NoiseAware: System for Real-Time Noise Monitoring in Smart Cities

Maurício Moreira Neto[2], **Francisco Gomes**[1,2,3], **Vitória Silvestre** [1]

[1]Group of Computer Networks, Software Engineering and Systems (GREat)
[2] Graduate Program in Computer Science (MDCC)
Federal University of Ceará – Fortaleza, CE – Brasil

[3] Engine Lab
Federal University of Ceará – Crateús, CE – Brasil

maumneto@ufc.br, almada@crateus.ufc.br, vitorianicolau3@gmail.com

Abstract. *The Internet of Things (IoT) is an emerging paradigm that combines several technologies. IoT provides the interconnection of physical devices for exchanging information among them and provides services to the users. The urban environment is susceptible to various kinds of noise from different sources. Therefore, a system is needed to monitor the urban environment's noise level. NoiseAware is a noise monitoring system for smart cities. NoiseAware monitors the noise levels of a particular location and sends an alert to the competent bodies to operate on site when the level reaches the allowable threshold. According to the literature review conducted, we did not find any work related to noise monitoring that has systems integration, accords real-time requests, and uses more than one communication protocol. NoiseAware has considerable service availability (approximately 81%) for system users, and the load tests show it has relevant request capability.*

1. Introduction

The emergence of the Internet of Things (IoT) can be attributed to the advancements in wireless sensor network technologies, embedded systems, and microelectronics [Junaid et al. 2017, Corotinschi and Găitan 2015]. IoT is a paradigm that facilitates the interconnection of physical devices to exchange information and provides services for users [Junaid et al. 2017, Bikmetov et al. 2017, Kang et al. 2016]. These devices monitor and act on the environment where they are placed, making IoT applicable in various areas such as smart cities, precision agriculture, e-health, and others.

Environmental risks, such as sound and air pollution, are prevalent in all countries and are treated as essential targets of current urban monitoring [Al-Saloul et al. 2015]. Traditional methods for measuring noise pollution levels involve noise measurement stations or wireless sensor networks, which have limitations regarding distance, maintenance, and management, making them expensive regarding human and financial resources. Therefore, there is a need for new low-cost strategies that can assist in measuring noise pollution levels in urban centers [Lokhande and Mohsin 2017, Hong et al. 2017].

To address the problems presented and improve urban noise monitoring in smart cities, we propose the NoiseAware system. NoiseAware is a noise monitoring system that utilizes sensor nodes throughout the city to monitor noise levels in various locations. If

the system detects that the noise level exceeds the threshold allowed by law, an alert will be sent to the relevant authorities to investigate the violator. This approach eliminates the need for local inhabitants to report infractions. In addition, it has a history of noise records in the locality.

The remainder of this article is organized into the following sections. Section 2 describes the NoiseAware system in detail, including its architecture and functionality. Section 3 presents the materials, goals, and discussion of the results of experiments conducted with the NoiseAware system. Section 4 discusses related work on noise monitoring in urban environments. Finally, Section 5 presents the conclusions and suggests future research directions.

2. NoiseAware Components

In this section, we provide a detailed description of the NoiseAware system. NoiseAware is an urban noise monitoring system, and its architecture comprises four modules, as illustrated in Figure 1.

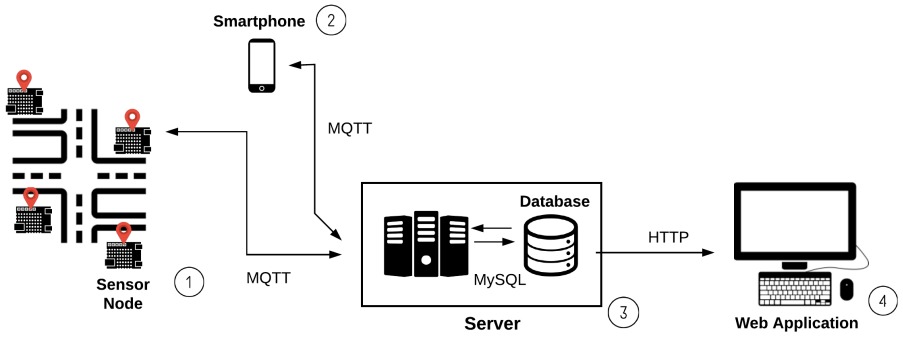
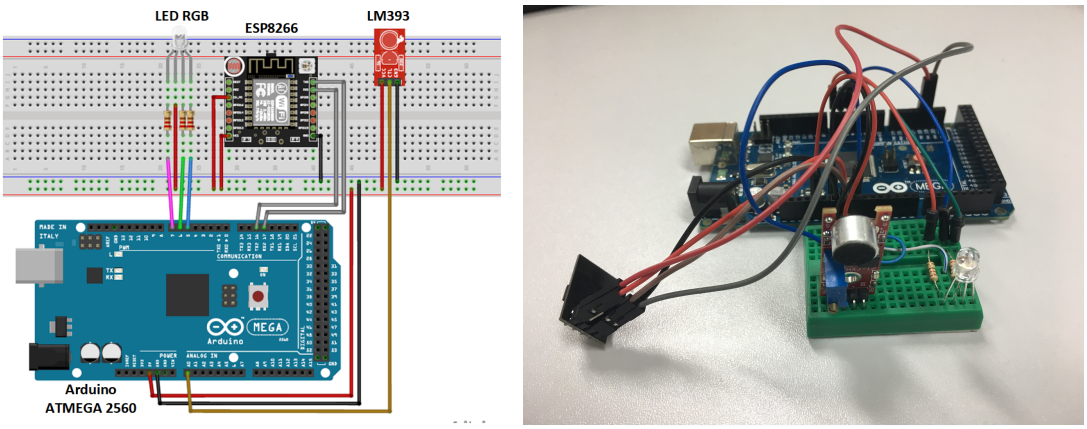


Figure 1. NoiseAware System Architecture.

The first module of the NoiseAware system is the noise monitoring module, where the sensor node is located. The NoiseAware sensor node is designed to capture the noise level of the locality and send the data to target users, such as inspection organs and ordinary users, via the internet. The sensor node communicates with the server using the Message Queue Telemetry Transport (MQTT) protocol. MQTT facilitates communication between the sensor node and the mobile application. Figure 2 shows the components used in the sensor node.

The NoiseAware sensor node comprises a prototype board with a processor, an ambient noise pickup module, a Wi-Fi communication module, and an indicator LED. In Section 3, we will present more details on the components used in the hardware. The green LED indicates the local noise level is within the permitted range. If the noise level changes and exceeds the allowed level, the red LED will light up to signal an infringement.

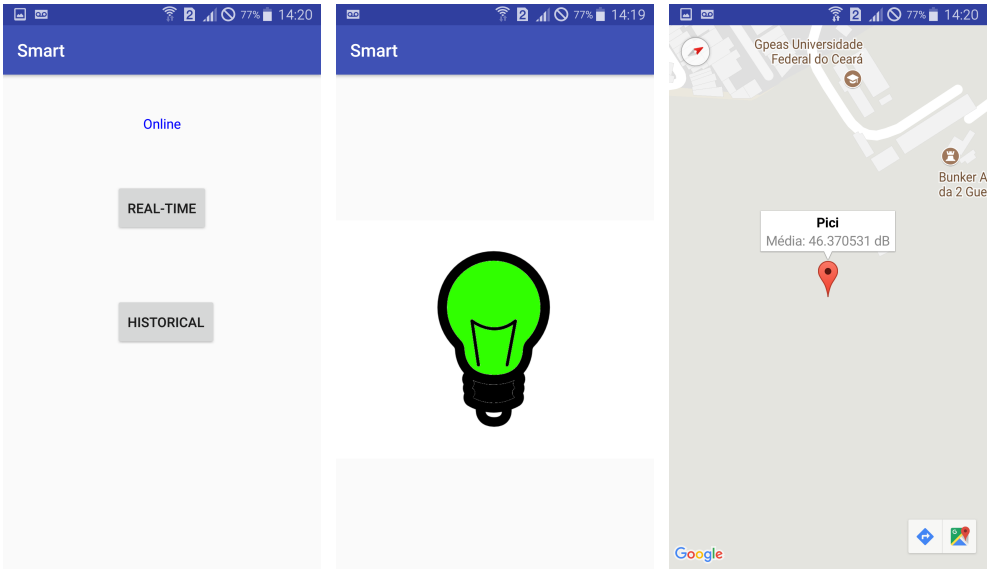
The second module is the mobile application, which is responsible for displaying the location's noise level data in real time and presenting the history of noise stories in the locations where the sensor nodes are installed. The server database is responsible for updating the history data of the locations where the sensor nodes are placed. The mobile application updates the history request using the Hypertext Transfer Protocol (HTTP).



(a) Hardware's schematic. (b) Hardware implemented.

Figure 2. Hardware of Sensor Node.

Figures 3(a), 3(b), and 3(c) show the screenshots of the mobile application. The home screen, represented by Figure 3(a), has two buttons in the center and a label at the top. The label at the top of the screen indicates whether the MQTT Broker is online or offline. When the Broker is activated, the application shows the real-time noise data of the monitored location via the "REAL-TIME" button. If the Broker is not activated, the application will only have the option to display the history of noise levels of the location using the "HISTORICAL" button.



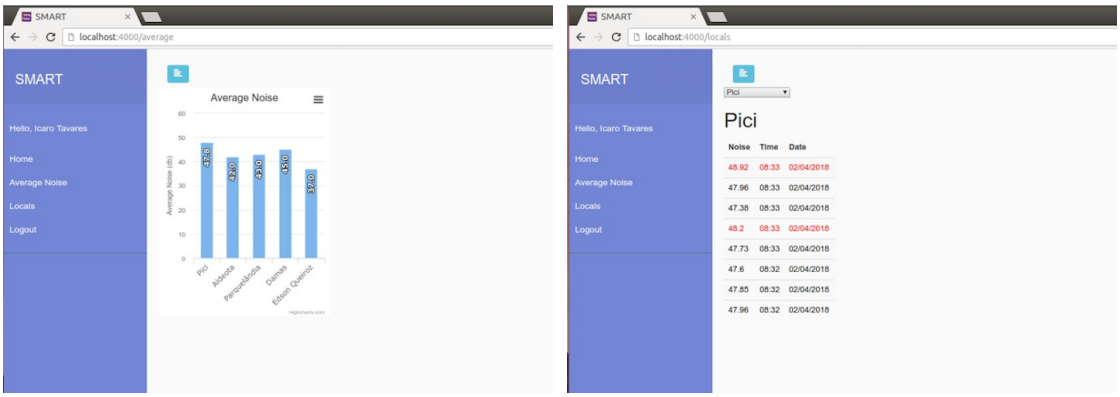
(a) App home screen. (b) REAL-TIME button. (c) HISTORICAL button.

Figure 3. Screenshots of the NoiseAware mobile application.

Figures 3(a) and (b) show if the decibel level of the location is adequate. This data is provided in real-time by the sound sensor. If the level is below the threshold set as appropriate, the LED will turn green. Otherwise, the LED will turn red, indicating the limit has been reached. Figure 3(c) illustrates the average noise level data of the locations where the various sensor nodes are placed.

The third module is the server, which is responsible for storing and processing the noise level data. The server stores the captured noise data in a MySQL relational database, which serves as input to feed the history of the mobile and web applications. The server sends the data in real time to the web application through the HTTP protocol. The Broker is installed on the server to use the MQTT protocol.

Finally, the fourth module is the web application, which is responsible for showing the inspection organ where the noise has exceeded the limit established by law. The web application indicates the location of the sensor nodes scattered throughout the streets, and when selecting the sensor node, the noise level data of the elected position can be obtained. Figure 4 displays the screenshots of the NoiseAware web application. This module was designed for the inspection organ to monitor, and based on [Marinov et al. 2017], we set the value of 48 dB as a threshold reference (red color) for the noise level.



(a) Average neighborhood noise. (b) Real-time noise data.

Figure 4. Screenshot of the Web Application.

3. Experiments and Results

This section will present the experiments performed to validate the proposed NoiseAware system. The objectives of the experiments are:

- To evaluate the availability of the MQTT Broker;
- To perform a server load test and evaluate its performance with an increasing number of clients.

It will describe the materials and methods used to implement the system and present the results obtained from each experiment. Lastly, we will discuss the results' implications and the study's limitations.

3.1. Materials

The materials used in our experiment were:

- The sensor node is composed of an Arduino ATMEGA2560 together with the LM393 sound sensor module, an ESP8266 communication module and an RGB led (Red-Green-Blue);
- The Broker used is the Mosquitto version 3.1, which is an open-source Eclipse Broker[1];

[1] An open Source MQTT - https://mosquitto.org/

- The mobile device used in our experiments was an LG G3 Beat smartphone with Qualcomm Snapdragon 400 MSM8226 Cortex-A7 1.2 GHz Quad Core, internal memory of 8GB and 1 GB RAM, running Android 5.0.2;
- The server was a laptop running Linux Mint 17.2 64 bit operating system, with 8 GB RAM and Core i5-4200U (1.6 GHz Quad Core) processor. The web service was implemented in Node.js 6.13[2]

3.2. Availability Experiment

System availability is a metric that measures the probability that a system is not failed or undergoing a repair action when it needs to be used. System availability is calculated by dividing uptime by the total sum of uptime and downtime:

$$Availability = \frac{Uptime}{Uptime + Downtime} \times 100\% \quad (1)$$

Where uptime is a computer industry term for the time during which a computer is operational, and downtime is the time when it isn't operational.

Figure 5 shows the noise level recorded by the sensor node in a controlled environment. It is evident from the graph that the noise level reached the threshold within a few minutes after 3 hours and 12 minutes (192 minutes) of measurement, and the average remained consistent with this threshold.

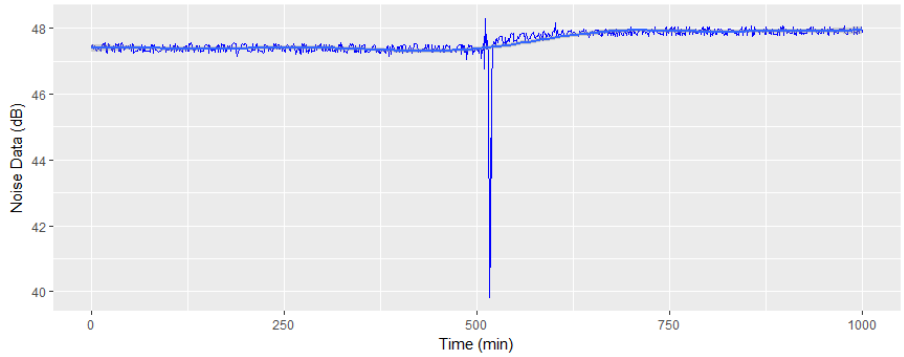


Figure 5. Noise level capture experiment using the NoiseAware system.

NoiseAware ran for 1000 minutes for this experiment. During the experiment, the broker disconnected around 5633 times. Each time this occurred, the NoiseAware sensor node took approximately 2 seconds to reconnect to the broker. Consequently, the NoiseAware was unavailable for a total of approximately 11266 seconds. Although the algorithm is designed to reconnect automatically, this high number of failures can result in data loss during sensor node operation. By converting the values to seconds and using Equation 1, we can determine the impact of these disconnections:

$$Availability = \frac{60000 - 11266}{60000} \times 100\% Availability \qquad = 81,22\% \quad (2)$$

[2]https://nodejs.org

Therefore, the availability was 81,22%, which shows that the NoiseAware has a simple noise capture environment, and due to intermittent capture, availability is impaired. Despite this, the system is helpful for what it proposes, capturing noise and notification when it exceeds the threshold.

3.2.1. Load Test

To conduct the load test, we utilized the loadtest library[3]. The test involved creating a specified number of clients with a concurrent parameter determining how many requests would arrive concurrently at the server. While the requests were not sent in parallel from different processes, they were sent concurrently, allowing a second request to be sent before the first one was answered.

The load test was executed 30 times, each running for 60 seconds. A total of 10, 100, and 1000 concurrent levels were tested in the experiment. Figure 6 depicts the charts of the load test, where it can be observed that the server was able to handle an average of 56220, 90716, and 91600 completed requests, respectively, for each concurrent level (see Figure 6(a)).

Regarding requests per second, the load test showed 937, 1512, and 1526, with an average latency of 10.7 ms, 66.1 ms, and 664.9 ms, respectively. Based on this data, when concurrent levels increase, the request per second increases, and the average latency increases in NoiseAware.

The errors encountered during the load test were 9961, 6800, and 5955, respectively, for 10, 100, and 1000 concurrent levels (see Figure 6(b)). To examine the relationship between the complete requests and errors data and verify the proportion between them, we calculated the following equation:

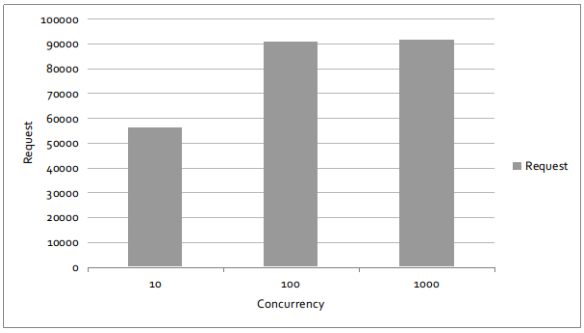
$$Result = \frac{CompleteRequests}{CompleteRequests + TotalErrors} \times 100\% \tag{3}$$

By applying Equation 3 to 10, 100, and 1000 concurrency levels, we calculated the percentages of completed requests and obtained values of 84,94%, 93,02%, and 93,89%, respectively (see Figure 6(c)). From this data, we can conclude that the system's ability to respond to requests correctly increases with the number of concurrent requests, and there is a direct relationship between the total requests and total error metrics. Therefore, the load test results indicate that NoiseAware can meet relevant request capacity requirements.

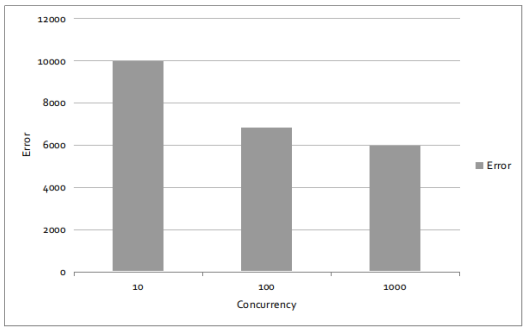
4. Related Work

To support the development of NoiseAware, a state-of-the-art review of noise monitoring systems was conducted. As a result, four papers that present NoiseAware-like applications were selected: City Soundscape [Zappatore et al. 2016], NoiseCo [Al-Saloul et al. 2015], NoiseSense [Qin and Zhu 2016], and Sense2Health [Hachem et al. 2015]. These systems are briefly described below.

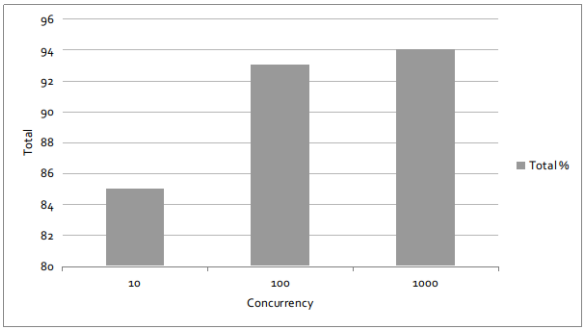
[3]https://github.com/alexfernandez/loadtest



(a) Completed Requests.



(b) Total Errors.



(c) Percentage of Requests for Errors.

Figure 6. Charts of Load Test.

Vanitha et al. (2021) proposed the work for automation of noise detection using Internet things. The work reduce the noise in the way of controlling noise pollution. The main focus of this system is to reduce human voices. The maintenance of the noise is made efficient, as all the recorded voice are stored in the secure database, through which data can be retrieved easily. By this system, the person who speaks in the enclosed area will get a mail to his/her mail address.

City Soundscape is an Android crowdsensing system part of a platform of the same name created by Zappatore et al. (2016) to identify and generate noise maps. The application sends location data and the noise level detected through the user's mobile device sensors to a server. Once in the server database, the platform checks the noise level averages of the regions and then measures the level of noise pollution at each location. This information is used by a noise management web application that generates a noise map and proposes solutions to reduce noise pollution in a region. The City Soundscape application also displays information about the local noise level and allows the customer to send comments about the noise in their location.

NoiseCo is an Android prototype system developed by Al-Saloul et al. (2015) that uses mobile device sensors to monitor the local noise level. The system uses a calibration formula to adjust measurements for different types of mobile devices, and specific techniques are used to smooth the measurements. The application also displays the series of measurements made by the device while the application is still running. Zhaokun and Yanmin (2016) developed NoiseSense, a mobile system for Windows Phone and Android, which is used to obtain noise level data in large cities. The application captures the noise level and sends the data to a web service, along with the device's location. These data are

processed using a measurement technique proposed by the authors to ensure greater precision. Using the data captured by mobile devices and data provided by other sources of measurement, the authors generated a noise map of the city of Shanghai with acceptable accuracy to prove the efficiency of their technique and the effectiveness of crowdsensing applications in monitoring noise pollution in urban centers.

In [Hachem et al. 2015], the authors introduce Sense2Health, a system developed for Android that sends data on the location and noise level identified by the user's mobile device to a server and presents a series of daily averages of measurements. On the server, a crowdsourcing application called GoFlow is responsible for processing the data and providing the averages for Sense2Health. Besides, GoFlow sends the noise level data and the location of the measurements to an urban noise monitoring system also proposed by the authors in the same work.

Table 1 shows a comparison between the related works and NoiseAware. According to the literature review, relevant characteristics of a noise monitoring system include an integrated environment, sensor node, real-time monitoring, and protocols. The integrated environment indicates if the system is composed of more than a single module. The sensor node indicates if there is any module responsible for noise capture. Real-time monitoring refers to whether the system provides monitoring information in real-time. Finally, the protocols column indicates the communication protocols used by the approaches for information exchange.

Table 1. Related Works.

Work	Integrated Environment	Sensor Node	Real-Time	Communication Protocol
Vanitha et al.	✓	✓	✗	HTTP
City Soundscape	✓	✓	✗	HTTP
NoiseCo	✗	✓	✓	HTTP
NoiseSense	✓	✓	✗	HTTP
Sense2Health	✓	✓	✗	HTTP
NoiseAware	✓	✓	✓	**MQTT / HTTP**

According to the table, it is clear that NoiseAware is the only work that possesses all the identified characteristics. It is a multi-module system with mobile, web, server, and sensor node components. It offers real-time and historical noise monitoring information and uses multiple protocols for communication, including HTTP and MQTT.

5. Conclusion

The NoiseAware system is a smart urban noise monitoring solution designed to monitor noise levels in specific city locations using sensor nodes. Whenever the system detects noise levels exceeding the established limit, it automatically alerts the inspection authorities to take appropriate actions, making the system autonomous. By doing so, NoiseAware proactively alerts the rules to excessive noise, reducing the need for residents to report noise pollution.

In future work, we plan to explore other IoT communication protocols, such as the CoAP (Constrained Application Protocol). Additionally, we aim to evaluate the system's

performance on a larger scale by using multiple sensor nodes to monitor noise levels across various locations.

References

Al-Saloul, A. H. A., Li, J., Bei, Z., and Zhu, Y. (2015). Noiseco: Smartphone-based noise collection and correction. In *Computer Science and Network Technology (ICCSNT), 2015 4th International Conference on*, volume 1, pages 369–373. IEEE.

Bikmetov, R., Raja, M. Y. A., and Sane, T. U. (2017). Infrastructure and applications of internet of things in smart grids: A survey. In *2017 North American Power Symposium (NAPS)*, pages 1–6.

Corotinschi, G. and Găitan, V. G. (2015). Smart cities become possible thanks to the internet of things. In *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 291–296.

Hachem, S., Mallet, V., Ventura, R., Pathak, A., Issarny, V., Raverdy, P.-G., and Bhatia, R. (2015). Monitoring noise pollution using the urban civics middleware. In *Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on*, pages 52–61. IEEE.

Hong, Z.-Y., Qiu, Z.-P., Zeng, S.-L., Wang, S.-D., and Sandrine, M. (2017). Research on fusion encryption algorithm for internet of things monitoring equipment. In *Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC), 2017 14th International Symposium on*, pages 425–429. IEEE.

Junaid, M., Shah, M. A., and Satti, I. A. (2017). A survey of internet of things, enabling technologies and protocols. In *2017 23rd International Conference on Automation and Computing (ICAC)*, pages 1–5.

Kang, B., Kim, S., Choi, M.-I., Cho, K., Jang, S., and Park, S. (2016). Analysis of types and importance of sensors in smart home services. In *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on*, pages 1388–1389. IEEE.

Lokhande, D. G. and Mohsin, S. A. (2017). Internet of things for ubiquitous smart home system. In *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, pages 314–320.

Marinov, M., Nikolov, D., Ganev, B., and Nikolov, G. (2017). Environmental noise monitoring and mapping. In *Electronics Technology (ISSE), 2017 40th International Spring Seminar on*, pages 1–7. IEEE.

Qin, Z. and Zhu, Y. (2016). Noisesense: A crowd sensing system for urban noise mapping service. In *Parallel and Distributed Systems (ICPADS), 2016 IEEE 22nd International Conference on*, pages 80–87. IEEE.

Vanitha, C., Sridhar, K., and Dhivakar, R. (2021). Automation of noise detection using internet of things. In *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, pages 184–189.

Zappatore, M., Longo, A., and Bochicchio, M. A. (2016). Using mobile crowd sensing for noise monitoring in smart cities. In *Computer and Energy Science (SpliTech), International Multidisciplinary Conference on*, pages 1–6. IEEE.