Mobile Networks and Applications

LADEA: a software infrastructure for audio delivery and analytics

--Manuscript Draft--

Manuscript Number:	
Article Type:	SM 287 - Smart Objects and Technologies for Social Good (GoodTechs 2020)
Keywords:	LoRa; MQTT; IoT; Sound analytics; Deep Learning; Autoenconder
Corresponding Author:	Pietro Manzoni (SM 268), Ph.D. Valencia, Valencia SPAIN
First Author:	Kiyoshy Nakamura
Order of Authors:	Kiyoshy Nakamura
	Daniel Hernandez
	José M. Cecilia
	Pietro Manzoni (SM 268), Ph.D.
	Marco Zennaro
	Juan Carlos Cano
	Carlos T. Calafate
Abstract:	The LoRa technology enables long distance links with reduced power consumption at low cost, the main limitation being the low bandwidth that it offers. With LoRa, remote locations, like rural areas, can benefit from connectivity based services that would otherwise be impossible. In this work, we describe a LoRa architecture that can include generic external data sources using an MQTT-based interface. We particularly focus on audio sources aiming to two basic services: a voice messaging system that allows users who cannot read or write to send voice notes, and an audio compression service to extract the main features from the audio input to use it for developing intelligent ML-based audio analytics.

Mobile Networks and Applications manuscript No.
(will be inserted by the editor)

LADEA: a software infrastructure for audio delivery and analytics

Kiyoshy Nakamura · Daniel Hernández · José M. Cecilia · Pietro Manzoni · Marco Zennaro · Juan-Carlos Cano · Carlos T. Calafate ·

Received: DD Month YEAR / Accepted: DD Month YEAR

Abstract The LoRa technology enables long distance links with reduced power consumption at low cost, the main limitation being the low bandwidth that it offers. With LoRa, remote locations, like rural areas, can benefit from connectivity based services that would otherwise be impossible. In this work, we describe a LoRa architecture that can include generic external data sources using an MQTT-based interface. We particularly focus on audio sources aiming to two basic services: a voice messaging system that allows users who cannot read or write to send voice notes, and an audio compression service to extract the main features from the audio input to use it for developing intelligent ML-based audio analytics.

Keywords LoRa, MQTT, IoT, Sound analytics, Deep Learning, Autoencon-der

1 Introduction

The global penetration rate of Internet usage increased from nearly 17% in 2005 to over 53% in 2019, according to the ITU/UNESCO Broadband Commission for Sustainable Development (2019) report. However, this ratio is not balanced on the planet. In fact, only 19% of the population in developing countries, such as many regions of Africa, are connected to the Internet while this ratio reaches up to 87% in most of the developed countries such as Europe.

Kiyoshy Nakamura, Daniel Hernández, José M. Cecilia, Pietro Manzoni, Juan-Carlos Cano, Carlos T. Calafate
Universitat Politécnica de Valéncia, Valencia, SPAIN
E-mail: pmanzoni@disca.upv.es

Marco Zennaro
ICTP, Trieste, ITALY
E-mail: mzennaro@ictp.it

In places with low connection rates there are many remote areas where providing connection services is a challenge. Therefore, citizens cannot benefit from any application with minimum bandwidth requirements such as simple messaging services. For example, Buehler et al. (2013) provides an analysis of the communication needs in rural health care in several developing countries. An example application is for pregnant women with cell phones who regularly receive regular SMS text messages, reminding them of appointments and allowing them to call health care providers for advice on specific problems, avoiding patients in isolated areas spending a lot of time and resources to reach the nearest hospital.

In this paper, we introduce LADEA, a generic software infrastructure for deliverying audio messages and enabling audio analytics. LADEA is partially based on our previous work (Nakamura et al. (2020)), where we introduced a voice messaging system based on LoRa (Chaudhari BS (2020)), with the possibility to include generic external sources of data using an MQTT based interface. There are in the literature other works that combine these technologies to provide connectivity related services, like Bhawiyuga et al. (2019) who proposed the design of the LoRa-MQTT gateway device for supporting the sensor-to-cloud data transmission in smart aquaculture IoT application, but also Spinsante et al. (2017); Huang et al. (2019); Paolini et al. (2020); Lachtar et al. (2020). In this work, we focus particularly on audio sources aiming to two basic services: a voice messaging system that allows users who cannot read or write to send voice notes, and an audio compression algorithm to extract the main features from the original audio to be used for developing intelligent cloud-based applications based on machine learning methods. We describe how the system was integrated in the existing platform and present some results on its performance.

The inclusion of illiterate people, i.e. people that cannot read and write, in this type of systems is still a critical goal. Our proposal combines a visual interface with an edge solution for data compression to include voice messages in the system. Audio analytics ML-based solutions are growing too, allowing the design of application like noise pollution (e.g., Cornelius et al. (2020)) or sound levels forecasting (e.g., Navarro et al. (2020)). We consider that these results are promising and describe a tool that can provide a useful service at a low cost. The paper is organized as follows. Section 2 describes the existing LoRa based messaging system providing details of the integration of a MQTT proxy in the system. It shows the architecture of the voice recording source and the data compression strategy used for machine learning analysis. Section 3 shows the experimental results of our architecture before concluding the paper and providing some direction for future work in Section 4.

2 LADEA description

This section introduces the architecture and main modules of LADEA. First of all, the software infrastructure based on LoRa is presented. Then, we briefly describe the module in charge of integrating external sources such as audio before introducing the main algorithms and applications developed for (1) sending audio messages to strength human interactions and (2) enabling further audio analytics.

2.1 System architecture

The overall architecture of the messaging system is detailed in Nakamura et al. (2020). At the core there are dedicated devices, called *hubs*, that create the connectivity spot inside an area. The hubs have both a WiFi (IEEE 802.1b/g/n), and a LoRa transceiver. The hubs work as standard WiFi access point to provide connectivity to close by devices. The interface with the messaging application is a web based system. The user can decide whether to send a text message to either a specific destination or to all reachable users, or to check for incoming messages stored in the hub. The hubs offer a REST interface to the connected devices to either send a message, or return previously received and locally stored ones.

Every user needs to "register" before exchanging any message. Registration is required to allow the system to localize end-points. When a user sends a message, the local hub "learns" that that user is connected through it, and creates an entry in a table. The first step is to discover where the destination user is located. To this end, the hub sends a broadcast message to all the surrounding devices and waits for the searched one to respond. A special *broadcast* user was included for messages that are to be delivered to all the registered users.

At this point, using a reliable unicast protocol, messages are transferred and stored in the destination hub. Once the user to whom the message is addressed to checks for available messages, he or she will receive the one stored in the local hub. The unicast protocol is based on a stop-and-wait ARQ approach with a dynamic and adaptive value for the re-transmission delay. The protocol ensures that information is not lost due to dropped packets and that packets are received in the correct order.

2.2 Integration of external sources

We integrated in the platform described before the data collection from external sources by introducing data from anything from any external source such as a simple temperature sensor or an audio/voice recorder. Actually, the latter has been the objective of this work. A dedicated service called `MQTTproxy` is defined in the platform. This service is attached to the system as a specific client. The general idea is that the external source has to package the content it is providing as a structured piece of information, and send it to the `MQTTproxy` as a message. The `MQTTproxy`will then (1) unpack the message, (2) build a proper MQTT message, and (3) publish it to the broker being used. The device that executes the `MQTTproxy` service has to be connected to a hub using WiFi and must clearly have a connection to the used broker, either through the Internet or through a direct TCP/IP link.

The set-up required is the one shown in Figure 1. The external source is attached to the "Hub 1" through a WiFi link. It is important to note that Bluetooth connection could be also possible. External sources are integrated in the platform and "talk" with hubs using a REST interface. The sequence is basically the same used by regular clients: there is first a registration phase followed by a "Push" phase. The transferred data is structured as a JSON object with the format indicated in Listing 1.

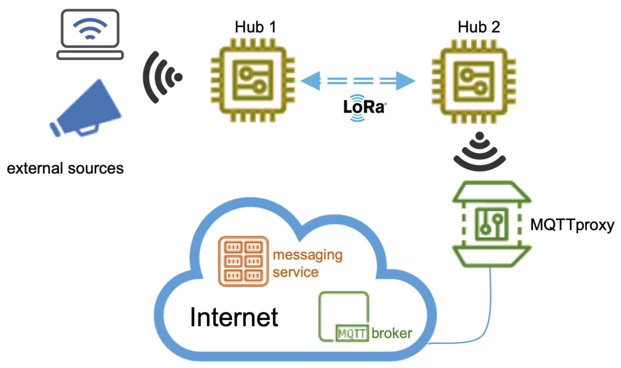

Fig. 1: Connections structure between the external source and the `MQTTproxy`.

```
{
    'DEV_ID': 'voice_recorder',
    'QOS': 0,
    'TOPIC': 'rasp1095/voicemessage',
    'VALUE': {'sender': 'pietro',
              'receiver': 'miguel'
              'message': <data file>}
}
```

Listing 1: The structure of the messages interchanged.

The data contained in the JSON object can have a variable size, limited only by low bandwidth of LoRa channel though. Topics average length can range between 10 to 50 bytes, while values can have a variable size, from a few bytes to hundreds of kilobytes, as in our case with the audio sources. Actually, Section 3 will show performance evaluation of our architecture with messages of up to 100kbytes.

The localization of the `MQTTproxy` is based on an "anycasting" approach. This means that there can be several `MQTTproxys` available in the area covered by any hub. As for regular clients, the hub that received the JSON message will start the search for an `MQTTproxy` as if they were regular end users. If multiple replays are received, the first one is selected. Other strategies could be adopted based, for example, on the detected load of a certain `MQTTproxy` device. The hub will forward the JSON message sent to it to the selected `MQTTproxy` hub using the standard procedure used by the messaging system. The `MQTTproxy`will periodically probe, using the REST endpoint, the hub that is connected with to obtain the data. Figure 2 graphically describes this operation.

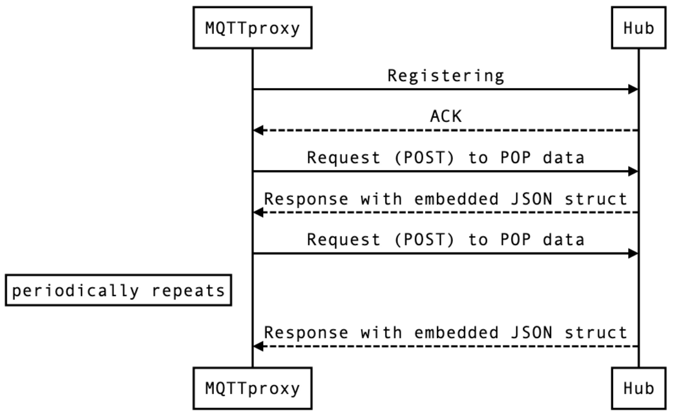
MQTTproxy
Hub
Registering
ACK
Request (POST) to POP data
Response with embedded JSON struct
Request (POST) to POP data
periodically repeats
Response with embedded JSON struct
MQTTproxy
Hub

Fig. 2: The MQTTproxy getting data from the hub.

Once the message is obtained, the `MQTTproxy` will extract the JSON and create a proper "publish" message to the connected broker. Published messages will be received by a specific "messaging service" available in the cloud (see Figure 1) that will forward it to the destination through email or other means. The obtained reply will be handled in the same way, by having the "messaging service" publish it to the broker and the `MQTTproxy` sending it back to the sender.

2.3 Sending audio message for human interactions

This section details the design and user interface of an external voice recording source. One of the main objectives of our infrastructure is to allow people who cannot read or write to send voice messages. To this end, we design an easy-to-use and effective interface. The prototype was designed on a Raspberry Pi 3 Model B+ device with an add-on 3.5-inch LCD touchscreen (LcdWiki[1]) (see

[1] https://www.lcdwiki.com/

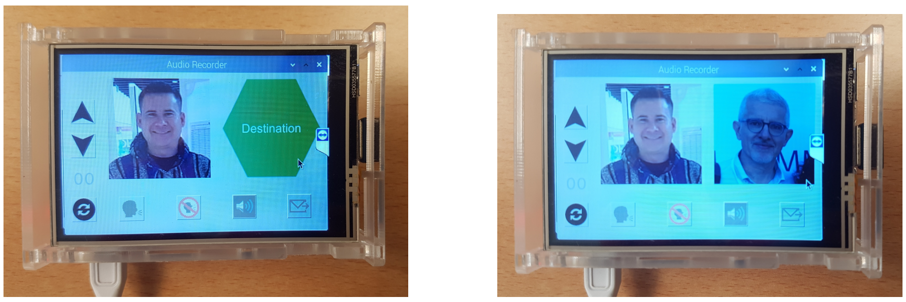

Fig. 3: Screens sequence when identifying the sender and the receiver.

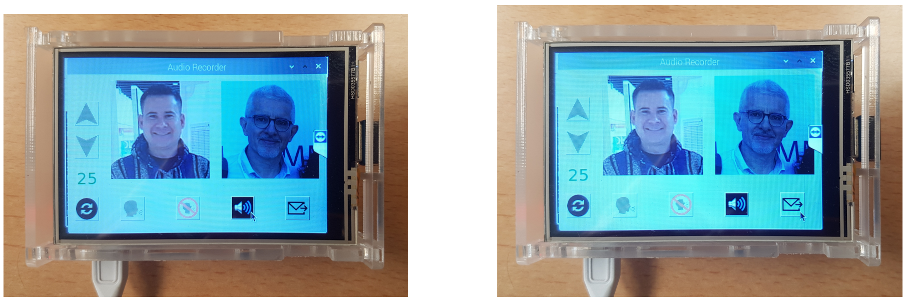

Fig. 4: Icon to play the voice clip (left) and icon to send (right) enableds.

Figure 3). This prototype has a graphical interface that displays a series of icons that will be enabled when necessary, to guide the user step by step.The user has to identify the source and destination of the recording for the message to be sent. With the help of the 'up' and 'down' icons, the user will navigate through a series of pictures of the already registered users. The user will first identify the sender, by clicking on their own picture, and then, in the same way, select the destination user (see Figure 3).

After identifying the source and destination users, the recording icon will be activated. Clicking it starts the recording process. The recording time is displayed and the icon is activated to stop this process. Voice clips of up to 50 seconds or less can be recorded. This value has been set based on the trade-off between the information that can be sent and the total time needed to send it. Once the message is recorded, the icons are activated to play the voice clip or to send it (see Figure 4).

The voice clip is stored in the microSD memory of the Raspberry as a WAV file; a 50 seconds message typically has a size of 1 MB. To compress the WAV file, we used the FFmpeg[2] framework to produce an mp3 file. The advantage of mp3 files is that they do not lose audio quality and their size decreases up to a factor of ten. The compression time clearly grows as the recording time increases. Tests were performed with audio clips of 10 s, 20 s, 30 s, 40 s and 50

[2] https://ffmpeg.org/

s; a 10 sec WAV file requires 0.69 sec up to the 1.40 sec required to compress a 50 seconds audio clip.

At this point the system already have all the required information, namely the source, the destination and the voice message, so that the JSON file can be prepared to be sent to the `MQTTproxy`.

2.4 Sending audio for data analytics

Compressing audio for data analytics is necessary to send it through a low-bandwidth connection like LoRa. The goal is to extract the main features from the original audio to use these features for developing intelligent applications based on machine learning methods. To this end, it is necessary to train a model capable of characterizing, compressing and reconstructing an audio recording. With the approach used in the previous subsection, the audio file suffers a quality loss that is not perceivable by the human ear, but can lead to a loss of performance in machine learning models (see Das (2020)). Therefore, we developed a lightweight deep learning auto-encoder model which compresses the sound stream from the original WAV audio so that limited "data loss" is obtained.

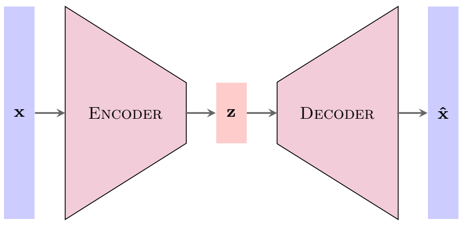

Fig. 5: The schema of the used Hinton and Salakhutdinov autoencoder.

An autoencoder Hinton and Salakhutdinov (2006) is a type of multi-layer neural network characterized by a certain structure formed by layers that decrease its dimensionality from the input layer to the central layer, and then symmetrically increase the number of neurons per layer until it ends up in an output layer of the same size as the initial one (see Figure 5). The central layer of an autoencoder can also serve as the input layer of a neural network that is trained for a different purpose than data compression. Moreover, autocoders are trained in an unsupervised way. The sender encodes the message so that it is smaller in size than its input. The main objective of the receiver is to reconstruct, in the best possible way, the data being received. Therefore, it is an online technique that has been used with files containing audio, to work by musical audio modeling by compressing the spectrogram Vedal (2019). The complete model is shown in the Figure 6. It is formed by linear layers totally connected with a normalization layer between each one of them in order to use a higher training rate and get the model to converge as soon as possible.

The training stage of the neural network previously described has been performed in a HPC server with two GPUs GeForce RTX 2080 using PyTorch lirabry[3]. The main parameters of the training procedure were 1280 epochs, Adam optimizer and learning rate of $1e^4$. The model has a total size of 256076000 trained parameters. The data used for the training has been taken from CSS10 dataset Park and Mulc (2019), which is a collection of single speaker speech datasets for 10 languages. Each of them consists of audio files recorded by a single volunteer and their aligned text sourced from LibriVox. All 6515 audio samples extracted for the training are made up of audios of about 10 seconds, which have been divided into sub-samples that fit the size of the input layer of the network.

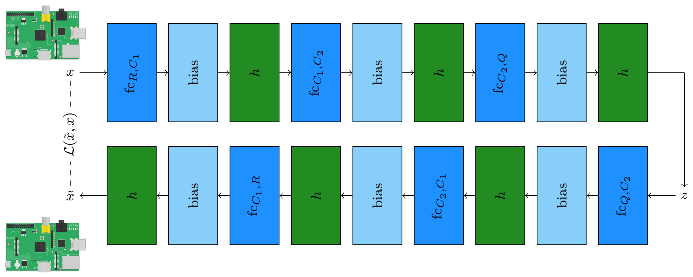

Fig. 6: Architecture of the autocoder sound compressor.

Each of the input audios has been preprocessed using the discrete Fourier transform of a real sequence to convert the sample data into time to frequency. The transformation into the frequency domain shows how the amplitude varies with frequency. This is often referred to as the frequency spectrum of the signal. A single frequency sine wave in the time domain will give a single line to that frequency in the frequency domain. A peak in the time domain will give a spread of frequencies in the frequency domain. And the reverse process has been performed for the reconstruction of the audio file after it has been processed by the auto-encoder

Once the model has been trained, it is compiled on the Raspberry Pi based platforms in order to run on these ARM architectures. After the training stage, the model is serialized and deployed simultaneously in the emitter and receiver (see Figure 6). In this way, the emitter will run the compression from the input layer to the smallest layer to compress the information. The receiver will run the part of the model from the middle layer normalization layer to the network output.

[3] https://pytorch.org

3 Evaluation and discussion

This section analyzes and discusses two different experiments. First, the performance results obtained with the vocal messaging system are presented by varying the distance between the hubs and the size of the sent messages. Second, the data compression strategy for machine learning development is evaluated under different compression rates.

3.1 Evaluation of the vocal messaging system

Two hubs were used to analyze the performance of our system as shown in Section 2. We tested various distances, namely 1m, 750m, and 6000m (6km). The 1m test is provided to have reference values to be compared with the longer distance tests. The 750m tests where performed in the "Ciudad de las Artes y las Ciencias" (Valencia, Spain), while the 6km test where performed between two viewpoints in Chiapas, Mexico. These scenarios are located in areas with a clear line of sight between the two points.

The system performance was measured using the successful transfer time (STT) metric. This metric measures the transfer time of a message from the sender's point of view, and is computed from the moment the first message fragment is sent, to the moment the last ACK of the last message fragment is received. All tests were conducted using a spreading factor of 7 (SF7) to minimize the time in the air. With this in mind, bursts of 10 messages were sent to determine the stability of the system. The system performance was stable and almost identical to that of the shorter distance tests. Re-transmissions were rare, even in the long-range experiments, having a negligible impact on the STT. It should be noted that delays are of the order of hundreds of seconds and therefore few more seconds do not affect the usability of the system, thus no effect was seen in the delivery of the voice messages.

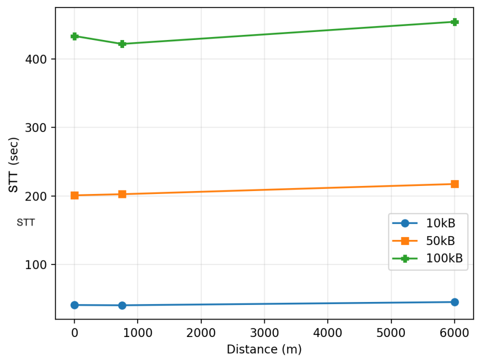

Fig. 7: Behavior of the STT versus distance between two nodes.

Figure 7 shows the behavior of the STT versus the distance between two nodes. An almost constant behavior can be observed in the results, although the STT clearly grows as long as the message size increases. The system is quite stable at increasing distance and very few re-transmissions were required during the experiments. For instance, the maximum delay obtained for 100KB messages was 457.56 seconds and the minimum delay was 451.49 sec (i.e. about 7.5 minutes).

Certainly, the worst aspect is the low performance obtained due to the use of the LoRa. This is compensated by the long range obtained and the low energy cost required by these devices, which makes it a frugal solution to a clear problem. We are aware of the limitations imposed by LoRa in relation to the use of the channel, which is called "duty cycle". 100kB messages may violate this rule if the duty cycle is applied to 1 hour slots, as suggested. In our case, we suppose, without loss of generality, that the targeted areas will not be too crowded; i.e. there will not bee too high density of devices, thus limiting the possible impact of these long messages. There is anyway the need, as future work, to provide some form of sending scheduling to avoid possible interference with other LoRa sources.

3.2 Evaluation of the audio compression

This section analyzes the data compression algorithm introduced in Section 2.4. The artificial neural network model designed was 980 MB in total. This is divided in two different neural networks that were deployed in the emitter and receiver respectively. The first neural network encodes the message and, therefore, it is deployed in the sender. The second neural network decodes the message and is deployed in the receiver. This means that each platform requires less than 500 MB for running the compression model, which is small enough to be run on a Raspberry-like platform. Our method provides up to 40% of compression rate of the original WAV file and this rate is kept constant as the size of input file increases.

Figure 8 shows the normalized waveform of the original audio record and the one obtained after decompression with our approach. It is important to note that the regenerated audio matches almost 95% the original record. Moreover, this procedure enables machine learning analysis a posterior, and thus additional data mining procedures could be developed.

Finally, the compression and decompression procedures with our approach have similar execution time, and they scale linearly along with the audio record size. For instance, for an audio of 29 seconds, with sample rate of 11025 and final sampling of 320828 values the encoder phase takes 32,21 seconds and the decoder needs 40 seconds.

4 Conclusions

In this paper we described a LoRa architecture that can include generic external data sources using an MQTT-based interface. We particularly focused

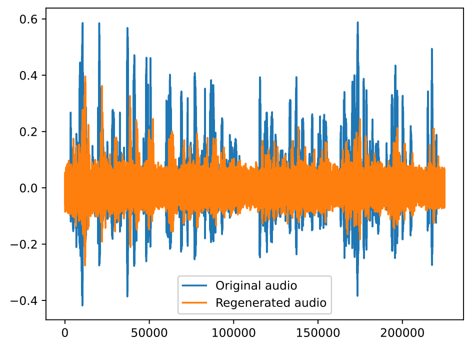

Fig. 8: Normalized waveform of original and predicted .

on audio sources aiming to two basic services: a voice messaging system that allows users who cannot read or write to send voice notes, and an audio compression service to extract the main features from the original audio to be used for developing intelligent ML-based audio analytics. The inclusion of illiterate people, that is people that cannot read and write, in this type of systems is still a critical goal, and simplifying the user interactions is highly demanded. Audio analytics ML-based solutions are growing too, allowing the design of application like noise pollution or sound levels forecasting. We described how the system was integrated in the existing platform and presented some results on its performance. We consider that these results are promising and describe a tool that can provide a useful service at a low cost.

Acknowledgment

This work has been partially supported by the Spanish Ministry of Science and Innovation, under the Ramon y Cajal Program (Grant No. RYC2018-025580-I) and under grants RTI2018-096384-B-I00, RTC-2017-6389-5 and RTC2019-007159-5, by the Fundación Séneca del Centro de Coordinación de la Investigación de la Región de Murcia under Project 20813/PI/18, and by the “Conselleria de Educación, Investigación, Cultura y Deporte, Direcció General de Ciéncia i Investigació, Proyectos AICO/2020”, Spain, under Grant AICO/2020/302.

References

Bhawiyuga A, Amron K, Primanandha R, Kartikasari DP, Arijudin H, Prabandari DA (2019) Lora-mqtt gateway device for supporting sensor-to-cloud data transmission in smart aquaculture iot application. In: 2019 International Conference on Sustainable Information Engineering and Technology (SIET), pp 187–190

Buehler B, Ruggiero R, Mehta K (2013) Empowering community health workers with technology solutions. IEEE Technology and Society Magazine 32(1):44–52, DOI 10.1109/MTS.2013.2241831

Chaudhari BS BS Zennaro M (2020) LPWAN technologies: Emerging application characteristics, requirements, and design considerations. Future Internet 12(3)

Cornelius K, Kumar NK, Pradhan S, Patel P, Vinay N (2020) An efficient tracking system for air and sound pollution using iot. In: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), pp 22–25, DOI 10.1109/ICACCS48705.2020.9074301

Das S (2020) Impact of Dithering in Compressed and Spectrally Distorted Speech for Emotion Recognition. International Journal of Advanced Research in Basic Engineering Sciences and Technology (IJARBEST) (August), DOI 10.20238/IJARBEST.2020.0610005

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507, DOI 10.1126/science.1127647

Huang A, Huang M, Shao Z, Zhang X, Wu D, Cao C (2019) A practical marine wireless sensor network monitoring system based on LoRa and MQTT. In: 2019 IEEE 2nd International Conference on Electronics Technology (ICET), pp 330–334

ITU/UNESCO Broadband Commission for Sustainable Development (2019) The state of broadband 2019. ITU/UNESCO, Report

Lachtar A, Val T, Kachouri A (2020) Elderly monitoring system in a smart city environment using lora and mqtt. IET Wireless Sensor Systems 10(2):70–77

Nakamura K, Manzoni P, Zennaro M, Cano JC, Calafate CT (2020) Adding voice messages to a low-cost long-range data messaging system. In: Proceedings of the 6th EAI International Conference on Smart Objects and Technologies for Social Good, Association for Computing Machinery, New York, NY, USA, GoodTechs '20, p 42–47, DOI 10.1145/3411170.3411238

Navarro JM, Martínez-España R, Bueno-Crespo A, Martínez R, Cecilia JM (2020) Sound levels forecasting in an acoustic sensor network using a deep neural network. Sensors 20(3):903

Paolini C, Adigal H, Sarkar M (2020) Upper bound on lora smart metering uplink rate. In: 2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC), pp 1–4

Park K, Mulc T (2019) CSS10: A collection of single speaker speech datasets for 10 languages. CoRR abs/1903.11269

Spinsante S, Ciattaglia G, Del Campo A, Perla D, Pigini D, Cancellieri G, Gambi E (2017) A LoRa enabled building automation architecture based on MQTT. In: 2017 AEIT International Annual Conference, pp 1–5

Vedal AH (2019) Unsupervised Audio Spectrogram Compression using Vector Quantized Autoencoders