Noise Monitoring System Development in a Library Based on The Internet of Things

1st Dania Eridani
Department of Computer Engineering
Diponegoro University
Semarang, Indonesia
dania@ce.undip.ac.id

2nd Adian Fatchur Rochim
Department of Computer Engineering
Diponegoro University
Semarang, Indonesia
adian@ce.undip.ac.id

3rd Alvin Zulham Firdananta
Department of Computer Engineering
Diponegoro University
Semarang, Indonesia
alvinzulham@students.undip.ac.id

***Abstract*— a library is one of the important places for the community, especially students. However, not all visitors know the library's rules and act arbitrarily to create noise that can disturb other visitors. This research is focused on the development of classifying and monitoring the unwanted noise in the library. The system is built with the Arduino Nano 33 BLE microcontroller using the DFROBOT Analog Sound Level Meter Sense sensor and ESP32-WROOM32U. The system is equipped with classification capabilities resulting from machine training using the Convolutional Neural Network algorithm by utilizing a Feature Extraction. The system is then connected to Wi-Fi to be integrated with websites created using the PHP programming language and the Laravel framework. Data from the monitoring will be stored in the MySQL database. The system can give a noise warning when a human or cell phone sound exceeds the threshold with an average of 82.78% classification accuracy and an ideal distance from the sound source, as far as 30-100 cm.**

Keywords—Noise Monitoring System, Internet of Things, Feature Extraction, CNN.

I. Introduction

A library is needed for students, lecturers, and researchers, which makes it an alternative place to search library sources and a place for group scientific activities. But, visitors often do not understand the ethics of being in the library room. Visitors who do not understand the rules make rowdy noises and speak too loudly, causing noise to other visitors and interfering with the concentration of reading or discussions being carried out. This is one of the main reasons for complaints submitted by library visitors to librarians. Librarians, as authorized officers, need to remind visitors to be calm and not to cause noise. Librarians must also carry out their work, from collecting new book data to maintaining books in the library. Of course, continuous warnings to different visitors will significantly interfere with the librarian's work [1][2]. Therefore, a system is needed that can automatically detect and simultaneously provides a notification signal to visitors not to make noise. One way to reduce noise in the library is to use a system that alerts visitors when it reaches the specified noise threshold. However, The problem found when creating a noise monitoring system is in the noise itself.

Sound is a condition of changing pressure or can also be described as the speed of oscillations or frequencies in Units of Hertz (Hz). There are three types of sounds, namely those that have frequencies between 20Hz to 20 kHz (can be heard by human ears), above 20 kHz, and below 20Hz (both of which cannot be heard by human ears) [3].

Noise can be interpreted as unwanted, disliked, and disturbing sounds or can also be interpreted as complex sound vibrations that have various frequencies and amplitudes that are periodic/non-periodic. Noise can be measured logarithmically by units of decibels (dB) which is the energy currency of the unity of the area. Noise can be classified into three types: engine, vibration, air movement, gas, and liquid [4].

The human sense of hearing can listen to sounds within 20 Hz-20kHz. Humans can easily distinguish the types of sounds without making any additional effort. If the machine wants to have the same ability to distinguish the types of sounds, extra effort must be made because the machine has a problem that is often called *machine hearing. Feature extraction* [5] can help the machine recognize sounds.

The system must distinguish which noise is produced by visitors and which is produced by the library environment. Adding Artificial Intelligence to the system allows it to recognize the noise around it. One of the branches of Artificial Intelligence is Machine Learning. Machine Learning has now arrived at the implementation of existing embedded systems [6][7]. Several studies were done on using artificial intelligence in voice recognition, and all use CNN as the algorithm [9]–[11]. However, The application of Artificial Intelligence to embedded systems is limited when operating on devices with limited capabilities [8].

There is also a study about noise monitoring systems in the library using an Arduino Uno-based sound noise detection and warning system made at the Amikom Purwokerto University Library. In the system, the GY-MAX4466 sensor was used as a noise sensor, and a 128x64 pixel OLED screen and speakers functioned as issuing warnings in the form of sound to library visitors around it [12]. The research also built a noise detection system in Arduino-based libraries. The system is built using an LM393D sensor and is connected to an LCD that will display the text and speakers used as sound output. The research built a library visitor noise detection tool with the ES8266 microcontroller. The microcontroller is connected to a KY-037 sensor and an LCD [13]. The study also built a website as a medium for monitoring and controlling noise detection [14].

These studies are similar because they built a monitoring system using a microcontroller connected to a sound sensor. However, no one has researched how the system can distinguish the noise produced by library visitors from the noise coming from the environment around the library. For this reason, this study has the main objective of how a system built on a microcontroller can have artificial intelligence that can identify sound noise.

II. Research Method

This research was built using several steps. Which are system concept, analysis of requirements, system design, system implementation, and system testing.

A. System concept

The system has a working principle of being able to receive sound input from the surroundings. The sound enters through 2 components, namely the noise sensor and microphone. The typical conversation or the minimum audible sound level is 60dB [15][16]. When the sound around the device exceeds the threshold of 60 dB, the system will run the microphone sensor and perform sound classification. Sounds will be classified into five categories: falling object, horn, human, cell phone, and siren. When the classification indicates a category of human or mobile phones, the notification LED will light up to commemorate the people around it. All tool readings in decibel values and classification results will be sent to the database, and a data summary will be displayed on the website.

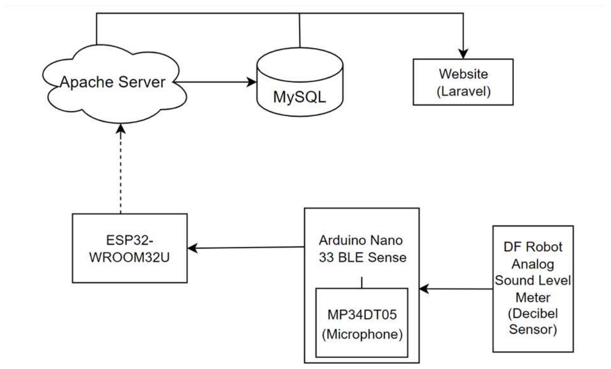
B. Analysis of requirement

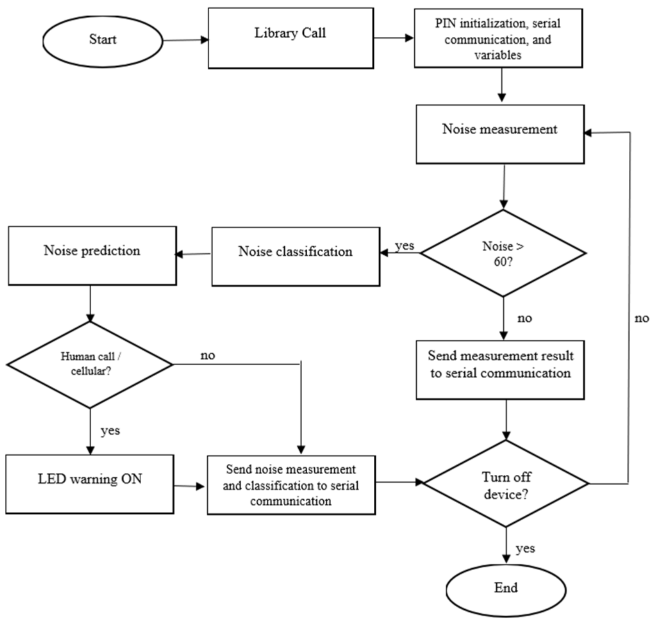
This system is built using two microcontrollers, Arduino Nano BLE Sense [17] as a classification microcontroller and ESP32-WROOM32U [18] as a microcontroller that connects with databases. Arduino Nano BLE Sense which already has a microphone type MP34DT05, is also connected to a noise sensor made by DF Robot with an Analog Sound Level Meter type and an indicator LED that functions as an indicator when the device is on and a warning LED for users around it. DF Robot Analog Sound Level Meter is a decibel measuring sensor produced by DF Robot. This sensor is also known as a decibel meter or *noise meter,* which measures the noise around the sensor [19]. The feature extraction results will be used for input in training using the Convolutional Neural Network (CNN) algorithm [10]. The data received from these sensors will enter the Arduino communication serial, which will be forwarded to the ESP32.

ESP32 will be connected to a previously determined WIFI so that it can communicate with the server. The data from Arduino nano sent to ESP32 is in the form of noise sensor reading data, and the classification results will be forwarded to the server to be stored in the database. So, data from the tool that has entered the database will be able to be displayed on the website. On the website, data can be displayed by year, month, and day. There is also an option to display reading data directly.

C. System design

Based on the analysis of requirements before, the system is shown in Fig. 1. When the device is connected to a power source, it immediately reads the required library. Next, the initialization of the pins, serial communication, and required variables will be carried out. Then, it is also checked whether the microphone is working or not. Otherwise, it will display an error warning. Then, the pin connected to the LED indicator will be turned on as long as the tool is on. It will then read the data from the noise sensor and go into conditioning. When the noise sensor reading exceeds the threshold of 60 decibels, a voice reading will be carried out through the microphone. Then the noise level reading and classification will be sent to the communication serial to be accepted on the ESP32. If the noise is not more than 60 decibels, then only the noise level measurement results are transmitted to the communication serial. This Noise Measurement and Sound Classification software will be embedded on the Arduino Nano 33 BLE Sense board with a flowchart, as shown in Fig. 2.


Apache Server
MySQL
Website (Laravel)
ESP32-WROOM32U
Arduino Nano 33 BLE Sense
MP34DT05 (Microphone)
DF Robot Analog Sound Level Meter (Decibel Sensor)
Fig. 1 Diagram Block


Start
Library Call
PIN initialization, serial communication, and variables
Noise measurement
Noise > 60?
yes
no
Noise classification
Noise prediction
Human call / cellular?
no
yes
LED warning ON
Send noise measurement and classification to serial communication
Send measurement result to serial communication
Turn off device?
no
yes
End
Fig. 2 Noise measurement and classification system

The AudioClassification library was created using Edge Impulse Studio [20]. This library is generated from machine training that generates a classification model based on the training carried out. In creating a classification model, a prior invocation of libraries is required in training. Then, a dataset will be loaded with 1071 sound samples divided into five categories called falling objects containing 216 sound samples representing the sounds of falling objects. Human contain 246 sound samples that represent noises coming from humans. A cell phone contains 225 sound samples, a collection of noise sounds from a cell phone. The horn contains 198 sound samples which are a class containing noise sounds caused by vehicle horns. The siren contains 186 sound samples, a type of sound produced by the sirens of ambulance, police, or fire engines. The fallingObj category contains sounds obtained from live recordings of falling objects by the author. The Phone category contains mobile phone voices obtained from open media such as YouTube. The Siren, Human, and Horn categories contain sounds taken from the UrbanSound8K dataset by sorting out sounds relevant to library conditions. Furthermore, sound wave extraction will be carried out aimed at improving the accuracy of the training results. The sound waves used for classification training are not recommended to be used directly because they will result in poor accuracy [21]. Then, after the extraction of sound waves, training is immediately carried out using the Convolutional Neural Network algorithm. The algorithm is made from several hidden layers to make it easier for the machine to recognize the type of sound expected. The hidden layers used are reshaped layer, two Conv/pool layers, flatten layer, a dropout, a dense layer, and drop out again. The flatter layer is true, the epoch used is 1000, and the learning rate parameter used is 0.005. After the training and testing are completed, the model will finally be saved and converted into a C++ library for easy use in programming on hardware.

In the data storage system, this will be done by ESP32-WROOM32U. First, the invocation of the libraries necessary for the running of the program is carried out. After calling the library, a serial communication initialization will be carried out to retrieve data from the Arduino Nano 33 BLE Sense, and initialize the variables and functions needed to be able to connect with WIFI such as SSID and password from the WIFI you want to connect. Furthermore, ESP32 will wait for the data sent from Arduino Nano until the data is received then the data will be sent to the server that was initialized at the beginning. The device will keep repeating the previously mentioned stages until the device is turned off. In addition to the configuration performed on the ESP32, it also needs configuration for the server to save the data to the database.

The database needs to be initialized to configure the MySQL database's table name, username, and password. In addition, there is also a variable initialization that will store data from ESP32. Furthermore, it will wait for a request from the hardware; when there is a POST request, it will be checked whether the API key owned is appropriate or not. This is necessary to avoid the entry of unwanted data from other parties. The received data will be sent to the MySQL database if the key is appropriate. Monitoring Website is a website that functions to display data statistics obtained from noise monitoring and classification.

III. Result and Analysis

The website functions to display data statistics obtained from noise monitoring and classification. A summary of the data will be displayed on the website. The dashboard of the website is shown in Fig. 3.

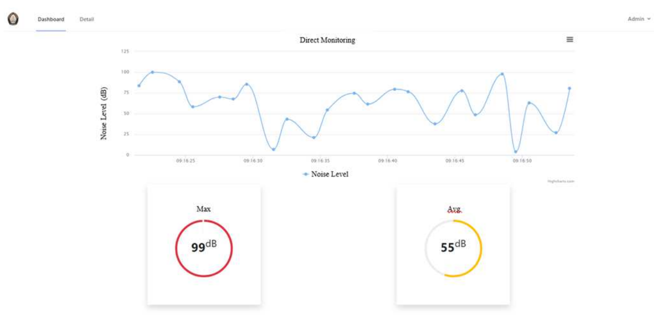


Fig. 3 Dashboard View of The Website

The monitoring system provides a login page used by admin and dashboard page to see a graph of the monitoring being carried out, the maximum data, and the average monitoring of the day. It also has a detailed page to see a summary graph of all monitoring carried out and the maximum, average, and highest-year data of the entire monitoring. Users can also see the details per year, month, week, and day.

A. Noise Monitoring System Distance Testing

Testing the noise monitoring system aims to see how it performs in monitoring the noise around it. Noise monitoring is done by taking noise measurements against the distance from the sound source. This test aims to determine how the DFRobot Analog Sound Level Meter sensor can measure sound noise at varying distances. One sound sample will be used, compared to the percentage loss from the initial noise read at a distance of 0 cm until a specific distance change is made. The calculation of the loss percentage can be defined by equation 1.

$$Loss = \frac{noise\ with\ x\ distance}{noise\ starting\ point} \times 100\% \qquad (1)$$

TABLE I. Noise Monitoring Systems Distance Testing

No.	Volume	Distance	Noise	Loss
1.	20 %	0 cm	90.4 dB	0%
2.		10 cm	78.5 dB	13,1%
3.		20 cm	69.6 dB	23,0%
4.		30 cm	64.5 dB	28,7%
5.		40 cm	60.6 dB	32.9%
6.		50 cm	60.3 dB	33,3%
7.		60 cm	59.9 dB	33,7%
8.		70 cm	59.8 dB	33,8%
9.		80 cm	58.7 dB	35,1%
10.		90 cm	58.3 dB	35,6%
11.		100 cm	57.7 dB	36,1%

The trial was conducted using one constant sound sample with a speaker volume of 20%, as seen in Table I. From the test, the initial noise at a distance of 0 cm was 90.4 dB. Then a shift of length of 10 cm is carried out. This can be used as a reference for the optimal measurement distance that can be done by the DFRobot Analog Sound Level Meter sensor.

B. Noise Monitoring System Data Delivery Testing

In testing the IoT system, measurements of the performance of receiving and sending data to the server will be carried out on the ESP32-WROOM32U board. Some aspects that will be tested include the accuracy and delay of data transmission. Data transmission accuracy can be defined as how much data is sent versus the amount of data received. Delivery accuracy can be formulated in equation 2

$$Accuracy = \frac{data\ accepted}{data\ delivered} \times 100\% \qquad (2)$$

Delivery accuracy testing needs to be done because when serial communication is carried out between Arduino Nano BLE Sense and ESP32-WROOM32U, there is a condition called lag that causes some data sent by Arduino to be received simultaneously by ESP32 which causes only 1 data to be sent to the server.

Delay is the time distance from data transmission to being received by the time during the data transmission process. Delay can also be defined as the time it takes from the sender to get to the recipient. Calculating the average delay can be done by dividing the total delivery time by the data received by the destination device with equation 3. The results of the accuracy and delay tests can be seen in Table II.

$$Delay = \frac{total\ time\ for\ sending\ data}{total\ accepted\ data} \times 100\% \quad (3)$$

TABLE II. DATA DELIVERY IN IOT SYSTEM TESTING

No.	Time Interval	Total Data Sent	Data Received	Accuracy (%)	*Delay* (ms)
1.	1 Minute	30	20	50%	3000 ms
2.	10 Minutes	300	159	53%	3774 ms
3.	30 Minutes	900	475	53%	3789 ms
4.	60 Minutes	1800	932	52%	3863 ms
Average				52%	3607 ms

Table II shows an average of 52% data accuracy and an average 3607 ms delay in the data delivery testing. Based on the delay column, sometimes there is a delay of approximately 3-4 seconds from the data send until the data received.

C. System-Wide Testing

This testing mechanism consists of two parts; the first is to check the CNN algorithm used in the library monitoring system in classifying the noise. The second one is to check the system's data flow from the input to the noise monitoring system.

In the first testing process, after the system builds using the CNN, the accuracy, F1 score, precision, and recall will be calculated based on the five noise categories in edge impulse studio, as shown in Table III.

TABLE III. ACCURACY, F1, PRECISION, AND RECALL RESULT

Confusion Matrix Parameter	Falling Object	Horn	Human	Cell Phone	Siren
Accuracy	68.3%	90.8%	93.8%	93%	68%
F-1	0.8	0.92	0.78	0.8	0.85
Precision	0.96	0.94	0.67	0.98	0.81
Recall	0.68	0.91	0.93	0.68	0.9

The test result showed that the system's average accuracy is 82.78%. As shown in Table III, horn, human, and cell phone sounds have a high accuracy value compared to falling objects and siren sounds.

The second testing process is testing the system built with two microcontrollers. The test was carried out by simulating the activities that occurred in the library with the distance of the tool to the user is 50 cm and it carried out for approximately 1 minute for each simulation. Testing is used to determine the system's capabilities starting from monitoring noise, classifying, storing, and displaying data, as shown in Table IV.

Fig. 4 also shows one of the scenarios where the system will notify the user when the sound input is greater than the threshold given. In the scenario, given one of the human sound which level is greater than the threshold. The system will process the input and provide a warning in the red LED.

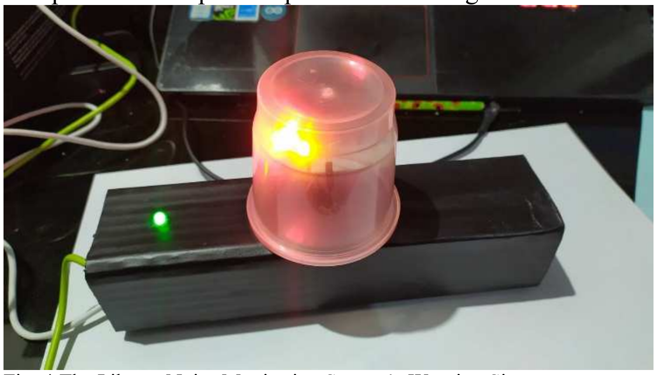


Fig. 4 The Library Noise Monitoring System's Warning Sign

TABLE IV. NOISE MONITORING SYSTEMS TESTING

Simulation	Qualitative Conditions	Noise Monitoring	Classification	Data Retention	*Website*
Silent state, there is only the sound of buzzing fans and air conditioners.	Calm	35-43 dB	-	The majority of important data is stored	Data may appear on the *website*
Chat in a fairly small voice	Calm	41-55 dB	-	The majority of important data is stored	Data may appear on the *website*
Laughs loud enough	Noisy	55-73 dB	Man, Falling Objects	The majority of important data is stored	Data may appear on the *website*
Coughing sound	Quite Noisy	53-68 dB	Humans, Falling Objects, Cell Phones	The majority of important data is stored	Data may appear on the *website*
Dropping objects	Quite Noisy	51-65 dB	Falling Objects, Cell Phones	The majority of important data is stored	Data may appear on the *website*
Horn sound	Noisy	58-85 dB	Horns, Sirens, Cell Phones	The majority of important data is stored	Data may appear on the *website*
Chat quite loudly	Noisy	57-75 dB	Man, Phone, Falling Object	The majority of important data is stored	Data may appear on the *website*
Phone ringing	Noisy	58-77 dB	Phone, Siren	The majority of impor-	Data may appear on the

				tant data is stored	*web-site*
Siren sound	Noisy	55-78	Sirens, Telephones, Horns	The majority of impor-tant data is stored	Data may appear on the *web-site*

Based on Table IV, from those 1 minute, some conditions are not right. However, the main function of recognizing human noise sounds is good enough because it can recognize some sounds produced by humans and cell phone.

IV. Conclusion

From the research that has been carried out, the system can give a noise warning when a human or cell phone sound exceeds the threshold with an average of 82.78% classification accuracy and an ideal distance from the sound source, as far as 30-100 cm. It is also able to monitor the noise in its surroundings. In IoT systems that utilize ESP32-WROOM32U hardware, they have excellent capabilities in transmitting data. Still, for data receipt from serial communication, there is a delay of approximately 3-4 seconds. Improving the accuracy of delivery can be done by increasing the delivery lag time performed by Arduino Nano 33 BLE Sense. Still, it will make noise monitoring less accurate due to the long lag.

Acknowledgment

This research was financially supported by The Faculty of Engineering, Diponegoro University, Indonesia through the Strategic Research Grant 2022.

References

[1] J. Lange, A. Miller-Nesbitt, and S. Severson, "Reducing noise in the academic library: the effectiveness of installing noise meters," *Libr. Hi Tech*, vol. 34, no. 1, pp. 45–63, 2016, doi: 10.1108/LHT-04-2015-0034.

[2] S. Gordon-Hickey and T. Lemley, "Background Noise Acceptance and Personality Factors Involved in Library Environment Choices by College Students," *J. Acad. Librariansh.*, vol. 38, no. 6, pp. 365–369, 2012, doi: 10.1016/j.acalib.2012.08.003.

[3] C. R. Pereira, "Basic Characteristics of Sound," 2015.

[4] G. Taraldsen, T. Berge, F. Haukland, B. H. Lindqvist, and H. Jonasson, "Uncertainty of decibel levels," *J. Acoust. Soc. Am.*, vol. 138, no. 3, pp. EL264–EL269, 2015, doi: 10.1121/1.4929619.

[5] G. Sharma, K. Umapathy, and S. Krishnan, "Trends in audio signal feature extraction methods," *Appl. Acoust.*, vol. 158, p. 107020, 2020, doi: 10.1016/j.apacoust.2019.107020.

[6] N. David, C. V. N. Anyika, I. N. Ejindu, and A. O. Abioye, "Library Sound Level Meter," *Quest J. Electron. Commun. Eng. Res.*, vol. 1, no. 1, p. 10, 2013, [Online]. Available: http://www.questjournals.org/jecer/papers/vol1-issue1/C112029.pdf.

[7] C. Banbury *et al.*, "Micronets: Neural Network Architectures for Deploying TinyML Application on Commodity Microcontroller," *Dict. Genomics, Transcr. Proteomics*, 2015, doi: 10.1002/9783527678679.dg13290.

[8] M. Z. H. Zim, "TinyML: Analysis of Xtensa LX6 microprocessor for Neural Network Applications by ESP32 SoC," no. June, 2021, doi: 10.13140/RG.2.2.28602.11204.

[9] H. S. Bae, H. J. Lee, and S. G. Lee, "Voice recognition based on adaptive MFCC and deep learning," *Proc. 2016 IEEE 11th Conf. Ind. Electron. Appl. ICIEA 2016*, pp. 1542–1546, 2016, doi: 10.1109/ICIEA.2016.7603830.

[10] A. Azarang, J. Hansen, and N. Kehtarnavaz, "Combining Data Augmentations for CNN-Based Voice Command Recognition," *Int. Conf. Hum. Syst. Interact. HSI*, vol. 2019-June, pp. 17–21, 2019, doi: 10.1109/HSI47298.2019.8942638.

[11] K. H. Lee and D. H. Kim, "Design of a Convolutional Neural Network for Speech Emotion Recognition," *Int. Conf. ICT Converg.*, vol. 2020-Octob, pp. 1332–1335, 2020, doi: 10.1109/ICTC49870.2020.9289227.

[12] C. M. Achsan and D. Krisbiantoro, "Design and Build an Arduino-Based Noise Detection and Warning Device (Case Study: AMIKOM Purwokerto University Library)," *SYMMETRICAL J.*, vol. 11, no. 2, pp. 551–559, 2021, doi: 10.24176/simet.v11i2.5013.

[13] Nurwati, "Noise Level Detection and Warning in Arduino-Based Libraries," *R. Natl. Semin.*, vol. 1, no. 1, pp. 1–4, 2018.

[14] Herianto and K. Hasnor, "Design and Build A Library Visitor Noise Detection Device Based on Sound Pressure Parameters using NodeMCU ESP8266," *J. Comput. Sci.*, vol. 10, no. 1, pp. 20–26, 2021.

[15] "Levels of Noise Levels of Noise Levels of Noise Levels of Noise." p. 140.

[16] "6 - Signalling systems and confirmed alarms.pdf." .

[17] Arduino, "Arduino® Nano 33 BLE Sense," *Arduino USA website*, pp. 1–12, 2021, [Online]. Available: https://docs.arduino.cc/resources/datasheets/ABX00031-datasheet.pdf.

[18] E. Systems, "ESP32-WROOM-32U," 2019.

[19] Jason, "DFROBOT SEN0232 Gravity Analog Sound Level Meter," 2017. https://wiki.dfrobot.com/Gravity__Analog_Sound_Level_Meter_SKU.

[20] Edge Impulse, "Recognize Sounds from Audio." https://docs.edgeimpulse.com/docs/tutorials/audio-classification.

[21] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, vol. 9781107057. 2013.