TEKNIK

Available online at: http://ejournal.undip.ac.id/index.php/teknik

TEKNIK, 46 (3), 2025, 295-305

Sound-Based Smart Toddler Monitoring System: AIoT Development with YAMNet on Raspberry Pi

Theresia Herlina Rochadiani [1*], Handri Santoso [2], Ito Wasito [3], Nadya Rudie Sucipto [4], Astria Febrian Anggraini [5], Ariya Panna [6]

[1, 4, 5, 6] *Informatics Study Program, Faculty of Science and Technology, Pradita University,*

[2, 3] *Magister of Information Technology, Faculty of Science and Technology, Pradita University,*

Scientia Business Park, Jl. Gading Serpong Boulevard No.1 Tower 1, Curug Sangereng, Kec.Kelapa Dua, Kabupaten Tangerang, Banten, Indonesia, 5810

Submitted: April 9th, 2025; Revised: July 10th, 2025; Accepted: July 20th, 2025; Available online: July 31st, 2025
DOI: 10.14710/teknik.v46i3.76484

Abstract

The safety of toddlers at home is paramount for parents, but constant monitoring is difficult due to busy schedules. The limitations of camera-based monitoring solutions, namely privacy concerns and heavy processing, drive the need to develop monitoring systems that utilize sound recognition. This research aims to develop Smart Guardian, an Artificial Intelligence of Things (AIoT) system that can detect risky or emergency sound patterns from children and send real-time notifications to parents' mobile phones. The applied method includes the development of a YAMNet-based sound recognition AI model, installed on a Raspberry Pi as an edge computing device, with a microphone functioning to record environmental sounds. This system is designed to identify crucial environmental sounds such as breaking glass, explosions, screaming, water, fire alarms, smoke detectors, in addition to infant crying. The results of prototype trials under laboratory conditions indicate that the fire alarm and smoke detector classes have extremely high confidence levels, around 0.95 and 0.83. However, the glass class showed varying confidence levels, around 0.5, while cough, explosion, water, and screaming had lower confidence levels with median 0.15, 0.13, 0.25, and 0.4, respectively. The conclusion from these findings is that Smart Guardian has great potential as a privacy-focused toddler monitoring solution, although further optimization is needed to improve the sound recognition performance of events with low and varying confidence levels.

Keywords: *AIoT; Raspberry Pi; sound detection; surveillance; toddlers; YAMNet*

1. Introduction

Children and the elderly are vulnerable to accidents and misfortunes. Especially children under five who are still in the period of growth and development. A high sense of curiosity makes them want to explore everything around them, so they move actively to and for without realizing and understanding the dangers around them (Asif et al., 2021). Therefore, parents or caregivers usually supervise and take care of them to ensure their safety. However, continuous supervision is impossible. This is because parents or caregivers may be careless, or also because of busyness so that supervision activities are conducted while doing other work.

The majority of accidents that occur at home happen to children. In fact, the number of children who entered the emergency room due to an accident at home almost touched 2 million (Putri, 2016). Accidents are the leading cause of injury or even death or disability in children. There are about 950,000 accidental deaths occurring each year (Gouda, Sorour, & Abdelaziz, 2022). In most studies of children's accidents, it was found that more than half of such accidents occurred in the home environment due to negligence in home safety (Silva, Fontinele, Oliveira, Bezerra, & Rocha, 2017). The factors that cause these accidents can be due to the child's age, gender and behavior, mother's education, family

*) Corresponding Author:
E-mail: th.herlina@gmail.com

socio-economic status, and home environment (Nageh, Abd El-Raouf, & Abd El-Mouty, 2020). Based on the study (Gouda et al., 2022) and (Mohammed et al., 2020) In Egypt, the three most common types of home injuries for children under the age of 5 are open wounds, burns, and fractures.

The strategy to prevent child injury is to combine the environment and behavior modification, which can be achieved by engineering to modify the environment so that the home becomes safer, with law enforcement by making rules and standards, and finally through the development of education and skills that include emergency health care (Jullien, 2021).

Information technology, especially in the field of IoT and Artificial Intelligence (AI) has come a long way. The use of this technology can be used to overcome existing problems, such as flood disaster management as part of the smart city feature (Sariffuddin, 2015), automatic toll payment (Syafei, Listyono, Prayogi, Darjat, & Hidayatno, 2019). Meanwhile, the use of information technology to supervise children in the home has been the goal of many studies. As a study conducted (Dewmini et al., 2024) which uses the emotion detection approach, body gesture detection, object detection through camera, and toddler crying analysis to monitor toddlers. The result of the study is a prototype that has an object detection feature with 89.9% accuracy, 85% emotion detection, determining the type of cry with 90.8% accuracy and a chatbot. Other studies (Prathyanga et al., 2024) building intelligent systems for daycare which can detect missing children, detect behavior and facial expressions, detect fallen children by utilizing cameras, and detect crying types. The system has high accuracy and with the implementation of this system there is an increase in safety and a quick response from caregivers of daycare. By leveraging blockchain and IoT technologies, the study (Golla, S, Swaroopa, S, & Kamat, 2024) build a monitoring system for babies that uses a humidity sensor to detect wet mattresses, a sound sensor to detect baby crying, a temperature sensor to detect the baby's and room temperature, and a camera to detect the baby's emotions. Based on the data captured by the system, then, the system will give action, such as when the baby cries, then the swing will be moved and the mother's sound will be played; when the room temperature is hot, the fan will be turned on. Other studies have also made use of AI, in addition to blockchain and IoT technology (Chauhan, Gupta, Gupta, & Haque, 2021) to create a smart swing system that can detect wet diapers and detect if children are not swinging. This system uses a sensor Force Sensing Resistors (FSR) for sensors that detect physical pressure, child pressure and weight and humidity sensors for detecting wet diapers.

In this study, the AIoT monitoring system for toddlers was built using a combination approach of AI and IoT where AI for detecting the type of sound and IoT for capturing the sound and transmitting information related to the sound to the smartphone as a notification to the parents. This study builds a sound-based AIoT that addresses privacy concerns when using camera image or video. In addition, sound was chosen because the sound detection processing process is lighter than image or video processing in edge computing. Many studies related to sound-based AIoT have been conducted, but with the scope to detect baby crying (Alam et al., 2023; Bhasha, Pavan Kumar, Baseer, & Jyothsna, 2021; Gamal, Radi, Yousef, & Ali, 2025; Lunawat, Adhduk, Sawant, & Sanghvi, 2023; Parvathavarthini, Bhuvaneswaran, Arun, & Sengottayan, 2024; Sutanto, Fahmi, Shalannanda, & Aridarma, 2021). In this study, the sounds detected were not only babies crying but sounds that often occur in home environments such as broken glass, explosions, coughing, burping, fire alarm, smoke detector, water, and screaming,

2. Materials and Methods

To build Smart Guardian, the sound-based smart toddler monitoring system there are steps that are conducted as illustrated in Figure 1. Start with collecting audio data, this study gathered audio data from AudioSet and YouTube. Then continue to the step for designing and developing system. This step includes AI modeling, IoT development, and mobile app development. The result of this step is Smart Guardian prototype. This prototype then evaluated by using Youtube audio to make sure that it runs well.

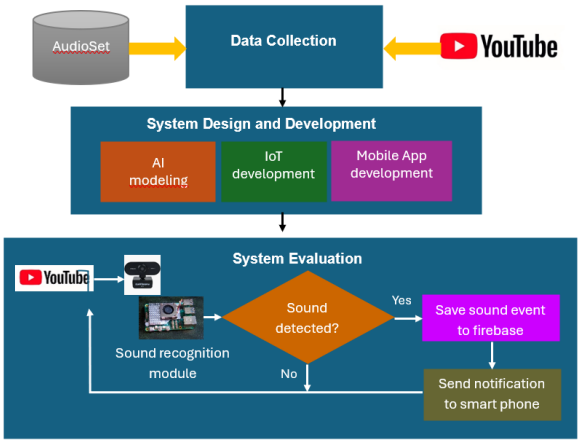

Figure 1. Research methodology.

Data Collection: Audio data for training and system testing is collected from two data sources, namely public datasets from AudioSet and YouTube. The AudioSet data, which is classified into 521 classes, becomes the

doi: 10.14710/teknik.v46i3.76484

training data in the YAMNet model. In this study, only nine relevant classes were used, namely Burping, Crying & Sobbing, Cough, Glass, Explosion, Water, Screaming, Fire alarm, Smoke detector. Figure 2 shows the log-mel spectrogram of each audio class. While YouTube data is recorded sound in the real environment, used as test data, which is played in the child's room and catched by using the built-in microphone of a Webcam device connected to a Raspberry Pi. Each audio sample was resampled to 16 kHz and 1 second in duration according to the preprocessing performed on the training data for the YAMNET model. The log-mel spectrograms of the testing data are illustrated in Figure 3.

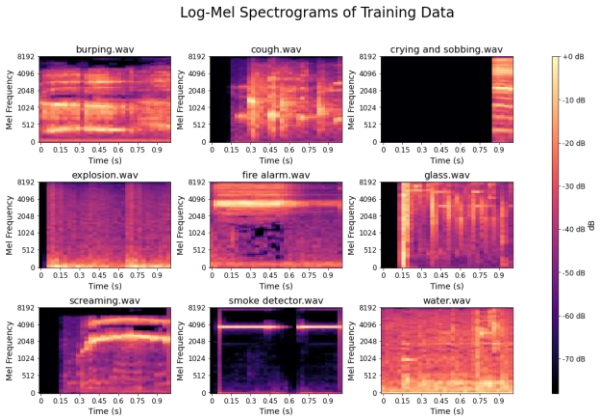

Figure 2. Log-mel spectrogram of training data.

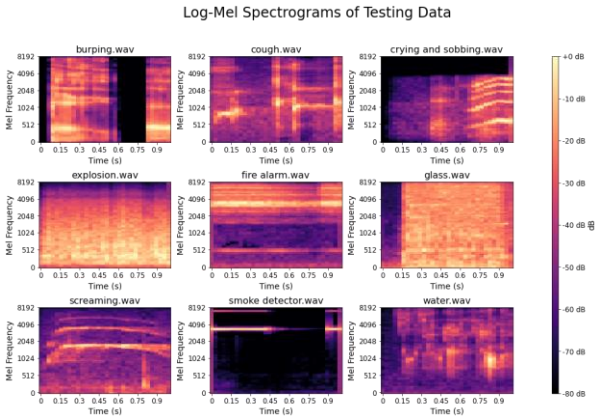

Figure 3. Log-mel spectrogram of testing data.

System Design and Development: Smart Guardian is designed as an IoT-based solution that is able to detect high-urgency sounds around children, as well as provide notifications to parents or caregivers in real-time. In addition, the system also provides visual access through a webcam camera if critical sounds are detected, so parents or caregivers can immediately confirm the baby's condition.

a. AI modelling

The system uses the YAMNet model developed by Google. This model serves as a sound classification based on the log-mel spectrogram extraction feature. This model is embedded in the Raspberry Pi via TensorFlow Lite to run locally. The real-time audio input is catched by microphone, then cut into short-duration frames, then sent to YAMNet for inference.

The output is a probability of the sound class, and if the sound is predicted to fall into a predetermined category, the system will mark it as a critical warning and trigger a system response in the form of sending a notification.

b. IoT Development

Figure 4 illustrates the development process for our Smart Guardian IoT device, detailing the sequence from hardware integration to the final alert mechanism. The process begins with the foundational step of integrating IoT components, which involves setting up and connecting the webcam centered around a Raspberry Pi 5. Once the hardware is assembled, we move to the core logic, which is coded onto the Raspberry Pi 5. First, we configured the system by filtering 9 of 512 classes that YAMNet can identify, focusing only on sounds relevant to security and safety, such as glass breaking, a baby crying, fire alarm, smoke detector. Then we embedded YAMNet model. The next step is receiving audio input from the device's microphone. This raw audio is then passed to the preprocessing stage, where it is formatted and the log-mel spectrogram extracted for analysis by YAMNet model. It is then fed into the embedded YAMNet model. The model then predicts the class of the incoming audio. If the detected sound matches one of our nine target classes, the final action is triggered: the system saves the sound event data to Firestore, our cloud database, and simultaneously sends a real-time notification to the user's device. This completes the loop, enabling the Smart Guardian to autonomously monitor the environment and alert the user to specific, critical events.

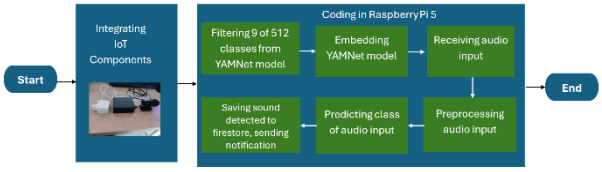

Figure 4. The process of IoT development.

c. Mobile-App Development

The mobile application was developed using the Flutter framework through a structured stage as illustrated in Figure 5. The development began with the UI/UX design process using Figma, exploring several variations of color schemes and layouts. From this stage, a mockup design

doi: 10.14710/teknik.v46i3.76484

was produced and served as a clear reference for the development of the user interface. Once the design process was completed, the next step was the preparation of the Flutter development environment. The tools used included Visual Studio Code as a text editor, along with the Flutter SDK and Android Studio. After the development environment was prepared, the next stage was the implementation of the features such as authentication, live-surveillance, and real-time log features. The implementation was conducted using the Dart programming language through the Flutter framework, where the user interface components and application logic were built according to the mockup design. Once the main features were implemented, the application entered the testing phase to ensure its functionality and performance. If bugs were found during testing, a bug-fixing process was conducted, and the application was evaluated again until the results met expectations. When the application successfully passed the testing phase, it proceeded to the build stage to generate the application package ready for deployment. This structured and iterative approach ensured that the application was developed systematically, in a measurable manner, and with a strong focus on the quality of the final outcome.

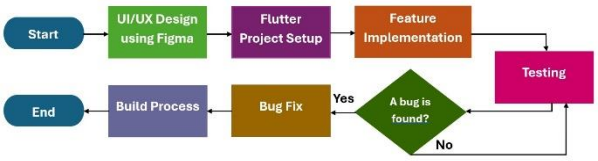

Figure 5. The stages of mobile-app development.

Here, we will elaborate on the detailed implementation of main features. The first feature is login that manages the user authentication. As shown by pseudocode in Figure 6, the implementation started by activating the loading state. Next, within the try block, it checks for notification permission; if it is not granted, the process stops after displaying an error message. If permission is granted, the function retrieves the Device ID from the input, performs the login process, and if successful, subscribes to a Firebase topic, stores the Device ID, displays a success message, and then navigates to the main page after a two-second delay. If authentication is unsuccessful, it displays 'the Device ID not found' error message. The catch block handles other unexpected errors, while the final block ensures the loading state is turned off regardless of whether the login process succeeds, fails, or encounters an error.

```
function login():
    Set isLoading = true

    Try:
        notifPermissionGranted =
          MessagingService.getAuthorizationStatus()

        If notifPermissionGranted is False:
            Show error message: "You need to grant
              notification permission..."
            Exit function

        deviceId = Trimmed text from deviceId input field
        authSuccess = AuthService.loginWithDeviceId(deviceId)

        If authSuccess is True:
            Subscribe to Firebase topic "deteksi-suara"
            Save deviceId to SharedPreferences
            Show success message: "Login successful!"
            Wait for 2 seconds
            Navigate to TabView screen

        Else:
            Show error message: "Device ID not found."

    Catch exception e:
        Show error message: "An error occurred: e"

    Finally:
        Set isLoading = false
```

Figure 6. Pseudocode of login feature.

The second feature is real-time sound events log. This real-time events log displays the sound detected with its timestamp. Figure 7 displays the pseudocode of real-time events log implementation. This pseudocode explains the logic for a date-filterable log page. The initializePage function sets the initial date to today and immediately calls loadLogs to access the data. When the user taps the date field, a date picker appears, and if a new date is selected, the loadLogs function is called again. The loadLogs function asynchronously retrieves the log data from storage, displays the loading status, and notifies users of success or failure. Finally, the filterLogs function filters the received log list to only display data that matches the `selectedDate` selected by the user.

```
function initializePage():
    selectedDate ← current date
    dateController.text ← format(selectedDate)
    call loadLogs()

function onDateFieldTapped():
    pickedDate ← showDatePicker()
    If pickedDate is not null:
        selectedDate ← pickedDate
        dateController.text ← format(pickedDate)
        call loadLogs()

function loadLogs():
    isLoading ← true
    Try:
        logs ← logRepository.getLogs(selectedDate)
        showSnackBar("Logs loaded successfully", color=green)
        _logs ← logs
    Catch error:
        print(error)
        showSnackBar("Something went wrong while loading
          logs", color=red)
    Finally:
        isLoading ← false

function filterLogs(logs):
    If selectedDate is null:
        return logs
    Else:
        return logs where log.timestamp matches selectedDate
          (year, month, day)
```

Figure 7. Pseudocode of real-time sound events log feature.

doi: 10.14710/teknik.v46i3.76484

The last feature is live surveillance feature. It provides real-time video streaming for users to monitor the toddlers. The implementation of this feature is illustrated by pseudocode in Figure 8. It describes the logic for a live surveillance page that displays a video stream from a URL. When the page loads (initState), it attempts to fetch and play the last saved URL (loadSavedUrl). The user can add a new URL and click the Connect button to invoke the connectToStream function, which then triggers initializeVideo. The main initializeVideo function manages the entire playback cycle: it sets the loading state, cleans up the old video controller, and within a try-catch block attempts to initialize a new stream, saving it if successful or changing the error state if not. The buildMainContent function dynamically displays the relevant user interface, whether it's the initial screen, a loading indicator, an error message with a retry button, or the video player itself, based on the current state. All resources such as the IP controller and video controller are gracefully cleaned up in the dispose function.

```
function initState():
    Initialize IP controller
    loadSavedUrl()

function loadSavedUrl():
    If not mounted → return
    Get saved URL from SharedPreferences
    If valid → set controller text and initializeVideo(savedUrl)

function initializeVideo(uri):
    If uri is empty or not mounted → return
    Set loading and reset error/frame flags
    Dispose old controller
    Try:
        Create and initialize VideoPlayerController
        If not mounted → dispose and return
        Assign controller, start playback, add listener
        Save URL, unset loading
    Catch error:
        Set error flag, unset loading

function disposeController():
    If controller exists → remove listener, dispose, nullify

function checkVideoStatus():
    If controller invalid or not mounted → return
    If not buffering and frame not ready → set frameReady = true

function connectToStream():
    If input is empty → return
    Unfocus keyboard, initializeVideo(trimmed input)

function buildMainContent():
    If no URL and not loading/error → show prompt
    Else if loading → show spinner
    Else if error → show retry UI
    Else if frame ready → show video player
    Else → show spinner

function dispose():
    Dispose IP controller and video controller
```

Figure 8. Pseudocode of live surveillance feature.

System Evaluation: To evaluate the performance of the Smart Guardian system in recognizing sounds in the real environment, tests were carried out using audio samples obtained from YouTube, such as the sound of broken glass, coughing, burping, and water. Each sound sample is evaluated on a YAMNet model that has been implemented on a Raspberry Pi 5 device. The evaluation is conducted by observing whether the system is able to detect and classify each sound accurately, with a high level of confidence according to the target class.

In addition to evaluating the AI model, testing was also conducted on supporting applications connected to the system. This test includes verifying whether the app is capable of displaying real-time notifications when critical sounds are detected by the system, as well as ensuring that the detected data including sound categories and time of events is stored into Firestore.

3. Results and Discussion

3.1. IoT System Architecture

The Smart Guardian system consists of three main components that are interconnected, namely sensors as physical inputs from the environment, artificial intelligence algorithms to process and classify audio data, and connectivity protocols for sending information to mobile applications. Figure 9 shows the Smart Guardian system architecture consisting of three main layers with interdependent functions. The perception layer acts as an environmental sensor, using a webcam and microphone connected to the Raspberry Pi 5 to capture real-time sound and image data from the child's surroundings that is then processed. The processed data is then collected and transmitted through the network layer, which is Wi-Fi, which connects the hardware in the home with the application services on the user's device and cloud database, Firebase Firestore, to save the collected data. Furthermore, the application layer acts as an interface for parents or caregivers to receive notifications, monitor their child's condition, and access monitoring data through a mobile application directly.

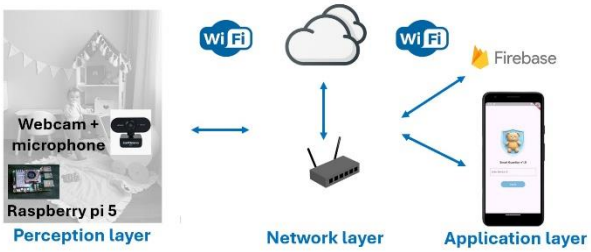

Figure 9. Smart Guardian System Architecture.

3.2. Sound Recognition

For the sound recognition required in the Smart Guardian system, this study adopts an existing model, namely YAMNet. YAMNet is a deep learning framework created specifically for audio classification. The model was developed based on the MobileNet architecture and trained with AudioSet, a large dataset that includes different types of annotated sound events. YAMNet's expertise is well suited to the purpose of this study as it can convert audio input into a mel-spectrogram and classify it into 521 different categories of sounds,

doi: 10.14710/teknik.v46i3.76484

including speech, music, background sound, and more. The use of YAMNet is expected to offer a more robust and common sound recognition solution for the Smart Guardian toddler surveillance system.

The architecture of the YAMNet model, as shown in Figure 10, consists of an input layer, a depthwise separable convolutional block, a global pooling layer, and a classification layer. The input layer receives data with dimensions of 96x64x1 which represents the number of mail bins x the number of time x the number of channels. Depthwise separable convolution blocks are the core of the YAMNet architecture, to separate convolution operations into two steps, namely depthwise convolution and pointwise convolution. Depthwise convolution uses a 3x3 filter that is applied to each channel separately. Pointwise convolution that follows depthwise convolution is a 1x1 convolution that combines the output of each channel. These two convolutions are conducted repeatedly throughout the network to efficiently extract features from inputs. The global pooling layer is found after a series of final convolutional blocks, where this layer takes the output of the spatial feature and reduces it to a single feature vector with dimensions of 1x1x1024, before passing it to the fully connected layer, which summarizes the spatial information of the entire feature map. The classification layer becomes the last layer of this architecture. This classification layer consists of a fully connected (FC) layer, a SoftMax classifier, and a confidence score. The FC layer performs a linear transformation to map features to lower latent spaces. Then the output of this FC is applied to a SoftMax function to produce the probability of each class which is referred to as a confidence score.

In order for the model to recognize sound input from the real environment, the sound input is preprocessed. The preprocessing flow is illustrated in Figure 8. It begins with the process of normalizing the captured audio signal. This normalization process changes the amplitude value of the audio signal in the range of -1.0 and 1.0 values, thus avoiding features that have the greatest amplitude dominating the model learning process. After the normalization process, the audio signal is resampled to 16 kHz according to the sampling rate of the audio data used to train the model. Next, the audio signal is transformed into a log-mel spectrogram representation with the steps shown in Figure 11. The final stage of preprocessing is to divide the log-Mel spectrogram into smaller segments.

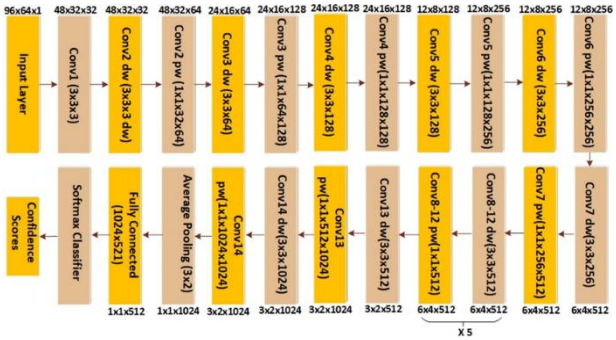

Figure 10. YAMNET Architecture (Mahum, Irtaza, Javed, Mahmoud, & Hassan, 2024).

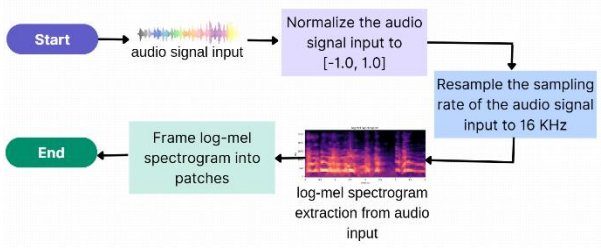

Figure 11. Preprocessing of audio inputs.

Each sound captured will be extracted with the log-mel spectrogram feature to be the input of the YAMNet model. The algorithm extracting the log-mel spectrogram feature is shown in Figure 12. First, the audio signal is resampled to 16 kHz if there is a difference. Next, parameters for the Short-Time Fourier Transform (STFT) such as window length (400 samples or 25 ms), hop (160 samples or 10 ms), and FFT length (1024) are set. Furthermore, the magnitude spectrum is calculated using STFT. A Mel filter bank with 64 bins is applied to this spectrum, then the result is a logarithmic transformation to produce a log-mel spectrogram as an output.

doi: 10.14710/teknik.v46i3.76484

```
Algorithm : generate_logmelspectrogram( audio, sr)

Input       : audio-waveform of audio, sr-sampling rate
Output      : log_mel
// change the sampling rate of audio input to 16 KHz
IF sr ≠ 16000:
        audio ← RESAMPLE(audio, orig_sr=sr, target_sr=16000)
//setting STFT parameters
window_length_samples ← 400     # 25 ms at 16 kHz
hop_length_samples   ← 160      # 10 ms hop
fft_length         ← 1024
//calculate STFT value
    stft_result ← STFT(audio,
                frame_length=window_length_samples,
                hop_length=hop_length_samples,
                fft_length=fft_length)
//calculate the magnitude of spectrogram
magnitude ← ABS(stft_result)
//apply mel filterbank
    mel_spectrogram ← MEL_FILTERBANK(magnitude,
                        num_mel_bins=64,
                        lower_freq=125,
                        upper_freq=7500,
                        sample_rate=16000)
//take the logarithmic value
log_mel ← LOG(mel_spectrogram)
return log_mel
```

Figure 12. The algorithm for extracting the log-mel spectrogram feature from the audio.

A spectrogram is a 2D representation of the squared magnitude of the STFT; hence, the formula for obtaining the spectrogram is shown in Equation (1).

$$Y(m,k) = |X(m,k)|^2 \tag{1}$$

Meanwhile, the amount of STFT value is calculated based on the equation formula (2)

$$STFT(x)(m,k) \equiv X(m,k) = \sum_{n=0}^{N-1} x(n+mH).w(n).e^{-i.n\left(2\pi\frac{k}{N}\right)} \tag{2}$$

where $X(m,k)$ is the value of STFT, m: signal frame, k:Frequency, $w(n)$: window functions, such as Hanning, Hamming, $x(n+mH)$: the signal segment where frame m is located. $e^{-in\left(2\pi\frac{k}{N}\right)}$: a complex exponential function that plays an important role in Fourier analysis, where i is an imaginary value, n is the time index, N is the FFT measure or the number of frequencies bins and k is the index of the frequency bin.

3.3. Smart Guardian Mobile App

The Smart Guardian mobile application is built to be used by users in monitoring children. This application will connect to the user's IoT device. Therefore, on the login page, shown by Figure 13, the user needs to enter the Device ID. This device ID is the user's IoT device that has been registered in the system database.

To help with monitoring, this application has two main features, namely Realtime logs and Live Surveillance. The Realtime log feature, which can be seen in Figure 14, displays the events of the detected sound with the order of the most recent occurrence placed at the top. Each event is equipped with data on the date and time of the event. Meanwhile, the Live surveillance feature is a feature that allows users to activate the camera to see real-time conditions after a sound event is detected, as shown in Figure 15.

Figure 13. Smart Guardian login page.

doi: 10.14710/teknik.v46i3.76484

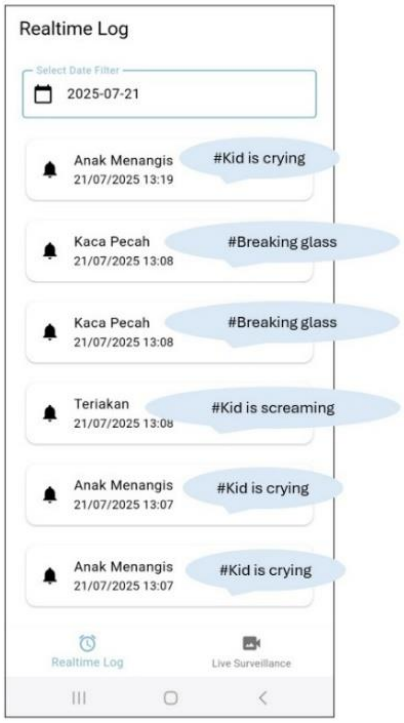

Figure 14. Realtime log page.

Figure 15. Live surveillance page.

3.4. Evaluation Results

The Smart Guardian prototype was evaluated in a laboratory environment. Each class is assessed with audio from three videos played from the YouTube app. The confidence level of each correctly recognized audio will be recorded, while for unrecognized audio, the confidence level value is given a value of 0. Using boxplot visualization, the test results are presented and analyzed.

From the visualization of the boxplot, seen in Figure 16, it can be seen that the fire alarm and smoke detector classes have high confidence and consistency. The fire alarm class showed the highest median confidence, almost reaching 0.95. The box is very tight, suggesting that most predictions are in the range of 0.92 to 0.98, with a whisker that is close to 1.0. This shows that the model is very reliable in recognizing fire alarm sounds. Like the fire alarm class, the smoke detector class also reflects high and stable confidence, with a median of around 0.83. As for the glass class, the median confidence is around 0.5, which reflects moderate uncertainty. However, the box is very wide, ranging from 0.38 to 0.62, and the whisker covers a large range from about 0.28 to 0.75. This shows that the model's confidence in the sound of glass breaking varies greatly.

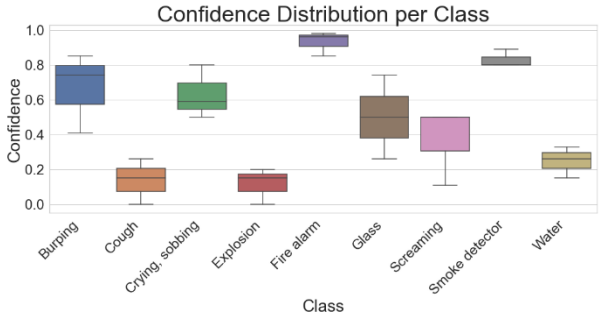

Figure 16. Boxplot of confidence level from test results for each audio class.

Some classes show that the model always has low confidence in its predictions, even though the variation is slight, namely the cough, explosion, and water classes. The cough class showed an extremely low median confidence, around 0.15, with a narrow box range between 0.08 to 0.25. This indicates that models are consistently less confident in assigning classifications for cough sounds. Like the cough class, the explosion class has a small median of confidence, about 0.13, with a very narrow range between 0.1 to 0.18. This indicates that the model looks very skeptical about the explosion sound. While the water class is not much different, this class has low confidence and is relatively stable, with a median of about 0.25 and a maintained range between 0.18 to 0.3. Models tend to be hesitant when predicting water sounds.

doi: 10.14710/teknik.v46i3.76484

The screaming class had a fairly low median confidence value, around 0.4. The box is quite wide, ranging from 0.3 to 0.5, and the bottom whisker stretches to about 0.1, showing some predictions with incredibly low confidence levels. Models seem to lack confidence in recognizing screaming sounds, and their confidence levels can vary significantly.

The data distribution is very dense, almost entirely above 0.8. This shows that the model has a very consistent and reliable prediction performance for smoke detector sounds.

3.5. Comparison with the existing works

As mentioned in the introduction, numerous studies have been conducted on the application of information technology to monitor children at home. Table 1 summarizes the comparison between the existing and the proposed systems. We can see that Table 1 demonstrates significant advantages of the Proposed Work compared to previous research on toddler monitoring systems. While existing studies have mostly focused on detecting infant cries, this study can recognize a much wider spectrum of emergency sounds. This includes crying, burping, coughing, fire alarms, smoke detectors, water, glass breaking, explosions, and screaming. The notification feature in the mobile app, along with enhancements such as live webcam monitoring and sound logging, strengthens this system as a more comprehensive and responsive monitoring solution for the diverse range of potential risks surrounding toddlers.

Table 1. Comparison between the existing works and the proposed work.

Work	Sound detected	Alert	Additional Features
(Durga, Itnal, Soujanya, Basha, & Saxena, 2021)	crying	Nx Siemens software	Using webcam to monitor the children; using sensors to monitor moisture and ambient temperature; smart cradle to help children sleep
(Zubair, Evans, Aterorimie, & Adedigba, 2021)	crying	Blynk	There are lighting, cooling fan, and lullaby player; using sensors to monitoring mattress wetness, body temperature, ambient temperature.
(Sundarajoo, Gwo Chin, Leong, & Fun, 2022)	crying	mobile app	Using white noise to comfort the baby; using LEDs and IP cameras; using sensors to monitoring temperature, mattress wetness, and moving objects.
Proposed	crying, burping, cough, fire alarm, smoke detector, water, glass breaking, explosion, screaming	mobile app	Webcam for real-time surveillance; realtime log of sound events

4. Conclusion: This research successfully achieved its goal of developing Smart Guardian, a sound-based Artificial Intelligence of Things (AIoT) system designed to detect risky or emergency sound patterns from toddlers and send real-time notifications to parents. The system leverages the YAMNet AI model implemented on edge devices such as the Raspberry Pi, offering a monitoring solution that prioritizes privacy and computational efficiency over camera-based approaches. Test results demonstrated the system's good ability to identify critical and distinct sounds such as fire alarms and smoke detectors, which were detected with a high level of confidence, around 0.95 and 0.83. However, detection of several other sound classes still showed low levels of confidence, such as cough, explosion, water, and scream classes which have confidence levels with medians of 0.15, 0.13, 0.25, and 0.4, respectively. It indicates the areas that need improvement. Overall, this study successfully demonstrated the feasibility and potential of Smart Guardian as an innovative toddler monitoring system, effective in providing early warnings and supporting child safety in the home environment.

Acknowledgements: Our gratitude goes to the Institute for Research and Community Service (LPPM) Pradita University for funding and supporting this research activity through the Even Semester 2024/2025 Internal Grant No. 002/LPPM-PT/SPK/PRADITA/III/2025. And we also thank the parties both directly and indirectly who have helped and supported this research activity from the beginning of development to prototype testing.

Bibliography

Alam, H., Burhan, M., Gillani, A., Haq, I. ul, Arshed, M. A., Shafi, M., & Ahmad, S. (2023). IoT Based Smart Baby Monitoring System with Emotion Recognition Using Machine Learning. Wireless Communications and Mobile Computing, 2023, 1–11. https://doi.org/10.1155/2023/1175450

Asif, R., Azam, N., Raza, F. A., Riaz, M., Zulfiqar, S., & Razzaq, M. (2021). Knowledge, Attitude and Practices Regarding First Aid against Domestic Injuries in Mothers of Children less than 5 Years of Age Attending Fauji Foundation Hospital Islamabad. Pakistan Journal of Public Health, 11(3), 151–157. https://doi.org/10.32413/pjph.v11i3.761

Bhasha, P., Pavan Kumar, T., Baseer, K. K., & Jyothsna, V. (2021). An IoT-Based BLYNK Server Application for Infant Monitoring Alert System to Detect Crying and Wetness of a Baby. In Advances in Intelligent Systems and Computing: Vol. 1312 AISC (pp. 55–65). https://doi.org/10.1007/978-981-33-6176-8_7

Chauhan, H., Gupta, D., Gupta, S., & Haque, M. J. (2021). A Smart Cradle System to Monitor Infants for Healthcare Baby Wards Based on IoT and Blockchain. 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), 606–609. IEEE. https://doi.org/10.1109/ICAC3N53548.2021.9725717

Dewmini, A., Shashindi, M., Uththara, K., Sathushka, G., Manathunga, K., & Karunathilaka, S. (2024). Todly: Multimodal Approach for Toddler Safety and Well-Being Using Advanced Machine Learning. 2024 6th International Conference on Advancements in Computing (ICAC), 67–72. IEEE. https://doi.org/10.1109/ICAC64487.2024.10850986

Durga, S., Itnal, S., Soujanya, K., Basha, C. Z., & Saxena, C. (2021). Advanced and effective baby care monitoring Smart cradle system using Internet of Things. 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), 35–42. IEEE. https://doi.org/10.1109/ICOSEC51865.2021.9591955

Gamal, M., Radi, I., Yousef, A., & Ali, A. G. M. (2025). Internet of Things-Based Smart Infant-Incubators Using Machine Learning Analysis. IEEE Access, 13(March), 77673–77685. https://doi.org/10.1109/ACCESS.2025.3564872

Golla, M., S, S., Swaroopa, P., S, S. M., & Kamat, S. (2024). Computational Intelligence based Smart Nest for Infants. International Journal of Innovative Research in Technology, 10(11), 1100–1109. https://doi.org/10.13140/RG.2.2.36226.70082

Gouda, A., Sorour, A., & Abdelaziz, M. (2022). Knowledge and Practice of Mothers Regarding First aids of Home Injuries Among Preschool Children. Zagazig Nursing Journal, 18(2), 206–218. https://doi.org/10.21608/znj.2022.269324

Jullien, S. (2021). Prevention of unintentional injuries in children under five years. BMC Pediatrics, 21(S1), 311. https://doi.org/10.1186/s12887-021-02517-2

Lunawat, S., Adhduk, A., Sawant, V., & Sanghvi, R. (2023). Smart Baby Cradle for Infant Soothing and Monitoring. In Practical Data Mining Techniques and Applications (1st ed., p. 16). Auerbach Publications.

Mahum, R., Irtaza, A., Javed, A., Mahmoud, H. A., & Hassan, H. (2024). DeepDet: YAMNet with BottleNeck Attention Module (BAM) for TTS synthesis detection. EURASIP Journal on Audio, Speech, and Music Processing, 2024(1), 18. https://doi.org/10.1186/s13636-024-00335-9

Mohammed, Z., Aledhaim, A., AbdelSalam, E. M., El-Setouhy, M., EL-Shinawi, M., & Hirshon, J. M. (2020). Factors associated with injuries among preschool children in Egypt: demographic and health survey results, 2014. BMC Public Health, 20(1), 595. https://doi.org/10.1186/s12889-020-08658-w

Nageh, H., Abd El-Raouf, S., & Abd El-Mouty, S. (2020). MOTHERS' KNOWLEDGE AND SUBJECTIVE PRACTICE TOWARD MOST COMMON DOMESTIC INJURIES AMONG UNDER-FIVE CHILDREN. Mansoura Nursing Journal, 7(1), 19–35. https://doi.org/10.21608/mnj.2020.175751

Parvathavarthini, S., Bhuvaneswaran, U., Arun, U. M., & Sengottayan, S. (2024). Baby Monitoring Using IOT and Deep Learning. 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), 1–9. IEEE. https://doi.org/10.1109/ICCCNT61001.2024.10725151

Prathyanga, A., Shyaminda, P., Chamikara, P., Lakshan, S., Thelijjagoda, S., & Kasthurirathna, D. (2024). Intelligent Daycare: Enhancing Child Safety with IoT and Machine Learning Innovations. 2024 9th International Conference on Communication and Electronics Systems (ICCES), 530–538. IEEE. https://doi.org/10.1109/ICCES63552.2024.10859472

Putri, B. J. (2016, September 15). 2 juta anak indonesia masuk IGD karena kecelakaan di rumah. Liputan6.Com. Retrieved from https://www.liputan6.com/health/read/2602688/2-juta-anak-indonesia-masuk-igd-karena-kecelakaan-di-rumah

Sariffuddin, S. (2015). PELUANG PENGEMBANGAN SMART CITY UNTUK MEWUJUDKAN KOTA TANGGUH DI KOTA SEMARANG (Studi Kasus: Penyusunan Sistem Peringatan Dini Banjir Sub Drainase Beringin). Teknik, 36(1), 32–38. https://doi.org/10.14710/teknik.v36i1.7823

Silva, M. F. da, Fontinele, D. R. D. S., Oliveira, A. V. S. de, Bezerra, M. A. R., & Rocha, S. S. da. (2017). Determining factors of domestic accidents in early childhood. Journal of Human Growth and Development, 27(1), 10. https://doi.org/10.7322/jhgd.127643

Sundarajoo, R. A., Gwo Chin, C., Leong, P. W., & Fun, T. S. (2022). A Remote Baby Surveillance System with RFID and GPS Tracking. International Journal of Engineering Trends and Technology, 70(11), 81–92. https://doi.org/10.14445/22315381/IJETT-V70I11P208

Sutanto, E., Fahmi, F., Shalannanda, W., & Aridarma, A. (2021). Cry Recognition for Infant Incubator Monitoring System Based on Internet of Things using Machine Learning. International Journal of Intelligent Engineering and Systems, 14(1), 444–454. https://doi.org/10.22266/IJIES2021.0228.41

Syafei, W. A., Listyono, A. F., Prayogi, A. S., Darjat, D., & Hidayatno, A. (2019). Pengembangan Perangkat Lunak Untuk Gerbang Tol Otomatis Yang Ramah Lingkungan Berbasis RFID Dengan Notifikasi Pembayaran Tanpa Kertas. Teknik, 40(1), 31. https://doi.org/10.14710/teknik.v39i3.22829

Zubair, A. R., Evans, U. A., Aterorimie, P. S., & Adedigba, A. . (2021). Development of a Baby Care System. Journal of Emerging Trends in Engineering and Applied Sciences, 12(6), 165–171.

doi: 10.14710/teknik.v46i3.76484