 feature tour,” 2021, https://picovoice.ai/blog/porcupine-wake-word-engine-v1-8-feature-tour/.

[50] L. Sun, T. Gu, K. Xie, and J. Chen, “Text-independent speaker identification based on deep Gaussian correlation supervector,” *International Journal of Speech Technology*, vol. 22, pp. 449–457, 2019.

[51] R. Milner and T. Hain, “DNN-based Speaker Clustering for Speaker Diarisation,” INTERSPEECH, 2016.

[52] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, “Deep neural networks for small footprint text-dependent speaker verification,” in *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4052–4056, ICASSP, May 2014.

[53] A. Geitgey, “natural language processing is fun,” 2025 pages, 2020, https://medium.com/@ageitgey/natural-language-processing-is-fun-9a0bff37854e.

[54] K. M. Malik, H. Malik, and R. Baumann, “Towards vulnerability analysis of voice-driven interfaces and countermeasures for replay attacks,” in *Proceedings of the 2019 IEEE Conference on Multimedia Information Processing and Retrieval*, pp. 523–528, MIPR, San Jose, CA, USA, April 2019.

[55] Y. Jung, Y. Kim, H. Lim, Y. Choi, and H. Kim, “Spatial pyramid encoding with convex length normalization for text-independent speaker verification,” *Interspeech*, 2019.