IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier xx.xxxx/ACCESS.2018.DOI

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

IVÁN FROIZ-MÍGUEZ[1,2], PAULA FRAGA-LAMAS[1,2], (SENIOR MEMBER, IEEE) AND TIAGO M. FERNÁNDEZ-CARAMÉS[1,2], (SENIOR MEMBER, IEEE)

[1]Department of Computer Engineering, Faculty of Computer Science, Universidade da Coruña, 15071 A Coruña, Spain
[2]Centro de Investigación CITIC, Universidade da Coruña, 15071 A Coruña, Spain

Corresponding author: Paula Fraga-Lamas (e-mail: paula.fraga@udc.es).

This work has been funded by the Xunta de Galicia (by grant ED431C 2020/15), and by grants PID2020-118857RA-100 (ORBALLO) and TED2021-129433A-C22 (HELENE) funded by MCIN/AEI/10.13039/501100011033 and the European Union NextGenerationEU/PRTR.

ABSTRACT

Systems with voice control are an attractive option for increasing technological integration, not only for people with little knowledge on technology or constrained Internet access, but also for people with certain disabilities. In addition, devices based on Alexa or Google Home provide an interesting alternative for interacting with Internet of Things (IoT) devices, but they usually rely on an Internet connection to a cloud server for their full operation. Furthermore, many voice-recognition systems are only available in a limited number of languages, which tend to be those with the highest number of speakers, thus excluding minority-language speakers. To address the previously mentioned issues, this article presents a solution based on Edge Computing and voice commands that carries out offline voice processing and that is able to interact with IoT-based systems. The proposed system performs local speech inference, providing a communication interface with IoT devices in a Bluetooth mesh, all in a fast way and without the need for an Internet connection. In addition, the proposed solution can be adapted easily for voice recognition of languages with few resources. Such a feature is demonstrated with the Galician language, which is spoken by less than 3 million people worldwide. In particular, different Automatic Speech Recognition (ASR) models based on three of the most popular ASR development frameworks (wav2vec2, DistilHubert, Whisper) were developed to transcribe short speech and to translate it into IoT commands that perform specific home-automation actions. Such models were fine-tuned for Galician with a corpus of approximately 20 hours and were evaluated in static and mobile opportunistic scenarios in terms of accuracy, energy consumption and latency on an embedded platform (that acts as an edge device) and on a cloud server. The obtained results show that inference is performed in less than 2 seconds on a Raspberry Pi 4 for the two smallest models and in less than 500 ms on a high-end Android smartphone when processing all data locally with CPU-only inference (i.e., without hardware acceleration or external processing). The results of the transcriptions are accurate enough to be able to use simple text distance algorithms to detect keywords in the speech and perform commands on IoT devices. In particular, a maximum success rate of 92% was achieved for detecting the indicated commands when using models optimized for being executed on embedded devices. For selected home scenarios, command actions were sent via Bluetooth with average response times of up to 113 ms.

INDEX TERMS ASR, Machine Learning, IoT, voice-assistant, Edge AI, Edge Computing, Home Automation, Opportunistic Communications.

I. INTRODUCTION

With the emergence of the Internet of Things (IoT), more aspects of everyday life are becoming interconnected and digitalized, providing people more comfort and greater control over their daily activities. However, most of such technological progress is only available for specific groups of society with some technological background. A large part of the population in developing countries, as well as the elderly, are usually not able to use traditional computing devices such as computers, smartphones or tablets. In this regard, voice recognition based systems are a good enabler for providing access to IoT technologies to a high percentage of the society.

The mentioned technological adoption, especially in the case of the elderly, is also beneficial for the whole society. The current trend indicates that, on average, people live longer [1], and, for people with partial loss of autonomy or at risk of losing it, the access to IoT technologies is able to improve their independence and to provide a better quality of life, avoiding premature admission to residential care homes with the associated healthcare costs. However, nowadays, unfortunately, many IoT solutions provide complex user interfaces to manage them. In this regard, voice-based interfaces are an intuitive and an accessible option that allows for a convenient and simple way to interact with IoT embedded systems.

The implementation of voice-recognition systems still presents certain challenges [2] [3]. In the last years Automatic Speech Recognition (ASR) systems have increased their presence in IoT-based Home Automation (HA) systems [4]. Systems such as Alexa or Google Home, although they are a reference in the HA market, depend on a constant Internet connection for their full operation. However, it is common for certain population groups like the elderly to have reduced or no Internet connection. In addition, many voice-recognition systems are always listening and the speech is sent without encryption, which represents a privacy risk [5] [6].

The main reason to process voice in a remote cloud is that audio transcription mechanisms are computationally expensive [7]. However, with the improvement in computational efficiency of embedded devices, and by reducing the transcription domain and using short voice commands, it is possible to obtain good results by doing the processing locally even though the achieved accuracy is lower than in a cloud [8].

Another problem is language availability in ASR systems. Traditionally, such systems need thousands of hours of labelled audio to train a model that delivers good transcription results [9]. Moreover, people usually do not memorize specific commands and tend to use small variations that are more natural to them instead of memorizing specific phrases [10].

There are several approaches to overcome the previously mentioned two drawbacks. On the one hand, in 2020 a framework for ASR was developed to allow transcription results to be obtained with near-human precision with much less transcription hours than the previous systems: wav2vec2 [11]. Keeping in mind that nearly 7,000 languages exist and that the vast majority of them has few labelled data or any of them available [12], wav2vec2 approach allows for expanding voice-recognition technology and thus making it more accessible. On the other hand, in order to overcome the limitation of imposing a specific grammar for communicating with a voice assistant, it is necessary to define a language grammar, since a literal transcription of the voice is actually not relevant (i.e., what is important is actually the meaning of the command, not the fact of saying specific words in a specific order). Thus, sets of keywords can be defined to build a grammar around them. In addition, it should be considered that a local or edge implementation requires reducing the language domain, which involves creating a model with a lower precision than the one that would be used in a cloud server.

This article presents a system that considers all the previously mentioned issues and tackles them by implementing a voice recognition system developed for the Galician language. The proposed system can be run without relying on an Internet connection and is able to execute specific commands on IoT devices. Galician has been chosen because it currently has few voice-recognition resources but it is used as a vehicular language by a high percentage of the elderly that live in Galicia (northwest of Spain), specifically in rural areas, where there is a clear lack of access to new technologies and where poor communications still prevail. Moreover, Galician population is immersed in an aging process that affects all Europe [13]. The average age continues to increase every year, with a total population of less than 2.7 million inhabitants, currently standing at 47.74 years [14]. The demographic situation in Galicia also presents a significant gap between a more urban and dynamic region and a more rural one where the aging process is even more aggravated and seems to lead to a depopulation also partly caused by the lack of economic incentives and services in these areas [15].

The following are the main contributions of this article:

- It describes a novel home IoT system based on low-cost embedded devices that makes use of Edge Computing and an optimized speech-recognition system for low-resource languages. Moreover, since IoT device communications are carried out through a Bluetooth mesh, it is not necessary to deploy any home infrastructure connected to the Internet.
- It includes a thorough analysis of the main Machine Learning (ML) models in the field of ASR to be executed on edge devices.
- It proposes different optimization techniques to run ML models on embedded devices.
- It presents a detailed evaluation of the developed IoT system in terms of response time, accuracy and energy consumption.

The remainder of this article is structured as follows. Section II reviews the state of the art of voice assistants

for HAS and healthcare, and continues with an analysis of the improvements provided by Natural Language Processing (NLP) in such domains. Moreover, Section II analyzes the suitability of using ML in Edge Computing devices, as well as the use of distributed communications for Edge Computing and the benefits they bring over traditional client/server architectures. Section III details the design and implementation of the proposed voice-assisted IoT system. In such a Section, an exhaustive analysis of the main HA technologies is carried out by focusing on the proposed use case, as well as on the main speech recognition development frameworks. Next, Section IV presents the results for a set of experiments performed for the two main subsystems (i.e., for the voice processing and the communications subsystem). Finally, Sections V and VI summarize the key findings and the most relevant conclusions.

II. STATE OF THE ART

A. VOICE-BASED SOLUTIONS FOR HAS

There are numerous papers and proposed solutions on the subject of voice interfaces focused on HA environments [16] [17] [18]. Many of such papers address the problem of running voice-based solutions on a server [16], others offer an edge or offline solution that provides faster response times and privacy [16] [18], while others include ML strategies for processing voice-related tasks [16] [17] [18].

The most accurate speech recognition models usually use billions of input parameters [19], which makes them computationally expensive and not suitable for IoT devices with power and processing constraints. However, a literal transcription is not necessary for the case of voice assistant systems [16] [17] [18]. For instance, a solution could consist in reducing the domain of the language and thus only detect certain keywords [16] [17] [18]. In this way, the end user can make use of a more natural language instead of learning exactly a specific set of commands, thus detecting specific keywords and building a grammar around these words. For instance, the SparkFun Edge [20] is a real-time audio analysis device that runs ML inference to detect keywords and responds accordingly, constituting a good example of voice recognition at the edge.

The deployment of ML models on all IoT end-devices is not adequate, since many of them are specific-task and low-power oriented. For such a case, an effective alternative would consist in placing additional computing devices at the edge of the network with enough computation to be Artificial Intelligence (AI) capable [21]. Such a strategy is known as Edge Intelligence (EI) or Edge Artificial Intelligence (Edge AI) [22]. The communication with such devices is much more immediate than making requests to a cloud, it considerably reduces Internet traffic, and, if the network is secured, there are no privacy issues when communicating within a private network.

B. VOICE-BASED SOLUTIONS FOR HEALTHCARE AND FOR THE ELDERLY

Advances in technology in recent years have enabled an improvement in healthcare towards a more specialized model focused on individual patient management. Moreover, the COVID-19 pandemic has forced the adoption of certain aspects of telemedicine [23]. However, accessing and using such novel technologies is often not straightforward for the elderly or for people with certain disabilities: it is challenging for them to adopt, to learn, and to interact with such tools that can normally only be accessed through modern Graphical User Interfaces (GUIs). Fortunately, current voice-based assistants allow users to interact with digital systems in a more natural way, which is especially useful for the elderly [17], [24], [25]. Some studies on the use of voice-based assistant solutions among elder population resulted on a preference to use voice-based interfaces rather than GUI-based touch screen inputs or keyboards [26]. Likewise, most of the tasks performed by the elderly are related to health or medical questions [26].

Voice-based assistants, when used in combination with the latest advances in technology, provide an improvement in the quality of healthcare services, especially in rural or remote areas where services are minimal [27]. This kind of applications normally rely on Edge Computing to deal with the critical response time of medical devices and data privacy [28].

In terms of voice-based applications, the healthcare sector is one with the fastest growing rate [29]. The development of voice-based healthcare applications is emerging together with telemedicine and advances in voice processing systems to provide high quality monitoring care and avoid excessive readmission rates [30]. Combined with IoT devices, voice-based applications provide smart systems with speed-up diagnosis, offer more accurate treatment and anticipate potential symptoms through data collection and processing thanks to the application of ML techniques for diagnostic monitoring [31], all leading to a better quality of life.

C. NATURAL LANGUAGE PROCESSING FOR VOICE-BASED SYSTEMS

When talking about voice assistants interfaces, another important issue to address is the reply offered by the system in terms of language semantic. For such a purpose, most virtual assistants are built to automatically answer questions performed by humans in a natural language, which are transformed into an internal representation. Then, the system extracts the essential information and delivers a response that considers the context.

Chatbots are an example of systems usually created with this kind of NLP technologies, which enable users to communicate in a natural and easy way [27], [32]. Most advanced speech-processing applications, instead of treating speech or text just as a sequence of symbols or by just looking for keywords inside the user question, they use pre-programmed

or acquired knowledge to decode meaning from factors such as sentence structure, context or idioms [33] [34] [35].

There are also voice assistants that use speech recognition and synthesis in addition to text-processing strategies based on NLP. Thus, if an embedded ASR model is used, the limitation of loss of transcription accuracy can be overcome, as this is not a determining factor for NLP mechanisms. The current state of the art for task-oriented semantic models use Bidirectional Encoder Representations from Transformers (BERT) [36] or Robustly Optimized BERT Pretraining Approach (RoBERTa) [37] as a pre-trained input. However, such models make use of huge memory footprints and are not suitable for embedded devices.

Luckily, there is a reduced version of Bert called DistilBert [38]. Bert based models store knowledge into models with millions of parameters. DistilBert uses a smaller general-purpose language representation pre-trained model that is later fine-tuned, which reduces the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster. For the fine-tuned part, there are numerous tasks with which a Bert-based model can be trained (e.g., question-answering, sentiment analysis) with the help of a training corpus. For instance, Fluent Speech Commands [39] is a dataset of people that interact with a smart home system in English spoken language. Such a dataset is adequate for developing voice-based assistants and, together with the use of BERT models as a transformer language model, it is possible to distil knowledge to build an acoustic model for intent classification [40].

All of the above is particularly interesting for edge devices. For instance, Lanyu Xu et al. [18] provide an ASR and a DistilBert model that operate on the edge. By employing a cache that reduces connections to the cloud and by pruning a pre-trained language model for using it in resource-constrained edge devices, latency and accuracy demands for efficient resource utilization are fulfilled. There are a few models based on Bert multilingual and monolingual to fine-tune a model for downstream grammar tasks with Galician (e.g., bert-base-multilingual-cased [41], bert-base-gl-cased [42], bertinho [43]). However, none of them are finetuned for detecting relevant tasks for the proposed system, such as question-answering, HA commands or spelling check (i.e., for the proposed system, the lack of a task-oriented corpus is a concern that needs to be addressed).

Also wav2vec2 has been ported and tested to be executed on the edge [8] [44]. In such a scenario the accuracy of the quantized model is worse than the one of the original model, but the used memory can be reduced considerably at the cost of a small increase in error. This type of model is best suited for systems that have to translate small segments of speech to commands where even small errors in the transcription do not impact significantly the performance of the system. It is worth noting that long transcriptions are not suitable for the embedded model due to the requirement of using large amounts of memory.

D. EDGE INTELLIGENCE AND GREEN IOT

The combination of Edge Computing with Machine Learning allows for collecting data from IoT-based devices and process them locally while making use of AI techniques, providing what is known as Edge Intelligence. Thus, Edge Intelligence enables to deliver an initial data analysis that avoids network-related problems that are usually present in cloud-based architectures [45]. In particular, edge processing allows for providing fast responses off the system, which is essential for voice-recognition based solutions, since users tend to quit if they do not receive an immediate response (due to the time required for processing the sent audio streams, the response of the cloud may not be immediate, especially in cases when the Internet connection is not reliable).

Regarding embedded architectures, they have experienced a major progress in the last years, considerably increasing their computational power, becoming more economical and improving their energy efficiency [46] [47]. Such embedded architectures are widely used in edge devices that perform tasks that were formerly performed exclusively by the cloud and which now can alleviate the workload of the growing IoT device ecosystem [48].

This trend of progressive replacement of cloud server-performed tasks by Edge Computing devices is also aligned with the objectives of the Green IoT (G-IoT) paradigm [49]. However, the use of ML in this kind of devices can consume a significant amount of energy [50]. While architectures like BERT or Generative Pretrained Transformer (GPT) [51] involve a significant consumption footprint, in recent years similar architectures have risen but with a much lower resource consumption [52] [53]. Also, the paradigm of AI acceleration through application-specific integrated circuits (ASICs) is emerging, providing much better performance for specific tasks than standard Graphical Processing Units (GPUs) [54]. For instance, Google has an ASIC-specific version for TensorFlow (Tensor Processing Unit (TPU) [55]). Moreover, Google provides TPU hardware specifically designed to run on the Edge (Coral-AI [56]).

Although ML inference, when performed on embedded devices, tends to be more efficient, the training of models remains computationally intensive [57]. Many frameworks provide several pre-trained models that can be fine-tuned to perform specific tasks, which need much less computational power but that remain being a heavy task [58]. There are also training frameworks that provide distributed training, but data movement between devices needs to be properly managed, or may otherwise become a performance bottleneck [59]. Thus, a properly defined parallelism strategy is needed in order to schedule computational tasks on inter-connected devices, thus minimizing communications cost, maximizing the use of computational and memory resources, and optimizing the computation-communication overlap for Large-Scale Training [60].

E. VOICE-BASED EDGE COMPUTING SYSTEMS

The Edge Computing paradigm can enable distributed communications, which has certain benefits for an IoT system, since it establishes a distributed topology that is ideal for environments with communications difficulties [61]. Such difficulties do not only include the fact of having a limited Internet access, but also the issues related to the implementation of star topologies, whose use is not efficient in environments where, to establish a direct connection of all the nodes with the central node, it is necessary to deploy numerous access points. In contrast, distributed topologies are able to overcome this limitation by allowing the different nodes of the network to forward the exchanged information.

Within a distributed network there may be different roles for the nodes. Some of them are oriented towards low-power consumption (mainly sensors and actuators), rely on batteries and perform specific actions periodically and then go into sleep or power-down mode. Other nodes operate as relays, thus forwarding data traffic among them and to the different low-power nodes of the network. In this latter case, relay nodes usually need a constant supply of energy to not only forward information, but to be also able to perform in a distributed way other tasks that would be performed by the central node in a star architecture.

This distributed strategy is not the typical approach followed by most voice-based assistants, which generally make use of a centralized star architecture deployed in a cloud or in a local central device that manages communications and that relays Internet-related tasks (in the case of locally deployed devices). This kind of solutions commonly rely on several smart speakers placed throughout a home or a building [24]. However, this approach implies a higher infrastructure cost, among other problems [62]. Moreover, even if the task of sending the audio to the central device is performed on a smart speaker that is placed in the room where the user is located, all the speakers are always active and listening, which is not energy efficient. A more efficient approach would consist in having only one device listening and always accompanying the user. For instance, smartphones are usually already carried by many people, but there are other portable devices with a minimum of computing capabilities, such as smart watches, smart bands or other types of wearable devices that are becoming smarter thanks to the progress made on embedded hardware and AI.

Finally, it is worth mentioning that some of the most popular IoT technologies for HA (e.g., Bluetooth, ZigBee, Thread, Ant+) already make use of a distributed mesh topology that includes Low-Power Nodes (LPN) that are asleep most of the time and relay nodes (among others roles). These intermediate relay nodes can manage a cache for LPN nodes with information to be sent to specific nodes when they wake up [63]. The problem is that this behaviour is intended for static nodes: if an LPN node wakes up and is in range with another relay node, the cache that was stored by the first relay node will be lost.

III. DESIGN AND IMPLEMENTATION

A. MAIN COMPONENTS OF THE PROPOSED SYSTEM

When it comes to designing a voice-based HA system, it is important to distinguish two parts:

- Voice-processing. It involves the initial preprocessing of the audio, thus generating the inference related to the ML model and detecting the specific commands from the speech. In the system described in this article, all voice processing will be performed exclusively by the Voice Assistant Device (VAD). Such a device has the following characteristics:
  - It operates on the edge without needing to connect to a server.
  - It is portable, since it is carried continuously by every user.
  - It is able to interact with the different deployed IoT devices by using voice commands.
  - It can also perform other types of general tasks that require connectivity (i.e., to search for specific information on the Internet, to retrieve news or the weather forecast, or to make phone calls).
- System communications. This part is in charge of managing the information exchanged between the deployed IoT sensors and the VAD.

Figure 1 shows the general architecture of the proposed system. Specifically, the different layers of the developed system are depicted:

- The IoT device layer includes the different sensors/actuators nodes of the system. It is worth noting that the layer includes nodes with constant energy supply that need fast response times while others do not have critical time-response requirements or have power supply restrictions (thus prioritizing energy saving over response time, becoming Low Power-consumption Nodes (LPN)).
- The gateway layer includes the different intermediate relay nodes, which provide the desired communications coverage to a house or building.
- The VAD layer essentially includes the voice assistant node. Since such a node runs on batteries, it is considered as an LPN node. The VAD will interact with the nodes of the gateway and IoT device layers to perform voice-induced actions. If necessary, it can also perform actions that involve using an Internet connection (e.g., check the weather, receive news, collect the user e-mail) or making calls through a cellular network, but, nonetheless, it must be noted that the presented architecture has been conceived and designed to be deployed in a house without any infrastructure with Internet access.

B. COMMUNICATIONS TECHNOLOGY SELECTION

Among the different wireless technologies that can be used for the proposed system, the main options on the market

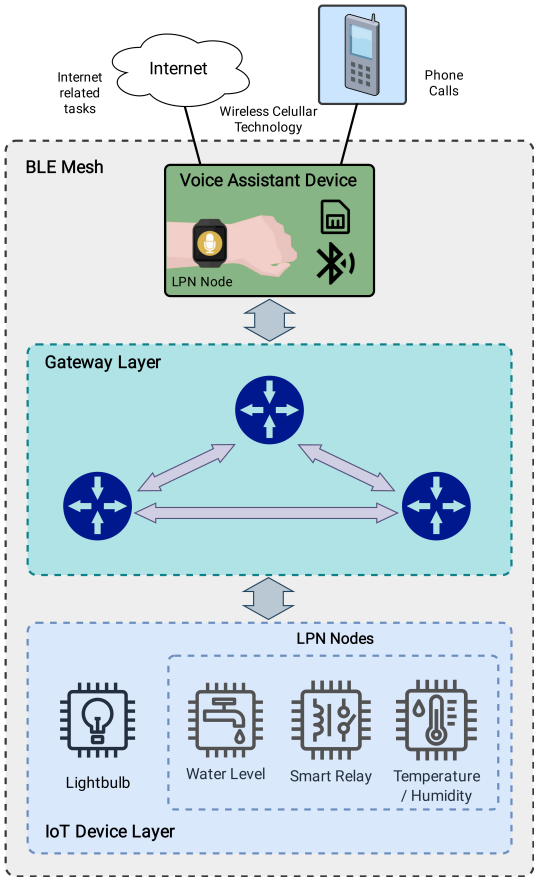


FIGURE 1: Main architecture.

were analyzed in terms of standardization body, operating frequency, maximum range, maximum data rate, modulation scheme, encryption, topology, latency, battery lifetime and cost. Table 1 analyzes such characteristics with the objective of allowing them to be compared among the main communications technologies.

a: Standardization Body

This feature is ideal for maintaining compatibility among different manufacturers. Most communications technologies are regulated by standards, companies or alliances that perform certification tests to ensure that they satisfy their requirements.

Some standards are proprietary while others are open, or even shared between various technologies such as IEEE 802.15.4. In this respect, open standards used in various technologies will lead to a higher level of interoperability, while using proprietary standards can sometimes lead to problems of compatibility as is the case of Insteon: with the closure of its services in April 2022, it left all its products out of operation [64].

b: Operating Frequency

This factor has a direct impact on several aspects of wireless communications such as transmission times, bandwidth, range and propagation. Most Wireless Personal Area Network (WPAN) technologies employ frequencies in the 2.4 GHz ISM band. This band do not have as much range as sub-GHz bands like those used in Low-Power Wide-Area Networks (LPWANs), but their data transmission rate is much higher than the latter, enabling its use in fast transmissions applications (moreover, since HA environments do not need to cover large communications distances, it is not advantageous to use LPWAN technologies).

Since the 2.4 GHz band is widely used, it can present saturation problems, especially in urban environments. In addition, its propagation through objects is worse than the one achieved with other technologies that operate in sub-GHz bands, such as ZigBee, Z-Wave or EnOcean, which usually provide an easy full coverage of a house. Other technologies such as WiFi or WiFi 6 can operate at frequencies above 2.4 GHz, offering a much wider bandwidth but with a shorter range and higher energy consumption. Nonetheless, for the proposed system, only short messages are involved, so a high bandwidth is not necessary.

c: Maximum Range

Range may vary considerably from a few metres (with short-range networks like BLE) to several kilometres (as with LPWAN technologies). Short-range networks allow data streams to be transmitted in real time with reduced energy consumption, while long-range networks are designed for punctual short-size transmissions, as the time on air of the messages is much higher. For instance, BLE is widely used in wearables or sensors for monitoring body constants due to its efficiency in short-range transmission streams.

For the system proposed in this article, the transmission of constant data streams is not required, but neither is it necessary to have a very large range that would considerably reduce data transmission. As a consequence, an intermediate range is enough. Luckily, most WPAN technologies are characterized as medium-range wireless technologies.

d: Modulation Scheme

Communications reliability depends mainly on the modulation together with coding rates that allow the creation of extra error checking bits. By creating a group of modulation and corresponding coding schemes (which is known as Modulation and Coding Scheme (MCS)), it is possible to adjust both parameters in order to cover larger distances. The more reliable options also have the slower transmissions (e.g., Binary Phase Shift Keying (BPSK) transmits only one bit per burst, which makes it an extremely slow but highly reliable modulation).

For instance, in Bluetooth 5 it is possible to use a Long Range modulation with two different coding schemes (S2 and S8) that allow the creation of extra bits for error correction. With respect to the the legacy modulation, S2 can decode a message at twice the distance, while with S8 four times the distance can be reached. This is achieved at the cost of lowering the bit rate to 500 and 125 Kbps, respectively, for

TABLE 1: Most relevant characteristics of the latest protocols and communications technologies for IoT in the HA market.

Technology	Standardization Body	Operating Frequency	Maximum Range	Max. Data Rate	Modulation Scheme	Encryption	Topology	Latency	Battery Lifetime	Cost
BLE (Bluetooth 4.2)	IEEE 802.15.1 (inactive)	2.4 GHz	>100 m	250 kbit/s, 1 Mbps	GFSK	E0 stream cipher AES-128, authentication: shared secret, data protection: 16-bit CRC	Point-to-point, piconet (scatternet), star, mesh	Around 3 ms (less than 10 ms)	Several years on a single coin-cell battery	Low
WirelessHART	Proprietary, IEEE 802.15.4 physical layer, IEC 62591:2016	2.4 GHz	<10 m	250 kbit/s	OQPSK	AES-128	Star, mesh	–	Several years	Expensive gateways
ZigBee	IEEE 802.15.4 (layer 1 and 2), Connectivity Standards Alliance	868-915 MHz, 2.4 GHz	30 m (indoor) and <100 m (outdoor) for 2.4 GHz, up to hundreds of meters outdoors for 868-915 MHz	20−250 kbit/s	BPSK (+ASK), OQPSK	AES-128, authentication: CBC-MAC (CCM), data protection: 16-bit CRC, error control/reliability: ACK, CSMA/CA	Star, cluster tree, mesh	Around 15 ms	Very low power consumption, 100-500μW, batteries last months to years	Low (gateways moderate)
Z-Wave	Z-Wave Alliance, proprietary	868-915 MHz	30 m (indoor), 100 m (outdoor)	40-200 kbit/s	FSK/GFSK	AES-128, data protection: 8-bit CRC, error control/reliability: ACK, CSMA/CA	Mesh	1 s	Alkaline batteries last months to years	Low
ANT+	Proprietary (ANT+ Alliance)	2.4 GHz (1 MHz channel bandwidth)	30 m	20 kbit/s (burst), 60 kbit/s (advanced burst)	GFSK	AES-128 and 64-bit key	Point-to-point, star, tree, mesh	–	Around one year	Moderate
Wi-Fi	IEEE 802.11b/g/n/ac	2.4–5 GHz (22 MHz channel bandwidth)	<150 m	Up to 600 Mbit/s	QPSK	AES block cipher, authentication: WPA2 (802.11i) and WPA3, data protection: 32-bit CRC	BSS, ESS	Less than 20 ms	High power consumption, 500 mW - 1 W (batteries usually last hours)	Moderate
Wi-Fi 6	IEEE 802.11ax	Between 1 and 6 GHz (2.5 and 5 GHz), 20 to 160 MHz channel bandwidth, 10 Gbit/s	<3 km	Around 1,200 Mbit/s	BSK to 1024 QAM	WPA 3	BSS, ESS	Around 10 ms	–	-
Insteon	Proprietary	902-924 MHz	50 m (outdoor)	13,165 Kbit/s	FSK	AES-256, rolling codes, publick key, error/control/reliability: 8-bit checksums	Full mesh	–	Several years	Moderate
EIB/KNX RF	KNX Association, ISO/IEC 14543-3	868 MHz and 2.4 GHz	150 m	16.4 kbit/s	FSK	AES-CCM	Line, tree and star	–	Several years	Relatively expensive transceivers
EnOcean	EnOcean Alliance, ISO/IEC 14543-3-1X, proprietary	868 MHz, 902 MHz and 928 MHz (280 KHz channel bandwidth)	30 m (indoors), 300 m (outdoors)	120 kbit/s	ASK	AES-CBC and AES-CTR	Mesh	40 ms (typical for transmitting three identical radio telegrams)	Very low consumption or battery-less thanks to using energy harvesting	Moderate
Thread	IEEE 802.15.4, Thread Group, OpenThread (open-source implementation by Google	2.4 GHz band, with a roadmap to sub-GHz bands	Up to 200 m	250 kbit/s	IEEE 802.15.4 modulations	AES-128. Commissioning uses standard DTLS with ECJ-PAKE	Mesh	–	Several years	-
NB-IoT	3GPP	700, 800 and 900 MHz (200 kHz carrier bandwidth)	10-15 km	200 kbit/s (typically 100 kbit/s)	BPSK, QPSK (OFDMA Half-duplex)	Yes (defined by 3GPP)	Star	<10 s	>10 years with a battery capacity of 5 Wh	High, SIM needed
Wi-Fi HaLow	IEEE 802.11ah	License-exempt bands around 900 MHz (20 to 160 MHz channel bandwidth)	<1 km	100 Kbit/s per channel, up to 347 Mbit/s	BSK to 256 QAM	WPA-3	Star-bus	100 ms (typical beacon interval)	Power consumption of 1 mW	-
Bluetooth 5	Bluetooth Special Interest Group (SIG)	2.4 GHz (2400-2483.5 MHz)	<400 m up to 1.2 km in Long Range mode at 0 dBm m	Up to 2 Mbit/s	GFSK	AES-CCM	Star-bus, mesh	<3 ms	Power consumption 1-20 mW	Low

each coding scheme. Apart from the range, this improves the link quality and gives a higher sensitivity to the RF module, which is especially interesting to provide coverage in large buildings or with considerable shielding.

e: Encryption

Most of the analyzed technologies provide some sort of encryption mechanisms, but it is important to mention that most devices that implement such technologies are resource constrained while traditional encryption mechanisms are computationally expensive [65]. One of the most used encryption mechanisms is Advanced Encryption Standard (AES), which is a symmetric algorithm that provides different key lengths. According to the National Institute of Standards and Technology (NIST), 128 bit is still secure for use until 2030 [66]. However, the progress of quantum computers might significantly reduce the strength of algorithms not specifically hardened against quantum computing [66]–[68].

f: Topology

This parameter primarily affects network architecture and deployment. For example, the use of a star topology like in NB-IoT (or in others based on 3GPP) implies that all nodes must be in direct communication with a central server. Considering that the system proposed in this article is intended to be used in rural areas with poor coverage, a centralized topology is not adequate.

Although it is possible to use centralized gateways that work on Edge or Fog Computing networks [69]–[71], this makes the product more expensive. In this respect, the best option is to deploy a distributed system with a mesh topology that allows to broadcast communications among nodes, thus overcoming range limitations. Most WPAN technologies enable this type of topology. However, distributed architectures

need certain considerations in their communications algorithms, since by nature they are more complex than traditional client-server communications. In this regard, technologies such as Thread, ZigBee or Ant+ are widely used in the HA market and make use of algorithms with routing tables. Thread even implements IPv6 connectivity in its upper layer, which is really interesting for IoT, specially in massively interconnected networks.

On the contrary, BLE is able to implement a mesh topology known as BLE Mesh that does not use routing tables but makes use of a controlled flooding algorithm that broadcasts messages throughout the network. This algorithm is simple and very efficient, especially in the case of one-to-many or many-to-many communication. Nonetheless, to avoid saturation problems, the deployed BLE Mesh networks must have minimal hops and deliver very short messages.

Since the system proposed in this article looks for controlling IoT actuators with basic commands (e.g., on/off commands) and for receiving information from sensors (in general, numerical values) in networks with a limited number of nodes, it is not necessary to make use of routing table algorithms.

Finally, it is worth noting that, in the same way that Thread provides support for IPv6, BLE can also be used through 6loWPAN, but since the environments contemplated in this article have limited connectivity, it has not been considered as a relevant factor in the design (moreover, IPv6 is still not widely implemented nowadays [72]).

g: Latency

Similarly to range or data rate, the technologies that allow for covering the longest distances with reduced transmissions also present the largest latencies. In this aspect, most of the WPAN technologies enable to develop applications with low latencies, even in real time. For the scenarios proposed in this article, it is important to note that latency is relevant, since there are cases in which an immediate response is essential (for example, lighting controls are expected to have a quick response).

h: Battery Lifetime

There are numerous factors that directly impact energy consumption, such as transmission power, data rate, modulation (more time on air implies more consumption) or topology (in multi-hop or mesh networks, consumption is higher when data forwarding is required), so it is necessary to consider all these factors to obtain a good balance between consumption and performance.

Although certain devices can be powered through the electrical grid, in the case of HA it is common to use devices with batteries for a simple deployment of the hardware involved. Since sensors and actuators usually have reduced dimensions, so is the battery that they incorporate.

Some technologies like EnOcean make use of energy harvesting mechanisms, while in other cases these mechanisms are not enough for the operation of the node, so it is important to minimize as much as possible the consumption of those that operate with batteries to maximize the lifespan and minimize charging times.

Battery-powered devices are normally used for tasks that do not require much data updating and consumption can be minimized by putting the modules in a deep sleep state. This feature is commonly used in commercial hardware, but it is also important to consider the consumption of the transmissions. For example, WiFi has a high consumption related to its transmissions and therefore it is not usual to make use of such a technology in low-power devices that operate with batteries.

i: Cost

It is necessary to consider different types of costs. First, the cost associated with the hardware. For instance, BLE modules are widely present in the market and their cost is low. Second, the cost of the data transmissions, since certain technologies like NB-IoT involve monthly fees. Third, it is also important to consider the cost of the required infrastructure (e.g., technologies like ZigBee make use of relatively cheap transceiver modules, but its gateways have moderate prices).

j: Final choice

Considering all the aspects analyzed in the previous subsections, it was decided to use Bluetooth 5. The main reason was the fact that this new version of the standard has better range, latency, bandwidth and coexistence than its predecessor, despite that it is not yet as widespread. Moreover, Bluetooth 5 is backward compatible, allowing the coexistence with other Bluetooth versions. Furthermore, the Bluetooth standard is open and widely documented, which enables more accurate and specific developments. In addition, the use of a mesh topology with BLE suits the proposed system (which requires a reduced network with simple messages without many hops), is energetically efficient and provides adequate response times.

C. BLE MESH COMMUNICATIONS

Internal communications are performed with BLE Mesh. Such communications are based on topics: performing an action means publishing a certain value in a specific topic, and consulting the status of a device means subscribing to a specific topic. There are nodes connected to a continuous power supply that require a fast response (such as lights), where there are also LPN nodes that operate with batteries or limited power supplies that need to be in periodic sleep cycles (the information addressed to these nodes is stored in intermediate nodes).

The BLE Mesh standard includes a feature called "friend node" that allows for storing information for the LPN nodes. However, friend nodes do not act as a distributed cache: when an LPN node loses its connection to the associated friend node, its cache is lost. This is not a problem for most LPNs, since they are static. Occasionally, in very specific cases, they

may lose the connection to their friend node, in which case they will look for another one that is in range to store the cache.

However, the VAD node, which is not an LPN node in the strict sense of the term, as it is always listening for commands: from the point of view of the Bluetooth subsystem, it works as an LPN node, sleeping most of the time and waking up occasionally to consult the cache or when a voice command is provided. To overcome this limitation, in the developed system, when a LPN node needs to send a message to the VAD node, it sends it to all the friend nodes. Thus, when the VAD connects to any node, it will read the message and the rest of the duplicate messages will be discarded.

D. COMMUNICATIONS SUBSYSTEM OPTIMIZATION

The main responsibility of the communications subsystem is to provide communications to the end device (i.e, to the device or devices involved in the voice command recognition process) to perform the operations indicated via voice commands.

Equation 1 indicates the factors involved in the calculation of the total latency of the system. As it can be observed, it is calculated as the processing time of the ASR model to generate the transcript plus the time to perform the action. Equation 2 shows the different partial times involved in the calculation of the time to perform an action, which is composed by the time that is needed to detect the particular command once the transcript is generated plus the time required to send it through the communications subsystem and the time consumed by the destination node to receive it and to execute the action.

$$t_{total} = t_{transcription} + t_{action} \tag{1}$$

$$t_{action} = t_{detection\_comm} + t_{sending} + t_{execution} \tag{2}$$

To minimize t_{action} it is necessary to consider that BLE Mesh communications were provided by Nordic boards based on a nrf52xxx System-on-Chip (SoC) [73]. BLE is widely supported by many IoT devices on the market and is especially efficient for those devices present in HA environments [74]. Nordic's SoC not only supports BLE, but also ZigBee, Ant+ and Thread, which are some of the most widely used technologies for HA. Moreover, Nordic's boards support diverse Bluetooth 5 modulations and extended advertisements that, although they are not officially supported by BLE Mesh, they can be used to obtain certain benefits over the Bluetooth legacy version (part of such benefits have been previously described in [75] [76]).

In this article, generic BLE Mesh models were used to communicate with sensors and actuators [77]. In particular, the generic On/Off and generic sensor models were used to represent actuators and sensors, respectively. Such BLE Mesh standard models are optimized to carry out communications with minimum overhead: the forwarded frames have a reduced size and the forwarding algorithm is a controlled flooding without routing algorithms, thus being very efficient especially for many-to-many and one-to-many communications [78]. However, to avoid system performance degradation, the following considerations need to be taken into account:

- The length of the system messages has to be reduced: if such messages are longer than 15 bytes, the frame would need to be segmented, which implies that each segment needs an ACK response and then all the segments need to be reassembled. Such a segmentation degrades the performance considerably. Luckily, the frames used for the selected models are very small (they transmit either an 'on/off' code or a specific numerical value).
- The network topology has to be simple: if there are too many nodes and hops, the controlled flooding algorithm is no longer efficient and the network can become saturated [79]. The only control mechanism that the algorithm contemplates is the use of a Time To Live (TTL) value to discard network packets. For the use case analyzed later in Section IV-B, a domestic network scenario was selected, where the number of nodes is usually small and with a few hops, so no issues should arise in relation to the network topology.

As it was previously described, the behaviour of the VAD node differs from the way standard BLE mesh nodes operate. Such a behaviour is related to the fact that the VAD node acts as an opportunistic node [80] that will move throughout space and time, so it will be associated to different friend nodes (which behave as described in Section III-C) [81]. This fact implies that LPN nodes that have to send information addressed to the VAD node will have to send it to all the available friend nodes (or at least to a subset of them that have a high likelihood of being in the range of the VAD node). Then, the VAD node will read the cache of the friend node to which it is associated at a specific moment and the cache addressed to the VAD of the rest of the friend nodes will be discarded. This is a very simple approach that implies redundancy and a higher data flow through the mesh, as there are no routing tables, and it is justified by the following assumptions:

- The described behaviour is exclusive for information addressed to the VAD node. The rest of the LPN nodes operate following the standard BLE Mesh functionality, which will be associated to a single friend node as a general rule.
- The messages addressed to the VAD node are not critical, so a fast response time is not essential. In fact, LPNs are designed to operate with batteries or with constrained power sources, which is why the definition of sleeping periods is a good practice to increase battery life.

E. SPEECH-RECOGNITION FRAMEWORK SELECTION

Speech-recognition frameworks have evolved remarkably in the last years. In 2011, Daniel Povey et al. [82] introduced a speech recognition toolkit called Kaldi that quickly became the ASR tool of choice for countless developers and researchers. Since Kaldi's introduction, several speech recognition frameworks have been released: wav2letter++, openseq2seq, vosk, SpeechBrain, Nvidia Nemo, Fairseq...

Despite having existed for more than a decade as a framework, Kaldi has relatively few open source models available [83]. This is a common problem in ML models for speech recognition based on supervised or semi-supervised learning. In supervised learning, data consist of labelled objects. An ML model is responsible for learning how to assign labels or values to objects (inputs), thus learning which speech audio object (input) is assigned to a specific transcription (output). Considering the complexity and diversity of different languages, this represents a major challenge for speech recognition.

In 2020, the wav2vec2 framework was presented [11]. For the first time, good transcriptions were achieved through self-supervised learning of meaningful representations of speech audio, followed by fine-tuning on the transcribed speech. The framework results outperformed those obtained by semi-supervised methods. The used type of self-supervised training strategy, used in other ML disciplines with remarkable results [36] [84] [85] [86], represents a considerable advance in the generation of models for languages with few available resources. From a pretrained self-supervised cross-language training, it is possible to obtain good results with a second training or fine-tuning, just using a small number of labeled text.

After the release of wav2vec2, similar stacks based on Self-Supervised Learning (SSL) were released, like HuBERT [87] in 2021 and Whisper [88] in 2022. HuBERT looks very similar to wav2vec2: both models use the same Convolutional Neural Network (CNN) followed by a transformer encoder. However, their training processes are very different and HuBERT's performance, when fine-tuned for ASR, either matches or improves wav2vec2. Whisper is an encoder-decoder Transformer model with Mel spectrogram inputs. Whisper's encoder feds the inputs into two CNN layers to finally output the Transformer component and the predicted tokens.

The main difference between the HuBERT/Whisper and wav2vec is that the pre-training stage, although it is also cross-lingual, it is not self-supervised. Wav2vec2 has only been pretrained on speech from audiobooks, which is a relatively narrow domain of clean, read speech. This has implications for model accuracy when processing noisy, conversational audio. Whisper, on the contrary, was trained in a supervised fashion on a very large corpus comprising 680,000 hours of crawled multilingual speech data. OpenAI refers to such a training as "weakly supervised" since the labels have not been verified by humans and thus are potentially noisy. The source and domain characteristics of the training data is unknown. Nevertheless, it is clear that the Whisper training corpus vastly surpassed the one used by wav2vec both in terms of scale and diversity.

Due to the previous analysis, for this article, the three mentioned architectures (wav2vec2, HuBERT and Whisper) were tested and adapted to the proposed edge-intelligent speech-recognition solution. Specifically, this article describes the different models that were trained, optimized and evaluated to suit the characteristics and constraints of performing ML tasks on edge devices.

F. SPEECH-RECOGNITION FRAMEWORK OPTIMIZATION AND FINETUNING

1) Main features

With respect to the VAD, a wav2vec2 pre-trained model, a distilled version and a Whisper model were used at the finetuning stage in order to adapt the final model to the Galician language and to use it on mobile architectures. Due to memory limitations and to the characteristics of the use case, the input audios are limited to 5 seconds. For convenience, the models were tested on a smartphone (One Plus 8T) and a Raspberry Pi 4. Nonetheless, the models can be used on other embedded devices that include a microphone and a speaker, and that must have a minimum of computational capacity and memory to be able to execute ML inference. In order to transform transcribed text into commands that can be understood by an IoT system, due to the lack of grammar corpuses in Galician for finetuning with Bert-based models for this domain, a set of keywords that trigger specific commands performed through publishing/subscribing specific topics was defined.

Keywords are detected from input transcription by performing text distance algorithms in order to determine the score of every word with a specific keyword. Each of such algorithms can create specific combinations that trigger the execution of a specific command, allowing as much as possible a natural speech for the creation of the command.

Overall, the speech-recognition subsystem performs all audio processing, speech and command detection. All processing is performed in the VAD: from the collection of the raw voice of the user to the decision of the command to be executed on the IoT system.

In order to carry out a fair evaluation of the speech and command-recognition inference, three key parameters were considered: system response time, accuracy and resource consumption. The following subsections analyze the different factors that impact such three parameters.

2) Inference runtime

Numerous frameworks exist for using mobile ML models to perform inference on embedded devices [89], although they are not as mature or functional as standard models. The main problem is the lack of support for the instruction set used in the model, as well as the different types of acceleration used in embedded hardware.

Depending on the type of hardware, there are different methods for using acceleration. For example, in Android there is the Android Neural Networks API (NNAPI), which is a C API designed for running computationally intensive operations for machine learning on Android devices [90]. NNAPI provides a base layer of functionality for higher-level machine learning frameworks. CoreML [91] is similar, but for iOS. It is worth noting that, in the case of the Raspberry PI, an onboard shared GPU is available (VideoCore VI).

The previously mentioned acceleration mechanisms allow for parallelizing instructions by reducing considerably the execution time. However, the set of instructions that they support is reduced, so, if there is a high number of operations of the model that are not supported, it will not experience improvement when executing the inference.

In this article, experiments were performed with four different ML stacks for mobile platforms: Pytorch Mobile, Tensorflow Lite, Onnx and NCNN [92] [93] [94] [95]. Pytorch mobile and Tensorflow Lite are frameworks that have been adapted for mobile platforms departing from some of the most popular used stacks for ML. Onnx is an open standard for ML interoperability that supports conversions that enable AI developers to use models with a variety of frameworks, tools, runtimes and compilers. NCNN is a high-performance neural network inference computing framework optimized for mobile platforms that is focused on an easy deployment of the most commonly used CNN networks.

3) Model optimization

The main way for reducing the size of a model is dynamic quantization, which consists in reducing the precision of the weights and/or activation from floating point (fp32) to integer (int8). This technique can be applied once the model has been trained and allows for a significant reduction in size and some improvement in speed of inference [96].

Post-training static quantization involves not just converting the weights from float to int, as in dynamic quantization, but also performing the additional step of calibration feeding with data through the network and computing the resulting distributions of the different activations. These distributions are then used to determine how the different activations should be quantized at inference time. As part of the wav2vec2 and Whisper feature encoders, the raw audio is passed to a 1-dimensional convolutional neural network (temporal convolution) followed by layer normalization and by the application of a Gaussian Error Linear Units (GELU) activation function, (as of writing, both PyTorch and Onnx do not support quantization for GELU activation [97] [98]).

Pruning the model is another option for improving the model efficiency. In general, neural networks are very overparameterized, so the process of removing weight connections in a network can increase inference speed and decrease model storage size. There are two types of pruning: structured and unstructured pruning. The difference between both comes from whether individual weights or groups of weights are removed together. Pruning generally requires setting a criteria on the weights of a model, and setting certain weights to 0 if they match such a criteria. For unstructured pruning, individual weight connections are removed from a network by setting them to 0. For structured pruning, groups of weight connections are removed together, such as entire channels or filters, thus changing the shape of the layers.

Setting weights to 0 will only be efficient and will speed up inference if the number of such weights is high and they have a large dispersion percentage. Sparsity tensors are still in beta phase and they are not the best option for speeding up model inference, being better to compress the size of the model [99].

The best way to increase the speed of inference is to distil the model. The original wav2vec2 pre-trained models are complex, presenting a large number of layers and parameters. Distilling a model means to generate a simplified model (student) from a more complex model (teacher) by distilling the knowledge from the latter to the former. Thus, distilling generates a model with fewer parameters and with slightly worse accuracy, but with a considerably increased inference speed.

As a consequence of the previous analyses, the model implemented by the system described in this article makes use of dynamic quantification and distillation as optimization mechanisms for reducing its size and for speeding up inference time.

4) WER and success rate

The Word Error Rate (WER) is a standard metric for ASR systems [100]. Although the WER is closely related to the performance of an IoT voice assistant, an exact transcription is not the expected goal (i.e., the objective is to execute an IoT command). This fact allows in part for overcoming the loss of accuracy that results from reducing the size of the model, since, regardless of the WER obtained by the model, the reduction and quantification processes will always worsen the accuracy of the model output slightly.

The main problem that the use of architectures like wav2vec2 and Whisper solves is the difficulty of performing speech recognition in different languages. This is a common problem in NLP: most corpus for specific tasks only support English. However, with such architectures it is much easier to obtain a transcription model in different languages. For example, the fluent speech corpus previously mentioned in Section II-C allows for building a specific grammar for HAs, but it needs to map the audio speech to a certain grammar, which is highly complex when performed for every available language. So, instead of generating a model that associates language-specific audios with specific intents related to HA (as it is carried out in Fluent Speech Commands), it is possible to use a model that associates speech recognition patterns with specific intents, which is much more straightforward, as it only involves using grammar, as many DistilBert models do.

Unfortunately, most of the language models generated with DistilBert for this kind of task are specific for the En-

glish grammar. Translating the corpus used to a specific language may be a valid option, but, to simplify the developed system, it was decided to apply text distance comparison algorithms to compare the output of the model with keywords defined for the IoT system. This is a simpler and efficient approach (i.e., less computational resources are needed) but it requires to be adapted to the specific commands and IoT components to be used. To perform this task, algorithms were used to measure the distance between texts by comparing the keywords with what was obtained in the transcription. As it is a speech recognition system, it is possible that phonetically similar sounds may be misinterpreted during the transcription stage, so a phonetic distance algorithm is more accurate than those based on tokens, which would give the same penalty to any character.

However, most phonetic-text distance algorithms are focused on the use of the English language. Phonetically, Galician has hardly any similarity with English, so other types of algorithms had to be used to measure distance. There is a wide range of algorithms used to measure text distances, falling into several categories with different characteristics [101]: Sequence, Edit or Token-based, Simple, Phonetic or Hybrid.

The algorithms based on Sequence or Edit have the advantage of simplicity and are suitable for short chains. Their use is more usual in the field of computer science [102]. Those based on tokens are used in general NLP problems, but are intended for long texts and take semantics into account, so they are not very suitable for single words or small sentences [103]. Hybrid algorithms combine an edit-based approach with a token-based approach and are also used in NLP [104]. Regarding the algorithms classified as 'Simple', as their name implies, they are really simple, comparing prefixes, suffixes or length. Therefore, simple algorithms are very efficient computationally, but they are too simple for the use case proposed in this article, which was evaluated by using two algorithms: Levenshtein (for Edit based) and Longest Common Subsequence (LCS) (for Sequence based). Both are used to compare the similarity of the written words with certain predefined keywords that are used to detect the commands to be executed. The success rate represents the number of commands that were detected correctly. Apart from the success rate, other metrics were used in the analysis of the command detection, such as the number of detected keywords or the false positives (representing the wrong detection of a keyword), as well as the average time of execution of the command detection for each used algorithm.

Equation 3 defines the detection of keywords between strings a and b using the Levenshtein distance and similarity.

$$det(a,b) = \frac{sim(a,b)}{lev(a,b)} \tag{3}$$

Equation 4 shows the expression used for obtaining the similarity between two strings a and b.

$$sim(a,b) = max(|a|,|b|) - lev(a,b) \tag{4}$$

Equation 5 shows the calculation of Levenshtein distance between two strings (a,b), where $tail(x)$ is a substring of all but the first character of x. As it can be observed, the second block of Equation 5 represents the deletion, insertion and substitutions performed for all substrings of (a,b).

$$lev(a,b) = \begin{cases} |a| & if|b| = 0, \\ |b| & if|a| = 0, \\ lev(tail(a), tail(b)) & if a[0] = b[0], \\ 1 + min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b) \end{cases} & otherwise, \end{cases} \tag{5}$$

Equation 6 defines the condition necessary for a detection, where α is a predefined threshold.

$$det(a,b) = 0 \vee det(a,b) \geq \alpha \implies detection \tag{6}$$

A simpler approach is applied for the similarity calculation with the LCS algorithm. Keyword detection is performed for such an algorithm through Equation 7 for two words a and b, being M the number of matches between the longest common subsequence of strings a and b.

$$det(a,b) = \frac{2 \times (|a| + |b|)}{M} \tag{7}$$

Finally, Equation 8 shows the condition necessary for a detection when using the LCS algorithm, where α is a predefined threshold normalized between 1 and 0.

$$det(a,b) \geq \alpha \implies detection \quad | \quad \alpha \in [0,1] \subset \mathbb{R} \tag{8}$$

IV. EXPERIMENTS

A. EXPERIMENTAL SETUP

In order to test the responsiveness, efficiency and success rate of the proposed system, different experiments were carried out by considering that the developed solution is divided into two subsystems:

- The first subsystem performs speech inference, in which an ASR ML model is applied to transcribe the commands and then, based on a predefined set of rules and keywords, the action to be performed is determined. Accuracy (determined by WER and success rate), latency and energy/requirement consumption were measured.
- The second subsystem is the communications subsystem. As it was previously described in Section III-B, the proposed system has been designed to work over BLE Mesh, whose performance has been evaluated in different scenarios for the subsystem in terms of response time.

The tests for both subsystems were carried out on different devices and with diverse embedded hardware in order to compare the performance of the created mobile ML models on different platforms. Specifically, tests were carried out by using an embedded device (a Raspberry Pi 4 with 2 GB

of RAM and a Quad-core Cortex-A72 (ARM v8) 64-bit System-on-Chip (SoC) at 1.5 GHz) and a smartphone (a One Plus 8T with 8 GB of RAM and an Octa-core Cortex-A77 (ARM v8) 64-bit (1x2.84 GHz, 3x2.42 GHz and 4x1.80 GHz Cortex-A55)). For the sake of fairness, instead of using the built-in Bluetooth interfaces incorporated on the Raspberry and the selected smartphone, the communications subsystem was evaluated by using independent BLE boards from Nordic Semiconductor (nRF5x based boards [73]).

For the smartphone tests, a basic Android application was developed. The main screen of the application is shown in Figure 2. With such an application it is possible to send voice commands by clicking the start button, which starts a 5 second recording. Then, the application performs the transcription inference with one of the different models exported to Android. Finally, the application shows at the bottom of the screen the detected keywords.

FIGURE 2: Main screen of the developed Android application for testing purposes.

B. EVALUATED USE CASE

In order to illustrate the operation of the proposed system and thus to evaluate its performance, a specific use case was devised. In such a use case a user sent voice commands in Galician to activate an IoT heating control. For this purpose, first, the user provided a sentence that included specific keywords such as “activate", “heating" or “temperature" (or synonyms of such keywords). Then, the ASR performed the transcription and, after applying the selected text-distance algorithm, the keywords were detected and triggered a specific rule from a set of predefined rules. Each rule executed one or more actions that sent through the communications subsystem using BLE mesh.

As an example, Figure 3 shows a sequence diagram that illustrates the communications performed between a user and several nodes. In the illustrated case, the rule for activating the heating through temperature involves two IoT nodes: a node that controls the heating (smart relay) and a temperature sensor. Since in the proposed scenario it is assumed that temperature changes inside a house are not abrupt, both nodes can operate as LPN, so they can be battery operated and will remain sleeping most of the time. Therefore, the defined rule will simply notify the smart relay when it wakes up that it should subscribe to the temperature topic and be operated (i.e., it should be able to receive on/off commands) depending on a specific temperature value.

With a target temperature defined by the user, the heating (smart relay) will switch on and off over time. It is also possible to receive a notification on the VAD when the heating is switched on or off. For such a purpose, the VAD node has to be in the range of any of the deployed IoT nodes and it will act a LPN node of the BLE mesh. Then, when the VAD would wake up, it would poll a friend node to look for messages for it.

Since an LPN must establish a friendship relationship with another node that supports the friend node functionality to reduce its receiver duty cycles and to save energy, it is important to determine which node should be chosen as a friend when there are several in range. When an LPN publishes a friend request message, this message is not retransmitted and is therefore only processed by friend nodes within direct reach. Friend nodes that receive such a message send certain parameters including the supported ReceiveWindow size, available message queue size, available subscription list size and the Received Signal Strength Indicator (RSSI) value measured by the friend node. The friend node transmits those parameters in a Friend Offer message back to the LPN. After receiving a Friend Offer message, the LPN selects a suitable friend node by applying a specific algorithm devised by the product developer. The algorithm is likely to consider various aspects: some devices may prioritize ReceiveWindow size to reduce power use as much as possible, while others may be more concerned about the RSSI value in order to ensure they can maintain a good quality link with the friend node [105].

However, since the VAD node behaves like a moving LPN node, any node with the friend node functionality can be its friend node at any given time, so it is necessary to replicate the cache of messages addressed to the VAD on all nodes acting as friends. As it is detailed later in Section IV-G1, the time degradation involved in propagating these messages across the mesh has been measured when bypassing the sleep cycles times indicated in Figure 3 and thus calculating the time spent forwarding the messages in scenarios with a different number of hops.

C. SPEECH-RECOGNITION ACCURACY

In order to find out how the different tested optimization mechanisms impact the accuracy of the used ASR model, WER was analyzed for a set of test audios. In particular, different models were generated by considering the use of dynamic quantization. In addition, since the used test audios

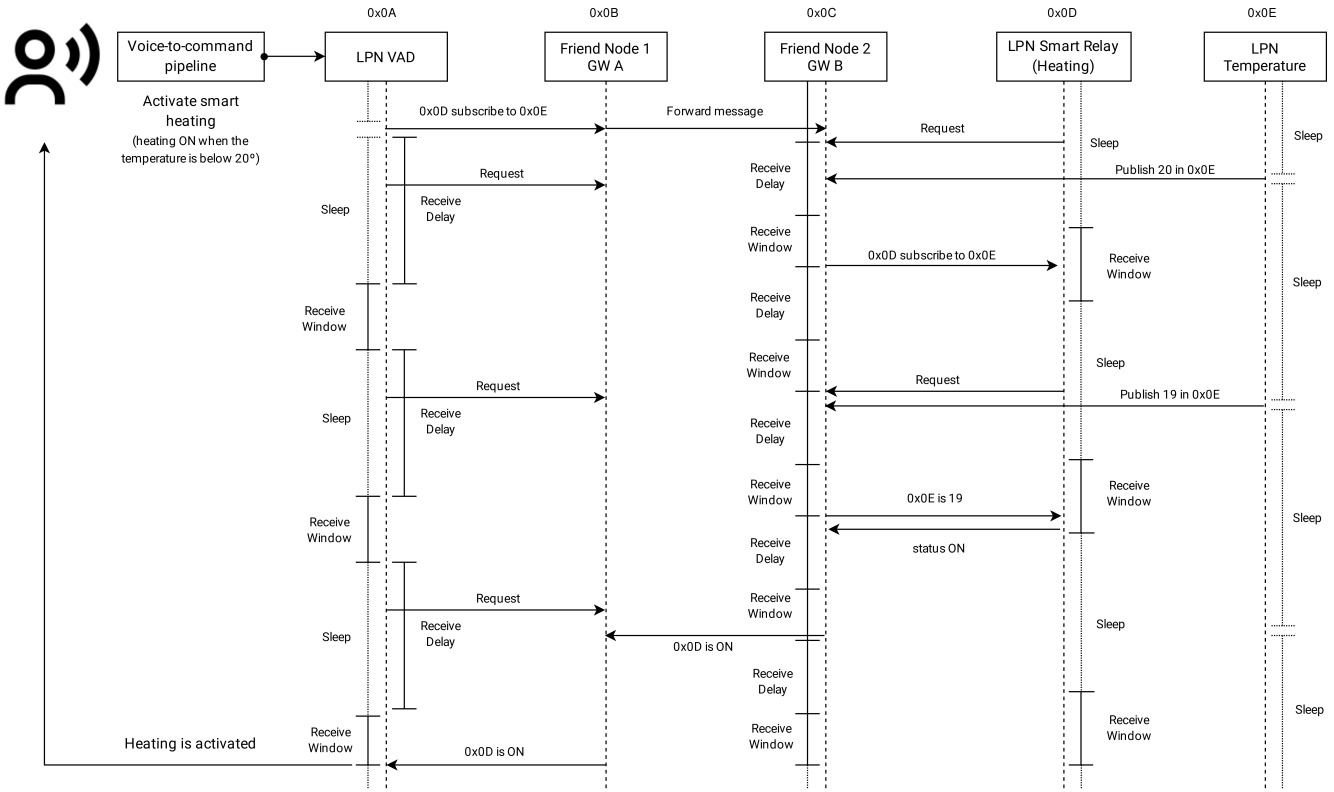


FIGURE 3: Sequence diagram of an example of opportunistic behavior of the Bluetooth subsystem.

have a fixed duration of 5 seconds but it is possible that the full 5 seconds are not used to provide a command, a preprocessing of the audio was carried out to trim the silences based on a silence length and a volume (in decibels) threshold, which such silences impact the obtained WER. Thus, four different models were created:

- A large model based on Facebook's multi-language pre-train: Wav2Vec2-XLSR-53 [106].
- A base model based on the multi-language pre-train Wav2Vec2-Base-VoxPopuli [107].
- A distilled pre-train model generated with DistilHuBERT, a novel teacher-student framework for speech-representation learning by multi-task knowledge distillation [108]. Even distilled, the generation of a pre-train acoustic model is a computationally expensive task that requires high-end Graphical Purpose Units (GPUs), so it was distilled on a student model via s3prl framework [109] with more than 1300 hours of unlabelled audio for Spanish, Italian and Portuguese from LibriSpeech [110], which are languages closely related to Galician.
- A tiny model based on the pre-trained model from openAI, trained on 680,000 hours of multi-language labelled data [111].

After creating the pre-trained models for acoustic representation, each model was finetuned with approximated 20 hours of Galician labelled data [112] [113]. The final finetuned models can be found ready to test on Hugging Face [114] [115] [116] [117]. In such a webpage, the result of the training can be observed on the Training Metrics section. It is important to note that the WER results shown there are for the training dataset, so they are better than the WER results obtained in the experiments presented in this article, which are depicted in Table 2 together with the number of parameters and the size occupied in disk by each model.

Specifically, Table 2 shows that the large model is heavy and complex, so it is not adequate for most current battery-operated IoT devices. Table 2 also shows that the simple fact of applying dynamic quantization (see the models with optimization 'QINT8') reduces considerably model size and allows for using it in devices with memory constraints. However, quantization does not reduce the number of used parameters. The base model is lighter and more suitable for embedded devices, however it is still heavy for low-performance IoT embedded devices.

Luckily, the distilled model reduces even more the complexity of the model and makes it more efficient, although trimming does not affect the structure of the model, since it is a very simple pre-processing that allows for reducing the size of the audio and therefore to accelerate the inference. Nonetheless, it can be observed that trimming can impact the transcription of certain words, as it is configured with a predefined threshold. The decibel threshold will depend on the noise of the environment: if there is too much ambient noise, setting a low threshold will not trim the audio at all,

while if it is too high, it may cut certain words from the speech.

ASR Model	Optimization	Parameters (M)	Size (MB)	WER (%)
Large	No	315	1203.66	14.18
	QINT8	315	303	16.55
	QINT8 and Trimmed	315	303	18.98
Base	No	94.4	361	23.19
	QINT8	94.4	117	25.36
	QINT8 and Trimmed	94.4	117	29.01
Distilled	No	38.3	147	28.89
	QINT8	38.3	73	31.03
	QINT8 and Trimmed	38.3	73	33.78
Tiny	No	37.8	144	20.93
	QINT8	37.8	116	22.12
	QINT8 and Trimmed	37.8	116	25.43

TABLE 2: Number of parameters, size and WER for the different models.

Regarding the WER, it can be observed how applying optimization methods progressively reduces the accuracy of the system, being the distillation with quantization and trimming the most imprecise but also the most efficient model. This is analyzed in the next section, where the created models are compared from the point of view of their inference time.

D. INFERENCE TIMES

1) Inference time with edge devices

Apart from accuracy, another determining factor that impacts the proposed system performance is the ASR model inference time. All the models described in Section IV-C support dynamic quantification and are optimized for mobile platforms, allowing the use of NNAPI on Android or CoreML on iOS. However, such solutions present a common problem: the support for ML on Edge devices is still in beta and is limited for NNAPI and CoreML. In the case of NNAPI, its execution is efficient when the required operations are supported (otherwise execution is slower than when using the CPU).

In the case of the Raspberry, one option to use GPU acceleration is the use of Vulkan [118]. The NCNN framework offers support for Vulkan, but the problem is the same as in the previous cases: the lack of support for some of the operations of the particular model. For this reason, CPU-based inference was used on both platforms.

The stack eventually chosen for the inference runtime was ONNX [95], since it was the one that provided the best results and it is widely portable, enabling to import/export from a wide range of ML frameworks, thus allowing the interoperability with many of the most popular frameworks (e.g., pytorch, tensorflow) and abstracting about the downstream inferencing implications.

For the experiments presented in this article, the following configuration was used for ONNX runtime execution:

- **options.graph_optimization_level**: it enables all optimizations. It was set to ORT_ENABLE_ALL.
- **options.intra_op_num_threads**: it controls the number of threads to use to run the model. It was set to the maximum cores available on the device.
- **options.execution_mode**: it controls whether the operators in the graph run sequentially or in parallel. It was set to ORT_SEQUENTIAL.

Usually, when a model has many branches, setting the option execution_mode to ORT_PARALLEL together with inter_op_num_threads to parallelize the execution of the graph (across nodes) will provide better performance. However, the used models do not present too many branches and no improvement was appreciated when using this execution mode.

Thus, using the previous runtime configuration, three different inference sessions were carried out (one for each model with quantification) for the transcription of a single audio and when executed on the Raspberry Pi. For each session, the different computational operations performed by the model were analyzed, including their total number and the amount of time dedicated to each one.

Figures 4a, 4b, 4c and 5 show, respectively, the results for the inference sessions for the Distilled, Tiny, Base and Large models. For the Base and Distilled models the convolution operations are the most critical. This is normal, since wav2vec2 is a CNN, so the Convolutional Layers are the most important layers in the ML model where the important features from the input are extracted and where most of the computational time is spent. In fact, apart from having a notable impact in these models, these layers are very sensitive to quantification (for CNNs it is recommended applying static quantization [119]). The results also show that, for the Base and Distilled models, quantifying the mentioned layers offers worse performance, especially in the distilled model. Thus, in the quantified version of the Base and Distilled models, all layers except the convolutional ones were quantified (a total of 8 layers for both models are convolutional). In contrast, Whisper is a RNN, which also has convolutional layers, but, as it can be observed in Figure 4b, the time consumed in them is much less than for the wav2vec2-based models.

The results for the inference session of the Large model on the Raspberry Pi show the amount of time spent on inference operations in comparison to the other three reduced models. Looking at Table 2 and Figures 4a, 4c and 5, it can be seen how the reduction of the model is what most impacts inference times. For instance, on the Raspberry Pi it is possible to achieve an improvement in the speed of more than 2.4x and 3.5x over the Large model in the Base and Distilled, respectively. The Tiny model, despite being slightly slower than the distilled model (approx. 100 ms), does not have the bottleneck related to the convolutional layers of the distilled model, which represent more than 58% of the inference time. Thus, in the case of the Tiny model, the operation that takes

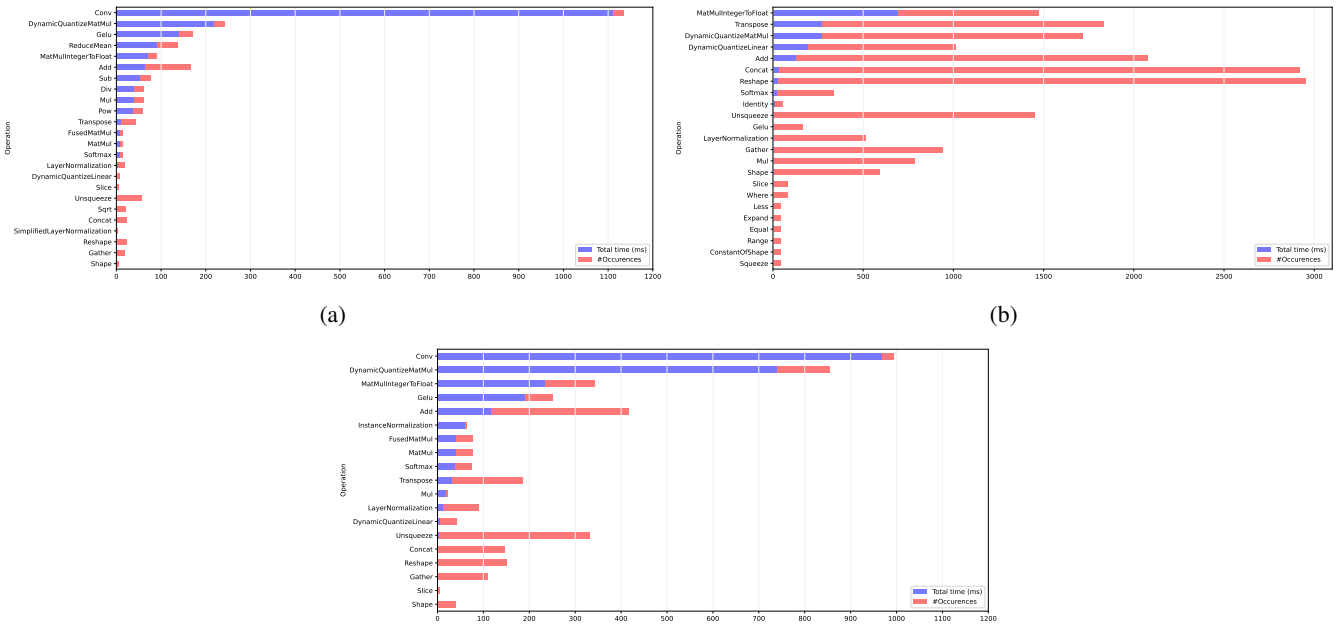


FIGURE 4: Total time and number of occurrences for each model operation in reduced models. Distilled (a), Tiny (b), Base (c)

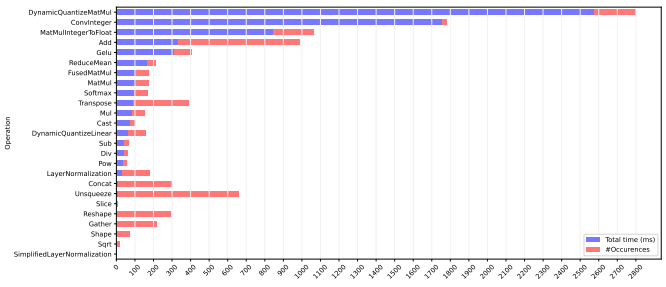


FIGURE 5: Total time and number of occurrences for each model operation in Large model.

most of the inference time consumes less than 20% of the total.

The obtained results can be complemented by comparing the performance of the two evaluated devices (the Raspberry Pi and the smartphone) when carrying out the same tasks. In such a case, it is worth noting that, for the calculation of the inference time, there is an identical pre-processing time for the different used models and a loading time that depends on each model. These two times elapse at the beginning of the execution only, therefore, for each inference, only the transcription time goes by. Thus, Table 3 shows the average time for load and transcription for the Raspberry Pi, while Table 4 shows the same times for the selected Android smartphone. To judge the results, it is important to remember that all audios have a 5 seconds duration except in the models with Trimmed options, where they are in general smaller.

Both for the Raspberry Pi and the smartphone, for the Large model it is indicated 'N/A' because without quantifi-

TABLE 3: Times obtained for the Raspberry Pi 4.

ASR Model	**Optimization**	**Average time** (s)		**Standard deviation**	
		Load model	Transcription	Load model	Transcription
Large	No	N/A	N/A	N/A	N/A
	QINT8	4.12	6.593	0.033	0.034
	QINT8 and Trimmed	4.08	5.37	0.039	0.738
Base	No	11.209	2.706	0.038	0.027
	QINT8	2.016	2.461	0.031	0.021
	QINT8 and Trimmed	1.988	2.006	0.029	0.307
Distilled	No	1.962	2.047	0.036	0.054
	QINT8	1.477	1.882	0.032	0.033
	QINT8 and Trimmed	1.443	1.529	0.027	0.211
Tiny	No	2.139	2.971	0.036	0.141
	QINT8	0.761	2.518	0.032	0.139
	QINT8 and Trimmed	0.759	2.128	0.027	0.211

ASR Model	**Optimization**	**Average time** (ms)		**Standard deviation**	
		Load model	Transcription	Load model	Transcription
Large	No	N/A	N/A	N/A	N/A
	QINT8	856.22	1076.1	3.18	31.2
	QINT8 and Trimmed	851.81	872.26	2.69	112.25
Base	No	993.04	714.56	3.11	29.22
	QINT8	311.5	504.09	2.71	25.31
	QINT8 and Trimmed	306.67	411.82	3.21	98.51
Distilled	No	382.12	516.12	2.49	30.91
	QINT8	179.41	424.48	2.19	27.89
	QINT8 and Trimmed	182.24	342.21	2.43	87.43

TABLE 4: Times obtained for the Android smartphone.

cation it is not possible to execute inference due to memory limitations (even on the selected smartphone, which has 8 GB of RAM). Therefore, the quantization of the model increases the inference speed significantly, especially when loading the model, partly due to the considerable reduction in space. In the case of trimming audio, improvement is achieved, but only in transcription times.

For the selected Android smartphone, the quantified Large model can be used in real applications in light of the times obtained. However, that is not the case of the Raspberry Pi, which has less processing capacity, especially considering that inference runs on the CPU. For a platform with hardware similar to the one of the Raspberry Pi, the best option would definitely be to use a lighter model like the Distilled, whose inference times are under 2 seconds.

2) Inference times when using a cloud server

The inference times obtained in the previous section are local times (i.e., times related to the execution on the mobile devices). To quantify the impact of Edge Computing on the response delay, in this Section, inference time is measured when the models are hosted in a cloud server. Since the cloud was a powerful server (with 2 AMD EPYC 7763 cores, 32 GB of RAM and an Nvidia TITAN A100 64 GB GPU), no optimization mechanisms were applied, so the Large model was used to provide high precision at the output. To carry out inference on the cloud, two API calls were performed against the remote server: one was in charge of sending the audio to the server, while the second one requested the processing of the audio and returned its transcription.

The tests were performed with a WiFi connection using IEEE 802.11n in the 2.4 GHz band. Figure 6 shows the channel occupancy (the access point for the connection to the server is represented in the figure as “SSID1"; the rest of the network identifiers have been renamed to protect their privacy).

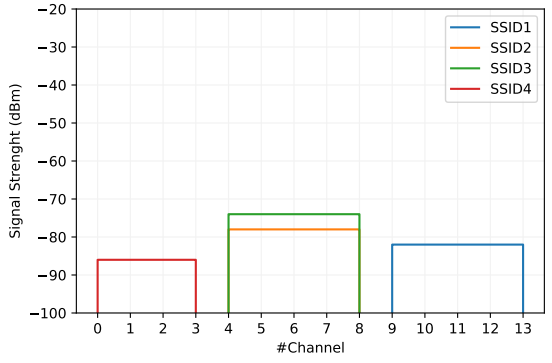


FIGURE 6: WiFi channel occupation for the 2.4 GHz band.

The REST API tests were performed in two different scenarios: one with good connectivity (Scenario A) and another one with poor connectivity (Scenario B). The quality of the connection was measured through the Round-Trip Time (RTT) of the desired server. Scenario A was considered as “good" since the performed tests showed 0 packet losses of a total of 50 delivered in 49,067 ms with a RTT whose minimum was 38 ms, the maximum was 50.7 ms and the average was 41.27 ms. The “poor" connectivity scenario (B) achieved the same packet reception in 49,079 ms with higher RTTs: the minimum was 44.15 ms, the maximum was 143.68 ms and the average was 56.38 ms.

Table 5 shows the times required to perform the two queries and the total time needed by the whole inference process for the scenarios A and B indicating the RTT of the connections. A total of 50 measures were performed.

As it can be observed in Table 5, as expected, times are smaller for the good connectivity scenario, being close to those obtained locally for the Android smartphone when using the distilled model (this is also in part due to the short duration of the audios). However, in scenario B times are clearly larger. Moreover, in scenario B, due to poor connectivity, significantly high values occur (e.g., for the 50 measurements performed, one of them reached 15,735 ms; without such a measurement the average decreases to 1212.15 ms).

E. COMMAND DETECTION

In the previous subsections the performance of speech inference process has been analyzed, but, as it was mentioned before, the ultimate goal of the proposed system is not to transcribe a speech literally, but to perform an action after processing the speech. It is therefore essential to measure the effectiveness and latency for detecting a command. For such a task, 25 voice commands with a 5 second duration were tested. Such commands represented a subset of the available commands. Each of the tested commands belonged to one of 8 different categories (turn on/off lights, turn on/off heating, turn on/off smart heating, show actual temperature, show notifications, turn on/off alarm, increase/decrease volume, call phone number) and all of them were expressed differently with natural spontaneous language and with a total of 28 different keyword predefined for the system, whose appearances in the utterance make it possible to determine the command to execute. For instance, the sentence *"Non escoito ben, fala máis alto"* (I do not hear well, speak louder) would be an example of a speech that includes several keywords. Specifically, the sentence includes two keywords "fala" (speak) and "alto" (louder), which fall under the category "increase/decrease volume".

It is worth mentioning that, as indicated in [120], the generated Galician model has some problems in the segmentation of words that can be improved by using a beam decoder with a language model [44]. However, since the default greedy decoder is more efficient, it was the one used. To overcome the mentioned problem, the system simply compares each word with the possible keywords and then the selected keyword is joined to the next and previous words. This provides a better comparison when a possible keyword is segmented into two words.

Scenario	A						B					
	Query 1		Query 2		Inference	RTT	Query 1		Query 2		Inference	RTT
Time	Connect	Total	Connect	Total			Connect	Total	Connect	Total		
Average (ms)	40.81	176.13	39.95	136.07	336.5	41.27	101.77	1301.57	275.62	614.16	1938.3	56.38
Standard deviation	2.33	10.27	1.4	7.9	10.95	2.04	195.67	1787.31	576.6	1485.73	3263.79	20.459
Minimum (ms)	39.33	169.82	39.16	127.25	323	39	44.59	602.42	58.38	175.21	817	44.15
Maximum (ms)	47.68	21.51	44.16	156.58	370	50.73	914.35	8813.27	2687.97	6897.31	15735	143.68

TABLE 5: Latency results obtained for transcription inference with large model hosted on the cloud when tested in scenarios A and B.

On the other hand, Whisper models add punctuation and accents that are not relevant for the selected home IoT scenario, so all these unwanted characters were escaped in the resulting transcription.

It should be noted that the detection of false positives does not necessarily imply an error in detecting a command. Likewise, there are commands that may be recognized without detecting all the keywords in the phrase. It should be also considered that certain keywords can also trigger other keywords that are synonyms.

Tables 6 and 7 show the number of errors, the number of detected false positive keywords and the processing times in relation to the execution of the model on the Raspberry Pi when using the Levenshtein and LCS algorithms.

ASR model	Optimization	Errors	Keywords detected	False positives	Success	Average time (ms)	Standard deviation
Large	No	N/A	N/A	N/A	N/A	N/A	N/A
	QINT8	0	68	2	25	268	0.0772
	QINT8 and Trimmed	0	68	2	25		
Base	No	3	62	1	22		
	QINT8	2	64	1	23		
	QINT8 and Trimmed	4	62	0	21		
Distilled	No	5	58	0	20		
	QINT8	5	58	0	20		
	QINT8 and Trimmed	6	57	0	19		
Tiny	No	2	63	1	23		
	QINT8	3	62	1	22		
	QINT8 and Trimmed	4	61	0	21		

TABLE 6: Command detection results obtained with Levenshtein algorithm.

As it can be observed in Tables 6 and 7, the large model, despite detecting several false positives, did not make any mistakes. In the case of LCS (Table 7) two different values are shown for the two model tests (i.e., for the quantified and quantified trimmed models): this is because one was performed with the same ratio as the rest of the tests (i.e., Base, Distilled and Tiny) while the other with a slightly higher ratio for the Large model. Thus, since the Large model

ASR model	Optimization	Errors		Keywords detected		False positives		Success		Average time (ms)	Standard deviation
Large	No	N/A		N/A	N/A	N/A		N/A		N/A	N/A
	QINT8	1	0	67	57	3	1	24	25	27	0.0068
	QINT8 and Trimmed	1	0	68	57	3	1	24	25		
Base	No	3		64		1		22			
	QINT8	3		64		1		22			
	QINT8 and Trimmed	4		63		1		21			
Distilled	No	5		58		1		20			
	QINT8	5		60		1		20			
	QINT8 and Trimmed	6		58		1		19			
Tiny	No	2		61		2		23			
	QINT8	3		60		2		22			
	QINT8 and Trimmed	4		59		1		21			

TABLE 7: Command detection results obtained with LCS algorithm.

is more accurate, with a lower ratio it detects more false positives, only making one error.

For quantified and quantified with trimming algorithms and for the large model, the results are almost identical. For the lower precision models, errors and lower keyword detection occur, but there are also no significant differences between using one or both optimizations or the original floating point model. With respect to the distance algorithm, in terms of time, there is a relevant difference between Levenshtein and LCS. Moreover, it can be concluded that the operation of the LCS algorithm is simpler but it also requires a finer adjustment of the detection rate.

F. POWER CONSUMPTION

Another interesting aspect to be considered in the proposed system is power consumption, since the inference will be performed on battery-powered devices. Figure 7 shows the power consumption of the Raspberry Pi 4 when executing the distilled model. Measures were taken with a Joulescope [121], which is a high precision power meter device with a 1.5 nA resolution and which is able to obtain measurements of voltage and current at 2 million samples per second and with a 250 KHz bandwidth.

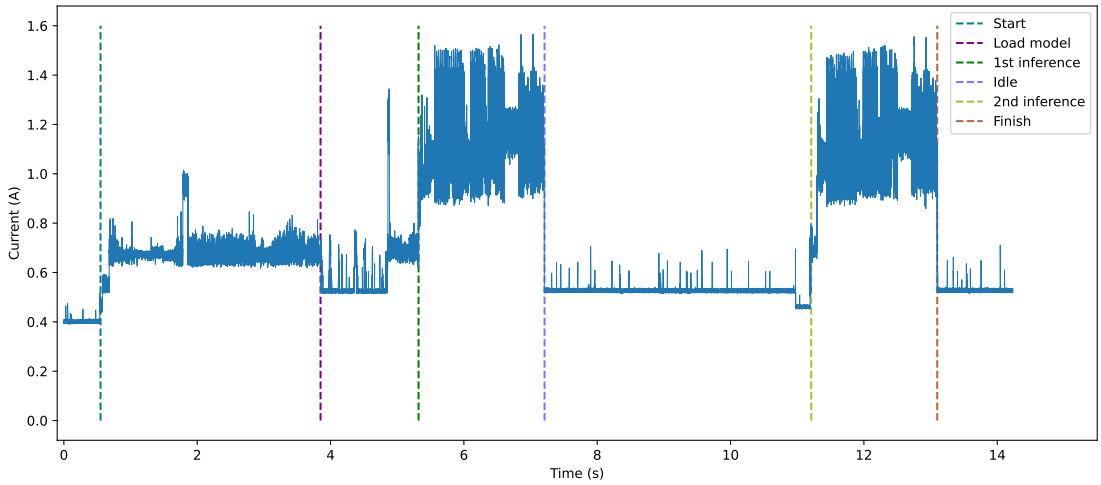


FIGURE 7: Regular operation of the Distilled model with 2 speech inferences.

Figure 7 depicts power consumption by distinguishing the different execution stages:

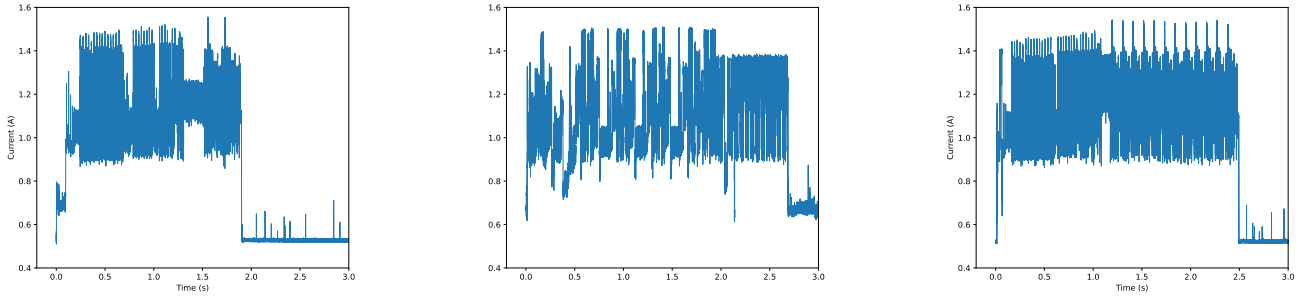
- Stage 1 - Pre-processing. It is common to all models, and requires an approximate average current of 675 mA.
- Stage 2 - Loading of the model. During the first part of this stage the system remains in idle, consuming approximately 520 mA. The second part is related to the actual model loading, which consumes an average of 731 mA.
- Stage 3 - First inference and transcription. For the sake of clarity, Figure 7 represents a first inference and transcription event, which requires an average of 1.2 A, followed by a 5-second delay that consumes roughly 525 mA (such a delay was included for visualization purposes, to distinguish more clearly the two performed inferences).
- Stage 4 - Second inference and transcription. The consumption of this stage varies significantly for the four tested models. Figure 8 shows the consumption for each of the reduced models while Figure 9 shows the values obtained for the quantified large model. The difference in consumption is noticeable: the event has a power consumption of approximately 3.13 Wh (626 mA) and 4.22 Wh (844 mA) for the Distilled and Base models, respectively, while the large model presents a consumption of 11.04 Wh (2207.9 mA). Therefore, in terms of consumption, the first two models are more adequate for embedded devices like the Raspberry Pi. The inference event of the tiny model, despite being less than 100 ms slower on average, has a lower average current consumption than the base one (1.13 A versus 1.23 A), being both adequate for low-resource devices.

Regarding the power consumption on the tested smartphone, unfortunately, it is not possible to use the Joulescope to obtain precise power consumption measurements as on the Raspberry Pi due to the embedded battery load regulator (i.e., it is not easy to measure the real power consumption without bypassing the battery-load regulator). Nonetheless, Android allows for exporting the system power consumption traces, which provide a general overview of the overall consumption. Thus, Figures 10 and 11 show such traces for battery and CPU usage, respectively.

In particular, Figure 10 shows the evolution of the battery on an execution of 22 inferences of 5 seconds when making use of the distilled model every 10 seconds. As it can be observed, a quite intensive use of the battery occurs. The results of the estimated power use of the developed application indicate that roughly 0.03% of the battery was consumed. Considering that the battery capacity of the selected smartphone is 4500 mA, an approximate consumption of 1.35 mA was required for the total execution (similar to which is observed in Figure 7 for the Raspberry Pi but with more speech inferences) and 0.061 mA for every individual inference event, which implies a battery level decrease of approximately 0.0013%.

Figure 11 shows the CPU usage required by the Android system during the execution of the application. In particular, an inference event for the distilled model of 556.4 ms is highlighted. It can be seen how the first 4 CPU cores manage Android internal tasks (which show a relatively high usage outside of the inference event interval), while the other 4 CPUs are dedicated specifically to the execution of the developed application, showing an intensive use during the inference events.

The obtained results, which should be considered as rough approximations, show a power consumption of 0.0411 mWh for the distilled model, which, compared to the 3.13 Wh obtained for the Raspberry Pi for the same model, suppose



(a) Distilled model. (b) Tiny model. (c) Base model.

FIGURE 8: Inference event current on reduced ASR models.

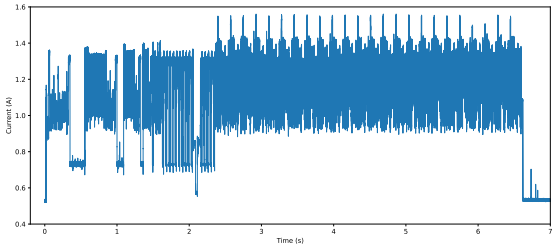


FIGURE 9: Inference event current consumption for the Large model.

a noticeable difference. However, it is important to note that:

- Both platforms make use of completely different hardware.
- No specific optimization mechanisms were implemented on the base OS.
- Measurements were taken in a very different way. The measurements for the Raspberry Pi 4 are more accurate than the estimations obtained for the Android smartphone.
- The Raspberry Pi 4 is more oriented towards a general use as a desktop computer and is not optimized for being used in low-power consumption solutions that run on batteries. Its consumption in idle is between 400 and 550 mA, which is quite high in comparison to the current consumed in idle by devices like Raspberry Pi Zero (around 80 mA). Nonetheless, there are several adjustments that can be made to optimize power consumption without limiting CPU frequency (e.g., by disabling the USB/Ethernet ports), but, since the performed tests were essentially focused on measuring the current consumed by inference events, the previously suggested improvements would not contribute to minimizing the overall solution energy consumption.

G. BLE MESH LATENCY

The use of BLE Mesh introduces communications latencies that have been estimated. For such an estimation, three different scenarios were considered and thus determine how BLE Mesh impacts latency when carrying out direct communications.

To perform this set of experiments, RSSI was used to distinguish among the scenarios. According to Nordic's datasheet for the used BLE Mesh board [122], the sensitivity in BLE 1M mode is -96 dBm. It is worth noting that, since the used board has a Power Amplifier/Low-Noise Amplifier (PA/LNA) module, the obtained range is better than the usually achieved with most Bluetooth devices.

It must be indicated that the three selected BLE Mesh scenarios are not related to the ones described in Section IV-D2. In this case, the scenario A had the best signal reception, with an RSSI equal to or higher than -55 dBm. Scenario B had a medium signal reception, thar was between -70 and -77 dBm. Finally, scenario C had the worst signal strength, with an RSSI equal or under -87 dBm.

Regarding the measured latency, it includes the period from the sending of the IoT command (once it has been determined) until the completion of the requested action by the sensor/actuator node. Times were obtained through a serial connection to the VAD and IoT nodes, which were previously synchronized through Network Time Protocol (NTP).

Scenario	A		B		C	
	Time (ms)	RSSI (dBm)	Time (ms)	RSSI (dBm)	Time (ms)	RSSI (dBm)
Average	78.49	-46.08	84.87	-71.72	113.41	-89.52
Standard deviation	2.96	3.1347	13.15	1.6713	59.1	1.3576
Minimum	73.54	-55	70.47	-77	73.32	-92
Maximum	84.57	-42	116.13	-70	281.33	-87

TABLE 8: Results obtained for the Bluetooth communications subsystem when testing and executing IoT commands in the three selected scenarios.

The results shown in Table 8 indicate that, on average, in the scenario with the worst signal (C) a latency of 113.41 ms was required, while only 78.49 ms for the best signal scenario (A). In any case, less than 282 ms were necessary.

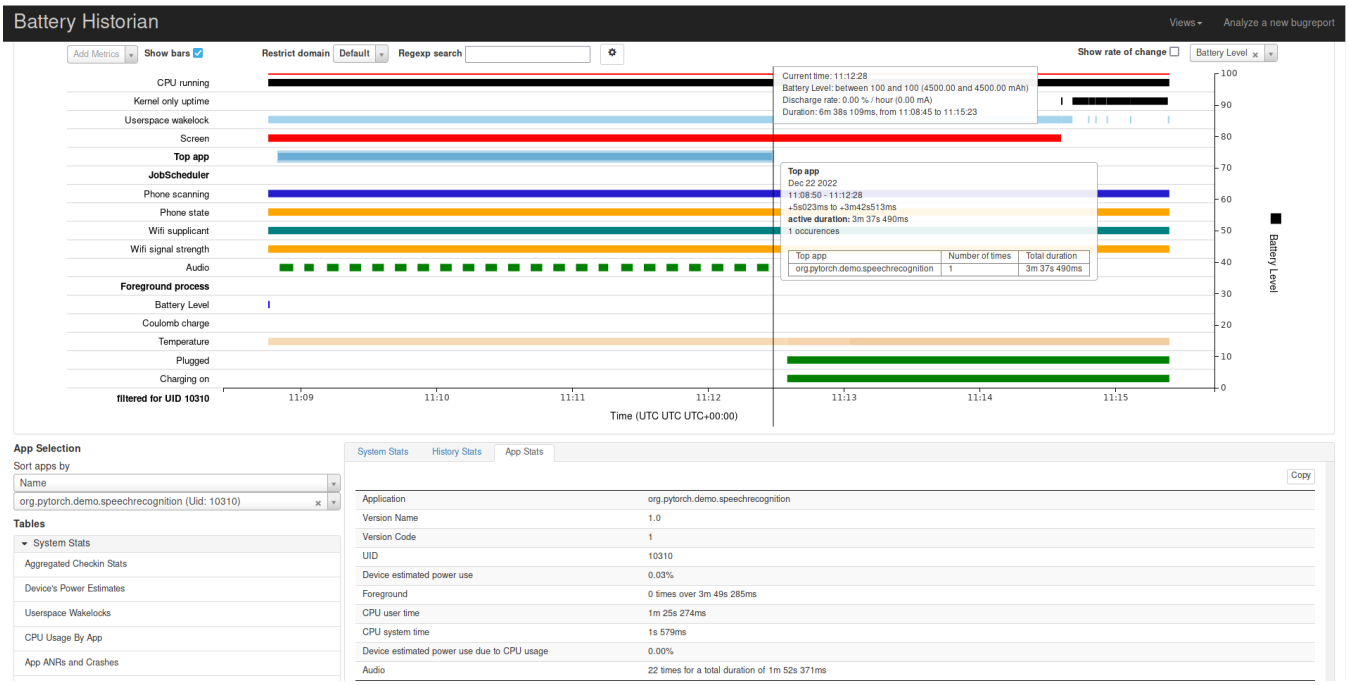


FIGURE 10: Android battery history for the Distilled model.

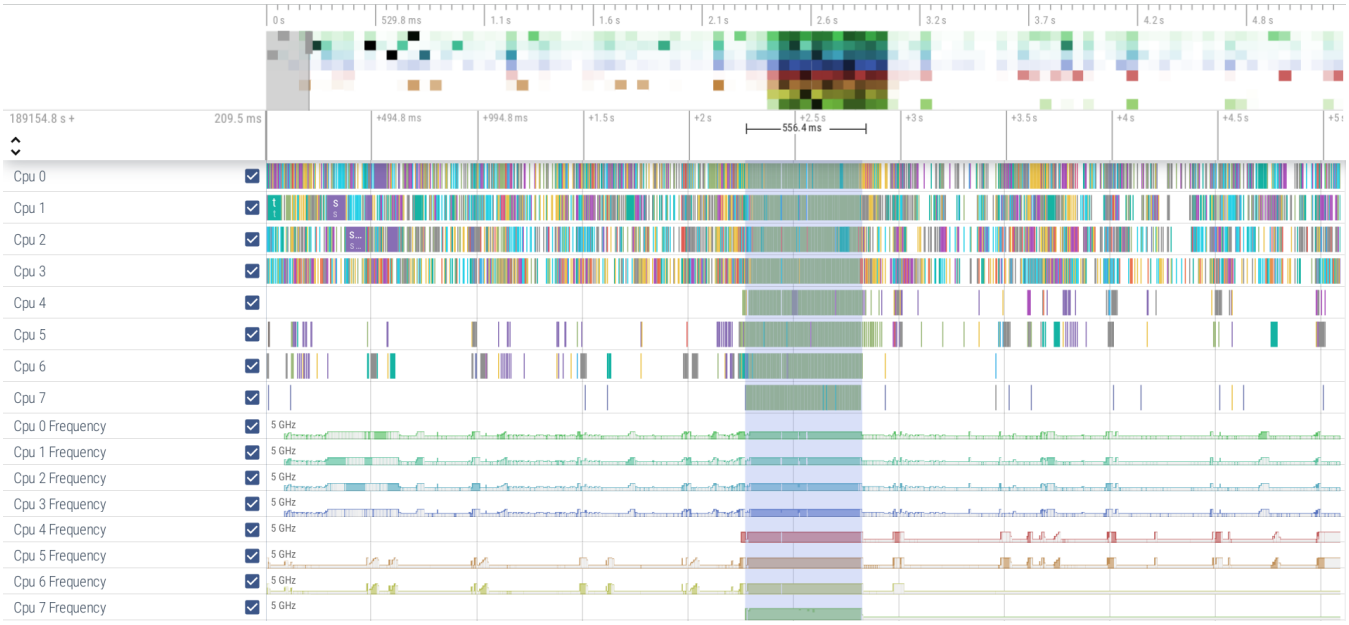


FIGURE 11: Android system trace Analyzer for the Distilled model.

These results show that BLE Mesh is an efficient protocol for performing communications with IoT devices when short messages are involved and when considering the total time as the time of transcription and command detection. For instance, in an Android-based system it would be possible with the fastest models to perform an action in less than 1 second, which is a reasonable latency for immediate-response tasks such as lighting control. For completion purposes, the next section goes further and analyzes latency for opportunistic scenarios, where no direct communications are available.

1) VAD opportunistic communications

a: Opportunistic scenario

Opportunistic systems complement traditional infrastructure-based communications by allowing mobile devices to communicate directly with each other when in range [123]. Due to their design, they are considered beneficial in urban systems where the network is overloaded, as well as in scenarios

where no communications infrastructure is available [124]. This last aspect is interesting for HA environments, since the cost of infrastructure to sensorizing a house can be high. By design, opportunistic systems are infrastructure-free, in which human-carried mobile devices (nodes) store and relay data when the opportunity arises.

In this case, the VAD node acts as a mobile node that can receive messages opportunistically from other devices. This occurs in contrast to the usual approach in HA environments, where voice assistants consist in smart speakers distributed throughout the different rooms of a house and whose communications are all centralized in a hub or server, with the involved costs. This approach is not usually considered by the main wireless HA protocols, which are focused on the use of static IoT devices that communicate with a specific relay/gateway node. In this case, being the VAD a mobile device, the solution is to replicate the information between the different relay nodes.

b: Opportunistic performance

In order to test how the opportunistic message strategy impacts the system, the delay involved in redirecting a message through several intermediate gateways was measured.

In this case, the total latency of the system is not a relevant performance indicator, since LPN nodes are most of the time asleep due to operating on battery or with a limited power supply and since the type of information they send to the VAD node is not critical. The example described in Section IV-B is a clear case of this kind of scenario: the activation of the intelligent heating system does not involve an intensive polling of the house temperature since there will not be significant variations in a short time (i.e., the heating will not raise the temperature quickly).

The total system response latency will be conditioned by the sleep cycles of the LPN node. However, it is possible to measure the time it takes to circulate an end-to-end message addressed to the VAD node when requiring a different number of hops (ignoring the sleep times). Table 9 shows the transmission times according to the number of hops (with a maximum of three) in different scenarios. These three scenarios are not physically the same as those considered in Table 7, since they contemplate opportunistic scenarios, but they follow the same RSSI ranges indicated in Section IV-G.

In fact, another aspect to take into consideration for testing is the increase of the number of intermediate nodes. It is necessary to take into account that the nodes are in range forming a daisy chain. This is important, since BLE Mesh does not make use of routing tables and it is possible that in the communication some intermediate nodes are skipped (when another node is in range). To avoid this issue and thus make sure that intermediate hops are not skipped, the transmission power levels of certain nodes were lowered.

c: Experiments

To calculate the time it takes from source to destination (being the origin the VAD node and the destination any LPN node in the system), a serial connection was used in the source and destination node as in the previous section and also in the next hop to the source node (VAD) to obtain the RSSI value. In this case, the different RSSI values for each scenario refer only to the source node, since is the only that is in movement and could experiment notable variations.

In order to calculate the time it takes from the source to the destination when several intermediate hops are involved, a serial connection is used to obtain the timestamp from the source node (i.e., the VAD node) to the destination node (i.e., a specific LPN node). In addition, this mechanism is also used to obtain the RSSI of the intermediate node that receives the message from the VAD and thus to correlate it with the three RSSI ranges defined in Section IV-G. Regarding the RSSI of the rest of the intermediate nodes, it is not specifically analyzed in the tests since they remain static, so they do not experience variations as remarkable as for the VAD node.

The tests were carried out indoors in a house and throughout 8 different rooms. A map of the indoor scenario in which the tests were carried out is shown in Figure 12. In such a Figure, GW1, GW2 and GW3 represent the intermediate gateways employed in the different scenarios (the gateway antenna orientation is also depicted). The destination node (LPN Dst) remained for all tests in the same position, while the intermediate gateway position varied according to the number of hops: for one hop only GW2 was used; with two hops, GW2 and GW1 were employed; and for three hops all the represented gateways were used. In addition, the intermediate gateways of the different scenarios were static, with stable RSSI values that ranged between -66 and -82 dBm. In contrast, the VAD node was placed in different positions depending on the scenario and considering the RSSI ranges defined in Section IV-G.

The results show minimal differences between the different levels of RSSI, similar to those shown in Table 8. However, as the number of hops increases, there is a noticeable increase in the measured times. In particular, increases in latency ranging from 20% to 42% are observed as the number of hops grows.

Finally, it should be considered that, when using the BLE Mesh standard, as a network grows in terms of complexity, its communications tend to become saturated. In such a case, the only control mechanism that is provided consists in sending addressed messages and in defining a TTL value. However, to avoid potential saturation problems, it is better to avoid the deployment of BLE Mesh networks that require a complex topology.

V. KEY FINDINGS AND FUTURE CHALLENGES

From all the tests and developments carried out throughout this article, the following conclusions were obtained:

- Most of the speech recognition models are heavy on resources and need to be run on a server, which leads to a potential lack of privacy, high costs and security problems. The main supported languages are the

Scenario	A				B				C			
	RSSI (dBm)	Time (ms)			RSSI (dBm)	Time (ms)			RSSI (dBm)	Time (ms)		
		1 Hop	2 Hop	3 Hop		1 Hop	2 Hop	3 Hop		1 Hop	2 Hop	3 Hop
Average	-52.09	151.06	201.51	258.12	-73.91	160.07	224.6	271.85	-88.9	207.61	272.3	349.5
Standard deviation	1.92	55.61	105.69	142.6	2.57	56.63	102.37	147.82	1.51	94.93	138.61	152.1
Minimum	-52	94.9	68.5	90.26	-70	95.46	98.22	99.8	-87	91.14	92.43	94.1
Maximum	-55	221.63	412.81	538.21	-77	221.36	472.18	567.4	-91	391.38	498.21	598.62

TABLE 9: Latency results obtained in multi-hop communication between the VAD and the IoT node.

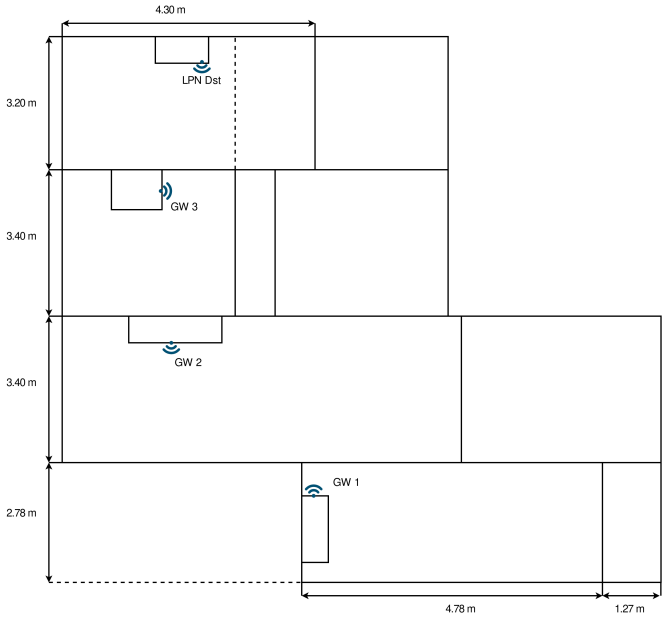


FIGURE 12: Node and gateway locations for the indoor tests.

most widely spoken, which traditionally require the use of corpuses with thousands of hours of audio associated with texts, which complicates the generation of language-specific corpuses.

- The main literature and solutions for NLP problems are specific for the English language, which makes it difficult to perform specific language processing tasks for different languages (mainly due to the lack of corpus in other languages).
- The latest ASR solutions allow for generating multi-language transcriptions with few labelled data hours for training, which enables to obtain an accurate acoustic model in specific languages with few resources.
- Text-distance algorithms are an elegant and efficient solution to overcome the lack of NLP corpuses for non-English grammars. However, for an acoustic model like the one generated, the ideal option would be to use phonetic-based algorithms, which are mostly based on English phonetics.
- The portability of embedded models for audio tasks is still low. They are too complex for being used in embedded architectures, at least in comparison to the most popular computer vision models for mobile architectures (e.g., MobilNet, Yolo), which are much lighter and can perform CPU inference with reduced times. For instance, the small model of MobileNetV3 has 1.7 M parameters [125], much less than for the Tiny and Distilled models generated for this article.
- The main technologies that allow for using embedded models on mobile platforms fail to support acceleration in many of the instructions for the generated models. For instance, the use of NNAPI acceleration with ONNX library lacks support for 7 different instructions with generated models [126]. One of such instructions is the convolution, which is the most critical in terms of processing time. In fact, acceleration in ONNX for Android only supports the 2-dimensional convolution (wav2vec2 and Whisper use 1-dimensional convolutions). Moreover, devices that are not based on Android suffer from a similar problem. For instance, the Raspberry Pi can make use of acceleration for inference through Vulkan, but the number of supported instructions is still limited.
- The obtained results are good when it comes to short audios, despite performing CPU inference. Even without the use of hardware accelerators like Google Coral, the Distilled, Base and Tiny models are undoubtedly the most suitable for an embedded device with low resources in terms of consumption and latency (like a Raspberry Pi). Specifically, the Distilled model is the most efficient. For the Android platform, even the large model with quantification produces acceptable inference times. Reduced models are more suitable for low-end smartphones, since such smartphones have larger capacity batteries (nonetheless, all of the generated models are valid in terms of energy consumption).
- The distillation of the Large model obtained the best results to speed up inference. Although applying quantification also provided a slight increase in speed, the most remarkable speed-up was obtained by distilling the knowledge of a large model into a small model with a much lower number of parameters and connections.
- Whisper-based model presents a better WER than the wav2vec2 and distilHubert based models. Despite using the same corpus for finetuning every model developed in the system, in relation to the number of parameters,

Whisper provides better results. In this case, the fact that the Whisper pretraining is supervised (weak supervision with labelled text from different domains (not only read text) with a higher number of hours than wav2vec2) makes it more robust. However, the time spent on transcription by Whisper is higher. This is in part due to the fact that it performs more tasks than wav2vec2 (e.g., capitalization, accents). Such tasks may be relevant for an ASR model, but they are not for a voice assistant.

VI. CONCLUSIONS

In this paper, an IoT home automation system with voice assistance has been presented. The system operates exclusively on the edge, without the need for an Internet connection, requiring a reduced deployment in infrastructure and allowing the use of low-resource languages like Galician. The system has been tested in static and mobile opportunistic scenarios, providing relatively fast and efficient response times, performing CPU-only inference for transcriptions, being suitable for edge devices with minimal computational capabilities and implementing a distributed architecture without the need for expensive gateways or hubs to manage the communications with the deployed IoT nodes.

Specifically, different multi-language ASR models were developed, validated and optimized for being used with a voice assistant. A relevant effort was put on the optimization and performance evaluation of reduced models that can be used by Android devices or by other types of resource-constrained embedded hardware. Moreover, the proposed system considers the use of mobile opportunistic devices, which is beneficial from the infrastructure point of view. Such an opportunistic approach was evaluated in terms of latency, obtaining the system response times in different scenarios. In terms of latency, with the most limited hardware, inferences of less than 2 s were achieved for the best case while accuracy for the worst case reached a success rate of 76%. Specific details have been provided regarding the energy consumption required by the transcriptions of each tested model. Finally, transmission latency was measured between the VAD and an IoT nodes, showing that it never exceeded 300 ms for the worst case in direct connection and 600 ms with 3 hops.

REFERENCES

[1] "United nations, world population ageing 2020 highlights." Available at https://www.un.org/development/desa/pd/news/world-population-ageing-2020-highlights (accessed on February 2023).

[2] P. Kaur, P. Singh, and V. Garg, "Speech recognition system; challenges and techniques," International Journal of Computer Science and Information Technologies, vol. 3, no. 3, pp. 3989–3992, 2012.

[3] S. Zaidi, V. Shukla, V. P. Mishra, and B. Singh, Redefining Home Automation Through Voice Recognition System, pp. 155–165. 05 2021.

[4] "Global automatic speech recognition market 2022." Available at https://www.linkedin.com/pulse/global-automatic-speech-recognition-market-2022- (accessed on February 2023).

[5] "Voice recognition tech privacy and cybersecurity concerns." Available at https://www.natlawreview.com/article/voice-recognition-technology-market-surges-organizations-face-privacy-and (accessed on February 2023).

[6] J. Lau, B. Zimmerman, and F. Schaub, "Alexa, are you listening?," Proceedings of the ACM on Human-Computer Interaction, vol. 2, pp. 1 – 31, 2018.

[7] A. Georgescu, A. Pappalardo, H. Cucu, and M. Blott, "Performance vs. hardware requirements in state-of-the-art automatic speech recognition," EURASIP Journal on Audio, Speech, and Music Processing, vol. 2021, 07 2021.

[8] S. Gondi and V. Pratap, "Performance evaluation of offline speech recognition on edge devices," Electronics, vol. 10, no. 21, 2021.

[9] C. Yi, J. Wang, N. Cheng, S. Zhou, and B. Xu, "Applying wav2vec2.0 to speech recognition in various low-resource languages," 2020.

[10] E. Guglielmi, G. Rosa, S. Scalabrino, G. Bavota, and R. Oliveto, "Sorry, i don't understand: Improving voice user interface testing," in 37th IEEE/ACM International Conference on Automated Software Engineering, ASE22, (New York, NY, USA), Association for Computing Machinery, 2023.

[11] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," Advances in neural information processing systems, vol. 33, pp. 12449–12460, 2020.

[12] "Language diversity index." Available at https://education.nationalgeographic.org/resource/language-diversity-index-map (accessed on February 2023).

[13] "Ageing Europe - statistics on population developments." Available at https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Ageing_Europe__statistics_on_population_developments (accessed on February 2023).

[14] "Instituto galego de estatística." Available at https://www.ige.gal/web/index.jsp (accessed on February 2023).

[15] F. Navarro Valverde et al., "Depopulation and aging in rural areas in the european union: practices starting from the leader approach," Perspectives on rural development, vol. 2019, no. 3, pp. 223–252, 2019.

[16] I. Irugalbandara, A. Naseem, M. Perera, and V. Logeeshan, "Homeio: Offline smart home automation system with automatic speech recognition and household power usage tracking," in 2022 IEEE World AI IoT Congress (AIIoT), pp. 571–577, 2022.

[17] N. Chumuang, M. Ketcham, S. Tangwannawit, W. Yimyam, S. Hiranchan, M. Rattanasiriwongwut, and P. Pramkeaw, "Development a home electrical equipment control device via voice commands for elderly assistance," in 2020 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), pp. 1–7, 2020.

[18] L. Xu, A. Iyengar, and W. Shi, "Cha: A caching framework for home-based voice assistant systems," in 2020 IEEE/ACM Symposium on Edge Computing (SEC), pp. 293–306, 2020.

[19] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, Q. Xu, N. Goyal, K. Singh, P. von Platen, Y. Saraf, J. Pino, A. Baevski, A. Conneau, and M. Auli, "Xls-r: Self-supervised cross-lingual speech representation learning at scale," 2021.

[20] "Sparkfun edge development board - apollo3 blue mcu - dev-15170 - sparkfun electronics." Available at https://www.sparkfun.com/products/15170 (accessed on February 2023).

[21] A. Ghosh, D. Chakraborty, and A. Law, "Artificial intelligence in internet of things," CAAI Transactions on Intelligence Technology, vol. 3, no. 4, pp. 208–218, 2018.

[22] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," Proceedings of the IEEE, vol. 107, no. 8, pp. 1738–1762, 2019.

[23] L. C. Schünke, B. Mello, C. A. da Costa, R. S. Antunes, S. J. Rigo, G. de Oliveira Ramos, R. da Rosa Righi, J. N. Scherer, and B. Donida, "A rapid review of machine learning approaches for telemedicine in the scope of covid-19," Artificial Intelligence in Medicine, vol. 129, p. 102312, 2022.

[24] M. Vacher, B. Lecouteux, and F. Portet, "Multichannel automatic recognition of voice command in a multi-room smart home : an experiment involving seniors and users with visual impairment," Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 09 2014.

[25] C. Chen, J. G. Johnson, A. Charles, K. abd Lee, E. T. Lifset, M. Hogarth, A. A. Moore, E. Farcas, and N. Weibel, "Understanding barriers and design opportunities to improve healthcare and qol for older adults through voice assistants," in The 23rd International ACM SIGACCESS Conference on Computers and Accessibility (Virtual Event, USA)(ASSETS'21). Association for Computing Machinery, Virtual Event, USA. https://doi. org/10.1145/3441852.3471218, 2021.

[26] A. Pradhan, A. Lazar, and L. Findlater, "Use of intelligent voice assistants by older adults with low technology use," ACM Trans. Comput.-Hum. Interact., vol. 27, sep 2020.

[27] A. S, N. Rajalakshmi, V. P. P, and J. L, "Dynamic nlp enabled chatbot for rural health care in india," in 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA), pp. 1–6, 2022.

[28] N. Mani, A. Singh, and S. L. Nimmagadda, "An iot guided healthcare monitoring system for managing real-time notifications by fog computing services," Procedia Computer Science, vol. 167, pp. 850–859, 2020. International Conference on Computational Intelligence and Data Science.

[29] S. Latif, J. Qadir, A. Qayyum, M. Usama, and S. Younis, "Speech technology for healthcare: Opportunities, challenges, and state of the art," IEEE Reviews in Biomedical Engineering, vol. 14, pp. 342–356, 2021.

[30] P. Suter, W. N. Suter, and D. Johnston, "Theory-based telehealth and patient empowerment," Population Health Management, vol. 14, no. 2, pp. 87–92, 2011. PMID: 21241182.

[31] S. U. Amin and M. S. Hossain, "Edge intelligence and internet of things in healthcare: A survey," IEEE Access, vol. 9, pp. 45–59, 2021.

[32] E. Kasthuri and S. Balaji, "Natural language processing and deep learning chatbot using long short term memory algorithm," Materials Today: Proceedings, 05 2021.

[33] J. Baptista, G. Fernandes, R. Talhadas, F. Dias, and N. Mamede, "Implementing european portuguese verbal idioms in a natural language processing system," Proceedings of europhras, pp. 102–115, 2015.

[34] Y.-C. Zhou, Z. Zheng, J.-R. Lin, and X.-Z. Lu, "Integrating nlp and context-free grammar for complex rule interpretation towards automated compliance checking," Computers in Industry, vol. 142, p. 103746, 2022.

[35] Q. Liu, M. J. Kusner, and P. Blunsom, "A survey on contextual embeddings," 2020.

[36] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of naacL-HLT, vol. 1, p. 2, 2019.

[37] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[38] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2019.

[39] L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio, "Speech model pre-training for end-to-end spoken language understanding," in Proc. of Interspeech (G. Kubin and Z. Kacic, eds.), pp. 814–818, 2019.

[40] Y. Jiang, B. Sharma, M. Madhavi, and H. Li, "Knowledge distillation from bert transformer to speech transformer for intent classification," 2021.

[41] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," CoRR, vol. abs/1810.04805, 2018.

[42] M. Garcia, "Exploring the representation of word meanings in context: A case study on homonymy and synonymy," in Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 3625–3640, Association for Computational Linguistics, 2021.

[43] D. Vilares, M. Garcia, and C. Gómez-Rodríguez, "Bertinho: Galician bert representations," Procesamiento del Lenguaje Natural, p. 13–26, 2021.

[44] S. Gondi, "Wav2vec2.0 on the edge: Performance evaluation," 2022.

[45] W. Sun, J. Liu, and Y. Yue, "Ai-enhanced offloading in edge computing: When machine learning meets industrial iot," IEEE Network, vol. 33, no. 5, pp. 68–74, 2019.

[46] S. Sen, J. Koo, and S. Bagchi, "Trifecta: Security, energy efficiency, and communication capacity comparison for wireless iot devices," IEEE Internet Computing, vol. 22, no. 1, pp. 74–81, 2018.

[47] "Arm's new cortex-a77 cpu micro-architecture: Evolving performance." Available at https://www.anandtech.com/show/14384/arm-announces-cortexa77-cpu-ip (accessed on February 2023).

[48] "How the iot ecosystem will look like in 2025." Available at https://medium.com/@sophiamcleod99/how-the-iot-ecosystem-will-look-like-in-2025-eb81af07402c (accessed on February 2023).

[49] P. Fraga-Lamas, S. I. Lopes, and T. M. Fernández-Caramés, "Green iot and edge ai as key technological enablers for a sustainable digital transition towards a smart circular economy: An industry 5.0 use case," Sensors, vol. 21, no. 17, 2021.

[50] M. Brandalero, M. Ali, L. Le Jeune, H. G. M. Hernandez, M. Veleski, B. da Silva, J. Lemeire, K. Van Beeck, A. Touhafi, T. Goedemé, N. Mentens, D. Göhringer, and M. Hübner, "Aitia: Embedded ai techniques for embedded industrial applications," in 2020 International Conference on Omni-layer Intelligent Systems (COINS), pp. 1–7, 2020.

[51] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language models are few-shot learners," Advances in neural information processing systems, vol. 33, pp. 1877–1901, 2020.

[52] K. Song, H. Sun, X. Tan, T. Qin, J. Lu, H. Liu, and T.-Y. Liu, "Lightpaff: A two-stage distillation framework for pre-training and fine-tuning," 2020.

[53] W. Liu, P. Zhou, Z. Zhao, Z. Wang, H. Deng, and Q. Ju, "Fastbert: a self-distilling bert with adaptive inference time," 2020.

[54] "Cloud tensor processing units." Available at https://cloud.google.com/tpu/docs/tpus.

[55] "An in-depth look at google's first tensor processing unit (tpu)." Available at https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu (accessed on February 2023).

[56] "Products | coral." Available at https://www.coral.ai/products/ (accessed on February 2023).

[57] K. Palanisamy, V. Khimani, M. H. Moti, and D. Chatzopoulos, "Spliteasy: A practical approach for training ml models on mobile devices," in Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications, HotMobile '21, (New York, NY, USA), p. 37–43, Association for Computing Machinery, 2021.

[58] "Fine-tune a pretrained model." Available at https://huggingface.co/docs/transformers/training.

[59] S. Alqahtani and M. Demirbas, "Performance analysis and comparison of distributed machine learning systems," 2019.

[60] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J.-R. Wen, J. Yuan, W. X. Zhao, and J. Zhu, "Pre-trained models: Past, present and future," AI Open, vol. 2, pp. 225–250, 2021.

[61] M. Elkhodr, S. Shahrestani, and H. Cheung, "Emerging wireless technologies in the internet of things : A comparative study," International Journal of Wireless 'I&' Mobile Networks, vol. 8, 11 2016.

[62] L. Schönherr, M. Golla, T. Eisenhofer, J. Wiele, D. Kolossa, and T. Holz, "Unacceptable, where is my privacy? exploring accidental triggers of smart speakers," arXiv preprint arXiv:2008.00508, 2020.

[63] "Low power node example (experimental)." Available at https://infocenter.nordicsemi.com/index.jsp?topic=\%2Fcom.nordic.infocenter.meshsdk.v3.2.0\%2Fmd_examples_experimental_lpn_README.html'I\&'cp=5_2_3_6_1 (accessed on February 2023).

[64] "The verge | insteon customers turned insteon's lights back on." Available at https://www.theverge.com/2022/6/9/23161803/insteon-customers-bought-company-restored-service(accessed on February 2023).

[65] M. Suárez-Albela, T. M. Fernández-Caramés, P. Fraga-Lamas, and L. Castedo, "A practical performance comparison of ecc and rsa for resource-constrained iot devices," in 2018 Global Internet of Things Summit (GIoTS), pp. 1–6, 2018.

[66] E. Barker and A. Roginsky, "Transitioning the use of cryptographic algorithms and key lengths," tech. rep., National Institute of Standards and Technology, 2018.

[67] T. M. Fernández-Caramès and P. Fraga-Lamas, "Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks," IEEE Access, vol. 8, pp. 21091–21116, 2020.

[68] T. M. Fernández-Caramés, "From pre-quantum to post-quantum iot security: A survey on quantum-resistant cryptosystems for the internet of things," IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6457–6480, 2020.

[69] P. Fraga-Lamas, P. Lopez-Iturri, M. Celaya-Echarri, O. Blanco-Novoa, L. Azpilicueta, J. Varela-Barbeito, F. Falcone, and T. M. Fernández-Caramés, "Design and empirical validation of a bluetooth 5 fog computing based industrial cps architecture for intelligent industry 4.0 shipyard workshops," IEEE Access, vol. 8, pp. 45496–45511, 2020.

[70] T. M. Fernández-Caramés and P. Fraga-Lamas, "Design of a fog computing, blockchain and iot-based continuous glucose monitoring system for crowdsourcing mhealth," Proceedings, vol. 4, no. 1, 2019.

[71] P. Fraga-Lamas, D. Barros, S. I. Lopes, and T. M. Fernández-Caramés, "Mist and edge computing cyber-physical human-centered systems for

industry 5.0: A cost-effective iot thermal imaging safety system," Sensors, vol. 22, no. 21, 2022.

[72] "Google statistics on ipv6 usage." Available at https://www.google.com/intl/en/ipv6/statistics.html (accessed on February 2023).

[73] "nrf52 series." Available at https://infocenter.nordicsemi.com/index.jsp?topic=\%2Fstruct_nrf52\%2Fstruct\%2Fnrf52.html (accessed on February 2023).

[74] "2022 bluetooth market update." Available at https://www.bluetooth.com/2022-market-update/?utm_campaign=bmu&utm_source=internal&utm_medium=web&utm_content=2022bmu-resourcepopup (accessed on February 2023).

[75] D. Pérez-Díaz-De-Cerio, Á. Hernández-Solana, M. García-Lozano, A. V. Bardají, and J.-L. Valenzuela, "Speeding up bluetooth mesh," IEEE Access, vol. 9, pp. 93267–93284, 2021.

[76] I. Froiz-Míguez, P. Fraga-Lamas, and T. M. Fernández-Caramés, "Power consumption analysis for the development of energy efficient bluetooth 5 based real-time industrial iot systems," in Science and Technologies for Smart Cities (S. Paiva, X. Li, S. I. Lopes, N. Gupta, D. B. Rawat, A. Patel, and H. R. Karimi , eds.), (Cham), pp. 188–206, Springer International Publishing, 2022.

[77] "Bluetooth mesh models - a technical overview." Available at https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-models (accessed on February 2023).

[78] A. Aijaz, A. Stanoev, D. London, and V. Marot, "Demystifying the performance of bluetooth mesh: Experimental evaluation and optimization," in 2021 Wireless Days (WD), pp. 1–6, 2021.

[79] "Bluetooth mesh network performance." Available at https://www.silabs.com/documents/public/application-notes/an1137-bluetooth-mesh-network-performance.pdf (accessed on February 2023).

[80] M. Conti and M. Kumar, "Opportunities in opportunistic computing," Computer, vol. 43, no. 01, pp. 42–50, 2010.

[81] "Bluetooth mesh networking: Friendship." Available at https://www.bluetooth.com/blog/bluetooth-mesh-networking-series-friendship/ (accessed on February 2023).

[82] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. K. Goel, M. Hannemann, P. Motlícek, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý, "The kaldi speech recognition toolkit," 2011.

[83] "Kaldi asr." Available at https://kaldi-asr.org/models.html (accessed on February 2023).

[84] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018.

[85] I. Misra and L. v. d. Maaten, "Self-supervised learning of pretext-invariant representations," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 6707–6717, 2020.

[86] O. Henaff, "Data-efficient image recognition with contrastive predictive coding," in International conference on machine learning, pp. 4182–4192, PMLR, 2020.

[87] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," 2021.

[88] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022.

[89] T. Sipola, J. Alatalo, T. Kokkonen, and M. Rantonen, "Artificial intelligence in the iot era: A review of edge ai hardware and software," in 2022 31st Conference of Open Innovations Association (FRUCT), pp. 320–331, 2022.

[90] "Neural networks api | android ndk." Available at https://developer.android.com/ndk/guides/neuralnetworks/ (accessed on February 2023).

[91] "Coreml overview - machine learning." Available at https://developer.apple.com/machine-learning/core-ml/ (accessed on February 2023).

[92] "An overview of the pytorch mobile demo apps." Available at https://pytorch.org/blog/mobile-demo-apps-overview/ (accessed on February 2023).

[93] "Tensorflow lite | ml for mobile and edge devices." Available at https://www.tensorflow.org/lite (accessed on February 2023).

[94] "Tencent/ncnn." Available at https://github.com/Tencent/ncnn (accessed on February 2023).

[95] "Onnx | home." Available at https://onnx.ai/ (accessed on February 2023).

[96] "Introduction to quantization on pytorch." Available at https://pytorch.org/blog/introduction-to-quantization-on-pytorch/ (accessed on February 2023).

[97] "Github pytorch / pytorch." Available at https://github.com/pytorch/pytorch/pull/34396 (accessed on February 2023).

[98] "Github microsoft / onnxruntime." Available at https://github.com/microsoft/onnxruntime/blob/main/onnxruntime/python/tools/quantization/registry.py (accessed on February 2023).

[99] "torch.sparse - pytorch 1.13 documentation." Available at https://pytorch.org/docs/stable/sparse.html#why-and-when-to-use-sparsity (accessed on February 2023).

[100] R. Errattahi, A. El Hannani, and H. Ouahmane, "Automatic speech recognition errors detection and correction: A review," Procedia Computer Science, vol. 128, pp. 32–37, 2018. 1st International Conference on Natural Language and Speech Processing.

[101] "textdistance - pypi." Available at https://pypi.org/project/textdistance/ (accessed on February 2023).

[102] "Comparison of the text distance metrics." Available at https://activewizards.com/blog/comparison-of-the-text-distance-metrics/ (accessed on February 2023).

[103] D. D. Prasetya, A. P. Wibawa, and T. Hirashima, "The performance of text similarity algorithms," International Journal of Advances in Intelligent Informatics, vol. 4, no. 1, pp. 63–69, 2018.

[104] S. Jimenez, C. J. Becerra, A. F. Gelbukh, and F. A. González, "Generalized mongue-elkan method for approximate text string comparison.," in CICLing, vol. 9, pp. 559–570, Springer, 2009.

[105] "Bluetooth meshnetworking : Friendship." Available at https://www.bluetooth.com/blog/bluetooth-mesh-networking-series-friendship/ (accessed on February 2023).

[106] "facebook/wav2vec2-large-xlsr-53 hugging face." Available at https://huggingface.co/facebook/wav2vec2-large-xlsr-53 (accessed on February 2023).

[107] "facebook/wav2vec2-base-100k-voxpopuli hugging face." Available at https://huggingface.co/facebook/wav2vec2-base-100k-voxpopuli (accessed on February 2023).

[108] H.-J. Chang, S.-w. Yang, and H.-y. Lee, "Distilhubert: Speech representation learning by layer-wise distillation of hidden-unit bert," 2021.

[109] "Github s3prl." Available at https://github.com/s3prl/s3prl (accessed on February 2023).

[110] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, "Mls: A large-scale multilingual dataset for speech research," ArXiv, vol. abs/2012.03411, 2020.

[111] "openai/whisper-tiny hugging face." Available at https://huggingface.co/openai/whisper-tiny (accessed on February 2023).

[112] "Mozilla Common Voice." Available at https://commonvoice.mozilla.org/gl/datasets (accessed on February 2023).

[113] O. Kjartansson et al., "Open-Source High Quality Speech Datasets for Basque, Catalan and Galician," in Proc. of SLTU and CCURL, (Marseille, France), pp. 21–27, May 2020.

[114] "ifrz/gl_wav2vec2-large-xlsr-53 hugging face." Available at https://huggingface.co/ifrz/gl_wav2vec2-large-xlsr-53 (accessed on February 2023).

[115] "ifrz/gl_wav2vec2_base hugging face." Available at https://huggingface.co/ifrz/gl_wav2vec2_base (accessed on February 2023).

[116] "ifrz/gl_wav2vec2_distilled hugging face." Available at https://huggingface.co/ifrz/gl_wav2vec2_distilled (accessed on February 2023).

[117] "ifrz/whisper-gl-tiny hugging face." Available at https://huggingface.co/ifrz/whisper-gl-tiny (accessed on February 2023).

[118] "The Khronos Group - ncnn brings neural network inference acceleration using vulkan." Available at https://www.khronos.org/news/permalink/ncnn-brings-nerual-network-inference-acceleration-using-vulkan-5c9200795dbf06.59317995 (accessed on February 2023).

[119] "Quantize onnx models | onnxruntime." Available at https://onnxruntime.ai/docs/performance/quantization.html (accessed on February 2023).

[120] I. Froiz-Míguez, Ó. Blanco-Novoa, P. Fraga-Lamas, D. Fustes, C. Dafonte, J. Pereira, and T. M. Fernández-Caramés, "Design and evaluation of a cross-lingual ml-based automatic speech recognition system fine-tuned for the galician language.," in Proceedings of V XoveTIC Conference, vol. 30, pp. 152–155, 2022.

[121] "Joulescopeusersguide_v1_1.pdf." Available at https://download.joulescope.com/docs/JoulescopeUsersGuide/JoulescopeUsersGuide_v1_1.pdf (accessed on February 2023).

[122] "nrf52833 - nrf52833_ps_v1.5.pdf." Available at https://infocenter.nordicsemi.com/pdf/nRF52833_PS_v1.5.pdf (accessed on February 2023).

[123] S. Trifunovic, S. T. Kouyoumdjieva, B. Distl, L. Pajevic, G. Karlsson, and B. Plattner, "A decade of research in opportunistic networks: Challenges, relevance, and future directions," IEEE Communications Magazine, vol. 55, no. 1, pp. 168–173, 2017.
[124] A. Martín-Campillo, J. Crowcroft, E. Yoneki, and R. Martí, "Evaluating opportunistic networks in disaster scenarios," Journal of Network and computer applications, vol. 36, no. 2, pp. 870–880, 2013.
[125] "Mobilenetv3-pytorch | github." Available at https://github.com/leaderj1001/MobileNetV3-Pytorch/blob/master/README.md (accessed on February 2023).
[126] "Android - nnapi | onnxruntime." Available at https://onnxruntime.ai/docs/execution-providers/NNAPI-ExecutionProvider.html#supported-ops (accessed on February 2023).

IVÁN FROIZ-MÍGUEZ received his M.Sc degree in Computer Engineering at the University of A Coruña (UDC) in 2016. From 2013 to 2019 he worked as DevOps and technical support engineer for companies such as Inditex, Sergas and Euskaltel. Since 2019 he is part of the Group of Electronic Technology and Communications (GTEC), Department of Computer Engineering (UDC) where he is attending the PhD program. His current research interests include Industry 4.0, wireless technologies, Internet of Things (IoT), Deep/Machine Learning, Fog and Edge Computing, Cybersecurity, Distributed Ledger Technology (DLT) and blockchain.

PAULA FRAGA-LAMAS (Senior Member, IEEE) received the M.Sc. degree in Computer Engineering from the University of A Coruña (UDC) in 2009, and the M.Sc. and Ph.D. degrees in the joint program Mobile Network Information and Communication Technologies from 5 Spanish universities: University of the Basque Country, University of Cantabria, University of Zaragoza, University of Oviedo and University of A Coruña, in 2011 and 2017, respectively. She holds an MBA and postgraduate studies in business innovation management (Jean Monnet Chair in European Industrial Economics, UDC), sustainability and social innovation (Inditex-UDC Chair of Sustainability). Since 2009, she has been with the Group of Electronic Technology and Communications (GTEC), Department of Computer Engineering (UDC). She has over 100 contributions in indexed international journals, conferences, and book chapters, and holds 4 patents. She has been included in 2019, 2020 and 2021 in the World's Top 2% Scientists, a study led by Stanford University that lists the 161,000 scientists worldwide with the highest impact publications. She has also been participating in over 40 research projects funded by the regional and national government as well as R&D contracts with private companies. She is actively involved in many professional and editorial activities, acting as reviewer, advisory board member, topic/guest editor of top-rank journals and TPC member of international conferences. Her current research interests include mission-critical scenarios, Industry 4.0/5.0, Internet of Things (IoT), Cyber-Physical Systems (CPS), Augmented/Mixed Reality (AR/MR), fog and edge computing, blockchain and Distributed Ledger Technology (DLT), and cybersecurity.

TIAGO M. FERNÁNDEZ-CARAMÉS (S'08-M'12-SM'15) works as a professor for the University of A Coruña (UDC)(Spain), where he obtained his MSc degree and PhD degrees in Computer Science in 2005 and 2011. His current research interests include IoT/IIoT systems, RFID, wireless sensor networks, extended reality, embedded systems and blockchain, as well as the different technologies involved in the Industry 4.0/5.0 paradigms. In such fields, he has contributed to more than 110 papers for JCR-indexed journal articles, peer-reviewed conferences and book chapters. Due to the impact of his publications, he has been included in 2019, 2020 and 2021 in the World's Top 2% Scientists, which lists the 2% scientists with most impact according to a study led by Stanford University (only 161,000 worldwide scientists are listed in the rank). In the same study, since 2020, he was among the 2% of scientists with more impact throughout their entire career. Moreover, due to his expertise in the previously mentioned fields, he has acted as peer reviewer and guest editor for different top-rank journals, and as project reviewer for the European Union and for national research bodies from Austria, Croatia, Latvia or Argentina.