EDUCACIÓN

iPSUMTEC
ISSN: 2594 - 2905

INTELLIGENT SYSTEM FOR MONITORING PHYSICAL VARIABLES IN BEEHIVES USING NEURAL NETWORKS

INTELLIGENT SYSTEM FOR MONITORING PHYSICAL VARIABLES IN BEEHIVES USING NEURAL NETWORKS

Carlos Humberto Montaño Alcalá[1], María Trinidad Serna Encinas[2*] , Fredy Alberto Hernández Aguirre[3]https://doi.org/10.61117/ipsumtec.v9i1.466

[1] National Technological Institute of Mexico/Hermosillo Institute of Technology,m16330894@hermosillo.tecnm.mx , https://orcid.org/0009-0006-3422-2877 [2*]National Technological Institute of Mexico/Hermosillo Institute of Technology,maria.sernae@hermosillo.tecnm.mx , https://orcid.org/0000-0002-2020-791X [3] National Technological Institute of Mexico/Hermosillo Institute of Technology, fredy.hernandeza@hermosillo.tecnm.mx, https://ocid.org/0000-0001-9208-5299

Abstract - Beekeeping is an essential activity for both honey production and plant pollination. In Mexico, beekeeping plays a significant role, as the country is the ninth-largest honey producer worldwide. To keep hives healthy and maintain high production levels, constant monitoring is necessary. Traditionally, this process involves manual inspections of the hive every 8 to 15 days to detect problems or diseases. As the number of hives increases, manual monitoring becomes inefficient, time-consuming, and logistically complicated, with the risk that some hives may not be inspected at the appropriate time. This article presents a system that monitors beehives using Internet of Things (IoT) technologies, integrating sensors that track variables such as temperature, humidity, weight, and audio signals from the hive with a pattern recognition algorithm used to detect events of interest to the beekeeper. The system also incorporates a mobile app capable of displaying alerts and sensor readings. The purpose of this project is to provide beekeepers with real-time data on the condition of the hive, as well as to support logistics and decision-making.

***Keywords*:** Beehives, IoT, Neural networks, Monitoring system, Physical variables.

Abstract: Beekeeping activities are important not only for honey production but also for pollinating plants. In Mexico, it is a significant economic activity, and the country ranks as the ninth-largest honey producer in the world. To maintain the health of the hives and ensure effective production, constant monitoring is necessary. Traditionally, this process involves manually inspecting hives every 8 to 15 days to identify any problems or diseases. As the number of hives increases, this type of monitoring becomes inefficient, time-consuming, and more difficult to schedule, and may even result in some hives not being inspected at the right time. This article presents a system that monitors hives using

Internet of Things (IoT) technology for monitoring. It integrates sensors that monitor variables such as hive temperature, humidity, weight, and sound, along with a pattern recognition algorithm to detect significant events requiring the beekeeper's attention. The system includes a mobile app that displays alerts and sensor data. The purpose of this project is to help beekeepers obtain real-time data on the status of their hives, thereby supporting their decision-making and management.

Keywords: Beehives, IoT, Neural networks, Monitoring system, Physical variables.

INTRODUCTION

Beekeeping is an economic activity focused on the management of beehives and the maintenance of their health, which requires the incorporation of technical methodologies involving the proper handling of queens, disease control, parasite elimination, and production optimization [1]. Ensuring the health and efficient production of beehives requires constant monitoring through inspections conducted by the beekeeper at intervals of 8 to 15 days, with the aim of detecting anomalies and diseases [2].

In addition to their productive function, beehives contribute to the pollination of surrounding vegetation; this is significant as an economic activity, since nearly 75% of food crops depend on the pollinating activity of bees [3].

Currently, there are technological solutions focused on monitoring beehives through the use of Internet of Things (IoT) systems. The primary focus of these systems is monitoring physical parameters such as temperature, humidity, and weight, as well as collecting acoustic signals, with the goal of using this data in analysis workflows, complemented by artificial intelligence (AI) algorithms [4], [5], [6], [7]. The solution proposed in this article consists of the development of an intelligent system based on an IoT framework that

collects data through sensor nodes, which is then transmitted to a server for processing, storage, and visualization. Additionally, the system includes an alert module based on neural networks, designed for the early detection of events relevant to hive management.

DEVELOPMENT

Methodology

The development of the system consists of five phases. The first phase involved reviewing the state of the art in beekeeping, mobile development, backend systems, IoT, and pattern recognition through conference papers, journals, theses, and relevant prior studies.

The second phase involved identifying the components and features required for the design and modeling of the monitoring system, including the selection of sensors, power sources, and the controller.

The third phase involved developing the processing system, including the processing of data from the monitoring system, pattern recognition, and data persistence.

The fourth phase involved implementing the mobile application, which is designed to process and visualize information from the service, as well as display detected alerts.

Finally, the fifth phase involved conducting comprehensive system testing, analyzing the results, and adjusting the system as needed.

System Microcontroller Algorithm

The microcontroller algorithm shown in Figure 1 represents the monitoring node cycle. The operational cycle runs on an ESP32 microcontroller.

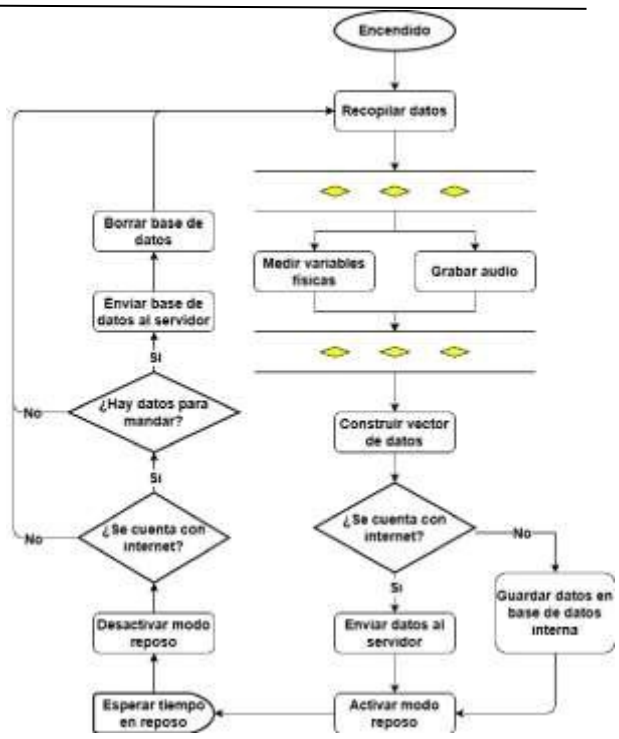


Figure 1. *Microcontroller algorithm.*
Source: Author's *own work (2026).*

The flow is structured as a cycle that coordinates data acquisition and synchronization with the server, subject to connection availability. The design includes local storage to tolerate network failures, as well as power-saving mechanisms that consist of alternating between active and low-power states. The graphical elements used and their function within the process are described below:

- Start (oval). Represents the starting point of the process.
- Decision (diamond). A node where the flow branches based on a condition; only one exit is selected.
- Wait (rectangle with a rounded corner). Indicates a pause in the process.
- Parallel Execution (nodes between two rectangles with yellow diamonds). A section where multiple nodes are executed simultaneously.

Data Models

The persistence layer is structured into three functional databases—*guard-database*, *nest-database*, and *mind-database*—designed according to the principle of separation of concerns.

The *nest-database* stores information related to the system's structure and sensor readings, enabling monitoring of the status of each hive.

The *guard-database* manages digital identities and access tokens.

The *mind-database* aggregates data derived from acoustic signal processing, enabling its use with machine learning models.

Figure 2 presents the complete data model of the proposed system.

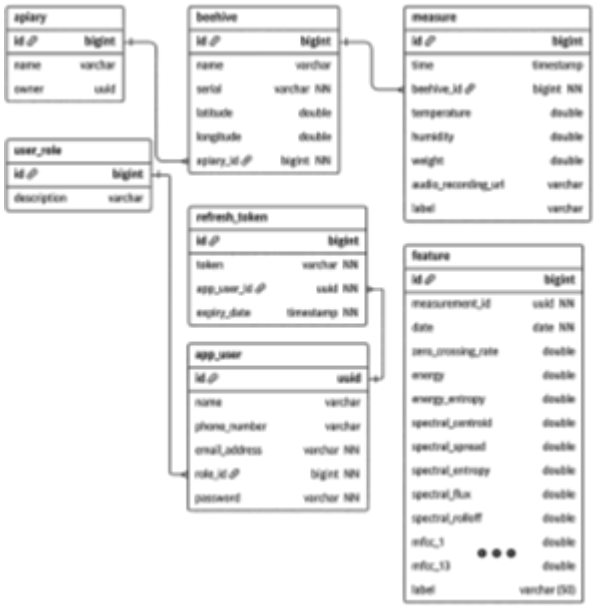


Figure 2. *Complete data model of the system.*
Source: Author's *own work (2026).*

Security database: *guard-database*

- *user_role* defines the roles available in the system.
- *app_user* represents registered users and their association with a specific role.
- *refresh_token* manages active sessions using tokens associated with each user, allowing sessions to be renewed or revoked.

Apiary database: *nest-database*

- *apiary* models groups of physical hives, allowing them to be organized by location or owner.
- *Beehive* represents each beehive within an apiary; it is the unit to be monitored.
- *Measure* stores the physical variables captured from each hive.

Feature database: *mind-database*

- *feature* stores the features extracted from the acoustic signals.

Software Architecture

The system is structured into three main components: the sensor node, the operational backend services, the audio processing service, and the mobile application. Figure 3 shows the complete system architecture.

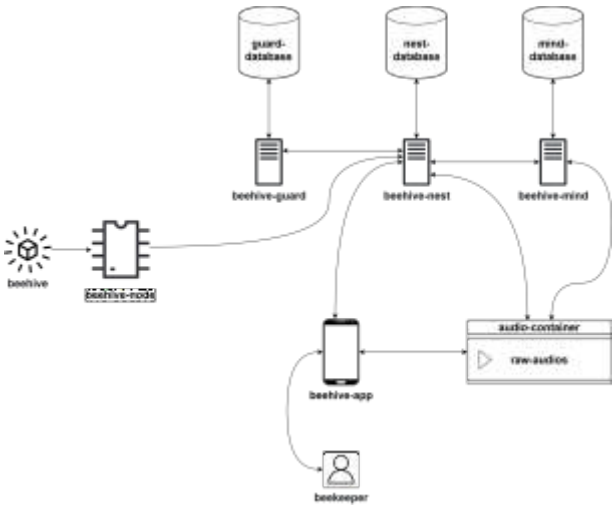


Figure 3. *Logical architecture of the proposed system.*
Source: Author's *own work (2026).*

Sensor node (beehive-node). A microcontroller-based component responsible for acquiring data from the hive; it integrates sensors to measure temperature, humidity, weight, and acoustic signals.

Backend services:

- *beehive-guard*: responsible for managing system security, including authentication and access control.
- *beehive-nest*: responsible for managing the system's operational information, including hives, measurements, and apiaries.
- *beehive-mind*: A server dedicated to extracting features from the audio data included with each measurement, as well as performing inference using the implemented model.

Audio container (audio-container): The persistence of audio recordings is implemented using an object storage service compatible with the S3 API. Under this approach, files are organized into logical containers called buckets. For the test environment, MinIO was used as a local implementation of the S3 standard [8].

Mobile application (beehive-app): This is the interface accessible to the beekeeper for monitoring the status of the hives in real time and receiving notifications. The ability to view the status of the hives in real time is achieved through periodic

requests to the server.

The main system workflows are described below:

1. The sensor node collects data from the hive and transmits it to the beehive-nest service.
2. The mobile app makes periodic requests to beehive-nest to obtain the most recent data.
3. The beehive-nest service sends the audio recording to the audio container.
4. The beehive-mind service processes the acoustic signals, extracts features, and classifies each recording as anomalous or normal. The result is stored in the database.
5. The beehive-guard service manages the authorization mechanisms applied to all interactions between the mobile app and the system.

Table 1 presents the main components of the proposed system architecture.

Table 1. *Main components of the system architecture.*

Component	Function	Persistence
beehive-node	Captures data from sensors	Micro SD
beehive -guard	Authentication and security	PostgreSQL (guard-data-base)
beehive-nest	Metadata and logs	PostgreSQL (nest-database)
beehive-mind	AI-powered audio analysis	PostgreSQL (mind-database)
audio-contai-ner	Storage of raw audio	S3/MinIO (raw-audio files)
beehive-app	API interface	Local persistence on mobile device

Source: Author's *own work (2026).*

Application Flow: The mobile app acts as the entry point to the system, allowing users to register, authenticate, access information, and send operational requests to the services. Figure 4 shows the app's main screen.

Figure 4. *Main screen.*
Source: Author's *own work (2026).*

As previously mentioned, Figure 5 shows the main screen, which is accessible immediately after logging in and allows users to view their information, manage associated hives, and log out.

The registration process requires the user to enter a password, an email address, a phone number, and a username. Once this is done, the user can log in using these credentials.

To add a new apiary, the user must go to the *"Beehives"* section from the main screen, use the "+" add button, and select the "Apiary" option. Meanwhile, to register a new beehive, the user must be within an available apiary and select the *"Add a new beehive"* option or use the "+" add button and select the *"Beehive"* option.

After the system validates the device, the hive's status can be viewed directly from the interface listing the apiaries.

The hive monitoring section displays the main parameters using visual indicators, whose color changes dynamically based on their values.

- **Humidity.** Represented by a drop icon; the indicator turns yellow when the value is below the recommended range and blue when it exceeds it.
- **Temperature.** Displayed via an indicator that turns blue if the value is below the established range and red when it exceeds it.
- **Weight.** Displayed as a constant numerical value, with no visual variation associated with its magnitude.

- **Audio.** Represented by a triangular icon, which allows the user to play the most recent recording associated with the hive.
- **Details.** Identified by a circular icon with an "I," which directs the user to the detailed view of the hive when selected.

Figures 5 and 6 show a detailed view of the hive.

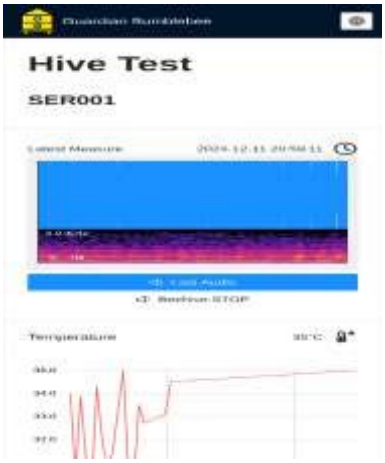


Figure 5. *Spectrogram of the collected audio.*
Source: Author's *own work (2026).*

Figure 6. *Latest measurements collected by the sensor node.*
Source: Author's *own work (2026).*

The figures above show a detailed view of the hive, displaying the spectrogram corresponding to the captured audio signal, along with the option to play it back. Additionally, the most recent measurements obtained by the sensor node and their temporal behavior are displayed through a graphical representation.

sampling rate **sr**; in the case of stereo audio, processing is performed independently per channel. The default parameters are n_fft = 2048, hop_length = 512, and a Hann window [9].

Zero Crossing Rate (ZCR) Algorithm [10], [11]:

1. Segment the continuous signal into fixed-length windows, for example, of 2048 samples.
2. In each window, evaluate the sign change between consecutive samples; each transition from positive to negative, or from negative to positive, is counted as a zero crossing.
3. Accumulate the total number of crossings within the window and normalize it with respect to its length to obtain a metric that is comparable across different signals.
4. Analyze the time sequence of these rates to detect regions of high activity, typically associated with noise or non-speech components of speech.

Audio Characteristics and Classification Audio Characteristics: This section describes the calculations and the algorithmic procedure to be implemented in Python. A monaural audio signal with a frequency

Energy
Algorithm [10], [12], [13]:

1. Divide the signal into windows of a defined duration.
2. In each window, calculate the square of each sample to quantify the signal's magnitude.
3. Sum the values obtained within the window; a higher value indicates greater sound intensity in that interval.

Energy Entropy Algorithm
[12], [14], [15]:

1. For each window of the signal, subdivide it into Q consecutive segments of smaller size.
2. Calculate the energy of each segment and normalize these values to obtain proportions whose sum equals 1.
3. Apply Shannon entropy to these proportions; high values indicate a uniform distribution of energy, while low values reflect its concentration in a few segments.

Spectral Centroid
Algorithm [10], [11], [15]:

1. Obtain the magnitude spectrum of each window of the signal.
2. Weight each spectral component by its corresponding frequency and calculate the resulting average, which allows us to estimate the center of mass of the spectrum.

Spectral dispersion
Algorithm [11],
[15]:

1. Determine the spectral centroid.
2. Calculate the standard deviation of the frequencies, weighted by their spectral amplitudes, to quantify the dispersion or width of the spectrum.

Spectral Entropy
Algorithm [10], [15], [16]:

1. Obtain the magnitude spectrum and normalize its components to construct a distribution.
2. Calculate the Shannon entropy of this distribution; high values indicate a uniform distribution of energy across the spectrum, while low values reflect its concentration in a small number of frequency bands.

Spectral flux
Algorithm [10], [17]:

1. Calculate the magnitude spectra of two consecutive windows and normalize them.
2. Calculate the difference between the current spectrum and the previous one, retaining only the positive differences.
3. Square these differences and accumulate the result; high values indicate abrupt transitions in the spectral content, associated with changes in timbre or sound attacks.

Spectral rolloff
Algorithm [11], [15]:

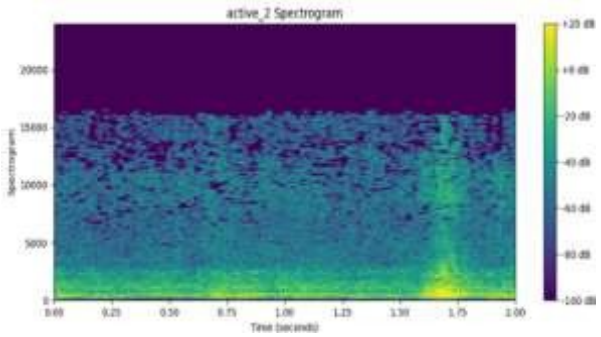
1. Sort the magnitudes in ascending order of frequency.
2. Find the frequency at which the accumulated sum reaches a fixed fraction—for example, 85%—of the total energy.

MFCC (Mel Frequency Cepstral Coefficients) Algorithm [10], [18], [19], [20]:

1. Segmentation and windowing: divide the signal into fixed-size frames and apply a window function to reduce discontinuities in the spectral analysis.
2. Transform and power: Obtain the spectrum of each frame and calculate its energy to represent how the signal is distributed across the frequency domain.
3. Mel filter bank: Project the spectrum onto a set of Mel-scaled filters to approximate how the human ear perceives sound.
4. Logarithmic compression: Apply a logarithmic transformation to reduce the signal's dynamic range.
5. DCT: Transform the obtained values to generate a small set of coefficients that describe the general shape of the spectrum.

Audio Classification

The following figures present a comparison between two spectrograms obtained from different beehives. Figure 7 shows a hive in its normal state, while Figure 8 shows a hive without a queen bee. The variations in



frequency distribution and spectral configuration indicate the presence of distinguishable acoustic patterns, which allows the problem to be addressed as a classification task [21].

Figure 7. *Audio spectrogram of a beehive in normal condition.*
Source: Author's *own work (2026).*

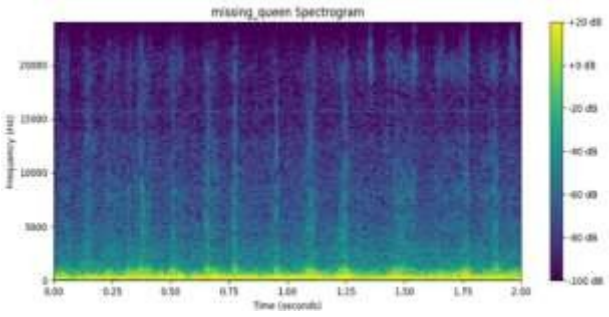


Figure 8. *Audio spectrogram of a hive without a queen.*
Source: Author's *own work (2026).*

Based on the differences identified in the acoustic signals, the feasibility of distinguishing between them is established. Based on this assumption, a neural network is designed and implemented that receives as input a vector of previously defined features and classifies the audio into normal or anomalous categories, considering conditions associated with colony stress.

Initial Dataset

The audio modeling is based on the integration of three public datasets, which include recordings from beehives under different conditions, locations, and recording devices, in order to construct a representative dataset.

- **Open Source Beehives Project–Audio Database as of 2-21-17.** It contains recordings organized by hive status and location; "Active" audio is considered normal, and the rest ("Missing Queen," "Pre-Swarm," "Queen Hatching," "Sick-Varroa," "Swarm") are considered abnormal [22].
- **Queen Loss Event in Africanized Honeybee Colony**. Includes feature vectors associated with queen loss events; QR is labeled as normal and QL as anomalous [23].
- **Beehive buzz anomalies**. Contains segments classified into various categories;

"Not a bee" segments are discarded, while "Bee" and "Missing Queen" are assigned as normal and anomalous, respectively [24].

Once the audio data and existing feature vectors were integrated and labeled, feature extraction was performed using 1-second segments per recording. The obtained features were combined with the previously available vectors to form a single dataset, in which each instance is represented by a feature vector and a label indicating whether it corresponds to a normal or anomalous condition. Figure 9 shows the distribution between the two classes.

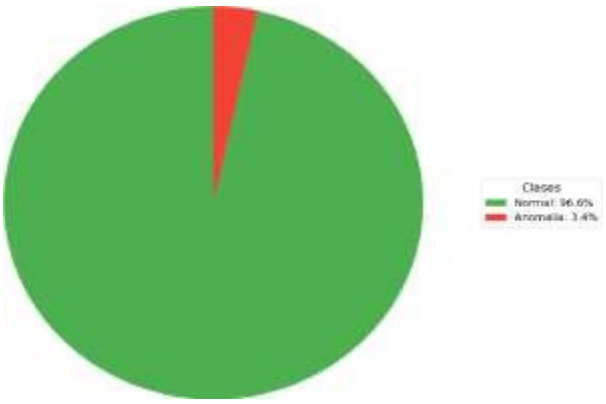


Figure 9. *Anomaly-to-normal ratio of the initial dataset.*
Source: Author's *own work (2026).*

As can be seen in Figure 10, the ratio of anomalous to normal segments shows a significant imbalance. In this context, a model that classifies all instances as *normal* would achieve a precision of 96%; this indicates that this metric is not valid for evaluating the model's performance, since it does not reflect its ability to detect anomalies. Taking this into account, and given that anomaly detection is the primary objective, the training process focused on optimizing the F1 score exclusively for the anomalous class. For this purpose, the F1 score is calculated for the anomalous class once each experiment is completed, allowing the score to be stored as the experiment's result.

Neural Network Design

The model was trained using 80% of the dataset, while the remaining 20% was used for validation. Hyperparameter selection was performed using a random search, evaluating multiple configurations generated within predefined ranges. In each iteration, the model is trained on the training set and evaluated on the validation set; this process is repeated iteratively, adjusting the search ranges based on the results of and prioritizing configurations with the highest F1 scores for the anomalous class.

Table 2 shows the results for the six experiments.

Table 2. *Summary of the best-performing test subsets used in each experiment.*

Experiment Number Experiment	Best Iterations	Iterations Total	Percentage (%)
1	20	200	10.00
2	20	200	10.00
3	60	600	10.00
4	17	1743	0.98
5	15	1557	0.96
6	15	1,500	1.00
Total	**147**	**5,800**	**2.53**

Source: Author's *own work (2026).*

The complete implementation of the system is available in a public repository [25].

DISCUSSION AND ANALYSIS OF RESULTS

Table 3 presents the maximum and minimum ranges of hyperparameters obtained in experiments 1 through 3, considering a total of 15 possible hyperparameters in the neural network architecture.

Table 3. *Hyperparameter ranges for experiments 1 through 3.*

Hyper-parameter	E1 min	E1 max	E2 min	E2 max	E3 min	E3 max
layers	2	2	4	4	2	4
units1	32	128	32	128	32	128
dropout1	0.2	0.5	0.2	0.4	0.2	0.4
units2	16	64	16	64	16	56
dropout2	0.2	0.5	0.2	0.4	0.2	0.4
units3	-	-	8	32	8	28
dropout3	-	-	0.2	0.5	0.2	0.4
units4	-	-	4	16	4	16
dropout4	-	-	0.2	0.6	0.2	0.5
loss_fn	focal	tversky and	focal	tversky	focal	focal
focal_gamma	1.0	5.0	1.0	5.0	3.0	5.0
focal_alpha	0.1	0.9	0.1	0.9	0.2	0.5
Tversky alpha	0.1	0.8	0.1	0.08	0.15	0.55
tversky_beta	0.1	0.9	0.15	0.08	0.15	0.65

lr	0.00020 8	0.008327	0.000258	0.003646	0.000312	0.003495

Source: Author's *own work (2026).*

To begin with, in Experiment 1, architectures limited to 2 layers were iterated, while in Experiment 2, only 4-layer architectures were evaluated. Due to the varied results, the number of layers was added as a dynamic hyperparameter for subsequent experiments. Additionally, among the best results from Experiment 3, only the focal loss function was observed; therefore, it was the only one considered for subsequent experiments.

Table 4 presents the maximum and minimum ranges of hyperparameters obtained in Experiments 4 through 6, considering a total of 12 possible hyperparameters in the neural network architecture.

Table 4. *Hyperparameter ranges for Experiments 4 through 6.*

Hyper-parameter	E4 min	E4 max	E5 min	E5 max	E6 min	E6 max
layers	2	4	4	4	4	4
units1	32	128	32	128	36	124
dropout1	0.2	0.4	0.2	0.4	0.21	0.4
units2	16	56	16	56	30	54
dropout2	0.2	0.4	0.2	0.4	0.24	0.4
units3	8	28	8	26	8	25
dropout3	0.2	0.4	0.2	0.4	0.22	0.4
units4	4	16	5	15	5	15
dropout4	0.2	0.4	0.2	0.4	0.2	0.37
focal_gamma	4	5.0	4.0	5.0	4.1	4.9
focal_alpha	0.2	0.4	0.2	0.4	0.21	0.39
lr	0.000407	0.003260	0.000497	0.002795	0.000549	0.002544

Source: Author's *own work (2026).*

Based on the results obtained, it can be observed that the optimal ranges, derived from the iterative process of the 6 experiments conducted, ultimately converge on a stable set of hyperparameters. Table 5 shows the results of the metrics for the 6 experiments.

Table 5. Ranges of metrics obtained in tests 1 through 6.

Experiment	Accuracy (min)	Accuracy (max)	f1 (min)	f1 (max)
1	0.9091	0.9963	0.4271	0.8445
2	0.9389	0.9964	0.4731	0.9470
3	0.9303	0.9964	0.4073	0.9467
4	0.9887	0.9967	0.8565	0.9507
5	0.9910	0.9967	0.8777	0.9518
6	0.9933	0.9969	0.9024	0.9546

Source: Author's *own work (2026).*

Table 5 presents the maximum and minimum intervals corresponding to the precision and F1 metrics obtained from the total number of models iterated in each experiment, highlighting that Experiment 6 achieves the highest values for both metrics, demonstrating the progress made in each experiment.

Table 6 shows the hyperparameters identified in Experiment 6.

Table 6. *Best hyperparameter configuration obtained in the search.*

Hyperparameter	Value
units1	88
dropout1	0.22
units2	48
dropout2	0.24
units3	13
dropout3	0.22
units4	8
dropout4	0.35
focal_gamma	4.65
focal_alpha	0.24
lr	0.001043400598498229

Source: Author's *own work (2026).*

Tables 3, 4, and 6 list the variables used in each experimental session. These correspond directly to attributes defined in the code used to run the iterations. Their meanings are described below.

- **layers**: number of hidden layers in the network.
- **units[n]:** number of neurons in hidden layer n (1 to 4).
- **dropout[n]**: proportion of deactivated neurons in layer n (1 to 4).
- **loss_fn**: loss function used during training.
- **focal_gamma**: gamma parameter of the focal loss function.
- **focal_alpha**: the alpha parameter of the focal loss function.
- **tversky_alpha**: the alpha parameter of the Tversky index.

- **tversky_beta**: beta parameter of the Tversky index.
- **lr**: learning rate of the optimizer.

Figure 10 shows the confusion matrix for the model selected in Experiment 6. On the validation set, this model achieves an accuracy of 99.6921% and a maximum F1 score of 0.954607 for the anomalous class.

Figure 10. *Confusion matrix of the selected neural network on the validation data.*
Source: Author's *own work (2026).*

On the validation set, the model achieves an accuracy of 99.6921% and a maximum F1 score of 0.954607 for the anomalous class.

When comparing the results obtained by the proposed system with previous research in the field of beehive monitoring, notable differences are identified in the method and scope of the proposed systems. The embedded system proposed in [5], based on the use of an ESP8266 microcontroller and an SHT21 sensor, focuses on the efficient monitoring of temperature and humidity, prioritizing primarily the optimization of energy consumption. However, its scope does not include any type of analysis of the obtained data, leaving this as a possible line of future research.

Similarly, the proposal discussed in [7] focuses on designing a system capable of withstanding extreme temperature conditions (-20 to 60 °C) and maintaining a high level of energy autonomy through the use of supercapacitors and solar panels. Similar to the previous proposal, the approach is based on data acquisition and transmission, without considering subsequent analysis.

Finally, the system proposed in [26] incorporates an automated analysis component, applying feature extraction techniques using MFCC and LPC, along with classification based on HMM/GMM machine learning models for swarm detection

from acoustic data. This approach demonstrates the viability of intelligent processing in the beekeeping context. The present work follows a similar line in terms of the integration of automated analysis, exploring the use of neural networks for the detection of anomalous events in the hive.

CONCLUSIONS

A microcontroller-based IoT architecture was developed for acquiring measurements and storing them in a cloud infrastructure. As part of the system, an electronic circuit was designed that integrates humidity, weight, and temperature sensors, as well as an audio capture module and data persistence mechanisms.

On the software side, a cloud service was implemented for managing apiaries, beehives, and measurements, structured using entities and operational relationships. Additionally, a user management and authentication service was developed, along with tools to explore and evaluate architectures with a focus on hyperparameter tuning and model performance optimization.

A cloud service was implemented to extract feature vectors from audio signals and classify them using a neural network. A mobile application was also developed as the system's interface, allowing users to view data and interact with the other modules.

Alert mechanisms based on thresholds for the monitored parameters were implemented, providing users with direct notifications.

Finally, by implementing the model corresponding to Experiment 6, a maximum accuracy of 99.6921% was achieved in the classification of anomalous events. However, due to the imbalance in the dataset, the most relevant metric in the validation set is the F1 score for the anomaly class, with a value of 0.954607. This result is particularly significant given the inherent variability of the analyzed acoustic signals.

Two main directions are proposed for future research. The first involves expanding the dataset through controlled sampling campaigns in production hives, correlating the acoustic signals with direct observations of the hive's condition, as well as incorporating additional sensors such as vibration, CO_2, and bee counting. This would allow us to capture the variability associated with climate, bee genetics, and colony dynamics;

; it would also enable the distinction between different types of anomalous events such as queen loss, heat stress, predators, and swarming.

The second line of research focuses on optimizing the classification model through the implementation of continuous learning, which allows the model to adapt to new field conditions without having to retrain it from scratch, as well as the parallel execution of multiple models under a voting scheme to increase robustness in the detection of anomalous events.

REFERENCES

[1] J. M. Flores and J. A. Ruiz, "Basic Principles of Modern Beekeeping," *Revista de Ciencias Agropecuarias*, vol. 38, no. 2, 2021, doi: 10.22201/fesz.23958723e.2021.2.82.

[2] A. Correa Benítez, I. Vásquez Valencia, N. Peña Haaz, L. León Luna, and R. Anguiano Báez, "Good livestock practices in primary honey production," 2018. [Online]. Available: https://atlas-abejas.agricultura.gob.mx/pdfs/Manual_BPP_en_la_Producci_n_primaria_de_Miel_octubre_2018.pdf

[3] A.-M. Klein *et al.*, "Importance of pollinators in changing landscapes for world crops," *Proceedings of the Royal Society B: Biological Sciences*, vol. 274, no. 1608, 2007, doi: 10.1098/rspb.2006.3721.

[4] S. Cecchi, S. Spinsante, A. Terenzi, and S. Orcioni, "A Smart Sensor-Based Measurement System for Advanced Bee Hive Monitoring," *Sensors*, vol. 20, no. 9, p. 2726, May 2020, doi: 10.3390/s20092726.

[5] M. Vidrascu and D. Svasta, "Embedded Software for an IoT Bee Hive Monitoring Node," 2017.

[6] M. Vidrascu and D. Svasta, "Maintenance-Free IoT Gateway Design for Bee Hive Monitoring," 2017.

[7] M. Vidrascu, P. Svasta, and M. Vladescu, "High-reliability wireless sensor node for bee hive monitoring," *Sensors and Actuators*, pp. 134–138, Mar. 2016, doi: 10.1109/SIITME.2016.7777262.

[8] "MinIO High Performance Object Storage — MinIO Object Storage (AGPLv3)." Accessed: Mar. 19, 2026. [Online]. Available: https://minio-docs.tf.fo/

[9] B. McFee *et al.*, "librosa: Audio and Music Signal Analysis in Python," *J. Open Source Softw.*, 2015, doi: 10.21105/joss.00022.

[10] M. Müller, *Fundamentals of Music Processing*. Springer, 2015. DOI: 10.1007/978-3-319-21945-5.

[11] G. Tzanetakis and P. Cook, "Musical Genre Classification of Audio Signals," *IEEE Transactions*

on Speech and Audio Processing, vol. 10, no. 5, 2002, doi: 10.1109/TSA.2002.800560.

[12] T. Giannakopoulos and A. Pikrakis, **Introduction to Audio Analysis: A MATLAB Approach**. Academic Press, 2014. [Online]. Available: https://www.mathworks.com/academia/books/introduction-to-audio-analysis-giannkopoulos.html

[13] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Prentice Hall, 2010.

[14] X. Li, K. Lee, J. Park, and S. Cho, "Audio feature extraction and classification using entropy-based methods," *IEEE Trans. Multimedia*, vol. 19, no. 2, 2017.

[15] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," Paris, 2004. [Online]. Available: https://recher-che.ircam.fr/anasyn/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf

[16] D. Jiang et al., "Spectral entropy for robust speech recognition," *ICASSP*, 2011.

[17] J. Foote, "Automatic Audio Segmentation using a Measure of Audio Novelty," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, New York, USA, 2000. doi: 10.1109/ICME.2000.869637.

[18] S. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Trans. Acoust.*, vol. 28, no. 4, 1980, doi: 10.1109/TASSP.1980.1163420.

[19] M. Slaney, "Auditory Toolbox Version 2," 1998. [Online]. Available: https://engineering.purdue.edu/~malcolm/interval/1998-010/AuditoryToolboxTechReport.pdf

[20] B. Logan, "Mel Frequency Cepstral Coefficients for Music Modeling," in *Proceedings of the 1st International Symposium on Music Information Retrieval (ISMIR)*, Plymouth, MA, USA, 2000. [Online]. Available: https://ismir2000.ismir.net/papers/logan_abs.pdf

[21] V. Kulyukin, S. Mukherjee, and P. Amlathe, "Toward Audio Beehive Monitoring: Deep Learning vs. Standard Machine Learning in Classifying Beehive Audio Samples," *Applied Sciences*, vol. 8, no. 9, p. 1573, Sep. 2018, doi: 10.3390/app8091573.

[22] J. A. Calvo, "Open Source Beehives Project - Audio Database as of 2-21-17," 2017. doi: 10.5281/zenodo.321345.

[23] icarodelimarodrigues, "Queen Loss Event in Africanized Honeybee Colony." [Online]. Available: https://www.kaggle.com/datasets/ica-rodelimarodrigues/queen-loss-event-in-african-ized-honeybee-colony

[24] yevheniiklymenko, "Beehive Buzz Anomalies," 2021. [Online]. Available: https://www.kaggle.com/datasets/yevheniiklymenko/beehive-buzz-anomalies

[25] C. Montaño, "Bumblebee Repository," 2026. [Online]. Available: https://github.com/carlos-montano-hub/bumblebee

[26] A. Zgank, "Acoustic Classification of Bee Swarm Activity for an IoT-Based Farm Service," Sensors, vol. 20, Dec. Dec. 2019. DOI: 10.3390/s20010021.

Table of Contribution Roles

Role	Author(s)
Conceptualization	Carlos Humberto Montaño Al-calá
Data Curation	Carlos Humberto Montaño Al-calá
Methodology	María Trinidad Serna Encinas
Project Management	Carlos Humberto Montaño Al-calá (co-author) - María Trinidad Serna Encinas (co-author).
Resources	Carlos Humberto Montaño Al-calá (joint) - Fredy Alberto Hernández Aguirre (joint).
Software	Carlos Humberto Montaño Al-calá
Supervision	María Trinidad Serna Encinas
Validation	Fredy Alberto Hernández Aguirre
Display	Carlos Humberto Montaño Al-calá (same) - María Trinidad Serna Encinas (same).
Editorial, original editor.	Carlos Humberto Montaño Al-calá (same) - María Trinidad Serna Encinas (same).
Writing, review, and editing.	Carlos Humberto Montaño Al-calá (same) - María Trinidad Serna Encinas (same) - Fredy Alberto Hernández Aguirre (same).