Ultra-Low-Power IoT Solutions for Sound Source Localization:

Combining Mixed-Signal Processing and Machine Learning

Daniel de Godoy Peixoto

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2019

ABSTRACT

Ultra-Low-Power IoT Solutions for Sound Source Localization:

Combining Mixed-Signal Processing and Machine Learning

Daniel de Godoy Peixoto

With the prevalence of smartphones, pedestrians and joggers today often walk or run while listening to music. Since they are deprived of auditory stimuli that could provide important cues to dangers, they are at a much greater risk of being hit by cars or other vehicles. We start this research into building a wearable system that uses multichannel audio sensors embedded in a headset to help detect and locate cars from their honks and engine and tire noises. Based on this detection, the system can warn pedestrians of the imminent danger of approaching cars. We demonstrate that using a segmented architecture and implementation consisting of headset-mounted audio sensors, front-end hardware that performs signal processing and feature extraction, and machine-learning-based classification on a smartphone, we are able to provide early danger detection in real time, from up to 80m distance, with greater than 80% precision and 90% recall, and alert the user on time (about 6s in advance for a car traveling at 30mph).

The time delay between audio signals in a microphone array is the most important feature for sound-source localization. This work also presents a polarity-coincidence, adaptive time-delay estimation (PCC-ATDE) mixed-signal technique that uses 1-bit quantized signals and a negative-feedback architecture to directly determine the time delay between signals in the analog inputs and convert it to a digital number. This

direct conversion, without a multibit ADC and further digital-signal processing, allows for ultralow power consumption. A prototype chip in 0.18µm CMOS with 4 analog inputs consumes 78nW with a 3-channel 8-bit digital time-delay output while sampling at 50kHz with a 20µs resolution and 6.06 ENOB. We present a theoretical analysis for the nonlinear, signal-dependent feedback loop of the PCC-ATDE. A delay-domain model of the system is developed to estimate the power bandwidth of the converter and predict its dynamic response. Results are validated with experiments using real-life stimuli, captured with a microphone array, that demonstrate the technique's ability to localize a sound source. The chip is further integrated in an embedded platform and deployed as an audio-based vehicle-bearing IoT system.

Finally, we investigate the signal's envelope, an important feature for a host of applications enabled by machine-learning algorithms. Conventionally, the raw analog signal is digitized first, followed by feature extraction in the digital domain. This work presents an ultra-low-power envelope-to-digital converter (EDC) consisting of a passive switched-capacitor envelope detector and an inseparable successive-approximation-register analog-to-digital converter (ADC). The two blocks integrate directly at different sampling rates without a buffer between them thanks to the ping-pong operation of their sampling capacitors. An EDC prototype was fabricated in 180nm CMOS. It provides 7.1 effective bits of ADC resolution and supports input-signal bandwidth up to 5kHz and an envelope bandwidth up to 50Hz while consuming 9.6nW.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank Prof. Peter Kinget and Prof. Fred Jiang for the opportunity, dedication, guidance and trust given to me during this research. The lessons they taught me will follow me far beyond the conclusion chapter.

I would also like to thank my defense committee members: Prof. Naveen Verma, Dr. Mudhakar Srivatsa, and Prof. Ioannis Kymissis for their careful review and comments on this work.

Thanks to all my colleagues from the CISL, ICSL, CLUE and other research groups at Columbia University and elsewhere with whom I had the great pleasure of working. Thanks to Prof. Kymissis and Aida Colón-Berrios for the opportunity to collaborate in their research on the monolithic integration of piezoelectric MEMS on CMOS. I would like to thank Prof. Nirjon, Bashima Islam, Md Tamzeed Islam and Stephen Xia for all the great collaboration effort put on the developement of *PAWS*. Thanks to Prof. Mingoo Seok and João Pedro Cerqueira from the Columbia VLSI lab for their assistance with the ASIC integration. Finally, I thank Prof. K. P. Pun and Erjia for a tremendous effort and commitment on the envelope-to-digital converter.

I would also like to thank all the Electrical Engineering department staff, especially Elsa Sanchez and Laura Castillo, who helped me countless times with all my issues.

I am also grateful to LASPAU, especially Aline Santos, who helped me greatly with my legal sponsorship during my studies.

To my family. I cannot thank my mother, father and sister enough for all the support and love they gave me during this process. All you did for me could not possibly be put in words here. This work is not only dedicated to you, it is yours.

And to my wife, Marcela, who heard every possible complaint with outstanding patience, pushed me to my best with relentless conviction every time it was needed, and shared all the achievements with amazing joy, I leave a reminder that this is only the beginning.

Chapter 1

Introduction

1.1 Defining IoT

Scrolling down through the proceedings of the most reputable conferences in integrated circuits, sensing networks, and embedded systems, the *Internet of things* (IoT) can be found as the motivation of countless works on a wide range of topics. But what is IoT? How can it drive research in ultra-low-power transmitters, low-jitter PLLs, and reliable speech recognition at the same time? Even the abstraction within its name, Internet of **things**, hints just how broad this subject can be. In reality, the IoT is not a new concept or a field of study that can be sustained on its own, but a collection (or integration) of required knowledge to make the IoT vision feasible. This vision, in its turn, promises that in a near future a massive network of electronic gadgets will surround our society, assisting us on our day-to-day activities [70]. This idea is so well accepted as an inevitable outcome of the development of current technologies that the academic society decided to prepare the technical environment required for it to happen. Consequently, researchers from various fields are trying to tailor their efforts to achieve this common goal.

To organize this confusion of IoT subjects, a subset of three domains can be used

to group the efforts of the academic work related to IoT applications, as shown in Fig. 1.1

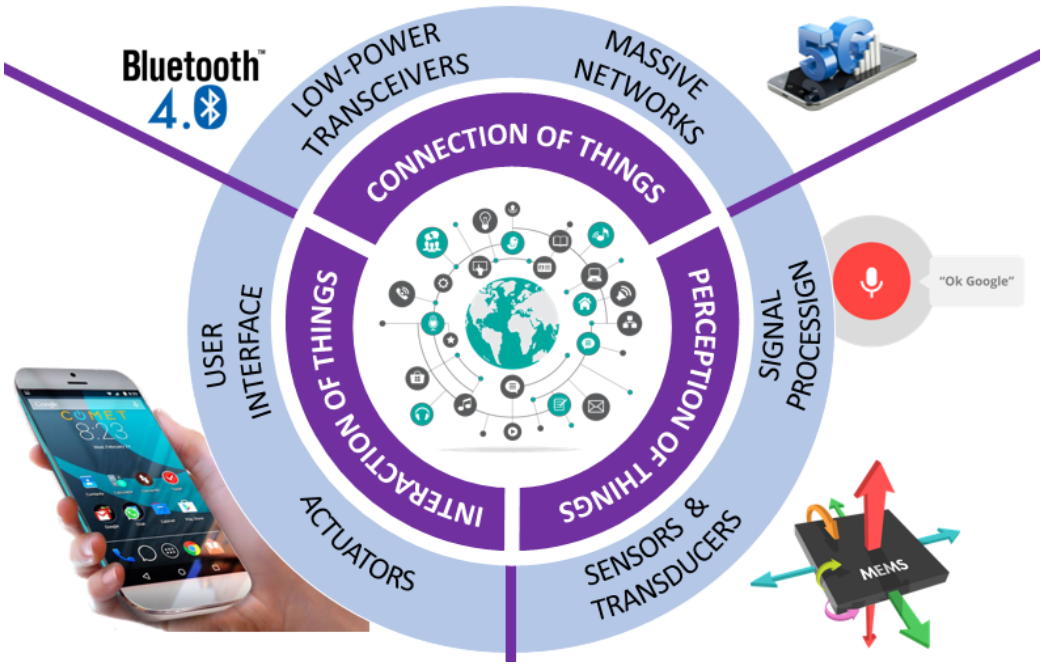

Figure 1.1: Segregation of the different fields of research within the IoT domain.

The connection of things embraces the communication challenges of a crowded machine-to-machine (M2M) network. One of the greatest challenges of the IoT vision is sustaining the ever-growing data-traffic requirements. Works related to low-power transceivers [48] and their subblocks [23, 46], as well as massive network architectures and protocols [16, 34] belong to this group.

The perception of things deals with systems that can understand or interpret the stimuli from the real world, hereby refereed as perceptive systems. Naturally, research related to sensors and transducers [37, 33, 84], data converters [20], and signal processing algorithms [21] fit in this category.

Finally, the **interaction of things** closes the loop by studying the how these

systems provide feedback to the users. The spread of smartphones opened several possibilities on how to translate the data gathered by the systems to intelligible information for the user in graphical or audio interfaces, yet new avenues such as haptic approaches are still being pursued to enhance this experience [45].

It is impossible to predefine the requirements of each IoT application, but some technical constrains should be common to most of the systems, such as power, bandwidth, size, cost, speed, reliability, and so on. Any progress that significantly reduces any of these requirements might provide the missing block for an IoT solution or enable the development of new IoT products.

1.2 Embedded IoT Perceptive Systems

Embedded IoT perceptive systems are the main target of this research. As rule of thumb, these systems run on battery, have limited processing and storage capabilities, have a reduced form factor, use wireless communication, and are expected to be affordable to the public. Table 1.1 provides a list of recently published end-to-end embedded IoT perceptive systems. They range from health monitoring to enhanced security in transactions to behavioral control, but they all share the same principle: detecting one or multiple events happening in the physical world and use them to provide some user feedback.

Each solution presents its own architecture, and deals with system-resource distribution in its own way, they all share a higher-level data-processing pipeline from

data to information, knowledge, and wisdom (DIKW) [60, 66]. The DIKW structure is used to hierarchically describe the different stages of the system from data to decisions and actions. **Data** are defined as symbols that carry properties of the event under observation, not always in a usable structure. **Information** is inferred, extracted from data; it is the means to provide quantitative or qualitative answers about the event. **Knowledge** is the application of information to provide answers; it is a deduced symptom of the event. **Wisdom** is the diagnostic or action made based on the knowledge of the event; it involves a higher understanding of facts beyond the observed event, it can be subjective to the user or action taker. When those notions are specifically applied to embedded perceptive systems, the transition blocks that change the data from a stage to another can also be labeled and analyzed as in Fig. 1.2.

The first challenge is to translate the real-world stimulus into an electrical signal; for that, we use **sensors**: microphones, photodiodes, capacitive touch sensors, thermistors. These are all components or structures that change their electrical behavior depending on physical changes in the environment. Even in the best-case scenario, when the physical change being measure is of an electrical nature as in an ECG [6],

Authors, year	**System purpose**
Jia et al. 2018 [38]	Low-power continuous ammonia monitoring
Yang et al. 2016 [80]	Enabling secure device paring
Nguyen et al. 2016 [56]	Diagnosing sleep disorder
Bui et al. 2017 [15]	Measuring blood oxygen level
De Godoy et al. 2017 [25]	Preventing sunburn and skin cancer
Adkins et al. 2016 [1]	Verifying smoking cessation
Goel et al. 2016 [28]	Measuring lung function

Table 1.1: Recently published end-to-end embedded perceptive systems.

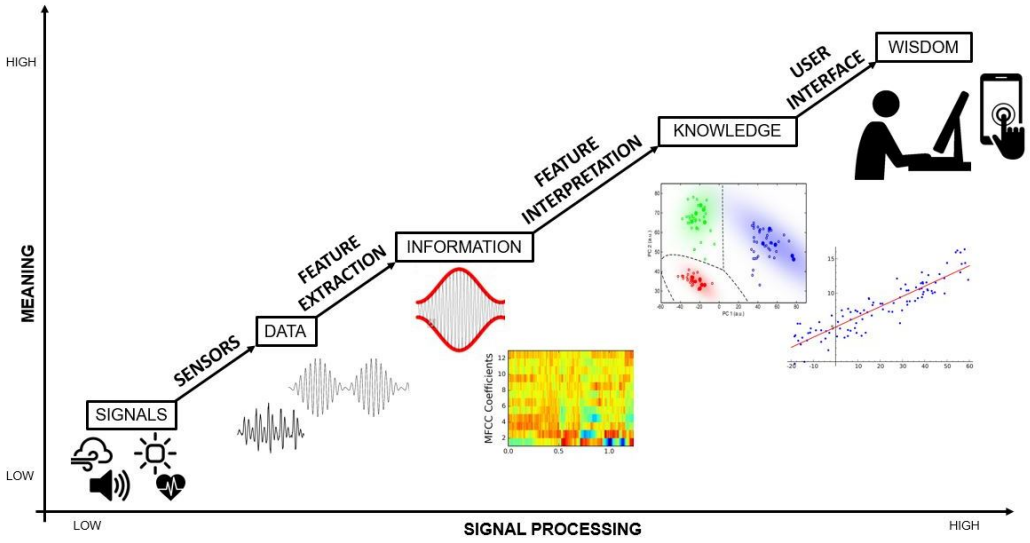

Figure 1.2: Stages of the signal-processing pipeline in a generic embedded perceptive system.

the electrodes used to capture the signal deserve equal attention [72].

The next step is to **extract** from the electrical signal generated by the sensors the **features** that carry information about the physical event. This process can be as straightforward as observing the amplitude of the input signal, or it can involve a more complex investigation of other signal characteristics, such as the behavior of specific spectrum components used in speech recognition [55].

Even after the corresponding feature is extracted from the sensor's signal, there is a gap between the values captured in the features and intelligible knowledge that can be shown to the user. The features must be **interpreted**. Just like the feature extraction, feature interpretation can vary in complexity depending on the application. A thermometer may simply match a voltage acquired from a thermocouple to a corresponding temperature using a linear equation, while an elaborate machine-learning classifier might be needed to identify potential health issues in a ECG reading [39].

Lastly the **user interface** is needed to transfer the information to the user. Unless the system itself intends to act upon the captured information, the whole process is useless if not externalized to the user. The visualization of the content happens in different ways, in real-time or though recorded log files, locally in the embedded device or on a host platform, by the user or by a third party. All these factors play a role in the final solution.

The first objective of this dissertation is to detail the development steps of a specific IoT perceptive system, an early-danger-detection wearable system using passive audio sensors to detect and localize approaching vehicles for pedestrian safety. It investigates the required signal-processing algorithms for a sound-source-localization solutions and shows an example of an end-to-end IoT implementation.

Furthermore, after implementing the system in traditional fashion, this work shows how to further optimize the performance of perceptive systems by combining mixed-signal processing techniques and machine-learning classifiers and proposing the use of analog-to-feature converters.

1.3 Analog-to-Feature Converters

"*The world is analog. ... Computers are digital*" [40]. The balance between the drawbacks and benefits of performing signal processing in either of these domains is a puzzling dilemma that bothered circuit designers for decades [61, 44, 77]. The clear choice of most modern systems is to rely heavily on digital-signal processing and use

the analog and mixed-signal components only to handle simple anti-aliasing, data conversion, and communication. However, while the flexibility and robustness of digital techniques guarantee its place in the core of complex algorithms, recent developments in supervised machine-learning classifiers brings new opportunities for mixed-signal processing in front-end signal handling.

In traditional architectures, feature extraction from the raw electrical sensor inputs proceeds in two steps: a data converter stage followed by digital feature extraction. The data converter, or analog-to-digital converter, is responsible for storing in digital code all aspects of the analog inputs as reliably as needed for the digital-signal processor to extract the features. This often involves preserving the spectral content of the signal—e.g., sampling faster than the Nyquist rate—and guaranteeing that the quantization noise will not exceed a critical threshold.

The main issue with this approach is that most of the effort of faithfully encoding all the information of the analog inputs add no meaning to the data; it can be seen as a horizontal step in the DIKW analysis (Fig. 1.3). Extra hardware, memory and processing resources are allocated to changing the encoding domain, or representation symbols, of the data just to be disregarded after the actual feature information is extracted.

In contrast with the traditional approach, this work proposes an investigation of a more direct conversion, an analog-to-feature converter. In the analog-to-feature converter a mixed-signal approach is used to only digitally encode the actual information that will required by rest of the processing pipeline, avoiding the need to sample

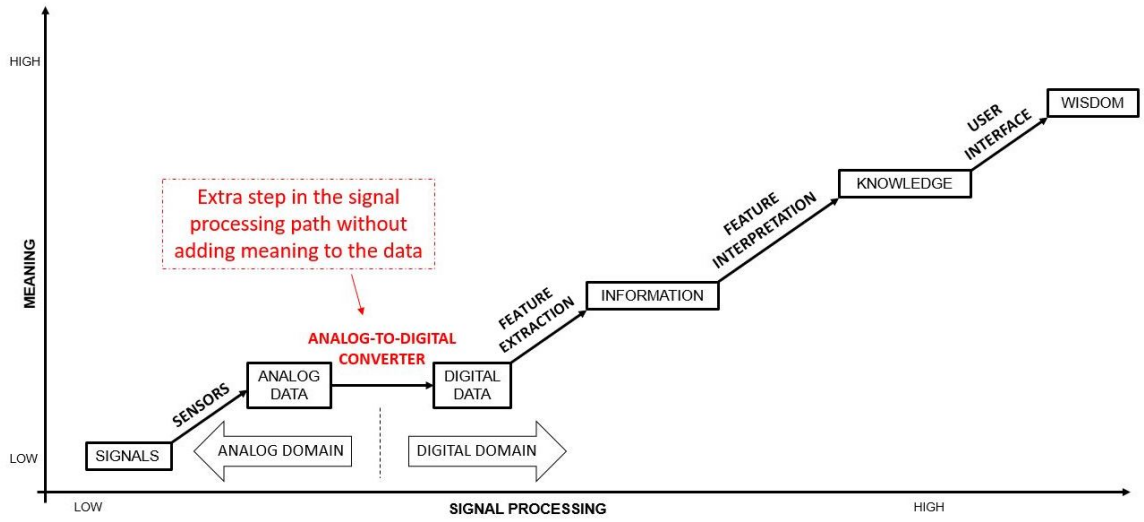

Figure 1.3: DIKW analysis including the data-converter step. The horizontal shift in the processing path shows that there is no increase in the data's meaning during the analog-to-digital conversion.

the input signal and store large amounts of data. This approach both deals with the analog/digital interface required by real-word/computer architectures and adds meaning to processed data. As shown in Fig. 1.4, the analog-to-feature converter fits in harmony with the DIKW analysis. Adding more mixed-signal processing to the architecture also further allows for ultra-low-power implementations, since analog techniques can be used to trade-off power consumption and accuracy. Also, a wider error margin and variations in the front-end processing can be handled with supervised machine-learning algorithms.

This architecture might not be ideal for all embedded IoT perceptive-system implementations, but examples are presented in this dissertation where ultra-low-power analog-to-feature ASICs were developed to extract both the arrival-time difference between multiple microphones and the amplitude envelope of a raw audio signal. The mixed-signal processing in the ASICs can be four orders of magnitude more

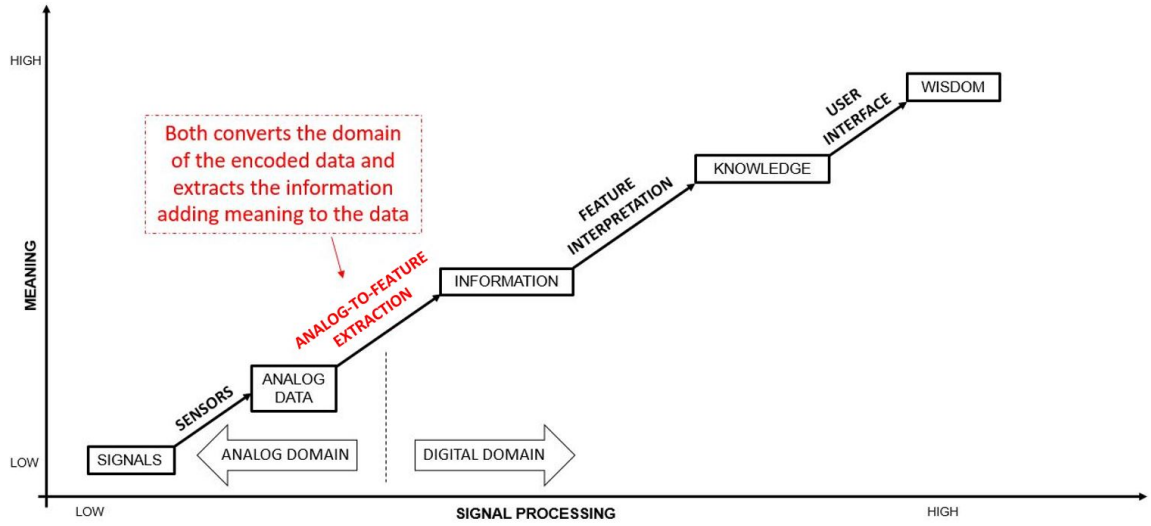

Figure 1.4: DIKW analysis of architectures with analog-to-feature converters. The data's meaning grows in each step of the processing pipeline.

efficient than the traditional data converter with a digital-signal processor without significantly degrading the final accuracy of the system.

1.4 Outline

This thesis explores the challenges of embedded implementations of IoT sound-source-localization systems and how it can combine mixed-signal processing and machine-learning algorithms to enhance the performance of perceptive systems. Chapter 2 presents a literature review on physical phenomena that enable sound-source-localization. Chapter 3 shows the implementation and characterization of an audio-based wearable IoT alert system for pedestrian safety that uses sound-source-localization algorithms. Chapter 4 introduces a 78nW ultra-low-power analog-to-feature time-delay estimation technique as an enhancement to the signal processing pipeline of the previously presented system. Chapter 5 presents a sub-10nW switch-

capacitor-based envelope-to-digital converter for audio applications. Chapter 5.3 summarizes the work and analyzes the analog-to-feature approach as a solution for a wide range of IoT perceptive systems.

Chapter 2

Sound-Source Localization Features Overview

2.1 Introduction

Object recognition and localization have been extensively explored in the literature. Nearly all the avenues explored mirror techniques present in nature, such as the use of stereo imaging [11], ultrasonic radar [9], and acoustic-source localization [22]. In vehicular tracking, video based approaches have been widely used [78, 11, 50]. More information can undoubtedly be extracted from images than from any other types of date; it is not by chance that humans learned to rely so much on their visual system. Commonality in vehicles' shapes and standardized road signs have enabled the use of sophisticated machine-learning algorithms to identify and predict the movement of cars [83]. Although such systems offer outstanding solutions for devices that can be hosted in large platforms, such as an autonomous car for collision prevention [71], they are less suitable for wearable systems. A major limitation is the computational requirements of real-time imaging processing and the feasibility of developing low-cost, power-efficient, rapid-response products. Another major issue is user privacy. As it was previously pointed out, constantly taking images of one's activities reveals an alarming amount of personally identifiable information.

Such active techniques as radar and LIDAR can certainly be used to detect obstacles and even some spatial behaviors [35, 17], but such solutions face great challenges in classifying the nature of those obstacles. This is particularly problematic in urban environments, with their abundant moving and stationary obstacles but relatively rare real threats to the user. On the implementation side, the inherently high power dissipation of active transducers discourages their use in most portable devices.

Passive audio sensors, on the other hand, provide enough information to allow classification and localization of the source with lower computational and power requirements. Audio classification has been used to detect such events as coughing [32], gun shots [18], human activity (talking, crying, running, etc.) [5], subtle sounds like keyboard typing and door knocking [73], and buses and trucks passing [51]. The rest of this chapter outlines the main principles allowing sound-source classification and localization. We investigate the physical phenomena that can be used to infer the position of a sound source as a background to understanding the feature-extraction decisions made in the subsequent chapters.

2.2 Physical Principles of Sound-Source Localization

Estimating the position or direction of acoustic sources is an essential skill for many living beings. Doing so accurately can be a life-or-death factor in the wild or in modern cities. Even though doing it is seamless to us, the actual mechanism involved

in this task are not yet fully understood. Neuroscientists and biologists examining the different aspects of the auditory system regarding sound-source localization have noticed that, in most animals, three main binaural cues are used to localize a sound source:

Interaural Time Differences

The *interaural time difference* (ITD) is the difference between the arrival times of the sound wave to each ear, as illustrated in Fig. 2.1. The correlation between the ITD and the source's localization is clear; if a source is to the left of the listener, the wave will reach the left ear earlier. Given the average human ear spacing, around 21cm, the ITD is to be most relevant for low-frequency sounds, with spectrum components below 1.5kHz [58]. Higher frequency components with wave length smaller than the space between the ears may lead to ambiguous estimations.

Interaural Level Differences

The next major spatial hearing cue is the *interaural level difference* (ILD), shown in Fig. 2.2. The ILD refers to the difference in power of the sound waves captured by each ear. It is also intuitive to predict that a sound source located to the left of the listener will be heard more loudly in the left ear. Notice, however, that the reason for this level change is mostly the *acoustic shadow* created by the human head, not the different lengths travelled by the sound wave to reach each ear. In a complementary fashion with the ITD, the ILD effect is more pronounced for high-frequency sounds

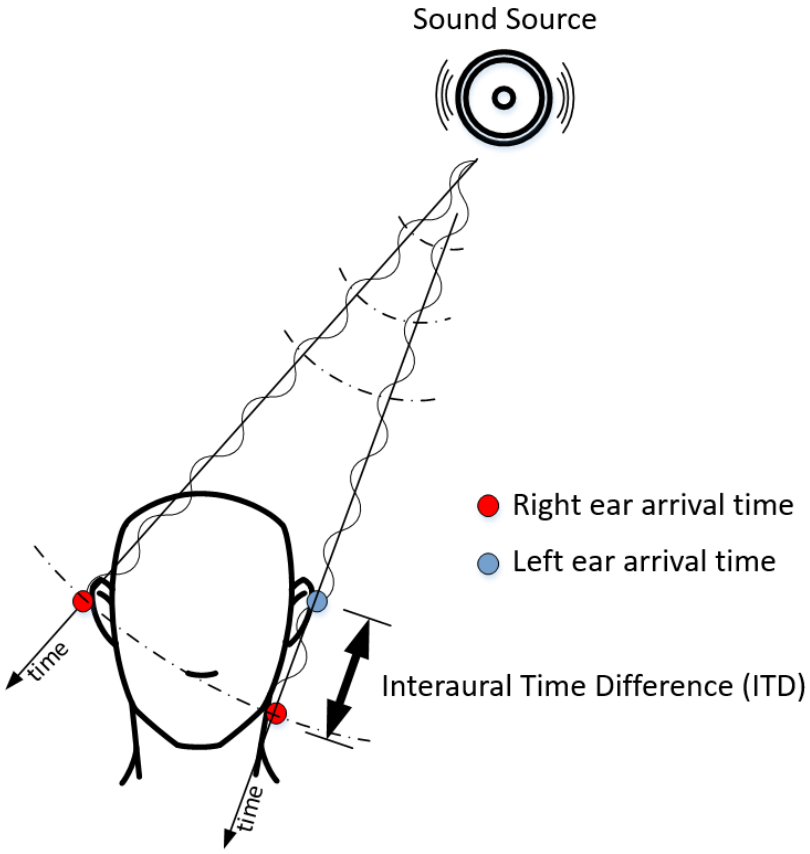

Figure 2.1: Illustration of the interaural time difference (ITD). The ITD is the most important spatial cue for sound-source localization, as the sound wave reaches each ear at a different time depending on the source's position.

[58].

Head-Related Transfer Function

With only two ears, ITD and ILD alone are not enough for humans to accurately localize the sound source. There are multiple points in space that will provide exactly the same ITD and ILD. This region of equal ITD and ILD is called the *cone of confusion* and is shown in Fig. 2.3. An asymmetric element is needed to distinguish the unique position of the sound source within the cone of confusion.

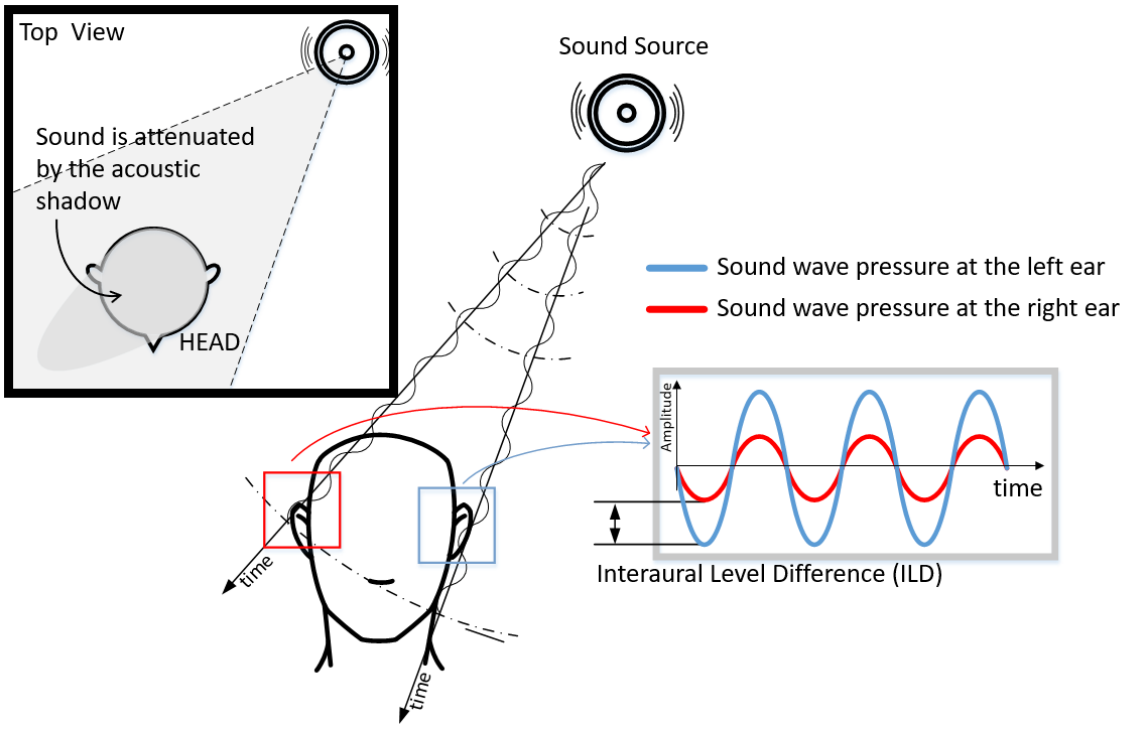

Figure 2.2: Illustration of the interaural level difference (ILD). Because of the acoustic shadow created by the head, the sound wave will reach each ear with different levels. The ILD is harder to characterize than the interaural time difference because the acoustic shadow depends heavily on the listener's mechanical structure.

The pinna, the outer part of the ear, is responsible for this distinction (Fig. 2.4). The sound wave will be guided to the inner ear differently depending on where the wave hit the pinna, leading to different *head-related transfer functions* (HRTFs). This subtle feature is so important for our binaural sound-source localization that most virtual reality technologies include the HRTF in their system [10].

2.3 Conclusion

This chapter presented the main features used in sound-source localization. Understanding the physical phenomena that naturally allow us to notice if a car is ap-

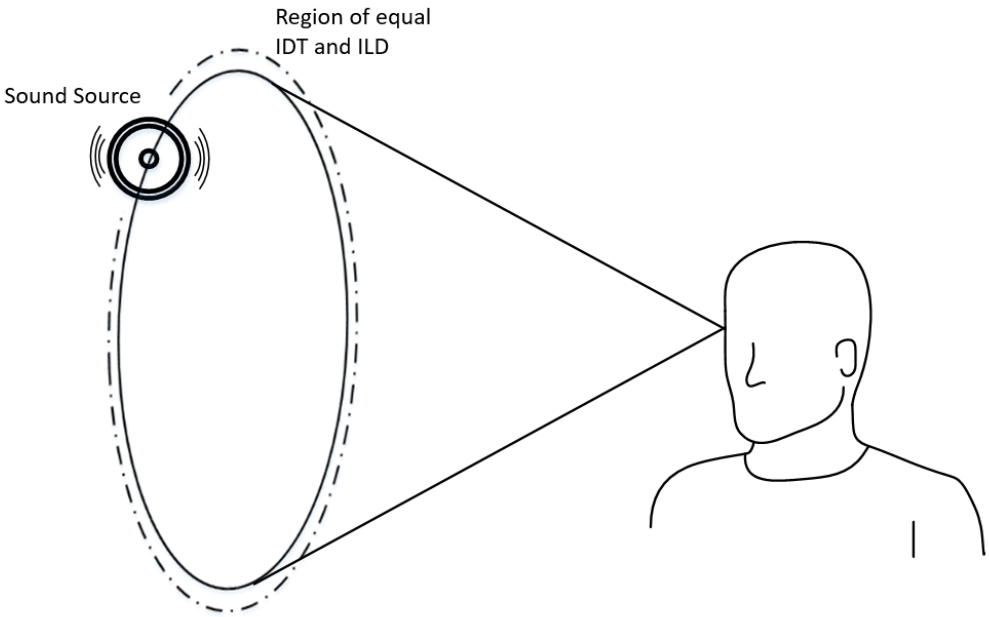

Figure 2.3: Illustration of the cone of confusion. Any sound source placed at the base of the cone will have the same ITD and ILD. This ambiguous region exists because humans only have two ears. In a system with arbitrary numbers of receivers, a thoughtful geometric distribution of the microphones could eliminate this region.

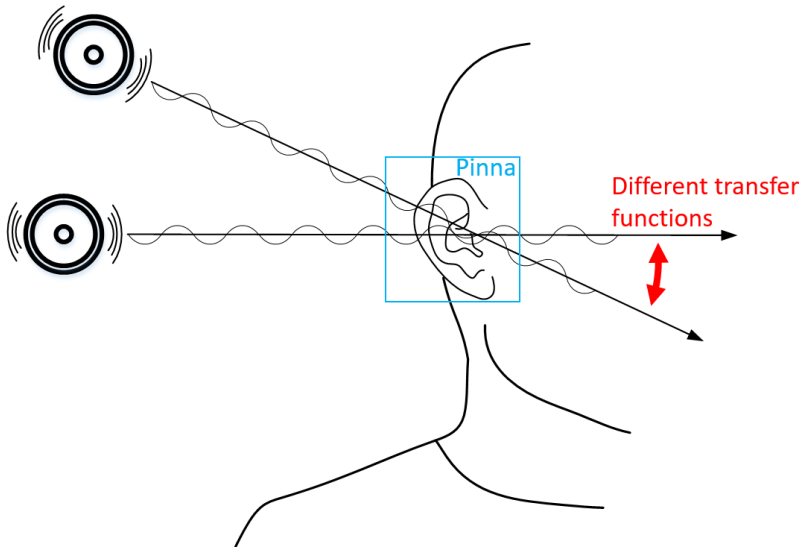

Figure 2.4: Illustration of the head-related transfer function. Different sound-wave incidence angles to the ear have different transfer functions due to the asymmetric shape of the pinna.

proaching from behind will help understand the work presented the next chapters.

From the features presented above, only the ITD and the ILD can be practically implemented on embedded systems. The mechanical-structural dependency of the HRTF places the investigation of its implementation outside the limits of this work. ITD and ILD, on the other hand, have been used on numerous embedded system and characterized in detail. They are used in Chapter 3 to implement an end-to-end embedded sound-source localization system with off-the-shelf components.

The complexity of extracting ITD features makes it worthy of a more detailed analysis. The phenomenon described here as ITD—or, more broadly, the interval difference required for a signal to reach multiple receivers—is referred in multiple works as time difference of arrival (TDoA). The study of TDoA is not limited to audio signals; it is commonly used in radio-frequency systems such as GPS and radar. There are different methods to extract the time delay between to two signals. Each algorithm has advantages and drawbacks that should be explored before choosing one to implement. Chapter 4 presents a method that combines cross-correlation with an adaptive negative-feedback loop. This method combines the low computational requirements of adaptive time-delay estimators with the principles of cross-correlation functions. Using an understanding of the cross-correlation function, the theoretical boundaries of operations are calculated and measured. Making it a reliable and viable alternative for sound-source localization in resource-constrained systems.

Chapter 3

PAWS: An Audio-Based CPS Solution for Pedestrian Safety

3.1 Introduction

Smartphones have transformed our lives dramatically, mostly for the better. Unfortunately, listening to music while walking has also become a serious safety problem in urban areas around the world. As reported by the *Washington Post,* pedestrians listening to music, texting, talking or otherwise absorbed in their phones are making themselves more vulnerable by tuning out the traffic around them [65]. Since a pedestrian is deprived of auditory input that would provide important cues to dangers such as honks or noises from approaching cars, he or she is at a much greater risk of being involved in a traffic accident. According to a study by Injury Prevention and CNN, the number of serious injuries and deaths involving pedestrians who were walking with headphones has tripled in the last seven years in the United States [57]. This global phenomenon is an important societal problem that we want to address by introducing advanced sensing techniques and intelligent wearable systems.

We tackle these challenges in PAWS, a *pedestrian audio wearable system* targeting

urban safety. PAWS is a low-cost headset-based wearable platform that combines four microelectromechanical system (MEMS) microphones, signal-processing and feature-extraction electronics, and machine-learning classifiers running on a smartphone to help *detect* and *locate* imminent dangers, such as approaching cars, and *warn* pedestrians in real time.

Newer smartphones being equipped with multiple built-in microphones, it may seem tempting to repurpose those microphones in software to localize cars based on ITD or other localization techniques. But these approaches require the user to hold the phone steady in their hand instead of keeping it inside a pocket [68]. It is unrealistic to expect users to constantly hold their phones steady and to not block the built-in microphone while walking. In addition, most built-in microphones are designed for voice and are often band-limited. These two limitations prevent the smartphone from capturing useful features produced by approaching cars in realistic urban environments.

This is a challenging problem as the battery-powered wearable platform must detect, identify, and localize approaching cars in real time, process and compute large amounts of data in an energy- and resource-constrained system, and produce accurate results with minimal false positives *and* false negatives. For example, if a user's reaction time is 500ms, the system has 360ms to detect a 25mph car and alert the user when it is 10m away from him. This problem is further compounded by high levels of mixed noise, typical of realistic street conditions in metropolitan areas.

To tackle these challenges, we develop a segmented architecture and data pro-

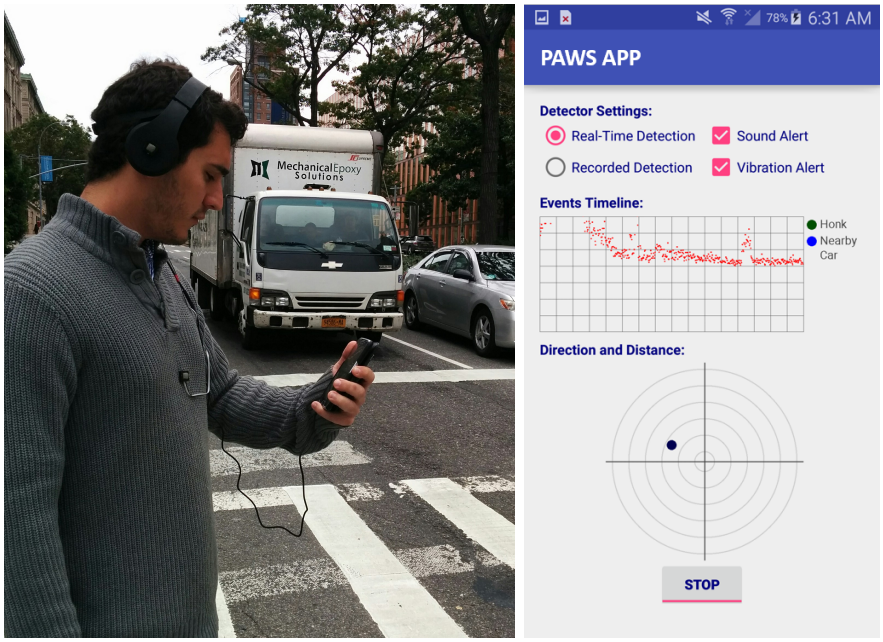

Figure 3.1: An inattentive pedestrian wearing a PAWS headset, and a screen shot of the PAWS application user interface.

cessing pipeline that partitions computation into processing modules across a front-end hardware platform and a smartphone. Four channels of audio are collected by a microcontroller-based front-end platform from four MEMS microphones strategically positioned on a headset. Temporospatial features such as relative delay, relative power, and zero-crossing rate are computed inside the front-end platform using the four channels and transmitted wirelessly to the smartphone. A fifth standard headset microphone is also connected to the audio input of the smartphone, and together with the data sent from the front-end platform, classifiers are trained and used to detect an approaching car and estimate its azimuth and distance from the user. We evaluate PAWS using both controlled experiments in parking lots and real-world deployments on urban streets. We make the four contributions in this work:

- We create an end-to-end, low-cost, wearable system and smartphone application

that accurately provides real-time alerts to pedestrians in noisy urban environments. Inattentive pedestrians can immediately benefit from our system.

- We develop a segmented architecture and data-processing pipeline that intelligently partitions tasks across the front-end hardware and the smartphone to ensure accuracy while minimizing latency.
- We propose a new acoustic feature designed to capture frequency-domain characteristics of such low-frequency noise as the sound of a car's tires against the road. We develop classifiers to recognize cars approaching the user and to localize approaching cars, with respect to the user, in real time.
- We share with the community our entire data set, which includes high-fidelity multichannel audio recordings of moving car sounds, honks, and street noises, that we have collected in a metropolitan area and in a college town.

The work presented in this chapter was performed in collaboration with Prof. Nirjon and his students in the University of North Carolina (UNC) at Chapel Hill. All members of the project worked on multiple parts and were fundamental to the system's integration. The UNC team focused on the smartphone signal processing related to the vehicle detection such as the new acoustic feature presented in Section 3.3.

3.2 Studying the Problem

We studied the car-sound recognition-and-localization problem using a validation platform before developing PAWS into a wearable system. The objective of this ex-

ercise was to analyze the feasibility and complexity of our proposed solution and to determine the specifications required to capture the necessary information: sampling rate, sensor placement, and relevant features for the machine-learning algorithms.

As shown in Figure 3.2, the validation platform directly connected eight MEMS microphones to a computer. The microphones were placed on a mannequin head to reproduce the physical phenomena of the final setup, such as the acoustic shadow of the human head [58] and the approximate spacing among sensors on a real user.

The study was performed in five locations in two different cites: a metropolitan area and a college town. The locations were two parking areas, a four-way intersection, and two multi-lane streets. We recorded audio from a total of 47 cars. We conducted our first set of controlled experiments in the parking areas with labeled distances, directions, and precise time keeping of honks and car passing. All other scenarios were uncontrolled.

Recording Specifications

To characterize the sounds of interest, such as an approaching vehicle's tire noise, engine noise, and honks, we conducted controlled experiments in two parking areas (Figure 3.2 shows one of the experiments). These results are later compared to the uncontrolled experiments for consistency.

Figure 3.3 shows the spectrogram of one recording from the controlled experiments. The top and the bottom spectrograms correspond to the same recording. Approximately 5s after the recording starts, a car honks, resulting in distinct sta-

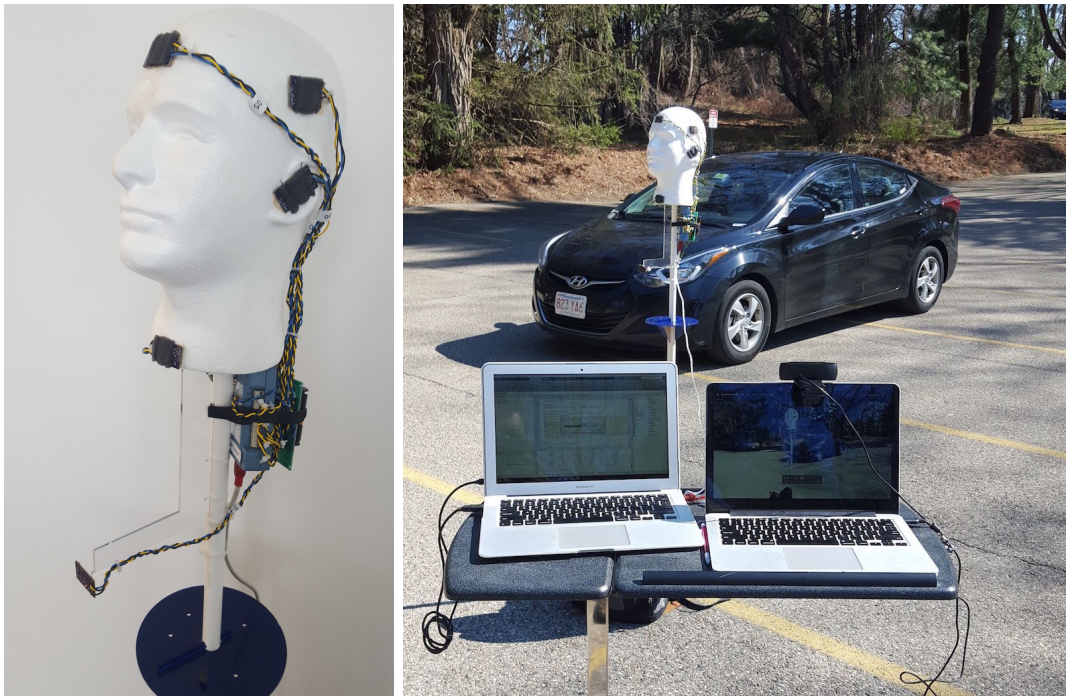

Figure 3.2: Validation system: Reference mannequin with eight MEMS microphones and data-acquisition board in a low-noise controlled-experiment setup.

tionary tones with fundamental frequencies near 500Hz. The vehicle then accelerates towards the mannequin. In the bottom figure, zoomed in on the lower part of the spectrogram, we see the engine noise. The engine noise follows the engine's speed (RPM). In a car with an automatic transmission, the engine noise is bounded between 60Hz and 200Hz. (notice the transmission shifting at the seven-second mark.) Once the vehicle gets closer to the mannequin, the friction noise from the tires and asphalt gets louder. This noise has a band-limited spectrum with more energy below 3kHz. When the car makes its closest approach to the system near the twelve second mark, a burst of air causes loud white noise. Similar spectrum components were found on several recordings of different approaching cars at similar speed (20–30mph) on dry asphalt.

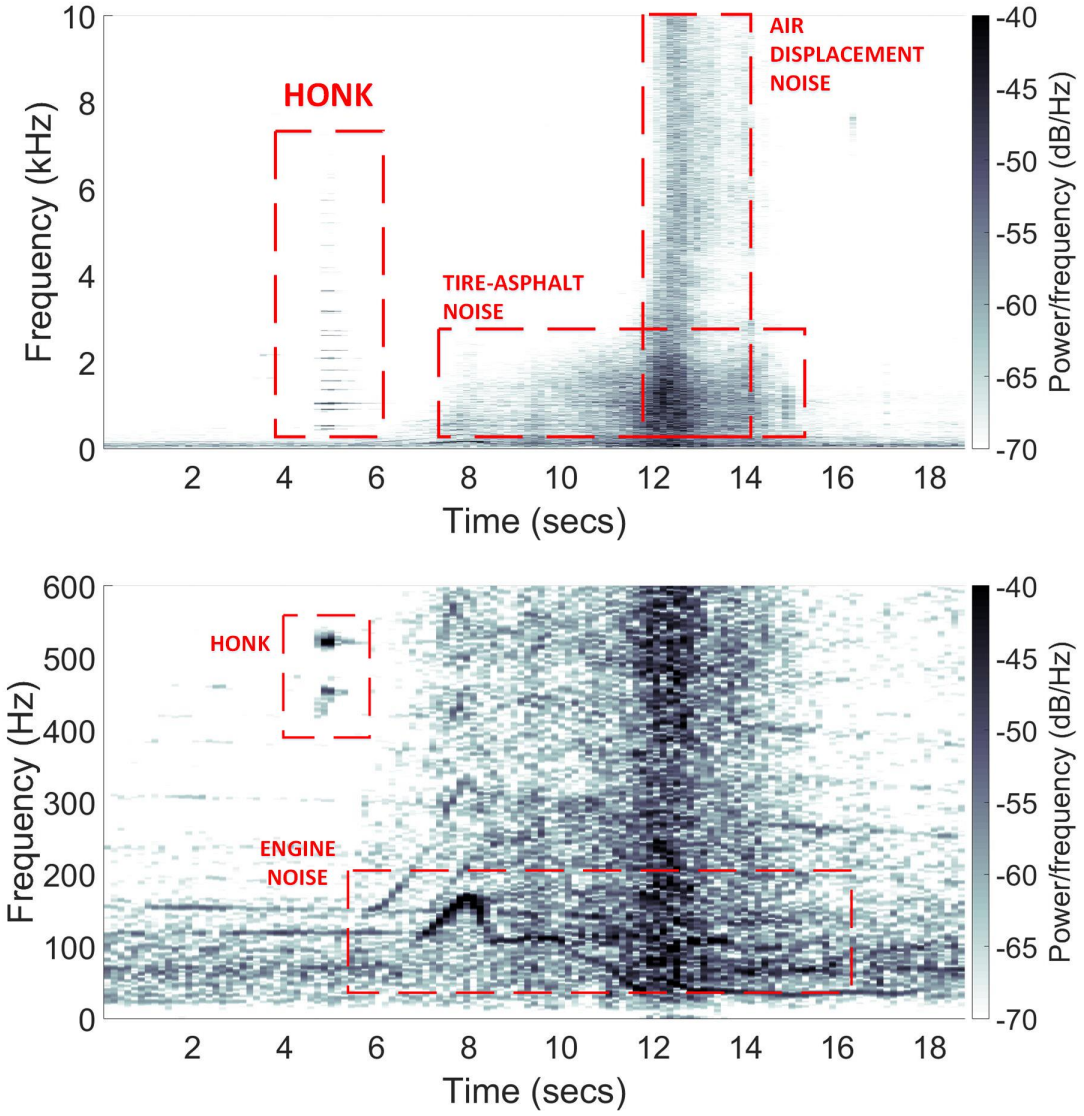

Figure 3.3: Spectrogram of one of the recordings from the controlled environment. The car was approaching the mannequin at 25mph.

These observations indicate that the audio system must reliably capture frequencies from 50Hz to 6kHz to accurately identify warning honks and vehicles that are still approaching the user. This requirement means that the system needs custom microphone drivers with a cut-off frequency of less than 10Hz (in contrast to standard headset microphones with a 100Hz cut-off frequency) and analog-to-digital converters with sampling rates above 12kHz.

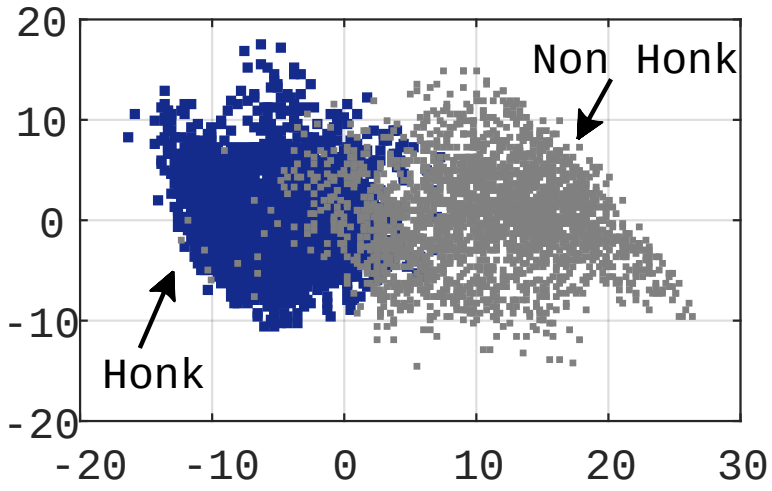

Figure 3.4: Distribution of honks and other types of sounds in a 2D feature space.

Presence of a Car

The presence of a car can be determined from high-energy, sharp sounds like honks, and from such low-energy, noise as the sound of tires on the road.

Honks are louder and easier to detect than tire or engine sounds. We analyze the Mel-frequency cepstral coefficients (MFCC) [53] of honks and compare them with other street sounds. The MFCC is one of the most commonly used acoustic features for detecting various types of sounds [49, 41, 59, 42] including car sounds [12]. For visualization purpose, we reduce the thirteen-dimension MFCC features to two dimensions (using PCA [54]) and the result is shown in Figure 3.4. We observe that honks are separable from other sounds as they cluster around a different point in the space. Honks are easily detectable using all thirteen coefficients.

MFCCs, however, are not effective in detecting other types of car noises, such as the sound of tires on the road. The fundamental reason behind this is that the Mel scale, expressed as $m = 2595 \log_{10}(1 + f/1000)$, was originally designed to mimic

human hearing of speech signals that maps frequencies $f < 1$kHz somewhat linearly, and maps $f > 1$kHz logarithmically. Our analysis of tire sounds shows that about 60% of the signal energy is attributed to frequency components below 1 kHz. Hence, to model such low-energy, low-frequency, noise-like sounds, we need to develop a new feature that captures these subkilohertz characteristics of audio signals. Section 3.3 describes this new acoustic feature.

Direction of a Car

To determine the direction to the car, we recorded audio of cars approaching from different directions and analyzed the signals captures by the different microphones. Some of these recordings also have honks in them. Intuitively, microphones that are closer to the sound source and are not obstructed by the human head should receive signals earlier, and the signals should be stronger. Hence, the relative delays and the relative energy of the received signals should be strong indicators of the direction from which a car is approaching.

In Figure 3.5, we plot the relative delays of the microphones with respect to the front microphone for left- and right-side honks. As expected, the relative delays change signs for left and right honks. We do similar tests in eight directions (each covering a 22.5° 3D cone surrounding the mannequin) to successfully determine the directions to honks near the user.

Similarly, we plot the microphones' relative delays as a car passes the mannequin from its left to right (Figure 3.6). We observe that the relative delays are quite random

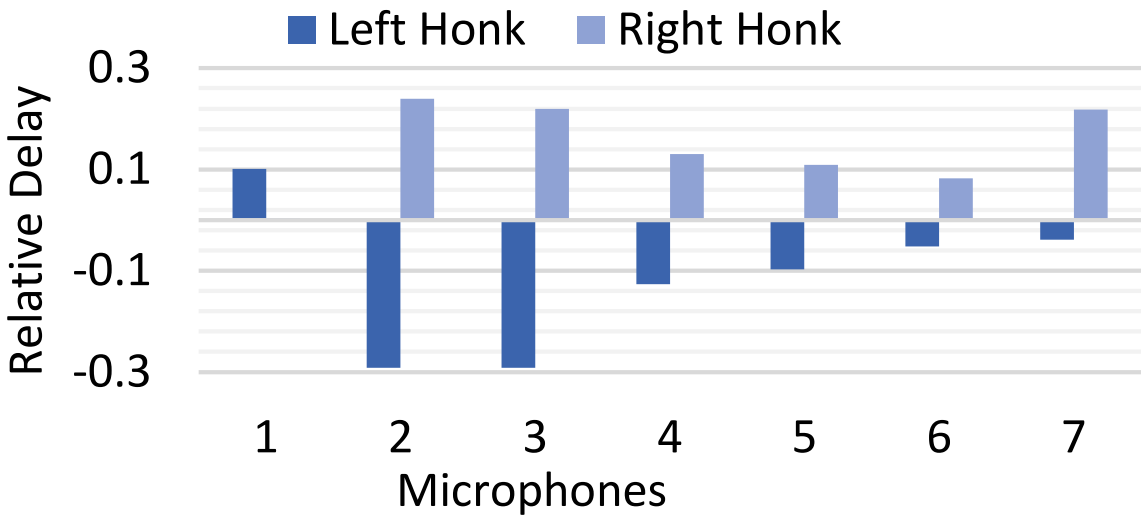

Figure 3.5: Normalized relative delays of the microphones for left and right honks.

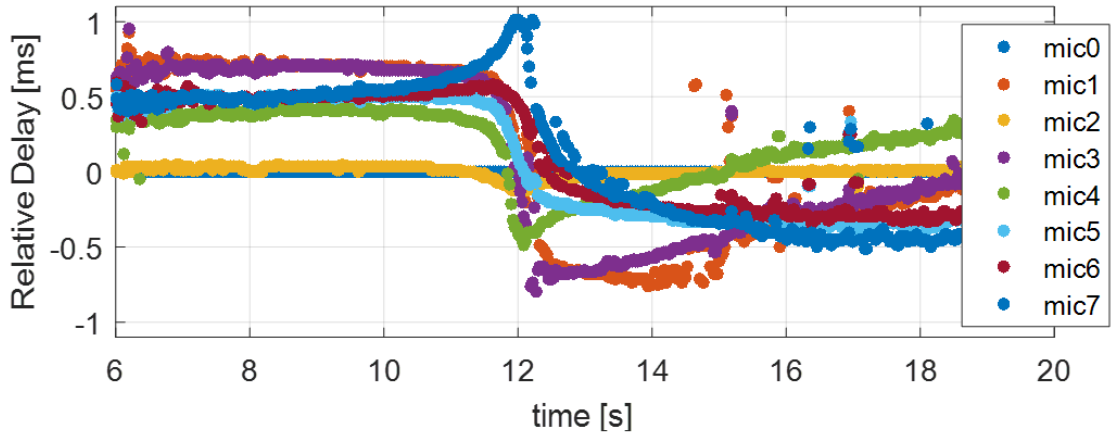

Figure 3.6: Relative delay versus time of a car driving past a mannequin from its left to right.

on both left and right ends. As the car approaches the mannequin, we see a trend in all the curves with one or more of them peaking. The trend reverses as the car passes the mannequin. This behavior suggests that patterns in relative delays (when they are looked at together) are useful to determine the direction of passing. Hence, by learning the trend and the point when the trend reverses, it is possible to differentiate a car on the left from a car on the right, as well as their angular directions.

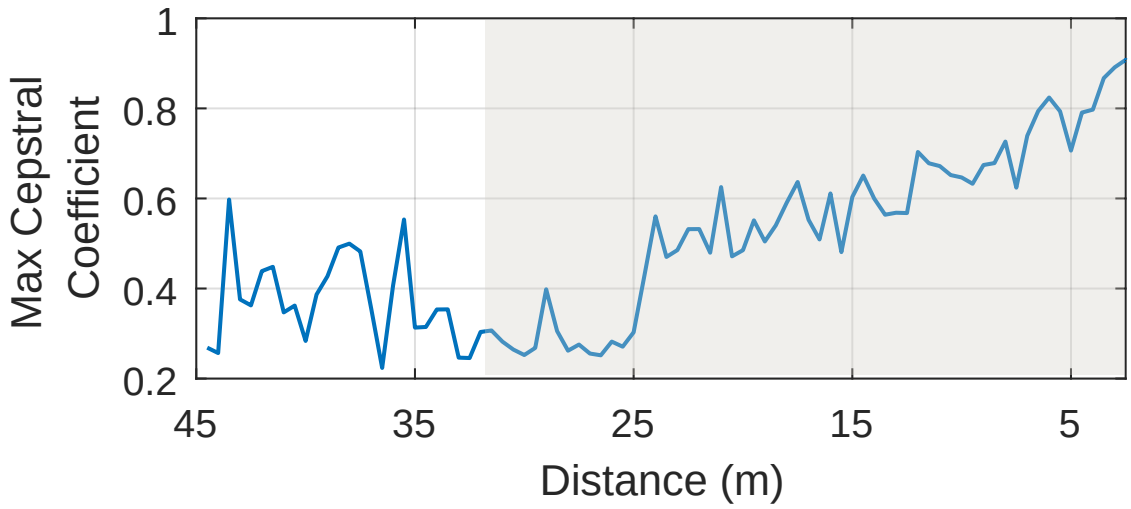

Figure 3.7: The maximum cepstral coefficient follows a trend when an approaching car is within about 30m from an observer.

Distance to a Car

To estimate the distance to the identified car, we formulate a regression problem that maps sound energy to distances. Later, we realized that due to environmental noise and the weakness of car sounds, a fine-grained location estimation is extremely inaccurate when the car is more than 30m from the audio recorder. When the car is within 30m, we find that the maximum values of the cepstral coefficients (computed every 100ms) are approximately linearly correlated with distance, as shown in Figure 3.7 for a car driven toward the mannequin. This relationship can be exploited to form a regression problem that maps maximum cepstral coefficients to distances.

For cars more than 30m away, although we are able to detect their presence and estimate their direction, a precise distance estimation results in a large error. However, the distance-estimation problem can be formulated as a multiclass classification task by dividing the absolute distances into a number of ranges such as (0, 30m], (30m,

60m], and (60m, 80m]. Each of these ranges can be characterized by signal-energy and zero-crossing rates, and can be classified accurately using a machine-learning classifier.

Therefore, PAWS uses a two-level approach for distance estimation. It first employs a classifier to determine a coarse-grained distance. If a car is detected within the nearest range, it applies regression to obtain a fine-grained distance estimate.

3.3 Overview of PAWS

PAWS is a wearable headset platform together with a smartphone application using four MEMS microphones and the smartphone microphone with a set of machine-learning classifiers to detect, identify, and localize approaching cars in real time and alert the user using audio/visual feedback on the smartphone.

The system consists of three main components: sensors and their drivers, front-end hardware, and a smartphone host (Figure 3.8). The four MEMS microphones, labeled MIC1 to MIC4, are distributed at the left and right ears, the back of the head, and the user's chest to provide relevant information about the sound source's location. The front-end hardware synchronously acquires analog signals from these microphones and locally extracts acoustic features used by the application running on the smartphone.

PAWS performs signal processing inside the front-end hardware to reduce the volume of data to be transmitted to the smartphone via a Bluetooth Low Energy

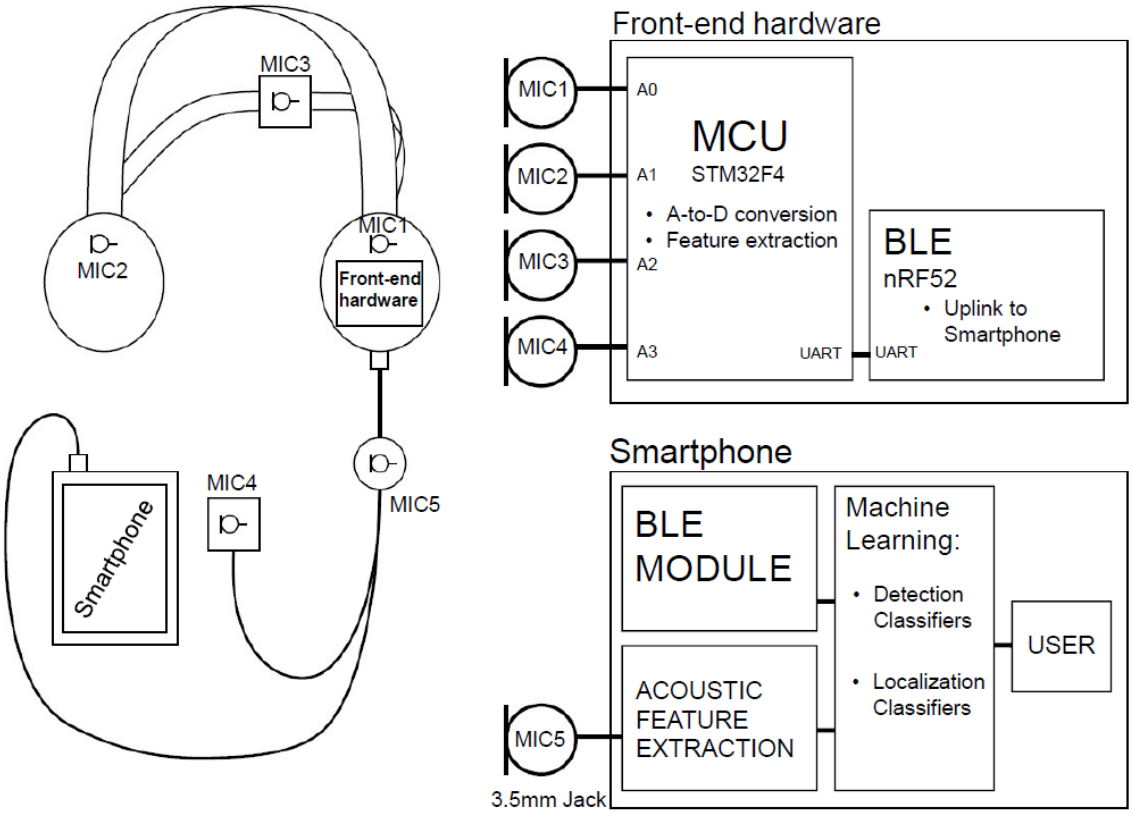

Figure 3.8: A block diagram of PAWS.

(BLE) connection. The headset's standard microphone (the fifth microphone, MIC5) is connected to the phone's 3.5mm audio input. Data from the fifth microphone is directly acquired by the smartphone. Using the features computed by the front-end hardware and an audio stream from the headset microphone as inputs, machine-learning classifiers running inside the PAWS application detect the presence of an approaching vehicle and estimate its position relative to the user. The decision to run the machine learning classifiers on a smartphone stems from insufficient processing power and memory for a single microcontroller to sample, extract features, and run classifiers for car detection and localization with reasonable latency.

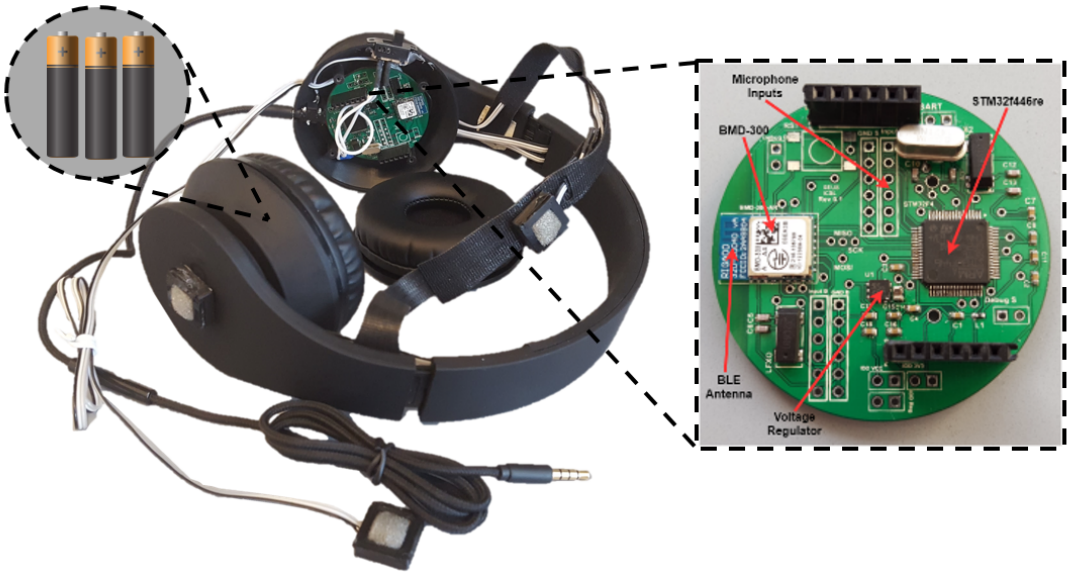

Figure 3.9: (Left) Teardown of the PAWS headset; the front-end hardware is exposed inside the left ear housing. (Right) Close up of the PAWS front-end hardware PCB.

Front-End Hardware

The front-end hardware is responsible for three blocks on the PAWS signal flow: synchronous ADC of microphone channels, embedded signal processing, and wireless communication with the smartphone. The integration of these blocks in a wearable resource-constrained system is challenging, and computational bottlenecks such as memory and data transfer require a careful distribution of resources.

To demonstrate PAWS's system architecture and algorithms, off-the-shelf components were used to build the system. As shown in Figure 3.8, four MEMS microphones are wired to a microcontroller unit (MCU). The MCU synchronously collects the signals, calculates the temporospatial features, and sends the result to a smart BLE module via a universal asynchronous receiver-transmitter (UART). The BLE module sets the link between the front-end hardware and the smartphone. The front-end

hardware is powered by standard AAA batteries and is designed to fit inside the left ear housing of a commercial headset, as shown on the left in Figure 3.9.

Front-End Signal Processing

This section outlines the operations processed by the front-end hardware. The MCU must sample the data from the four MEMS microphones and perform feature extraction, while the BLE module is responsible for transferring the calculated features to the smartphone. Since cars may be traveling at high speeds, fast response times and low latency are critical. PAWS uses a Cortex-M4 MCU to perform data acquisition and processing in real time. The design choices and evaluation are explained in detail in Section 3.4.

Sampling Data

Audio is captured from four microphones at 32kHz with an 8-bit successive-approximation ADC and a four-channel analog multiplexer running in the microcontroller. The sampling frequency was chosen as a compromise between the lowest rate necessary to capture the spectral content, as explained in Section 3.2, and the performance enhancement achieved by a delay estimation with finer granularity.

Feature Extraction

Running the feature-extraction algorithms in real time on a Cortex-M4 is challenging due to the complexity and number of computations required across the four channels.

To service a continuous stream of incoming data, it is imperative that one feature extraction finishes before the next window of data is completely received. The feature extraction calculations were simplified to achieve low latency; complex multiplication and division were avoided. The following features were calculated on the acquired four-channel data: relative power of each channel with respect to MIC1, relative delay with respect to MIC1, and zero-crossing rate of each channel. These features are calculated for every time window of 100ms with 50% window overlap.

The relative power ($Rp_{N,1}$) is calculated by summing the difference of squares between samples from each microphone to the reference microphone, MIC1.

$$Rp_{N,1} = \sum_{i=1}^{W_L} (X_N^2[i] - X_1^2[i]), \tag{3.1}$$

where N is the channel number, W_L is the window length (in this case 3,200 samples), X_N is the channel signal, and X_1 is the reference MIC1 signal.

The relative delay is calculated using cross-correlation ($XCORR_{N,1}$). The lag between the channels is defined as the index where the cross-correlation is maximum.

$$XCORR_{N,1}[d] = \sum_{i=0}^{W_L} X_N[i-d].X_1[i] \tag{3.2}$$

This is the most computationally expensive calculation of the front-end system. Since the physical separations of microphones are limited, e.g. the average spacing between ears is $\sim$25cm, the range of valid relative delays is bounded, making it possible to compute and compare the $XCORR$ only for $d \in [-40, 40]$. These limits

on the cross-correlation interest interval make the time-domain calculation of the cross-correlation more efficient than frequency-domain approaches [75].

The zero-crossing rate (ZC_N) is the number of times a signal changes sign within a given time window.

$$ZC_N = \sum_{i=1}^{W_L} (|\operatorname{sign}(X_N[i]) - \operatorname{sign}(X_N[i-1])|) \tag{3.3}$$

Data Transfer

The BLE module gathers the resulting ten-element feature values and sends them to the smartphone following a custom protocol in 40-byte packets. The protocol consists of a validation header (3 bytes), followed by a set of hardware configuration flags (1 byte), payload size (1 byte), and the feature values: 1×3 bytes for relative delays, 8×3 bytes for relative powers, and 2×4 bytes for zero crossings of MIC2–4.

Smartphone Data Processing

The PAWS smartphone app receives a 44.1kHz, single channel audio stream from the headset via the standard microphone jack and the ten-element acoustic features over BLE, and processes them in real time in a service. The application includes a graphical user interface to start and stop the service, configure alerts, and display a timeline of approaching cars along with their distances and directions.

Figure 3.10 shows the data processing pipeline of the PAWS smartphone application. The application's two-stage pipeline detects and localizes cars.

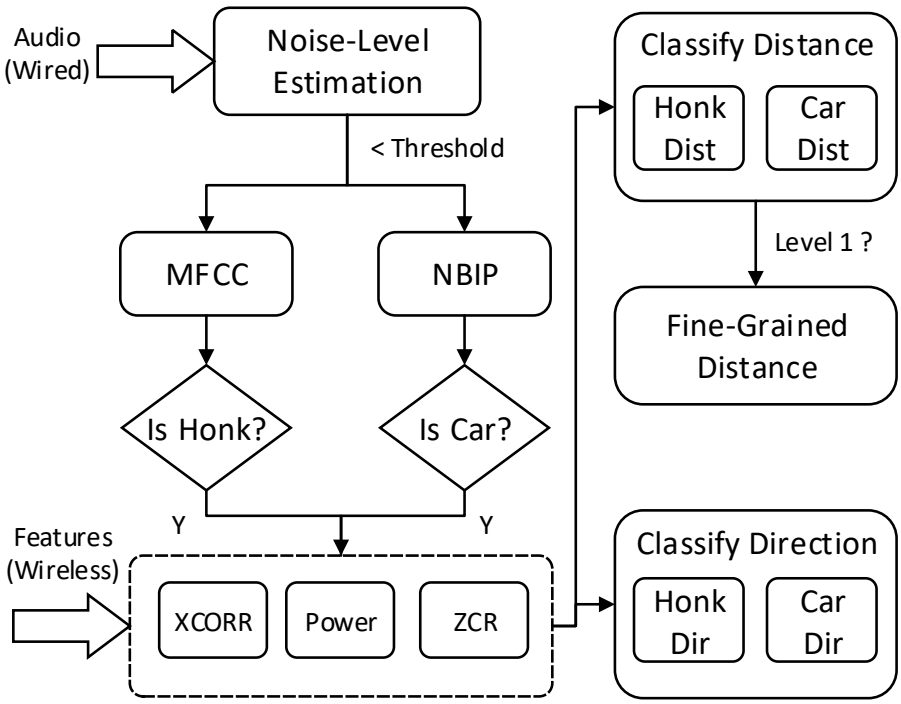

Figure 3.10: Smartphone data processing.

Car Detection Stage

In the first stage of the data processing pipeline, the single-channel audio stream is assessed to determine the severity of the surrounding noise level. If the level is below a safe threshold, the rest of the pipeline is executed to detect the presence of an approaching car. If the surrounding noise is extremely high, the user is alerted that the system is not in a suitable operating environment. We empirically determined that when the received signal strength is above 0.03dB, the SNR becomes very low, and our system's performance deteriorates from its ideal level. In such noisy environments (e.g., too many cars and honks), PAWS alerts the user of its ineffectiveness.

Two offline-trained classifiers are used in this stage to detect car honks and engine/tire sounds. The first classifier uses standard MFCC features to detect the pres-

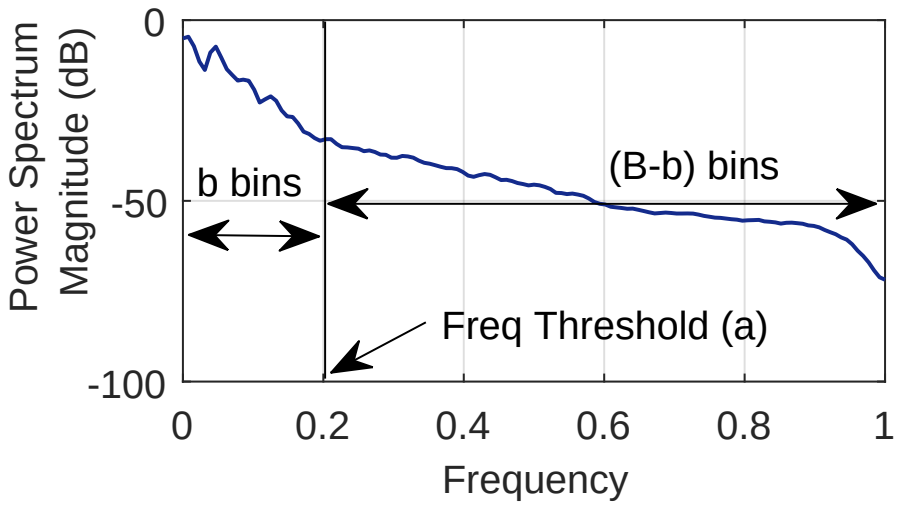

Figure 3.11: The basic idea of nonuniform binning of spectral energy.

ence of car honks. For the other type of car noises, we propose a new acoustic feature, the *nonuniform binned-integral periodogram* (NBIP), that divides the frequency scale nonuniformly to capture the variation at the lower end of the frequency spectrum characteristic of car noises. Five steps compute the NBIP features:

- **Step 1:** The FFT of each audio frame $x(t)$ is computed to obtain the Fourier spectra $X(f)$. Only the left half of this symmetric spectra is retained.
- **Step 2:** The periodogram of $x(t)$ is obtained from $X(f)$ by normalizing its magnitude squared and then taking its logarithm.

$$P_x(f) = 20 \log_{10} \left(\frac{1}{F_s N} |X(f)|^2 \right)$$

  F_s and N denote the sampling frequency and the signal length, respectively.
- **Step 3:** The frequency range is divided into a total of B bins. The frequencies below a threshold a are equally divided into b bins, and the higher frequen-

cies are equally divided into $B - b$ bins. The binning process is illustrated in Figure 3.11. The optimal values of the parameters B, a, and b are empirically determined, which we will describe shortly.

- **Step 4:** $P_x(f)$ is integrated in each bin to obtain a B dimension feature vector $v = (v_1, v_2, \ldots, v_B)$.

$$v_k = \begin{cases} \int_{(k-1)\Delta_1}^{k\Delta_1} P_x(f)df, & \text{if } 1 \leq k \leq b \\ \int_{a+(k-b-1)\Delta_2}^{a+(k-b)\Delta_2} P_x(f)df, & \text{otherwise,} \end{cases}$$

where, $\Delta_1 = \frac{a}{b}$ and $\Delta_2 = \frac{1-a}{B-b}$ are the bin sizes for frequencies below and above the threshold a, respectively.

To find the optimum values of parameters a and b, we vary the parameters $0 \leq a \leq 1$ and $1 \leq b \leq B$ in small increments and compute the vector difference between features of car noises and all other sounds. Figure 3.12 shows the search space for a and b for a fixed value of $B = 20$. We observe that $a = 0.3$ and $b = 18$ maximizes the vector difference between the car noise features and other sound features. A quantitative comparison of NBIP and MFCC is given in Section 3.5. We see that for noise-like sounds, NBIP provides much more accuracy in detection than traditional MFCC.

The features described above are used to detect approaching cars' engine and tire noises only. As honk is not a noise-like sound, we cannot use the proposed NBIP for its detection. We use MFCC in this case. For both types of classification (honks

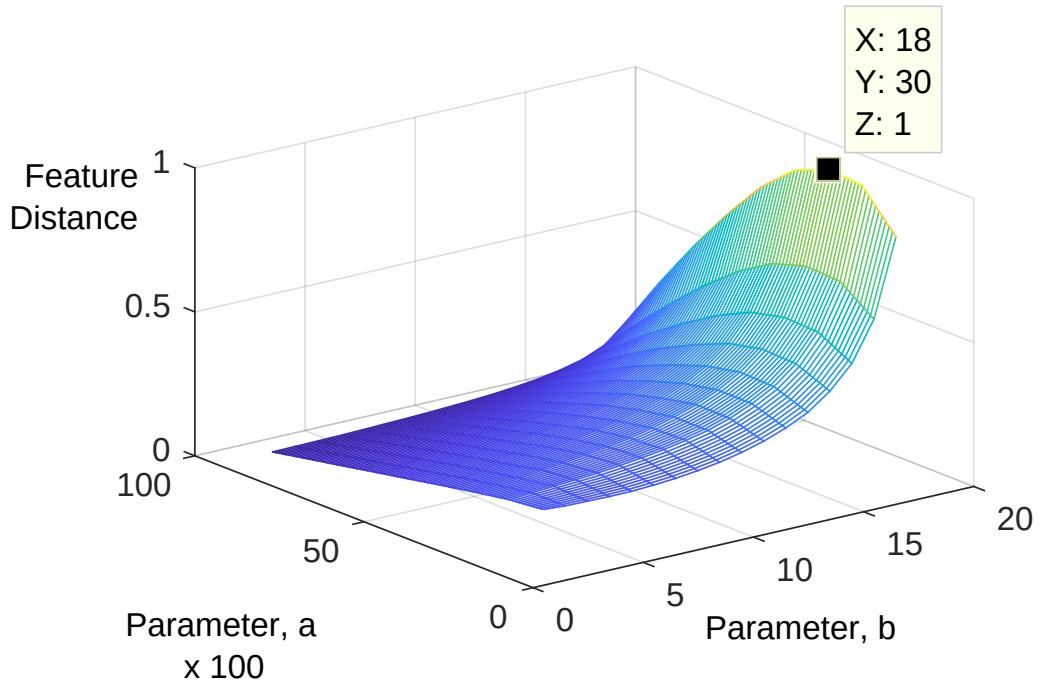

Figure 3.12: NBIP search space for parameter optimization.

and engine/tire noises), we train separate random-forest classifiers [14] to perform significantly better than other classifiers (e.g., support vector machine [19]).

Car-Localization Stage

If a car is detected, the second stage of the pipeline is executed. In this stage, the smartphone acquires and uses the four-channel acoustic features received from the embedded front-end system to estimate the distance and direction of the car. Four multiclass random-forest classifiers are used to classify eight directions and three distance levels based on honks and engine/tire sounds. Because the feature vectors are only of 10 dimensions, we feed all the features into both classifiers for a simpler implementation. However, a principal-components analysis (PCA) reveals that relative delay and relative power are more relevant for direction classification, whereas rela-

tive delay combined with ZC and relative power are relevant for distance estimation. Relative delay is relevant for direction because the microphone closer to the sound source will receive the audio signal sooner than the other microphones.

In addition to determining one of the three levels of distances, when a car is detected within the nearest level (within 30m), PAWS runs a linear-regression-based fine-grained distance estimator. This step includes computing the cepstral coefficients and then fitting the maximum value to an actual distance in meters. This step does not add any significant cost as we obtain the cepstral coefficients as a byproduct of MFCC computation during the car-detection stage.

Alert Mechanism

The application alerts a user with audio/visual feedback. If a car is detected within a user-configured distance range (e.g., 30m) – the phone vibrates, reduces the volume, and beeps. It can also be configured to play a customized message, e.g., "a car is {approaching, honking} on your {direction, left, right}". The application also visually shows the location and direction of the car on its user interface, as shown in Figure 3.1.

3.4 Platform Evaluation

Real-Time Performance

In this section, we discuss the system's real-time performance, the timing constraints involved, and the design choices to meet them. Response time is crucial for our system,

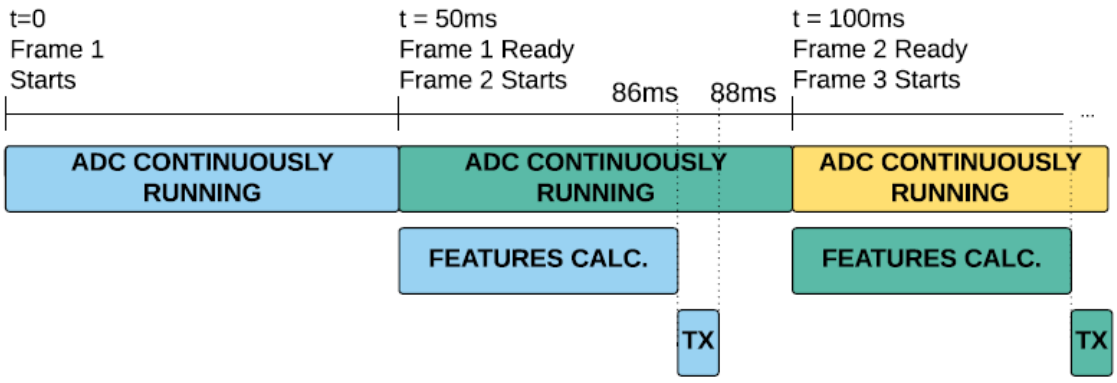

Figure 3.13: Pipeline of the MCU processing. The "Feature Calc." block represents all the operations involved in the feature extraction, and "TX" represents the UART communication between the MCU and BLE module.

as milliseconds can make a difference in saving the user's life. The embedded front-end hardware is handling 32kHz with eight bits per sample for each of the four MEMS microphones. To minimize latency, we compute features in 100ms windows every 50ms in a pipeline fashion. This means that features are being calculated every 50ms with 50% window overlap. The MCU uses a dedicated ADC module with direct memory access (DMA) to leave more CPU cycles available for feature calculation. The ADC continuously samples the audio and stores samples in RAM while features from the previous frame are calculated. Data are transferred from the MCU to the BLE module via a dedicated UART module. For this pipeline to work in real time, all features from the current frame must be calculated before the following frame is acquired, and the UART module must finish sending the current feature vector before the next feature is ready. Figure 3.13 presents the timing of the different parts of this pipeline. Feature calculation consumes 36ms of the available 50ms in each time slot, and the UART module transmits each feature vector in 1.9ms.

Another crucial timing aspect of the system is the transmission latency from the

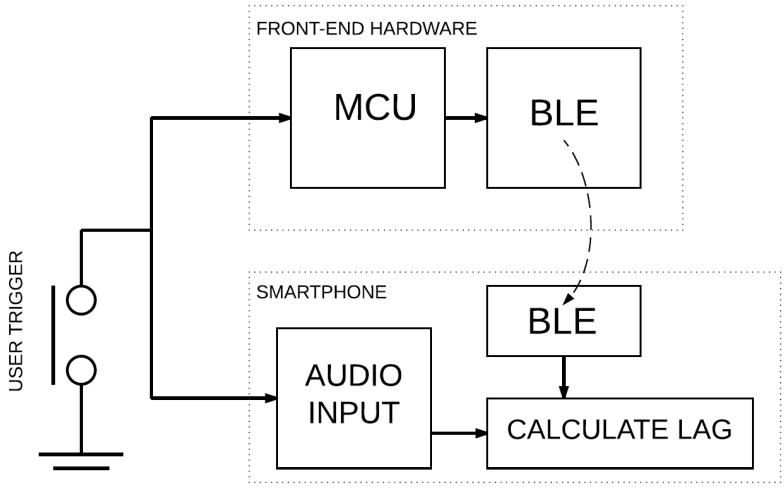

Figure 3.14: Block diagram of the test setup for the latency between the features from the front-end hardware and the smartphone.

BLE module to the smartphone. This latency will not only lengthen the system's response time, but it can also cause a mismatch between vehicle detection (based on the audio signal from the wired microphone) and its localization. If the temporospatial features calculated in the front-end hardware take too long to reach the smartphone, the location estimation displayed to the user might refer to a different sound source than the vehicle that the system just detected.

To verify that the smartphone receives the data within an acceptable time interval, an adaptation to the system was made, as shown in Figure 3.14. A button was simultaneously connected to one of the inputs of the front-end hardware and the smartphone's microphone input (as the regular microphone button). A verification app was developed to compare the difference between the time when the button-press event was detected by the smartphone application and when the smartphone received the data packet containing the same event. All aspects of the MCU and the BLE module firmware remain equivalent to the setup for standard operation. The

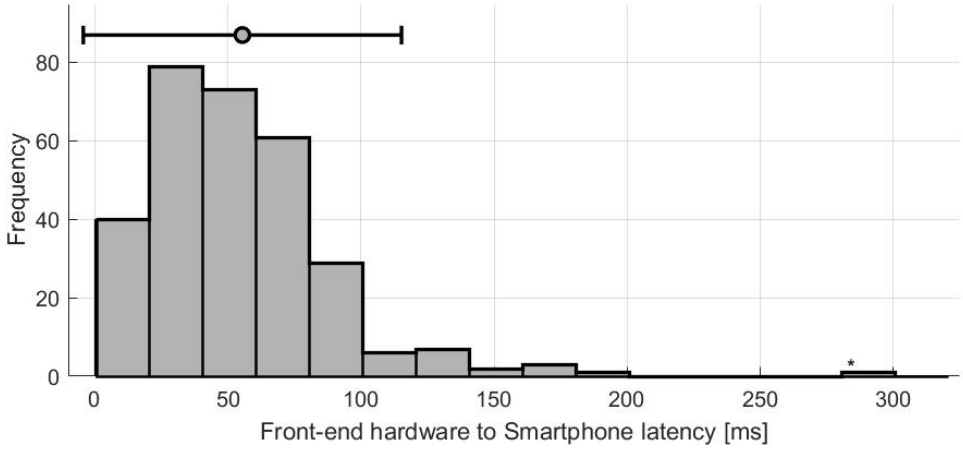

Figure 3.15: Histogram of the front-end-to-smartphone latency acquired with the Figure 3.14 test setup.

average delay is on the order of 55ms as shown in Figure 3.15. Since the event can be captured by the MCU anywhere within the 50ms sampling windows, this latency is not expected to be lower than the 38ms required for the calculation and transmission. However, due to randomness in the delay on the smartphone path, a few samples on the histogram have lower latencies.

The front-end hardware and the smartphone will be close together as both of them will be on the user's body. This small distance will ensure very little effect on the connection due to the presence of multiple Bluetooth devices in the environment unless there is major interference.

Figure 3.16 shows the execution times of various components inside the smartphone application. The application runs four threads in parallel. Thread 1 is responsible for getting audio data from the single-channel microphone. We take ten frames per window (448ms) for robust feature calculations. Thread 2 is responsible for receiving acoustic features over BLE. Thread 3 runs the car detector, which takes 86ms. The

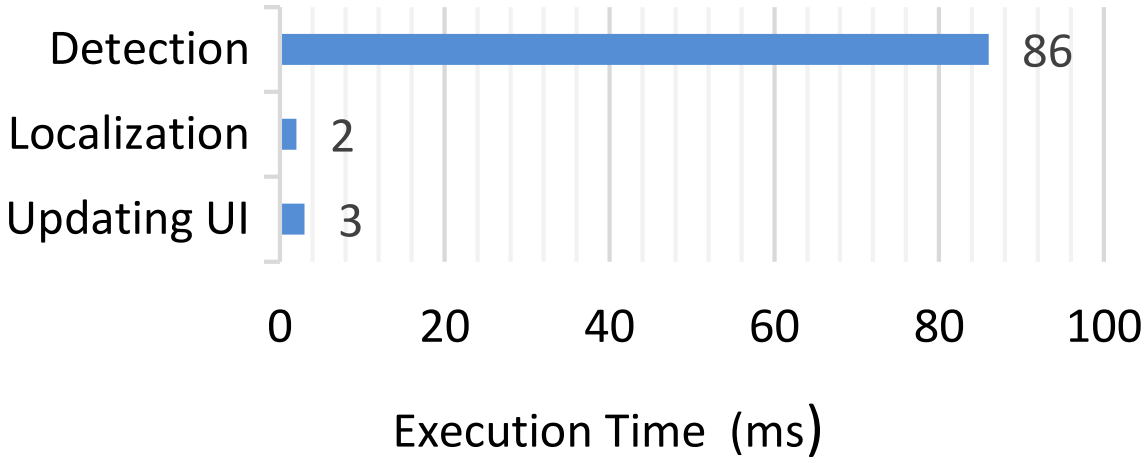

Figure 3.16: Execution times of various components of the PAWS smartphone app.

distance and direction estimators, which also runs in Thread 3, takes a mere 2ms since these classifiers use precomputed features. The UI thread (Thread 4) takes 3ms to update the UI and notify the user. The worst-case execution time for the PAWS app is 91ms. Because we use a 50% overlap between successive windows, the PAWS app runs the full classification pipeline every $448/2 = 224$ms, and detects and localizes cars in 91ms (i.e., in real time).

Power Consumption and Price Breakdown

We evaluate PAWS's energy consumption by measuring the power consumption for both the embedded platform and the smartphone during idle and active states. In the active state, data is processed, features are computed, and results are transmitted to provide danger feedback to the user. In the idle state, the smartphone application is not connected to the headset and most of the clocks in the embedded front-end platform are turned off to conserve power. The sole purpose of the idle state is to conserve power when the user is not using the system (e.g. when the headset is not

Table 3.1: Power Consumption and Price Breakdown

	Idle [mA]	**Active [mA]**	**Unit Price [US$]**
MCU (STM32f4)	4.37	50	3.20
BLE Transceiver (nRF52)	0.46	7	6.40
MEMS Mics	0.48×4	0.48×4	0.40×4
Amplifiers	2.34×4	2.34×4	1.60×4
3.3V regulator	0.1	0.1	0.50
Total	16.21	68.4	18.10

paired with the phone).

The embedded platform uses an STM32f4 Cortex-M4 chip to sample and extract features and a BMD-300 as the BLE transceiver. Operating at 180MHz, the STM32f4 consumes the most power, 50mA when active. While not in active use, the power can be reduced to 4.37mA. The Cortex-M4 architecture provides a familiar environment for firmware development with an acceptable energy footprint and a low cost of US$3.20 at major parts suppliers. The BMD-300 BLE transceiver module transmitting at 0dBm power consumes 7mA when active and 0.46mA when idle and transmitting only advertisement packets. The BMD-300 module integrates the Nordic nRF52 BLE chipset and antenna in a small-footprint component that fits this application for a low price of US$6.40. The other components of the front-end hardware are the 3.3V regulator, the MEMS microphones, and the preamplifiers. The overall power consumption of the system is below 70mA, allowing for 17 hours of continuous operation when powered by three standard AAA Alkaline batteries. As shown in Table 3.1, the total retail cost for the main electrical components is around US$18.00 per board.

For the smartphone, the most energy-consuming component is the display, which

Table 3.2: CPU and Memory Footprint.

	CPU (%)	**Memory** (KB)
STM32f4 (Active)	75.8	84.916
STM32f4 (Idle)	0	6.908
PAWS App (Active)	17.88	32,580

is only used to configure the app. Therefore, it is not necessary to keep it on at all times. The BLE communication consumes about 0.2mA. The energy consumption for the rest of the application is between 0.3µA h to 0.8µA h per frame.

CPU and Memory Footprint

We measure the CPU and memory footprints of both the front-end data acquisition system and the smartphone application. The portion of the embedded front-end that consumes the greatest resources (memory and CPU cycles) is the feature extraction process on the STM32f4.

Table 3.2 shows the average CPU and memory usage of the STM32f4 chip in PAWS. The CPU usage is almost 76% when the system is actively sampling and extracting features from audio sampled at 32kHz in 50ms time slots. However, when the system is idle, the CPU usage reaches 0%. This is because, when the system is idle, the STM32f4 CPU and all of its main clocks are shut down; the system is only able to wake up again when it receives an external event from the BMD-300 BLE module, which is generated by the smartphone application on demand. Because of this, PAWS saves energy and CPU/memory resources when not actively in use.

3.5 Empirical Data Analysis

This section describes the performance of the car-detection and localization algorithms in controlled settings. The full system is evaluated in an uncontrolled environment in Section 3.6.

Empirical Dataset

Detection Data collection consisted of recording single-channel audio samples using our embedded front-end platform. We used 47 different cars including sedans, SUVs, and convertibles from the street. We then listened to each audio stream and labeled the presence of cars. Later, we postprocessed the data to add Gaussian noise at 10dB (low noise), 30dB (medium noise) and 50dB (high noise) to the collected audio sample.

Localization We could collect the precise location of the cars on streets. So, we conducted a controlled data-collection experiment to record four-channel audio samples using the initial prototype of the embedded front-end platform. We used three standard automatic sedan cars in two different empty parking lots, where we marked different points on the ground and precisely measured the distances and angles as we drove each car towards the mannequin honking occasionally. The experiment area was about 120m $\times$ 100m. Table 3.3 lists the datasets, their purposes, and the number and types of audio clips. We also used a video camera to record the entire session. After the data collection, each sound clip was manually labeled to precisely mark the duration of honks and approaching car positions by listening to each clip

Table 3.3: Summary of the Empirical Dataset.

Dataset	Purpose	Clips	Characteristics
Honk	Honk Detection	720	1-ch, 1s
	Honk Localization	998	4-ch, 1s
Moving Car	Approaching Car Detection and Localization	63	1-ch, 15s, ∼25MPH
City Noise	Noise Injection	397	60dB–66dB

while watching the video as needed. In addition to honks and moving-car sounds, we also recorded city noises from the streets of a busy metropolitan area, which are used to inject controlled noise into sound clips to be able to test our algorithms' robustness. The datasets are available online.[1]

Comparison of Features

We compare the proposed NBIP features with standard MFCC features in terms of their ability to distinguish the two classes of sounds (car tire/engine sounds vs. noncar noises). Figures 3.17 and 3.18 show the mean and standard deviation of each component for the two feature vectors (NBIP and MFCC) for the two classes of sounds. We observe that most of the NBIP feature components (the first ten components) are very dissimilar for the two classes, whereas the MFCC features for both classes are very similar. Unlike the MFCCs, NBIPs are designed to maximize their vector representations for car engine/tire vs. noncar sounds, which helps them accurately recognize cars.

[1] http://icsl.ee.columbia.edu/projects/seus/

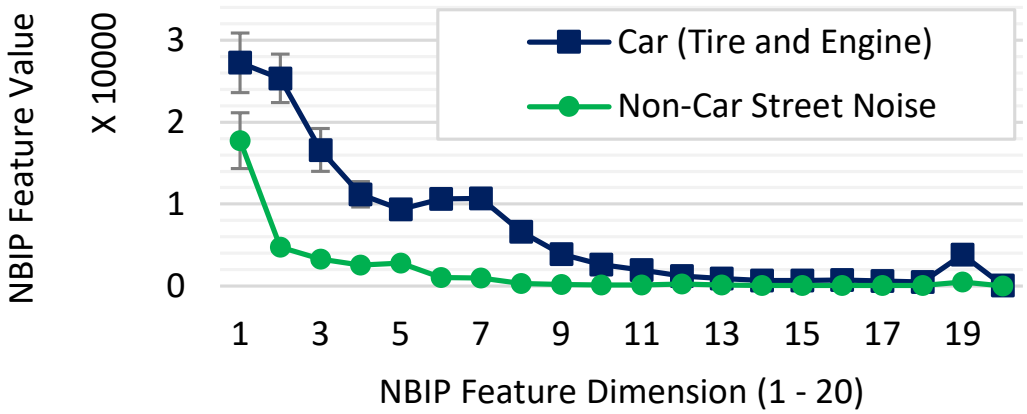

Figure 3.17: The proposed NBIP feature vector for car tire and engine sounds are designed to maximize their dissimilarity from noncar street noises like human chatter, human motion sounds, machine sounds, and loud music. Error bars on the data points indicate the standard deviations.

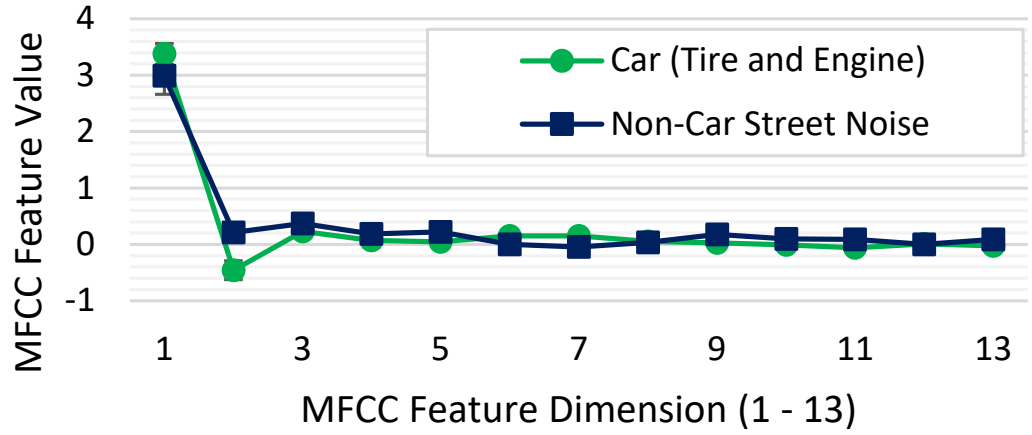

Figure 3.18: Standard MFCC features are less effective at separating the two classes than the NBIP features. Error bars on the data points indicate the standard deviations.

Accuracy of Car-Presence Detection

We measure the precision and recall of our honk detector as well as the approaching-car detector for various levels of injected noise across all distances and angles between the car and the mannequin. For these experiments, we use 80% splitting of dataset and tenfold cross validation. Figure 3.19 shows that the precision and recall are 99%

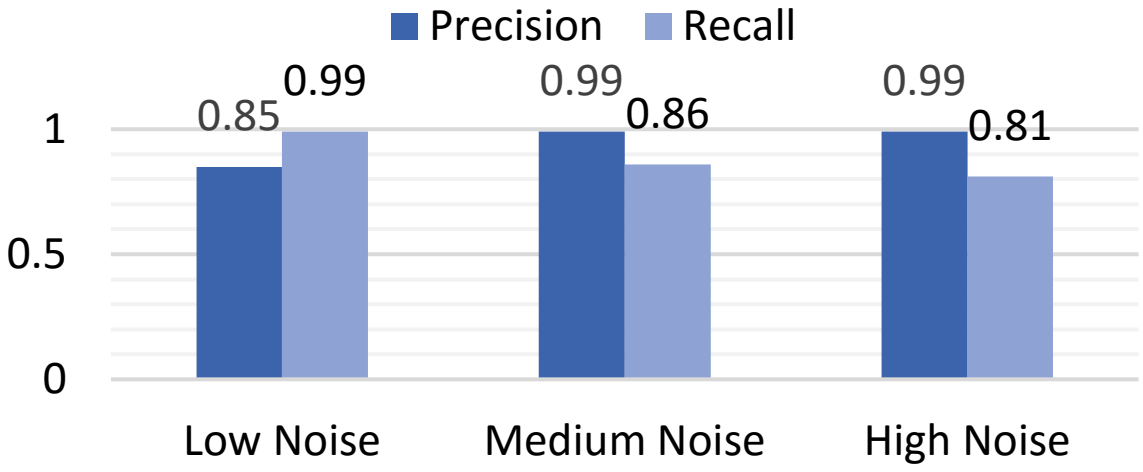

Figure 3.19: Honk-detection accuracy.

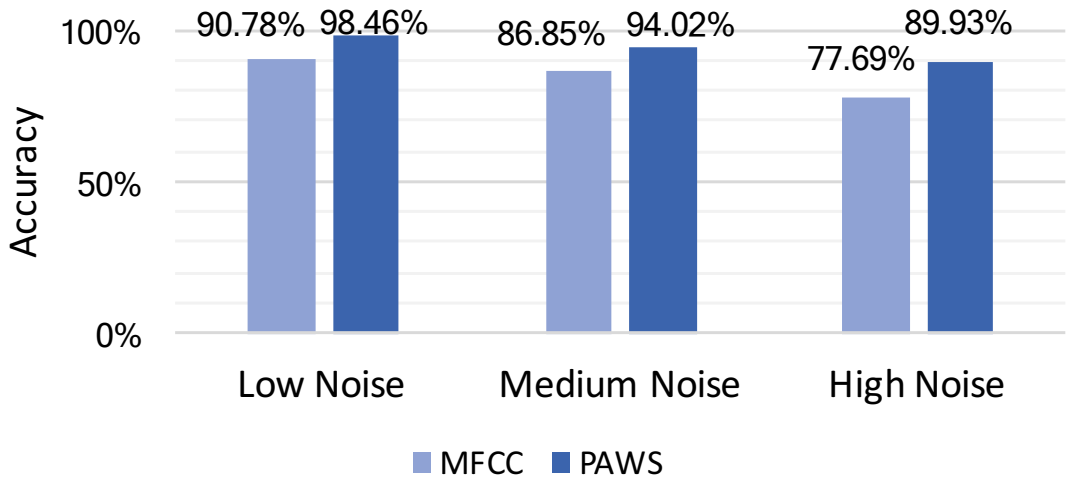

Figure 3.20: Approaching-car-detection accuracy.

and 81% for honk detection, even under extreme noise. Figure 3.20 shows that for extreme noise, the accuracy of PAWS for approaching-car detection is 89%, whereas the accuracy of MFCC is 78%. The average precision and recall of PAWS for all noise levels are 96% and 92%. PAWS has a very low false negative rate in detecting approaching cars which is extremely important for such a safety-critical system.

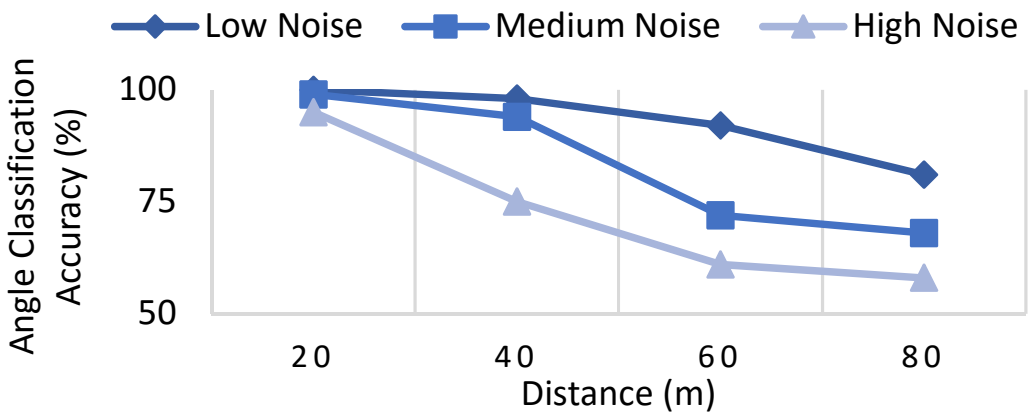

Figure 3.21: Honk-angle classification.

Direction-Estimation Accuracy

Figure 3.21 shows the accuracy of the angle detector when the distance and the level of noise are varied. Recall that this classifier detects one of the eight 45° 3D cones around a person. Its accuracy drops below 75% under extreme noise or when the distance is more than 40m. Under low-to-medium noise levels and <40m distance, the accuracy is 95%–100%. In urban areas with 35–45mph speed limits, this gives a person 2–3 seconds to react after the system detects a honk. For approaching cars, as seen from Figure 3.22, PAWS can determine whether a car is approaching from the left or the right with over 90% accuracy under extreme noise, with accuracy as high as 99.4% on less-noisy roads. Section 3.6 breaks down direction-estimation accuracy for all eight angles.

Distance-Estimation Accuracy

We measure PAWS's classification accuracy for inferring the distance of an approaching car and show the confusion matrices in Figure 3.23 for honk-based and approach-

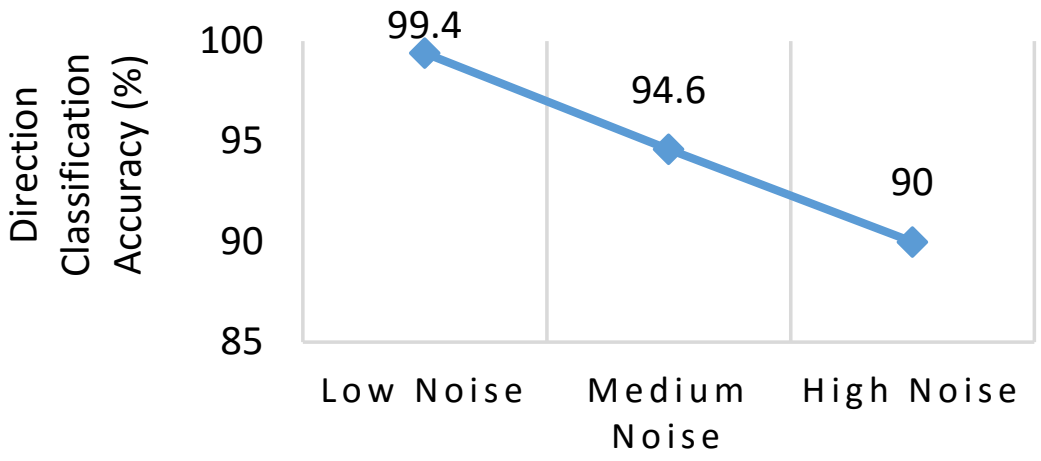

Figure 3.22: Direction classification for approaching cars.

Honks

	Predicted d1	Predicted d2	Predicted d3	
d1	**95**	4	0	94%
d2	5	**100**	28	77%
d3	1	26	**40**	59%
	96%	75%	60%	

Cars

	Predicted d1	Predicted d2	Predicted d3	
d1	**222**	20	6	90%
d2	36	**183**	29	74%
d3	7	34	**207**	83%
	84%	77%	86%	

Figure 3.23: Confusion matrices of distance classification for honks (left) and approaching cars (right).

ing cars' tire and engine sound-based distance classification, respectively. The three distance levels are $d_1 < 30$m, $30\text{m} < d_2 < 60$m, and $60\text{m} < d_3 < 80$m. When the car is either honking or is within 30m of the system, the accuracy of the two classifiers is 90%–94%. The confusion between levels d_2 and d_3 is higher, so the estimated distance at those distances may be erroneous in a strict sense. However, due to the high precision and recall of the car detector, PAWS will still be able to detect the car at those distances and warn the user in time.

Additionally, when a car is within 30m, PAWS's fine-grained linear-regression-based distance estimator is able to estimate distances with an average error of 2.8m.

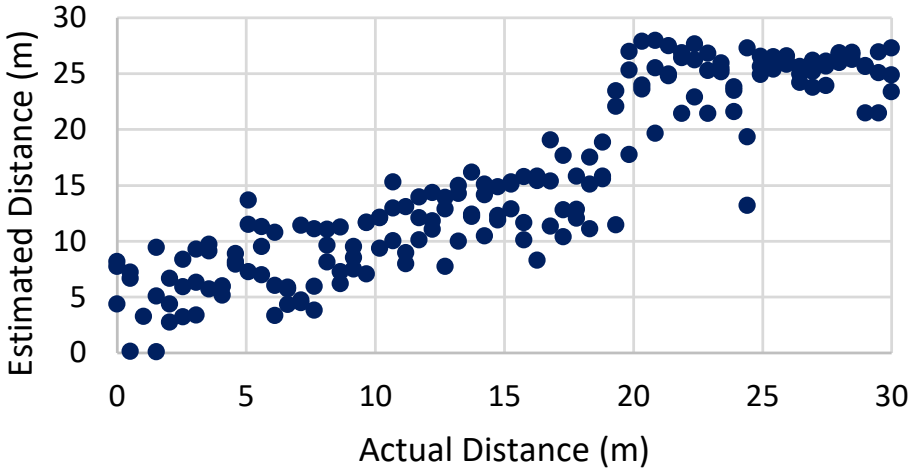

Figure 3.24: Estimated distances for cars within 30m are, on average, 2.8m off of actual distances.

Figure 3.24 shows a scatter plot of actual distances and estimated distances when cars are coming toward a person wearing the PAWS headset.

3.6 Real-World Deployment

Experimental Setup

To evaluate the end-to-end performance of the complete PAWS system in realistic settings, we conducted experiments in two different environments: a metropolitan area and a campus street. Unlike Section 3.5, we perform no postprocessing such as imposing noise or controlling the cars.

In the metropolitan area, three subjects participated in the experiments in multiple sessions acting in the roles of ground-truth collector, PAWS user, and distracted user, respectively. The ground-truth collector uses a special version of the PAWS application, "PAWS Clicker," to point to a location and direction of a car on the screen

as he sees it or hears its honks. The PAWS user carries a phone with the PAWS app running and also logs the honks and cars with another phone that runs the PAWS Clicker app. The third user mimicking an inattentive person is like the ground-truth collector except that he is also listening to music. During this experiment, PAWS is exposed to about 60 unique vehicles of various types—from trucks and buses to sedans and SUVs with different speeds and trajectories. Real urban noise, such as pedestrians, bikers, and wind were also present in the environment. After performing a sanity check on human-logged data, we realized that there were inconsistencies since participants have their own perceptions of car locations, and there were human errors in locating a car on the UI as well. As the distracted user has worse performance than the other two users, we consider the reports that are consistent in the ground-truth collector and attentive user's log and compare the results with PAWS.

The second experiment was performed on a campus street. It had fewer cars, but the numbers of pedestrians, bikers, and campus buses were higher. The weather was also windy due to the hurricane season. Because the ground-truth collection with the PAWS Clicker app proved to be problematic, this time, we used three fixed markers (yellow cones) on the sidewalk, and every time a vehicle passed a cone, a volunteer raised a flag and the event was logged in the original PAWS app. The setup is shown in Figure 3.25. The experiment was repeated multiple times. Each time, the user faced the road at a different angle, θ, to the accuracy of the direction estimation for as many different angles (3D cones) as possible.

Table 3.4 provides some statistics of the deployment in both environments. For

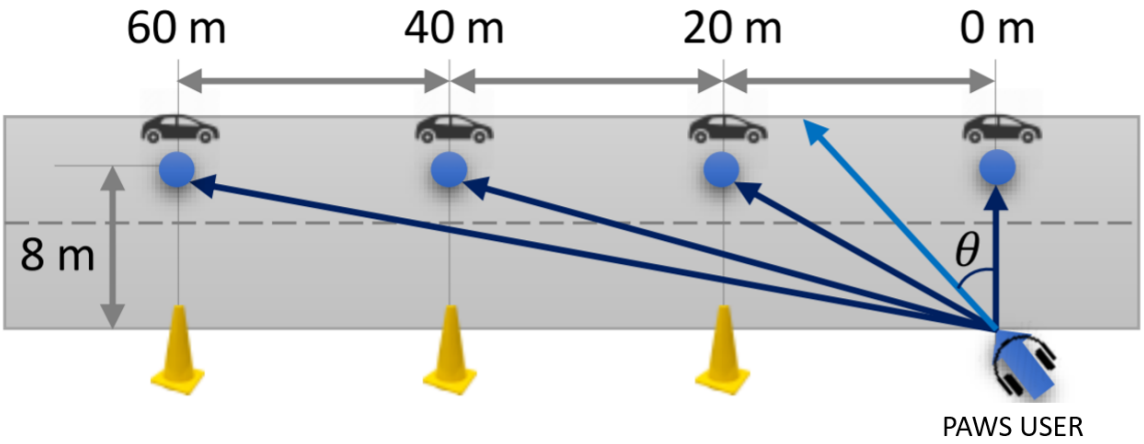

Figure 3.25: Experiment scenario in a campus street.

Table 3.4: Summary of Deployment Events.

Deployment	User (Facing Angle)	Honks	Car Events
City	0°, ±45°, 180°	48	165
Campus	0°, ±45°, 90°, ±135°, 180°	0	97

each one, the table shows how the PAWS user faced the road, and the number of logged honks and car events. Each honk is logged once, but each car is logged 3-5 times as it passed the participants. Since the second deployment was in a campus, we could not exercise the honk detection feature of PAWS in this environment.

Results

We measure PAWS's car-detection accuracy and compare its performance with the ground-truth collector's and distracted user's reports. Figure 3.26 compares the exact counts of total logged honks and approaching-car events for both environments. We see that PAWS logged the cars logged by the ground-truth collector, whereas the distracted participant missed about 26%–36% of them. This shows that PAWS is a

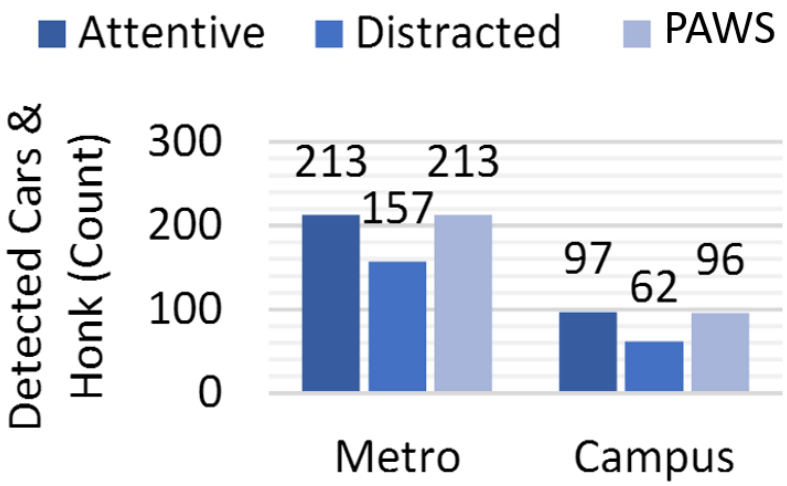

Figure 3.26: Car-detection performance.

highly efficient system for detecting and alerting pedestrians of approaching cars.

We also compute PAWS's distance and direction accuracy and show the results in Figure 3.27 for both environments. We assume that the distances and directions reported by the ground-truth collector are accurate. Each reported distance and direction is first mapped to the corresponding distance level and direction class and then compared with PAWS's classification results to compute the accuracy numbers. We observe that the overall accuracy of the distance classifier is 86%–87%, and that

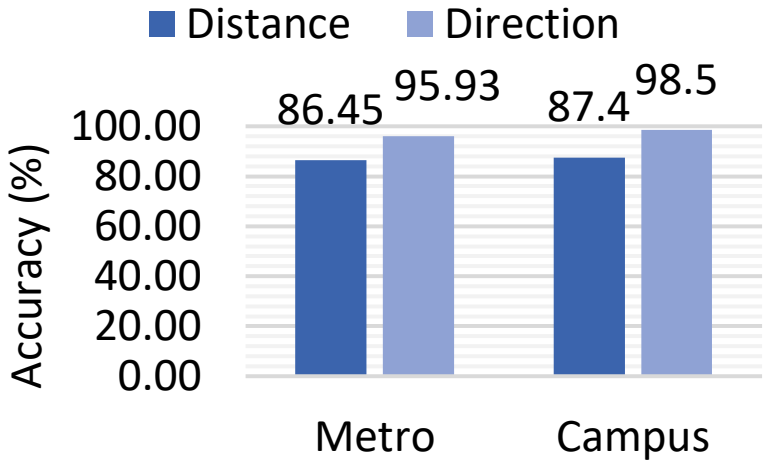

Figure 3.27: Car-localization accuracy.

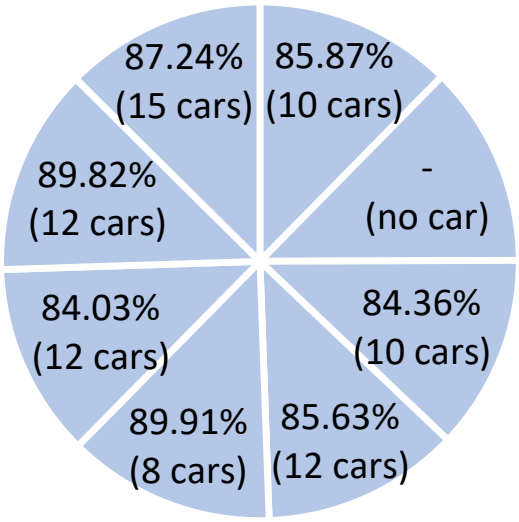

Figure 3.28: Accuracy for different directions.

of the two-level direction classifier (left vs. right) is 96%–98.5%. Upon investigating the cases where PAWS makes mistakes in distance estimation, we notice that the majority of these are where the car was at the furthest ends of the street. There are also some cases where a user logged the location to an area close to the boundary between two classes. Such human errors are also a reason for loss of accuracy. Figure 3.28 breaks down the direction estimation result for different cones in 360°, with an average accuracy of 86.7%. This is lower than the left–right detection because the participants naturally yaw their head about ±22.5° as they stand by the street. This effect could have been neutralized had we used an IMU to determine the user's staring angle with respect to his body. We leave this enhancement for future work.

3.7 Limitations and Future Work

We acknowledge that some scenarios can reduce the accuracy of the current system. Their effects on the predictions are discussed below.

Noisy Streets

PAWS is designed to detect the presence of cars in real-world environments. Streets may contain diverse kinds of noise, some of which are vastly different from those for which we have trained our system. PAWS should be trained in as many scenarios as possible to be able to handle these new types of sounds. For now, as a fallback mechanism, we turn off PAWS when the noise level is high, as described in Section 3.3.

Nearby Cars

The current PAWS design considers only the positions of vehicles relative to the user, not their trajectories. We can foresee occasions where a pedestrian is walking parallel to a busy road, and the system is giving warnings, even though the user is not in danger of being hit. A system to take into account the trajectory of both the vehicle and the user is under development.

Multiple Approaching Cars

The presence of multiple cars at the same time can impair the localization of vehicles. PAWS localizes the loudest source. However, the loudest source may not be the most relevant vehicle to the user. Sound-source separation and multiple sound source localization techniques are being investigated to improve the system.

3.8 Conclusion

This chapter presents PAWS, a wearable system that uses multiple audio sensors to protect pedestrians by identifying and localizing approaching vehicles. PAWS is carefully designed to recognize the honks and noises of an approaching vehicle. Using machine-learning algorithms, PAWS is able to identify honks and tire/engine sounds with greater than 80% precision and 90% recall. It further provides feedback on the azimuth of the sound source with up to 99% accuracy and predicts the distance from the user with up to 94% accuracy. As technology evolves and new distractions and dangers permeate modern cities, innovative safety systems must and will arise as solutions to balance the common citizen's welfare.

Chapter 4

A Sub-100nW Three-Channel Time-Delay-to-Digital Converter

4.1 Introduction

In a cyberphysical system (CPS), data converters are key blocks connecting the digital signal-processing or control blocks to the real world by encoding the sensors' responses

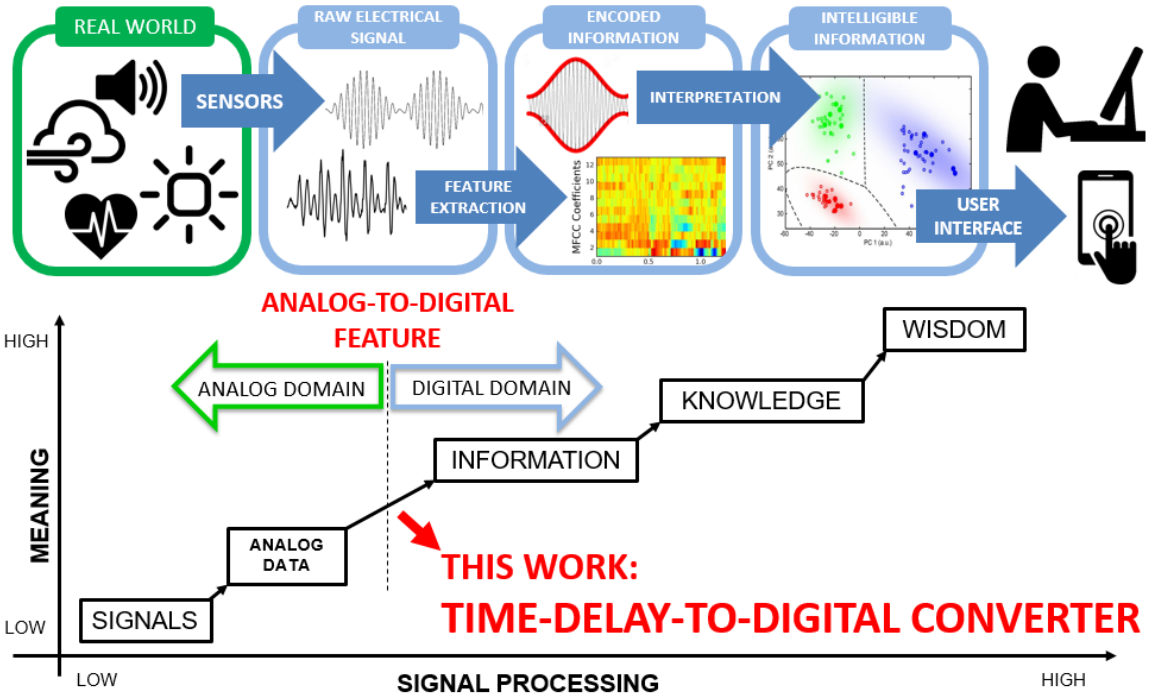

Figure 4.1: This work in the DIKW processing flow. The analog-to-feature converter, here exemplified as time-delay-to-digital converter, combines in a single block the analog–digital domain conversion and an increment in meaning of the resulting data.

into a format that can be easily manipulated by the digital blocks. In a CPS with a machine-learning classifier back end, the digital back end is only interested in the features present in the sensor signals. A traditional analog-to-digital converter (ADC), however, converts the complete, raw sensor signal into a digital signal which is then processed by digital feature-extraction blocks. In the *analog-to-feature* approach the sensor interface is specifically optimized to extract the features in the analog domain and only digitizes those features, as shown in Figure 4.1. In this chapter, we demonstrate the analog-to-feature approach in the context of a sound-source localization CPS detecting cars.

Vehicle awareness systems are of significant interest to the smart-city community [65, 30]. The ability to localize vehicles in an urban area can be the first step to reducing the number of traffic accidents involving pedestrians. Large-scale systems, integrated in traffic lights or in smart vehicles, use various techniques to detect cars: from LIDAR [35, 17] to stereo vision [11] to radio-frequency networks [8]. Such systems, however, must have access to large power sources, either the vehicle's battery or the power grid. Designing a wearable vehicle-aware system powered only by small batteries and with a reduced physical footprint [27] is still technically challenging. Audio-based integrated systems have demonstrated encouraging progress towards reaching ultra-low power consumption [37, 62], since the low frequency of the acoustic signal allows the integrated circuits to operate with a reduced clock frequency and supply voltage.

Time-delay estimations (TDE) are used to extract the arrival-time difference be-

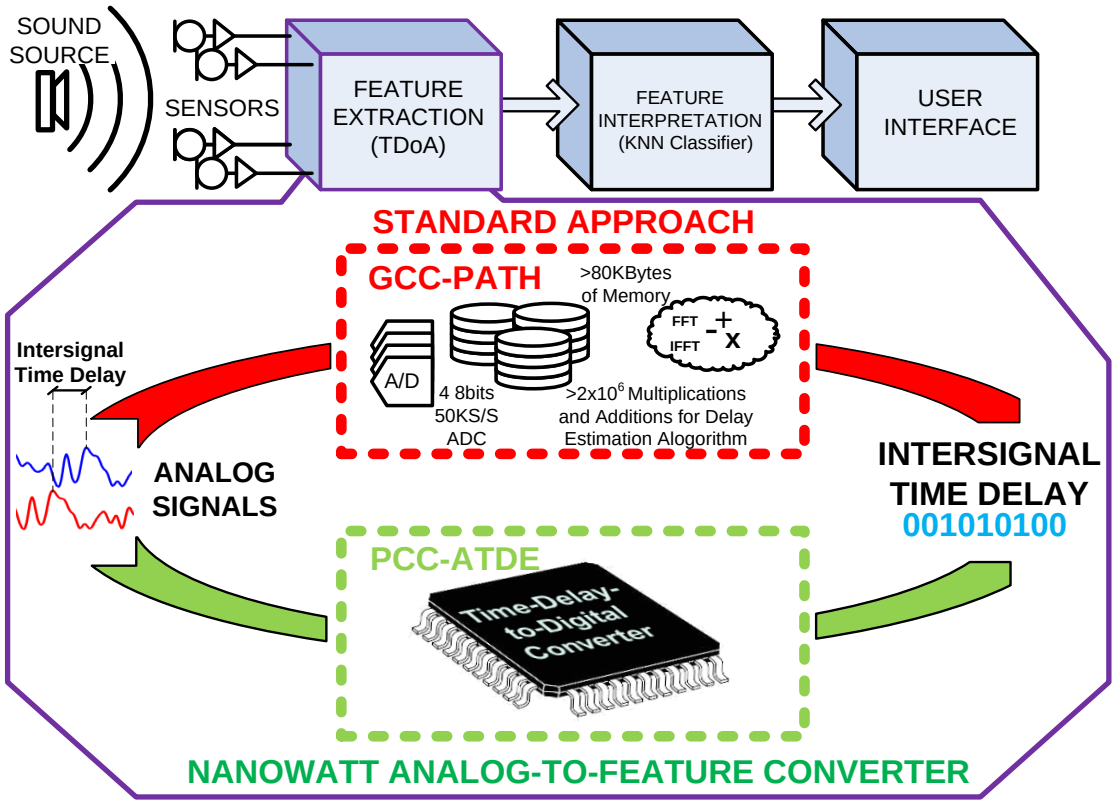

Figure 4.2: Example of how polarity-coincidence-correlation, adaptive, time-delay estimation (PCC-ATDE) can be used to simplify a sound-source-localization IoT system. PCC-ATDE is used to directly extract to digital the arrival-time difference between the analog signals of the microphones domain. It uses significant fewer resources than traditional approaches like generalized cross-correlation phase transform.

tween microphones in sound-source-localization systems [31, 29, 13, 22]. The standard approach uses a direct cross-correlation (DCC) function. For each TDE, time frames of the input signals are stored and all the points of the DCC function are calculated. The argument of the maximum DCC corresponds to the intersignal time delay. With a sampling frequency above 50kHz, the storage of the frames and the arithmetic operations to calculate the DCC values are a roadblock to achieving a submicrowatt implementation [29] as is needed in mobile, wearable, or IoT applications.

The low complexity of adaptive time-delay estimation (ATDE) techniques, such

as least-mean-square TDE (LMS-TDE) [69], makes them an attractive approach, but they still require a high-resolution ADC after the sensors. The bio-inspired silicon cochlea in [81] estimates the time difference by translating the audio stimulus into asynchronous events, but its power consumption is still in the microwatt range.

We present a 78.2nW 50kHz time-delay-to-digital converter with four audio input channels and three 8-bit delay outputs [26]. The presented architecture requires neither a multi-bit ADC, memory blocks to store frames or intermediate results, nor any computationally expensive algorithm.

Section 4.2 presents the negative-feedback tracking-loop architecture of the proposed TDE. Section 4.3 analyses the discrete-time implementation of the method. Section 4.4 introduces a delay-domain model of the loop, used in behavioral simulations to analyze and validate the proposed method. Section 4.5 describes the silicon implementation of the ultra-low-power TDE prototype, and its characterization is presented in Section 4.6. Section 4.7 compares our performance to previous work. The prototype is used to build a sound-source localization system, tested in Section 4.8 in a controlled indoor environment, and used in Section 4.8 to detect the bearing of approaching cars on the streets of New York city.

4.2 Feedback Time-Delay Estimation

When designing a TDE block, the considerations for selecting the sampling frequency are different from other feature-extraction blocks, where it is typically set by the signal

bandwidth. In a TDE, the sampling frequency of the data converters, F_S, defines the resolution. In this work, we support a -1ms to 1ms delay range with 8-bit resolution for noise-like sources with dominant spectral components below 250Hz (noise from approaching automobiles or other vehicles). For that case, the audio signal needs to be sampled at $>$ 50kHz, $\sim 100\times$ the Nyquist rate.

TDE with Direct Cross-Correlation

Consider the outputs of two microphones, $M_1(t)$ and $M_2(t)$ at different positions in space, which capture the signal of a single source $x(t)$:

$$M_1(t) = x(t) + n_1(t) \tag{4.1}$$

$$M_2(t) = (1 + \epsilon) \cdot x(t - D) + n_2(t) \tag{4.2}$$

where D is the time delay the algorithm must determine, $n_1(t)$ and $n_2(t)$ are random noise, and ϵ is the gain (or attenuation) difference in the microphones. The estimation of D requires computing the direct cross-correlation

$$DCC_{M_1,M2}(\tau) = \frac{1}{T}\int_0^T M_1(t).M_2(t-\tau)\mathrm{d}t \tag{4.3}$$

for many different τ and then determining the argument of the peak:

$$D = \mathrm{argmax}(DCC_{M_1,M2}(\tau)) \tag{4.4}$$

There are multiple ways to compute $DCC_{M_1,M2}(\tau)$: in the time domain, in the frequency domain, or by applying weighting functions to emphasize the peak as done in the generalized cross-correlation phase transform method (GCC-PHAT) [13].

TDE with Polarity-Coincidence Correlation

An alternative to reduce the complexity of the $DCC_{M_1,M2}(\tau)$ calculations is to use the polarity-coincidence correlation (PCC) function [24]:

$$PCC_{M_1,M_2}(\tau) = \frac{1}{T}\int_0^T \text{sign}(M_1(t)) \cdot \text{sign}(M_2(t-\tau))\text{d}t. \tag{4.5}$$

It has been proven [76] that

$$PCC_{M_1,M2}(\tau) = \frac{2}{\pi}\sin^{-1}\left(\frac{DCC_{M_1,M2}(\tau)}{\max(DCC_{M_1,M2}(\tau))}\right). \tag{4.6}$$

Hence, $\text{argmax}(PCC_{M_1,M2}) = \text{argmax}(DCC_{M_1,M2})$. Note that the computation of PCC requires only one-bit signals, in contrast to DCC which requires multibit signals. The one-bit quantization of the PCC also makes it less sensitive to the microphones' gain difference ϵ.

Regardless of how you obtain $DCC_{M_1,M2}(\tau)$ or $PCC_{M_1,M2}(\tau)$, their computation involves storing a large frame of both $M_1(t)$ and $M_2(t)$. Finding D requires calculating and storing the cross-correlation for the various τ within the TDE range and, finally, searching for the argument of the peak.

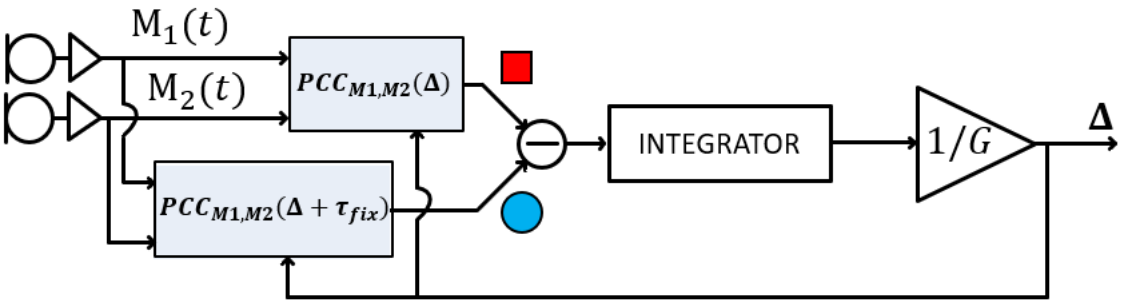

Figure 4.3: Simplified block diagram of the negative-feedback tracking loop used by the PCC-ATDE loop to estimate the intersignal delay Δ.

Polarity-Coincidence, Adaptive, Time-Delay Estimation

We present the polarity-coincidence-correlation, adaptive, time-delay estimation (PCC-ATDE) approach. The PCC-ATDE uses only two values of the PCC function to close a negative feedback loop that continuously tracks the intersignal delay D. Figure 4.3 illustrates the principle of the PCC-ATDE. Two points of the $PCC_{M_1,M_2}(\tau)$ are calculated, one with argument Δ, marked with a red square, and the other at $\Delta + \tau_{\text{fix}}$, marked with a blue circle.

To search for $D = \text{argmax}(PCC_{M_1,M_2})$, the loop takes the difference between the two PCC_{M_1,M_2} values. Figure 4.4 presents three possible cases. If the current Δ is sufficiently close to the argument of the peak, the difference between $PCC_{M1,M2}(\Delta)$ and $PCC_{M1,M2}(\Delta + \tau_{\text{fix}})$ indicates whether Δ is smaller or larger than the argument of the peak. The integrator will continuously increase or decrease the value of Δ until $PCC_{M1,M2}(\Delta)$ and $PCC_{M1,M2}(\Delta + \tau_{\text{fix}})$ have equal values, locking the loop at $\Delta = D - \tau_{\text{fix}}/2$, which gives a measurement of the desired intersignal delay, D. The attenuator $1/G$ sets the speed and bandwidth of the loop. Section 4.4 details its effect on the TDE. A practical problem for the architecture in Figure 4.3 is that to calculate

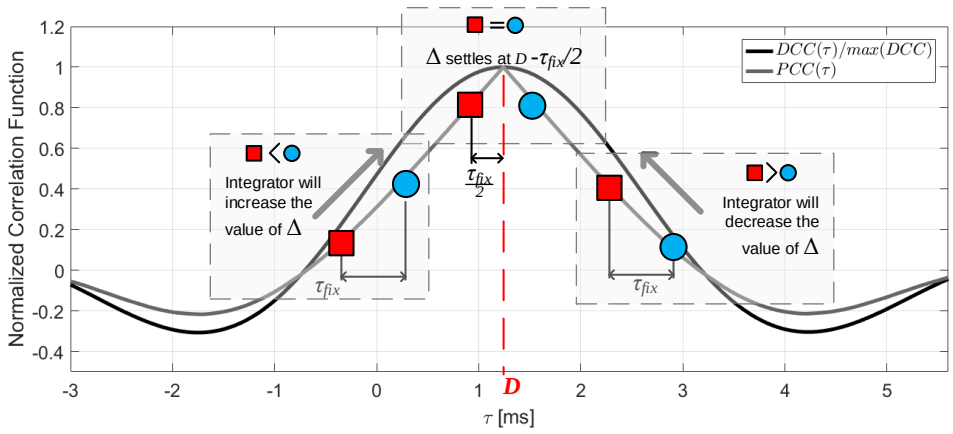

Figure 4.4: The polarity-coincidence correlation function, $PCC(\tau)$, and the normalized direct cross-correlation function, $DCC(\tau)/\max(DCC(\tau))$, of band-limited noise delayed by $D = 1.2$ms. Marked as red triangles and blue circles are three possible $PCC_{M_1,M_2}(\Delta)$–$PCC_{M_1,M_2}(\Delta + \tau_{\text{fix}})$ pairs in the PCC-ATDE loop in Figure 4.3.

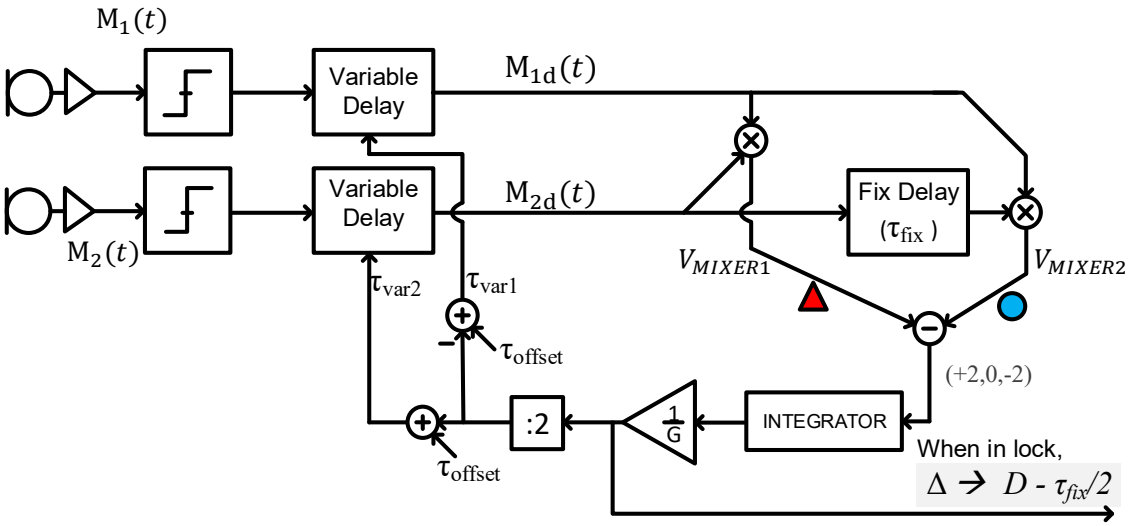

Figure 4.5: Block diagram of the proposed PCC-ATDE loop.

the PCC_{M_1,M_2} value pair, the system must store a large frame of each input signal.

The block diagram in Figure 4.5 shows how $PCC_{M_1,M_2}(\Delta)$ and $PCC_{M_1,M_2}(\Delta + \tau_{\text{fix}})$ can be extracted in the PCC-ATDE without storing signal frames. The analog microphone signals are connected directly to a comparator acting as a 1-bit ADC. Then each signal goes through a variable-delay cell, τ_{var1} and τ_{var2}:

$$M_{1d}(t) = \text{sign}(x(t - \tau_{\text{var1}}) + n_1(t - \tau_{\text{var1}})) \tag{4.7}$$

$$M_{2d}(t) = \text{sign}((1 + \epsilon) \cdot x(t - \tau_{\text{var2}} - D) + n_2(t - \tau_{\text{var2}}))$$

As shown in Figure 4.6, the variable delays, τ_{var1} and τ_{var2}, are defined such that $\Delta = \tau_{\text{var2}} - \tau_{\text{var1}}$. Since a variable-delay line can only introduce positive delay values, an offset τ_{offset} is added such that τ_{var1} and τ_{var2} are always positive. When the loop settles, Δ corresponds to the time-delay estimation between the inputs and can assume both positive and negative values depending on which input is ahead.

Next, $M_{1d}(t)$ and $M_{2d}(t)$ are multiplied to create $V_{\text{MIXER1}}(t)$. The average of $V_{\text{MIXER1}}(t)$ is the same as $PCC_{M_1,M_2}(\Delta)$:

$$\begin{aligned} \frac{1}{T}\int_{t-T}^{t} V_{\text{MIXER1}}(t)\text{d}t &= \frac{1}{T}\int_{t-T}^{t} \text{sign}(M_1(t)) \cdot \text{sign}(M_2(t - \Delta))\text{d}t \\ &:= PCC_{M_1,M2}(\Delta) \end{aligned} \tag{4.8}$$

$M_{2d}(t)$ is further delayed by a fixed value, τ_{fix}, and then multiplied by the upper

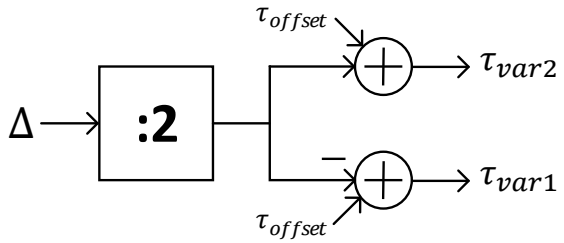

Figure 4.6: Block diagram of how τ_{var1} and τ_{var2} are obtained from Δ. τ_{offset} guarantees that neither assume negative values.

branch to create V_{MIXER2}. The average V_{MIXER2} is $PCC_{M1,M2}(\Delta + \tau_{\text{fix}})$.

Note that the averages of the multiplier's output, V_{MIXER1} and V_{MIXER2}, need not be explicitly calculated. The loop integrator providing a low-pass feedback loop performs the averaging and attenuates the higher frequency components. $1/G$ controls the loop bandwidth, guaranteeing that the average value is properly calculated.

4.3 Discrete-Time PCC-ATDE Loop

So far, we have presented a continuous-time PCC-ATDE loop. However, realizing programmable variable-delay lines for audio signals is very difficult; Implementing a discrete-time realization offers substantial design simplifications because it only requires conventional building blocks typically available in a signal-processing library.

Figure 4.7 shows a discrete-time realization of the PCC-ATDE loop outlined in Figure 4.5. The output $\Delta[n]$ at time step n is given by

$$\Delta[n] = \Delta[n-1] + \frac{1}{G} \cdot (\mathit{Mixer}_2[n] - \mathit{Mixer}_1[n]). \tag{4.9}$$

As in (4.8), since $\tau_{\text{var1}}[n] - \tau_{\text{var2}}[n] = \Delta[n]$, we can show that the average value of Mixer_1 is the same as $PCC_{M_1,M_2}(\Delta)$:

$$\begin{aligned} \frac{1}{T} \sum_{k=n-T}^{n} \mathit{Mixer}_1[n] &= \frac{1}{T} \sum_{k=n-T}^{n} \text{sign}(M_1[k - \tau_{\text{var1}}[k]]) \cdot \text{sign}(M_2[k - \tau_{\text{var2}}[k]]) \\ &:= PCC_{M_1,M_2}(\Delta[n]). \end{aligned} \tag{4.10}$$

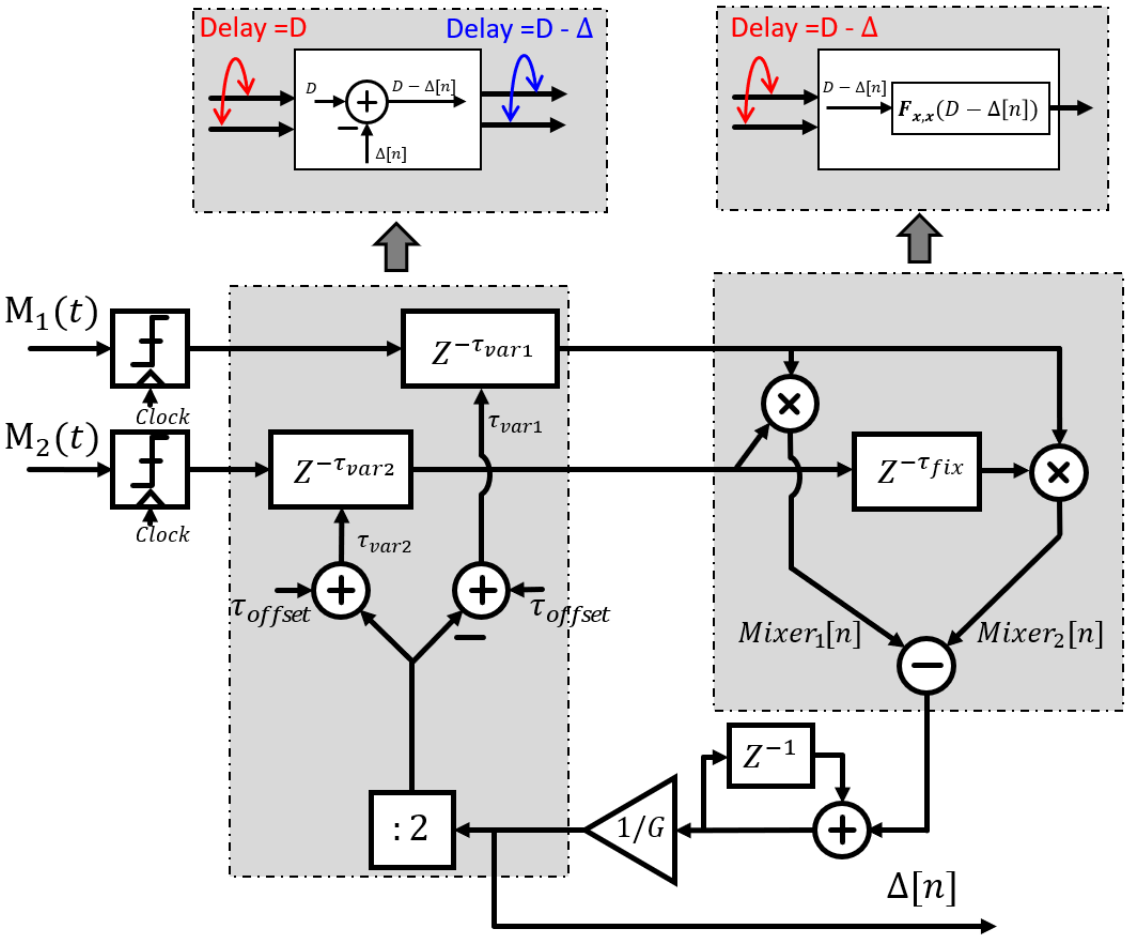

Figure 4.7: Block diagram of the discrete-time implementation using latched comparators, digital delay cells, XORs, adders, and dividers. The two shaded areas have different functionalities: The section on the left is used to add a delay of $\Delta[n]$ to the microphone signals. On the right is part of the loop responsible $F_{x,x}$, which is a function of the intersignal time-delay of its input.

Consequently, the average of $Mixer_2 = PCC_{M_1,M_2}(\Delta + \tau_{fix})$. Since the average values of both $Mixer_1$ and $Mixer_2$ are functions of Δ, and assuming PCC_{M_1,M_2} and τ_{fix} are static, the average of $Mixer_2[n] - Mixer_1[n]$ is $F_{M_1,M_2}(\Delta[n])$:

$$\frac{1}{T}\sum_{k=n-T}^{n}[Mixer_1[n] - Mixer_2[n]] = PCC_{M_1,M_2}(\Delta[n] + \tau_{fix}) - PCC_{M_1,M_2}(\Delta[n])$$
$$= F_{M_1,M_2}(\Delta[n]). \tag{4.11}$$

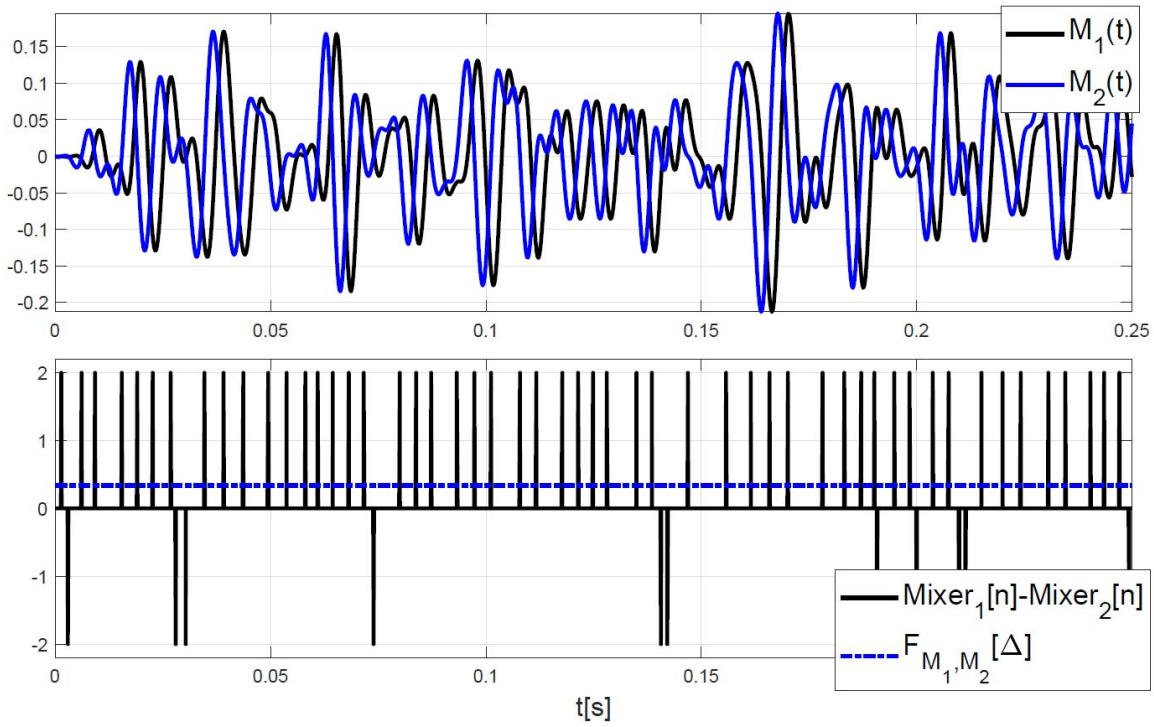

Figure 4.8: Simulated results of $M_1(t)$ and $M_2(t)$ and the resulting $Mixer_1[n] - Mixer_2[n]$, a fast-switching signal with much higher frequencies than the PCC-ATDE loop. Its average, $F_{M_1,M_2}(\Delta)$, is a function of the microphones' intersignal delay.

Figure 4.8 shows a simulated example of $Mixer_2[n] - Mixer_1[n]$ a fast-switching digital signal that can only assume values $\{-2, 0, 2\}$. Its AC components can be expressed as $e[n]$ and will be attenuated by the PCC-ATDE loop's much lower cut-off frequency. Rewriting $Mixer_2[n] - Mixer_1[n]$ as its DC and AC components, we have

$$Mixer_2[n] - Mixer_1[n] = F_{M_1,M_2}(\Delta[n]) + e[n]. \tag{4.12}$$

We can now substitute $F_{M_1,M_2}(\Delta[n])$ and $e[n]$ in (4.9):

$$\Delta[n] = \Delta[n-1] + \frac{1}{G} \cdot F_{M_1,M_2}(\Delta[n]) + \frac{1}{G} \cdot e[n]. \tag{4.13}$$

$e[n]$ does not introduce a DC error in $\Delta[n]$, but contributes as noise at the output. If we neglect $e[n]$ and focus on the loop's low-frequency output, we obtain a nonlinear feedback loop that continuously adjusts $\Delta[n]$ to keep $F_{M_1,M_2}(\Delta[n]) = 0$.

4.4 Discrete-Time PCC-ATDE Loop Analysis

We now analyze the PCC-ATDE loop's behavior. We focus specifically on the case where $M_1(t)$ and $M_2(t)$ come from the same source $x(t)$, but are differentially delayed due different signal paths to the microphones as expressed in (4.1) and (4.2).

Delay-Domain Model

Based on (4.13), we now propose a delay-domain model to predict the behavior of the PCC-ATDE loop. Similarly to phase-locked loops (PLL), the PCC-ATDE operates across multiple domains. Like phase-domain models used to analyze and design PLLs, a delay-domain model can assist in the PCC-ATDE design. In a PLL, the swap between the time domain and phase domain is accomplished by the phase detector (PD). It takes the reference and VCO signals as inputs and outputs a value corresponding to their phase difference. In the PCC-ATDE, the function $F_{M_1,M_2}(\Delta)$ is responsible for the domain swap. The value of $F_{M_1,M_2}(\Delta)$ is directly dependent on the difference between the estimated time delay, Δ, and the intersignal time delay from the microphones D.

Since the only correlation between $M_1(t)$ and $M_2(t)$ comes from the source $x(t)$,

the PCC function between M_1 and M_2 can be approximated by the auto PCC function of $x(t)$ shifted by the intersignal delay D:

$$PCC_{M_1,M_2}(\Delta) \approx PCC_{x,x}(D - \Delta). \tag{4.14}$$

We can use this approximation as the explicit contribution of the microphone delay D to the values of F_{M_1,M_2}, resulting in the function $F_{x,x}$:

$$\begin{aligned} F_{M_1,M_2}(\Delta) &\approx PCC_{x,x}(D - \Delta - \tau_{fix}) - PCC_{x,x}(D - \Delta) \\ &= F_{x,x}(D - \Delta) \end{aligned} \tag{4.15}$$

The approximation in (4.14) and $F_{x,x}$ are illustrated in Figure 4.9. Examples of PCC_{M_1,M_2} and $PCC_{x,x}$ are overlapped for a band-limited noise $x(t)$ and a sinusoidal $x(t)$. Next to each of them is the resulting $F_{x,x}(D - \Delta)$. Notice that $F_{x,x}$ only depends on the source signal $x(t)$ and the fixed delay τ_{fix}. Since it is defined by the difference of two $PCC_{x,x}$ values spaced by τ_{fix}, it can be understood as the derivative of $PCC_{x,x}$.

Introducing $F_{x,x}$ into (4.13), we now have a direct relation between the output of the loop, $\Delta[n]$, and the intersignal delay, D, that was previously implicit in F_{M_1,M_2}:

$$\Delta[n] = \Delta[n-1] + \frac{1}{G} \cdot F_{x,x}(D - \Delta[n]) + \frac{1}{G} \cdot e[n] \tag{4.16}$$

Using (4.16) provides the delay-domain model in Figure 4.10. This model is a power-

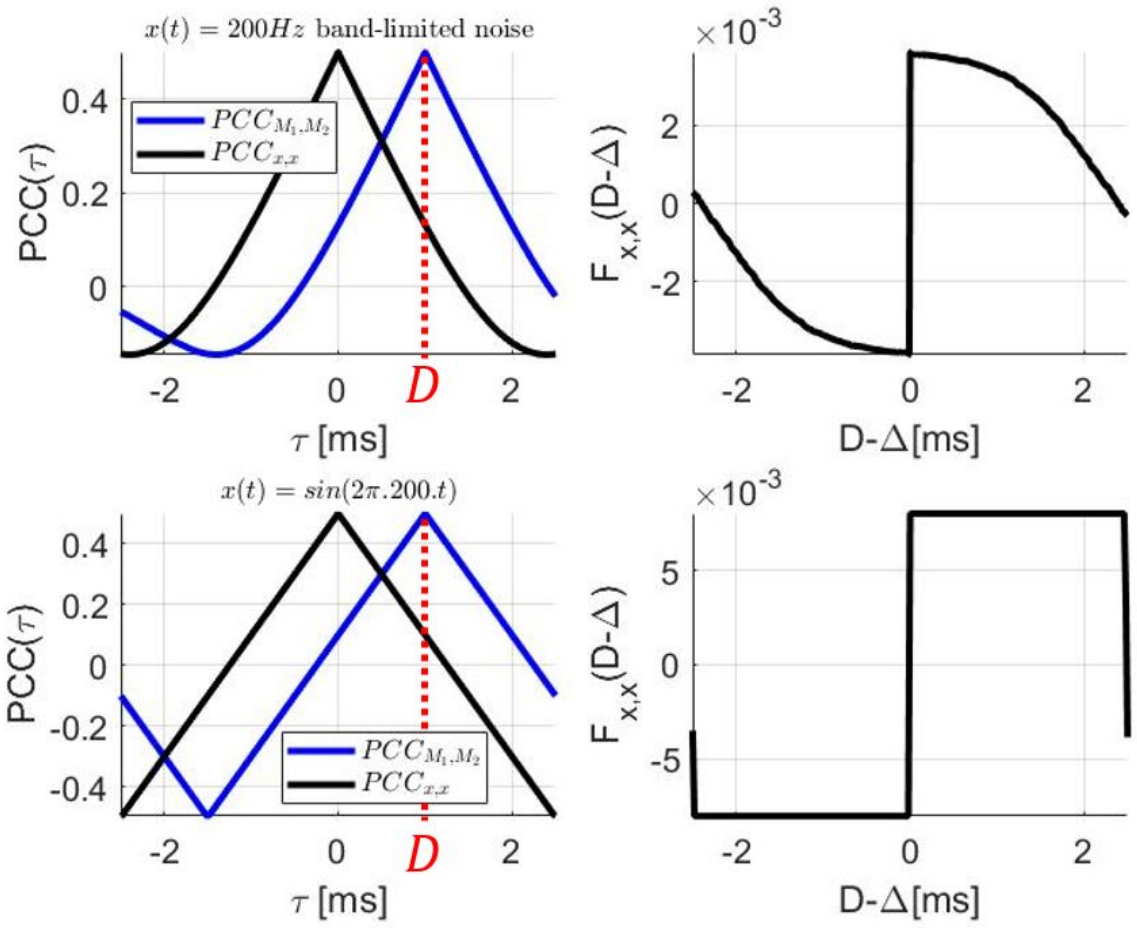

Figure 4.9: Simulated PCC_{M_1,M_2}, $PCC_{x,x}$, and $F_{x,x}$ for band-limited noise and sinusoidal source signal $x(t)$. PCC_{M_1,M_2} peaks at the intersignal delay D, while $PCC_{x,x}$ always peaks at 0, both with similar shape since the only correlated factor of M_1 and M_2 is $x(t)$. $F_{x,x}$ is the derivative of $PCC_{x,x}$.

ful tool for the extraction of the system's transient and steady-state responses, and also to determine the boundaries for the loop's correct functioning. The low number of elements in the model and the first-order negative-feedback architecture create a misleading impression that the PCC-ATDE will follow a conventional analysis. The nonlinear, $x(t)$-dependent function $F_{x,x}$ is a complex mathematical element that affects all the system parameters, from the range of converter to the settling time and bandwidth.

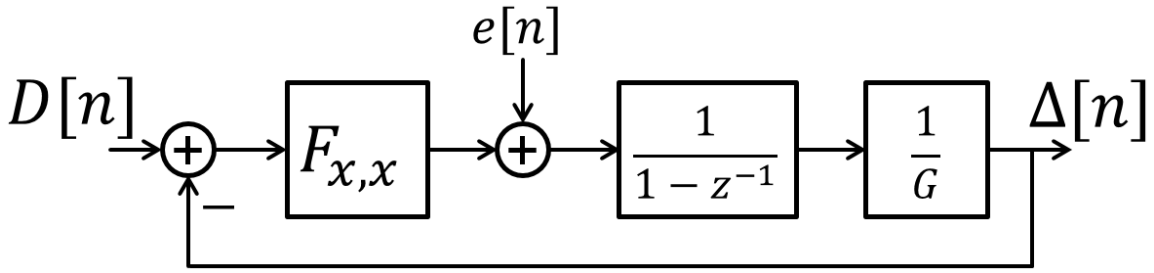

Figure 4.10: Delay-domain model of the PCC-ATDE. The input for this models is the intersignal time delay between the microphones $D[n]$. Since $F_{x,x}$ is the DC component of an operation, the remaining undesired high-frequency components are expressed as an error, $e[n]$.

Analog Input Bandwidth and Converter Range

To maintain the negative feedback and guarantee convergence to the correct time-delay estimate, $F_{x,x}[D-\Delta]$ has to have positive values for $\Delta < D$ and negative values for $\Delta > D$. Since $F_{x,x}(D-\Delta)$ is defined as the difference of two consecutive values of $PCC_{x,x}(\tau)$, see (4.15), the equivalent condition is that the derivative of $PCC_{x,x}(\tau)$ is positive for positive τ and negative for negative τ.

Figure 4.11 shows the $PCC_{x,x}(\tau)$ of band-limited noise signals. The local minima that limit the convergence condition for the PCC-ATDE, are indicated as τ_{MAX} away from the peak of $PCC_{x,x}(\tau)$ at $\tau = 0$. Higher bandwidth analog input signals have their local minima closer to the origin, so they have a smaller τ_{MAX}.

To define a range for the system, we must ensure that any possible time delay between the input signals D differs from any current output values $\Delta[n]$ by less than τ_{MAX}:

$$|\Delta[n] - D| < \tau_{\mathrm{MAX}}. \tag{4.17}$$

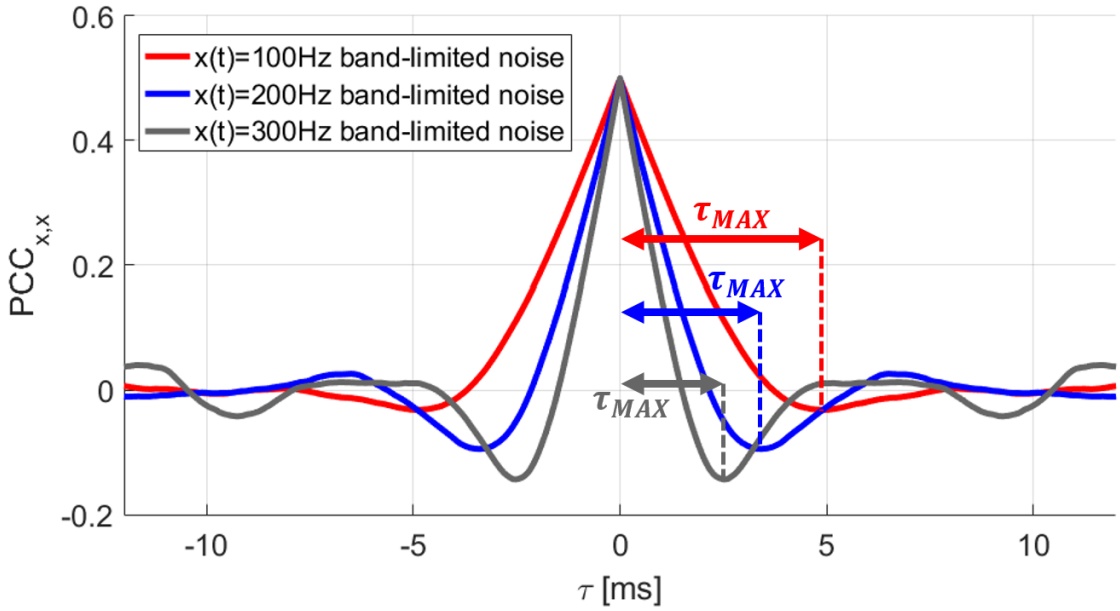

Figure 4.11: Example of the resulting $PCC_{x,x}(\tau)$ for inputs $x(t)$ with different bandwidth. τ_{MAX} indicates, for each bandwidth value, the maximum difference between $\Delta[n]$ and $D[n]$ that the PCC-ATDE can tolerate and still correctly track the delay.

This is an important design parameter for sound-source localization systems, where the maximum time delay between the input signals is limited by the spacing of the microphones. If the microphones are separate by approximately 35cm, the intersignal delay will be always $|D| < 1ms$. Applying a boundary to the output $|\Delta| < 1.5ms$ will guarantee that the loop stays within the covered range for $x(t)$ sources with bandwidth lower than 200Hz that has a $\tau_{\text{MAX}} > 2.5$ms. Low-pass filters can be used before the PCC-ATDE to limit the bandwidth of $x(t)$.

Response to a Step in the Intersignal Time Delay

As highlighted in the delay-domain model, the input to the PCC-ATDE feedback loop is the intersignal time delay D of the analog signals $M_1(t)$ and $M_2(t)$. Hence, to analyze the system's step response, we vary the delay between two identical 200Hz

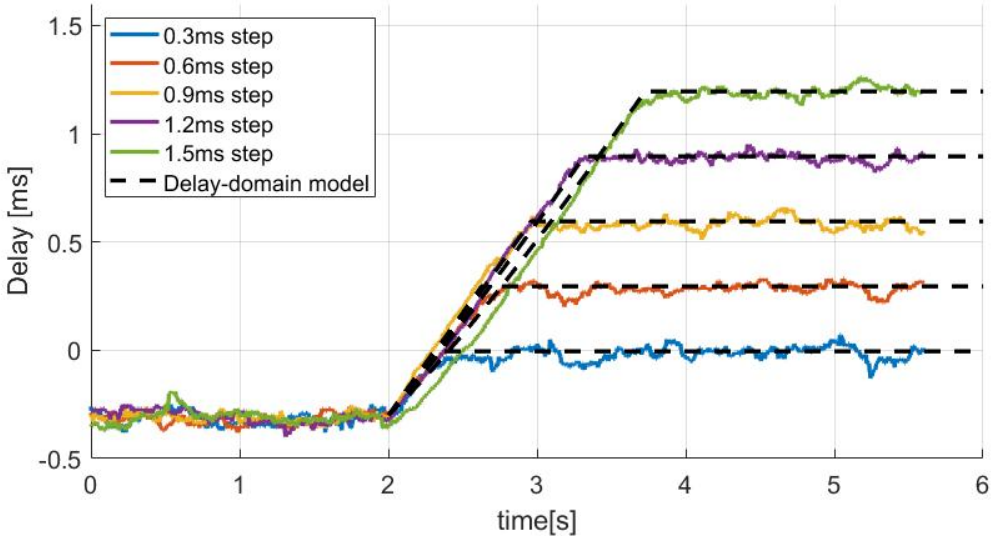

Figure 4.12: Step response of the time-delay-to-digital converter with different step amplitudes. Colored continuous lines represent experimental data and dashed lines are simulated using the delay-domain model. The close match of the results validate the delay-domain model's transient response.

band-limited signals applied to the microphone inputs. Figure 4.12 shows the step responses simulated with the proposed delay-domain model when steps of 0.3ms, 0.6ms, 0.9ms, 1.2ms, and 1.5ms are applied in the time delay two seconds after the beginning of the simulation. The experimental data is also shown and will be discussed in Section 4.6.

The settling times of the responses vary from 0.36s for a 0.3ms step to 1.72s for a 1.5ms step and depend on the step amplitude. Looking back at the delay-domain model, Figure 4.10, we see that this behavior comes from the slope limitation caused by the nonlinear element, $F_{x,x}$:

$$\Delta[n] - \Delta[n-1] = \frac{1}{G} \cdot F_{x,x}(D - \Delta). \tag{4.18}$$

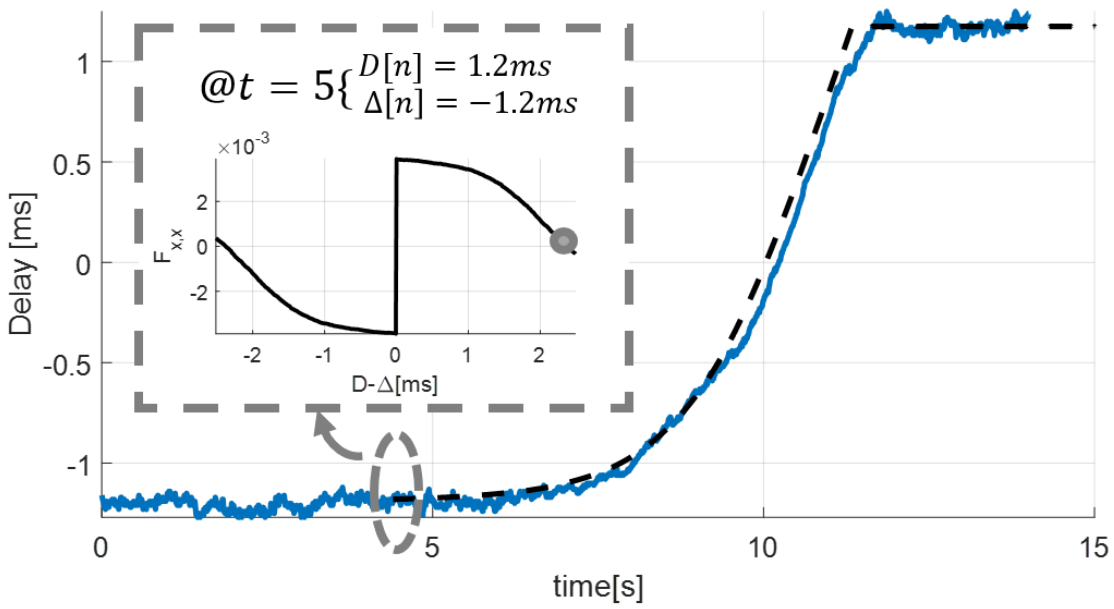

Figure 4.13: Isolated 2.4ms step. The colored continuous lines represent experimental data and the dashed line is simulated using the delay-domain model. The large step is used to highlight the effect of $F_{x,x}$, illustrated inside the dashed box, on the transient response. At $t = 5$s, $D - \Delta = 2.4$ms making $F_{x,x}$ very small. As shown by the gray dot in the plot, this reduces the output's slope.

For small amplitudes of $D - \Delta$, we can approximate $F_{x,x}$ as a step function, in the case of this simulation with limits $\pm 0.004 T_{\text{LSB}}$. T_{LSB} is defined by the PCC-ATDE sampling frequency as $1/F_s$. With that assumption, we can use (4.19). Using $G = 4$ and $|F_{x,x}| = 0.004 T_{\text{LSB}}$, we can calculated a settling time of $T_{\text{set}} = 0.3s$ for the input step $A_{\text{step}} = 0.3ms$, fairly close to the measured result.

$$T_{set} = \frac{|A_{step}| \cdot G}{|F_{x,x}(0)|} \tag{4.19}$$

As the amplitude of the input step increases, the shape of $F_{x,x}$ will affect the settling time of the system. To show that, a similar plot with an isolated $2.4ms$ step is presented in Figure 4.13. In a large step, the initial value of $F_{x,x}(D - \Delta)$ is small,

so the slope of the curve is also small. As Δ gets closer to D, $F_{x,x}(D-\Delta)$ approaches its maximum, increasing the slope of the TDE.

Both properties of the step response, namely the amplitude-dependent response time and its overall shape, are captured by the delay-domain model.

Response to a Sinusoidally Varying Intersignal Time Delay

Next, sinusoidal variations on the analog signals' intersignal time delay are applied to verify the PCC-ATDE's steady-state response. As with bang-bang PLLs [74] and slew-limited amplifiers [3], the slope limitation caused by the nonlinear element is expected to affect the PCC-ATDE's steady-state response. In the steady state, assuming the PCC-ATDE is able to track the input, the loop operates around $\Delta[n] - D[n] = 0$, allowing us to predict the a maximum increment the loop is able to track:

$$\max(|\Delta[n] - \Delta[n-1]|) = \frac{1}{G} \cdot |F_{x,x}(0)|. \tag{4.20}$$

The loop can handle a high-frequency signal with low amplitude, but will distort large low-frequency signals. If we assume a sinusoidal input for the intersignal delay, we can use (4.21) to calculate maximum amplitude–frequency product before we reach slope saturation:

$$\max(2\pi \cdot f_{\text{delay}} \cdot A_{\text{delay}}) = \frac{1}{G} \cdot |F_{x,x}(0)|. \tag{4.21}$$

Figure 4.14 shows the outputs from three 0.25Hz sinusoidal intersignal time-

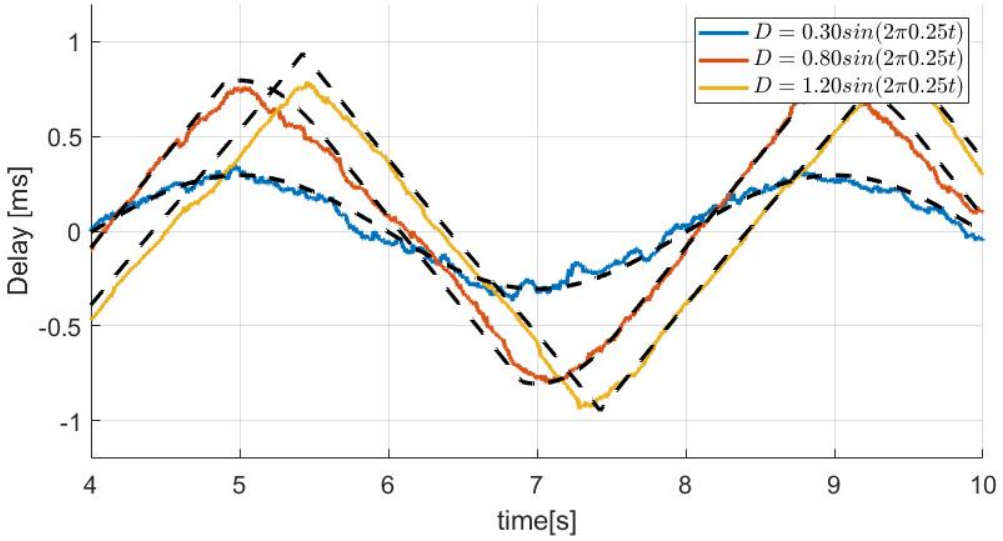

Figure 4.14: Steady-state response of the PCC-ATDE for sinusoidal delay inputs with three different amplitudes and the same frequency. Colored continuous lines represent experimental data and dashed lines are simulated using the delay-domain model. The response to high-amplitude signals are distorted due to the slope limitation.

delay inputs with different amplitudes. For this simulation $G = 4$, and $|F_{x,x}(0)| = 0.004T_{\mathrm{LSB}}$. The continuous lines are measurement data, and the dashed lines are the results simulated with the delay-domain model. Only the signal with $A_{\mathrm{delay}} = 0.3$ms is correctly tracked by the PCC-ATDE; the slope overload clearly distorts the other two responses where the amplitude–frequency product exceeds 0.001. The delay-domain model is also able to faithfully capture the loop's steady-state response.

Using the maximum allowable rail-to-rail amplitude (A_{MAX}) in (4.21) we can find the PCC-ATDE's power bandwidth (PBW):

$$PBW = \frac{1}{G} \cdot |F_{x,x}(0)| \cdot \frac{1}{2\pi \cdot A_{\mathrm{MAX}}}. \tag{4.22}$$

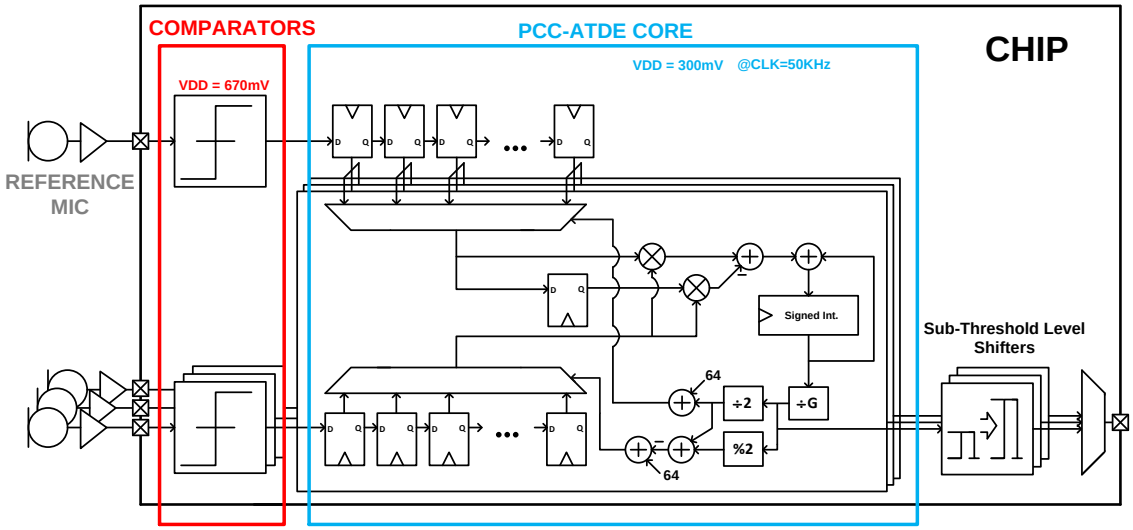

Figure 4.15: Block diagram of the CMOS prototype. The system has two main blocks: four latched comparators that receive the input from the microphones and the ultra-low-power core that outputs three digital time-delay values. Subthreshold level shifters interface the low-voltage signals with the digital I/O pads.

4.5 CMOS Prototype Implementation

Figure 4.15 presents the block diagram of the CMOS prototype of the discrete-time implementation. The chip has four analog inputs connected to a microphone array. One of the microphones provides the reference for the time-delay estimation; the chip outputs the time delay of the other three analog signals relative to this reference microphone.

Figure 4.16 shows the PCC-ATDE die photo in standard 0.18µm CMOS technology with a total area of $1mm^2$. Digital blocks were synthesized from subthreshold CMOS logic cells and the overall power consumption of the three-channel PCC-ATDE time-delay estimator is 78.2nW.

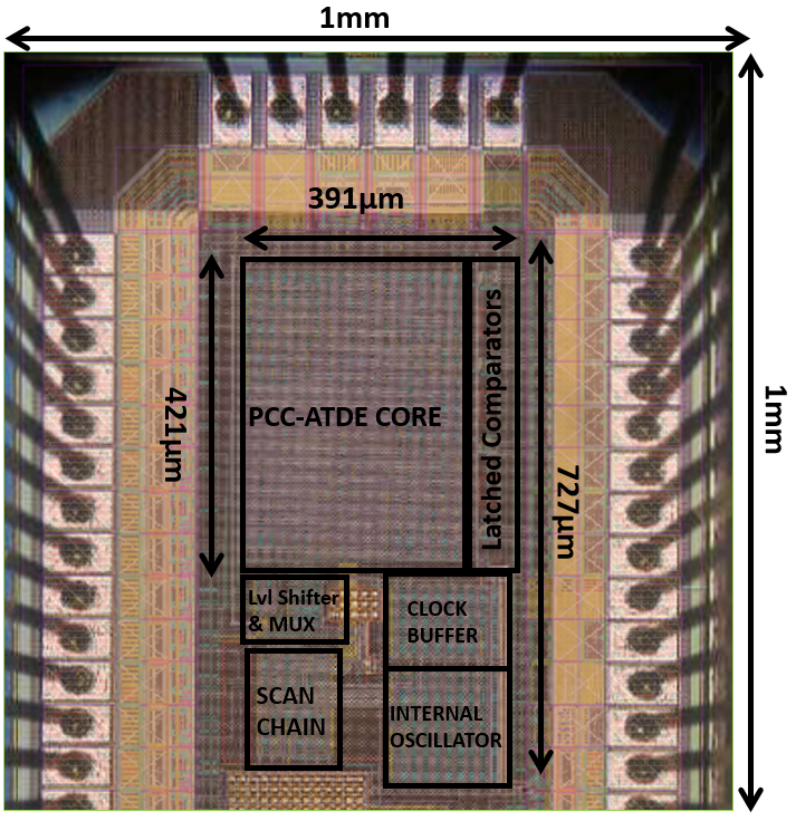

Figure 4.16: Microphotograph of the PCC-ATDE CMOS prototype.

Front-End Comparator

The latched comparator shown in Figure 4.17 is based on [20, 82]. It consumes no power in reset mode, leading to a total power consumption of 3.1nW per comparator at 670mV when operated at 50kHz. The front end is designed with thick-oxide transistors for better ESD robustness. It supports differential inputs, but, to simplify the integration with the off-the-shelf microphones and preamplifiers, it was used in a single-ended fashion in the final system.

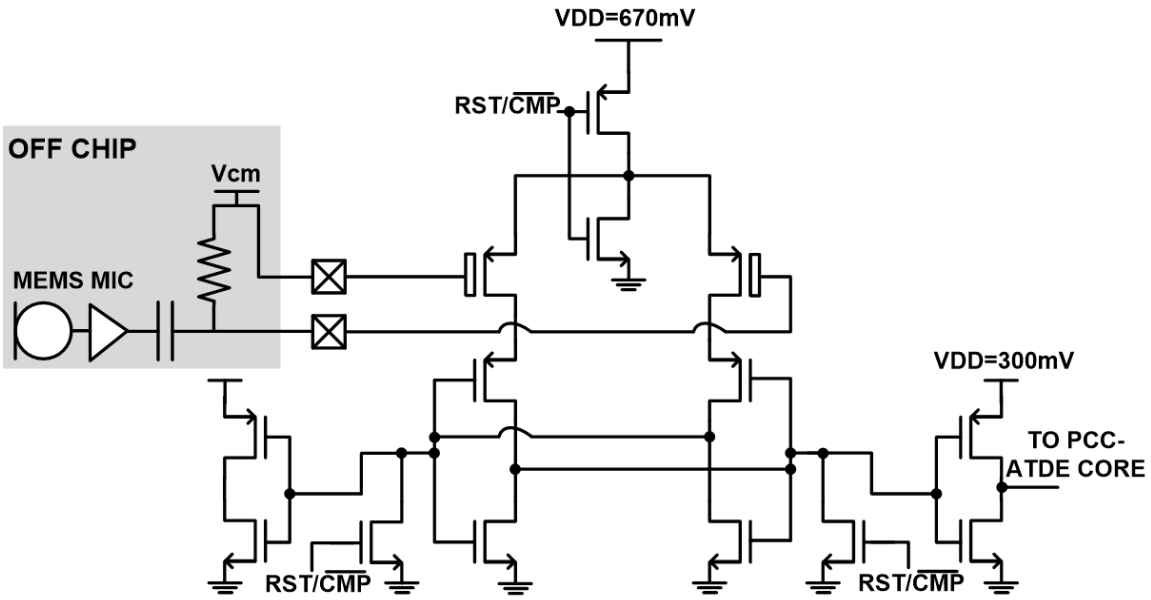

Figure 4.17: Schematic of the latched comparator. The shaded area illustrates the microphone connection to the circuit.

Ultra-Low-Power Processing Core

A 0.18µm CMOS technology was selected for its low leakage current, while easily meeting speed and density requirements. The core of the PCC-ATDE was synthesized with subthreshold CMOS logic [63, 2]. Reducing the power supply voltage helps to decrease both dynamic and static power consumption.

The variable-delay cells are implemented with multiplexed chains of 128 flip-flops. Delays are chosen by selecting a stage of the flip-flop chain. After the 1-bit quantization, multiplications are computed with XOR logic gates. An extra flip-flop after the upper multiplexer provides the fixed delay, τ_{fix}. A 10-bit register and adder accumulates, and the $1/G$ attenuation is realized with arithmetic shifts of 0, 1, or 2 bits. The output is divided by two to evenly distribute $\Delta[n]$ to both variable-delay cells. In the case of a odd $\Delta[n]$, one is added to the value of the lower variable-delay cell.

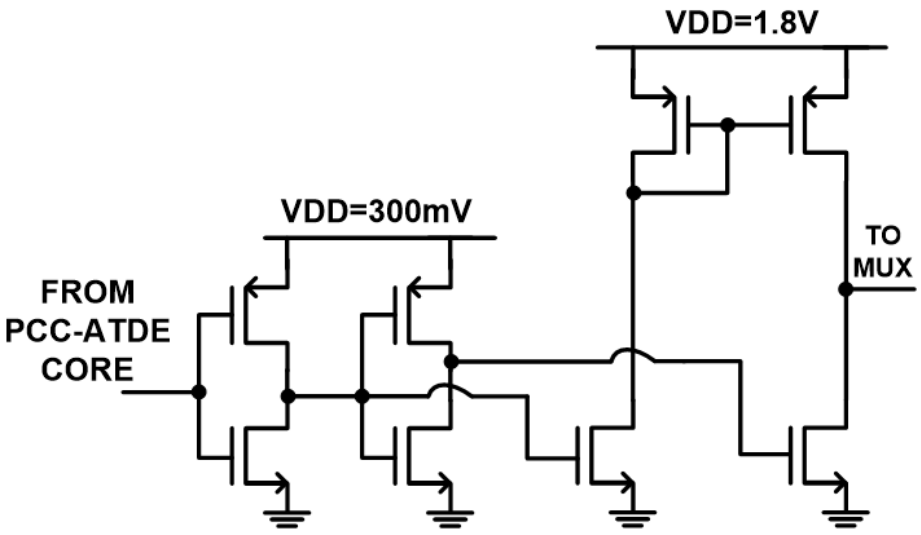

Figure 4.18: Schematic of the subthreshold level shifter used to convert the 300mv digital signal from the ultra-low-power core to the 1.8V level of the I/O pads.

Subthreshold Level Shifter

For this experiment, the 300mV signals from the PCC-ATDE core are converted to 1.8V I/O levels with the subthreshold level shifters shown in Figure 4.18. The current-mirror level shifters [79] guarantee the conversion of the subthreshold logic signals. In a fully integrated system implementation, these level shifters are not required, so their power consumption has not been included in the power-consumption assessment.

4.6 Experimental Characterization of the PCC-ATDE Operation

For the experimental performance characterization, arbitrary waveform generators (AWG) are used to provide the analog inputs. The AWGs output a 600mVpp 60Hz–200Hz band-limited noise signal to simulate the sound of approaching vehicles. All

AWGs are synchronized and under software control so that the delays between them can be precisely set for accurate measurements. Similar measurements were made to obtain Figures 4.12–4.14, but, for the characterization plots, the system operates at the optimal FOM sampling frequency, $F_S = 50\text{kHz}$, as shown in Section 4.7.

Step Response

The first measurement, shown in Figure 4.19, is the step response of the PCC-ATDE with different attenuator (G) values. As illustrated inside the dashed box, the step function is in the intersignal time delay between the analog inputs. Before the 5s mark, Input 2 is $D = -1\text{ms}$ delayed of Input 1. After the 5s mark, the delay changes to $D = 1\text{ms}$. The -1ms-to-1ms step response varies from 514ms when $G = 1$ to 2.05s when $G = 4$. As detailed in Section 4.4, the step response is amplitude dependent, a ± 1ms step was used since it fits a reasonable microphone spacing of 35cm for sound-source-localization devices.

Steady-State Response

The nonlinear, slew-limited behavior does not allow us to provide a straightforward number for the PCC-ATDE's bandwidth. Instead, we use the *power-bandwidth*, also known as full-power bandwidth, to characterize the system. It is defined by the maximum frequency of a rail-to-rail input the loop can handle. The calculated PCC-ATDE power-bandwidth (4.22) with a ± 2.52ms range is 0.252Hz, 0.126Hz, and 0.063Hz, for $G = 1$, $G = 2$, $G = 4$.

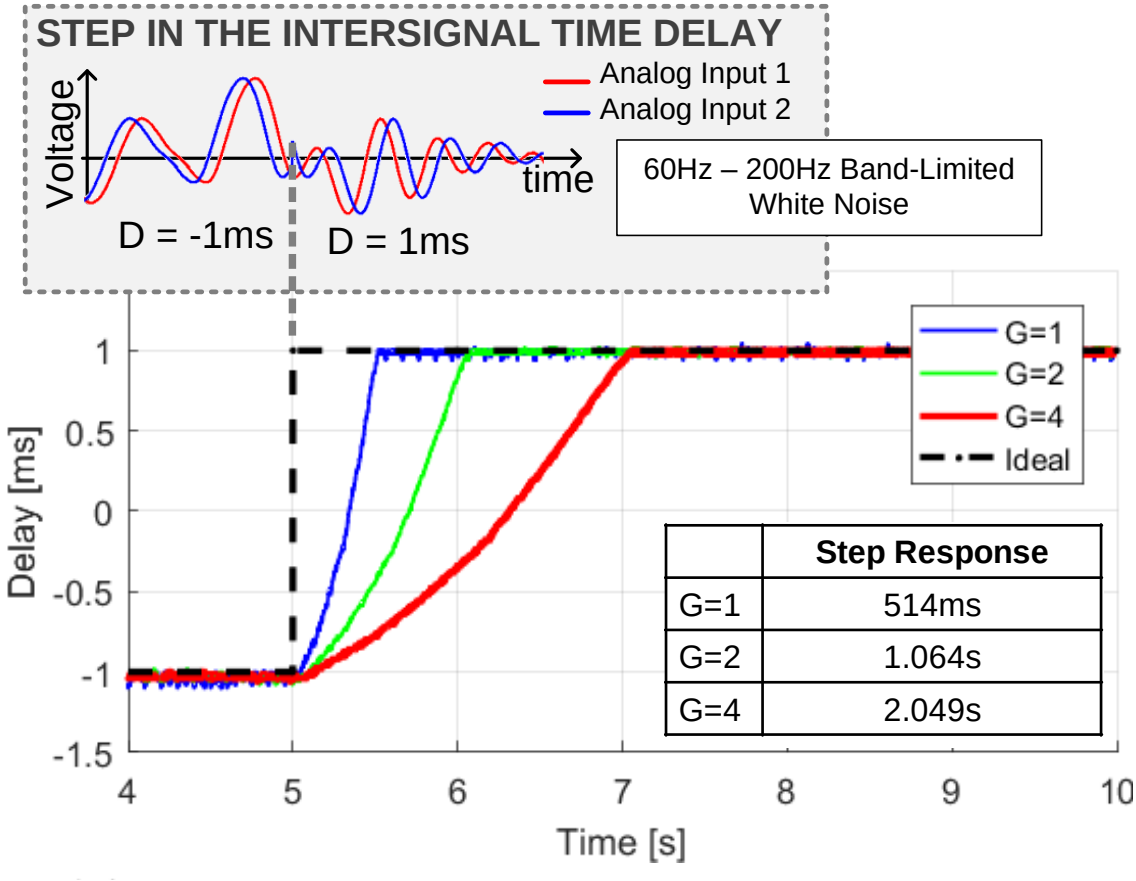

Figure 4.19: Measured PCC-ATDE ±1ms step responses. As shown in the shaded area, the step is in the intersignal time delay, not in amplitude. The dashed lines shows the delay switching from $D = -1$ms to $D = 1$ms at the 5s mark. The step response for the same input changes with the attenuation value, G from 514ms to 2.049s.

Linearity

The rail-to-rail pure-tone response is plotted in Figure 4.20 and is used to extract the expected number of bits (ENOB), 5.41bits to 6.06bits.

We also conducted static linearity tests. The y-axis of Figure 4.21 shows the digital codes obtained for analog inputs delayed by the corresponding x-axis values. The algorithm's clock improves linearity with a peak INL of $-1.57/1.33$LSB and peak DNL of $-0.85/0.97$LSB with $T_{\text{LSB}} = 20$µs with no need for calibration.

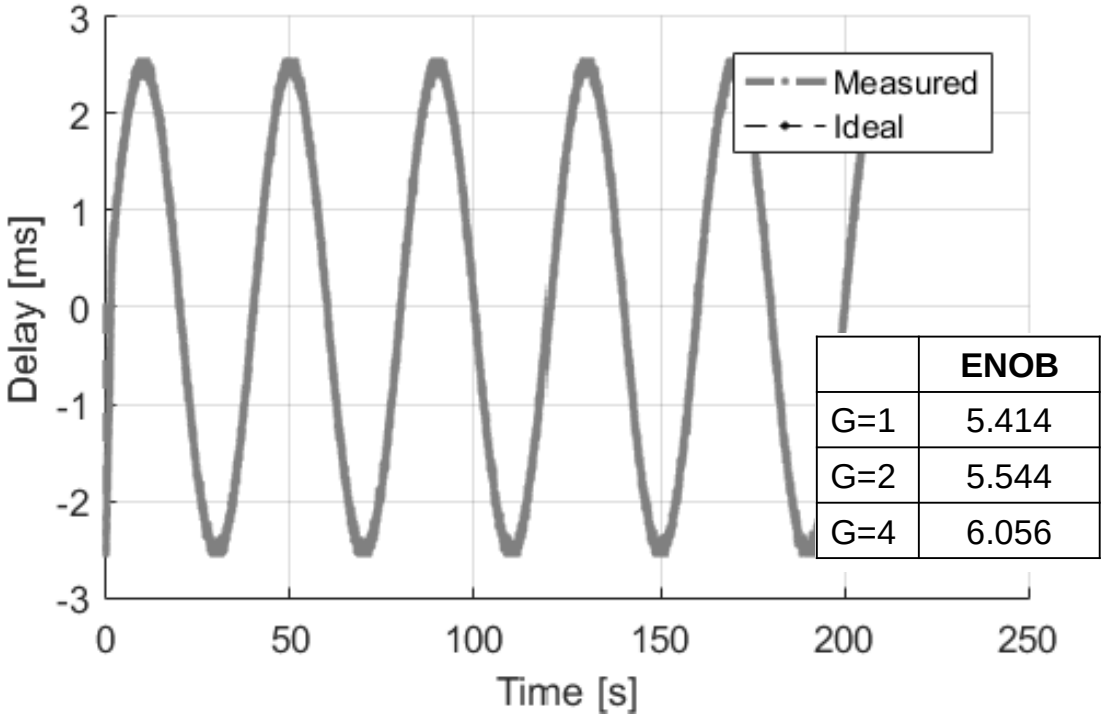

Figure 4.20: Steady state measurement used for ENOB calculations. A low-frequency rail-to-rail input was used to avoid slope saturation as detailed in Section 4.4.

4.7 Performance Comparison

Power Consumption Figure of Merit

To establish a metric that captures most of the PCC-ATDE's design aspects, we use a figure of merit (FOM) similar to that to compare ADCs:

$$FOM = \frac{Power}{\#channels \cdot 2^{ENOB} \cdot F_s} \tag{4.23}$$

The plot in Figure 4.22 shows the comparator's and ultra-low-power core's contribution to the FOM at frequencies from 10kHz to 800kHz. In each of the measurements, the power supply was adjusted to its minimum value to sustain normal operation at

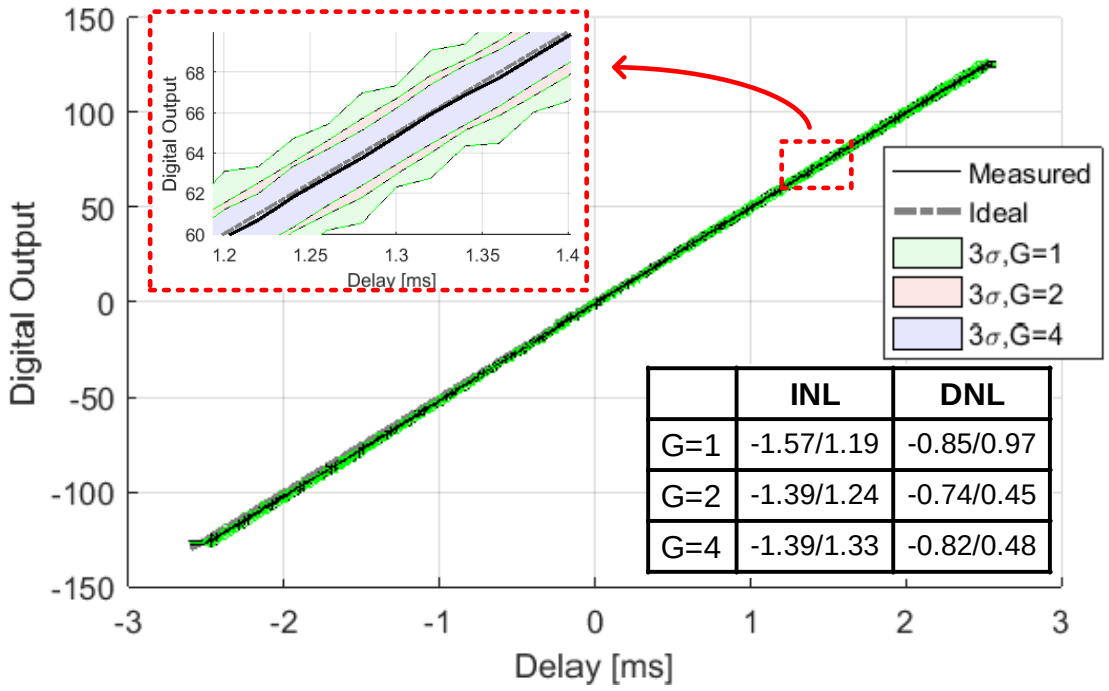

Figure 4.21: Linearity plot of the PCC-ATDE. The continuous black line is the ideal linear response. The dashed gray line is the measurement data. The colored areas around the plot are 3σ regions for different attenuation values, G.

the given F_s. Running at $F_s = 50$kHz the system's FOM reaches an optimal value of 7.84fJ/Conv.-Step. Operating with higher clock frequencies reduces T_{LSB} and can be used to enable TDoA calculation in higher frequency applications, such as ultrasound.

Comparison to the State of the Art

The LMS-TDE is the most commonly implemented adaptive time-delay estimation. Yet, to the best of our knowledge, no silicon implementations are available in the open literature. Details on the performance of the LMS-TDE were extracted from simulations presented in [69]. Similarly to the PCC-ATDE, the LMS-TDE uses feedback to perform the time-delay estimation, but the algorithm still requires a front-end ADC to capture microphone audio and significantly more arithmetic operations.

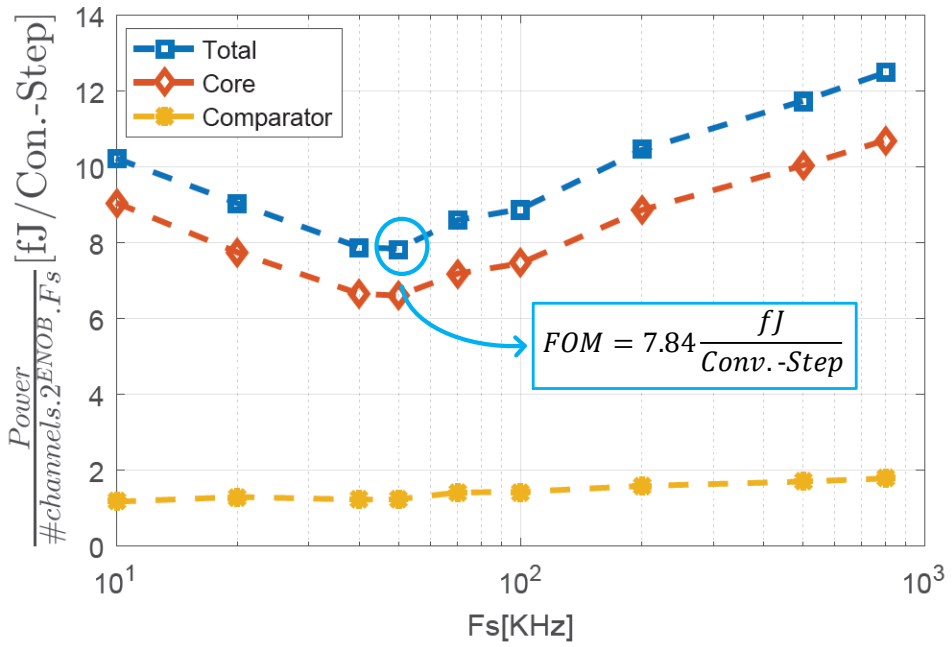

Figure 4.22: Figure of merit (FOM) plot over sampling frequency. The selected FOM is plotted as the clock frequency of the system is changed. For each FOM measurement, the supply voltages of the blocks are adjusted to the minimum possible value to sustain correct operation.

Cross-correlation based time-delay estimation is the conventional approach. Time-delay estimation is presented in 0.18µm, CMOS with subsample ($T_{\text{LSB}} < 1/F_S$) [29]. To store the audio frames and intermediate results from the algorithm, this solution uses 20kB of memory, in contrast with the 257 DFF required by the PCC-ATDE. The calculations also involve taking the FFT and iFFT of 1,024-point vectors that are much more complex than the basic XORs and adders present in the PCC-ATDE. The result is a normalized area more than six times larger and a power FOM $10^5\times$ higher than for the PCC-ATDE.

The binaural silicon cochlea uses address-event representation (AER) to estimate the time delay between microphones [47]. The AER also provides information on the spectral content, and more recent work on AER silicon cochlea has been published

Table 4.1: Comparison table of low-power sound-source-localization solutions. Highlighted are the parameters in which this work excels: low arithmetical complexity; low power per conversion step; and small normalized area.

SOUND SOURCE LOCALIZATION SOLUTIONS				
Method	LMS-ATDE	GCC-PATH	Silicon Cochlea	PCC-ATDE
Reference	[14]	[11]	[15]	[This Work]
Technology	**Simulation**	180nm CMOS	350nm CMOS	180nm CMOS
Sampling Frequency	1KS/s	20KS/s	~20Keps	50KS/s
Interface with Analog Inputs	8-bit ADC	8-bit ADC	BPF bank, Half-wave Rectifier, Pulse-frequency Modulator	Comparators
Memory Required per TDE	31B	20-kB	-	257 Flip-Flops, 1 10bit register
Arithmetical Complexity per ATDE	362 Additions, 373 Multiplications	2 1024-point FFT, Maximum Likelihood Search	Peak Seach on Events Matching Histogram (~150KOperations/s)	2 XORs, 4 Adders, 1 Arithm. Shifter
Range	-	[-1.5ms,1.5ms]	[-700µs,700µs]	[-2.52ms,2.52ms]
T_{LSB}	1ms	5µs	10µs	20µs
Number of TDE Channels	1	1	1	3
TDE ENOB	-	-	-	6.06bits
Total Power	-	28.98mW	357µW	78.2nW
TDE Energy per Conversion Step per Channel [1]	-	~2.4nJ/Conv.-Step	~127pJ/Conv.-Step	7.84fJ/Conv.-Step
Area		6.25mm²	16.2mm²	1mm²
Normalized Area [2]	-	193µ(mm/nm)²	132µ(mm/nm)²	31µ(mm/nm)²
Portable to Off-the-Shelft Solutions	Yes	Yes	No	Yes

1. $FOM = \frac{Power_{Total}}{\#Channels.\,2^{ENOB}.\,Fs}$ 2. $A_{norm} = \frac{A_{Total}}{\lambda^2}$

[81, 4], but the selected paper [47] has more details on its sound-source localization performance. The silicon cochlea needs many operations to convert the AER into a time-delay estimation. Combined with the AER circuitry, the solution is still more than four times as large as the PCC-ATDE and has $10^4\times$ the power FOM.

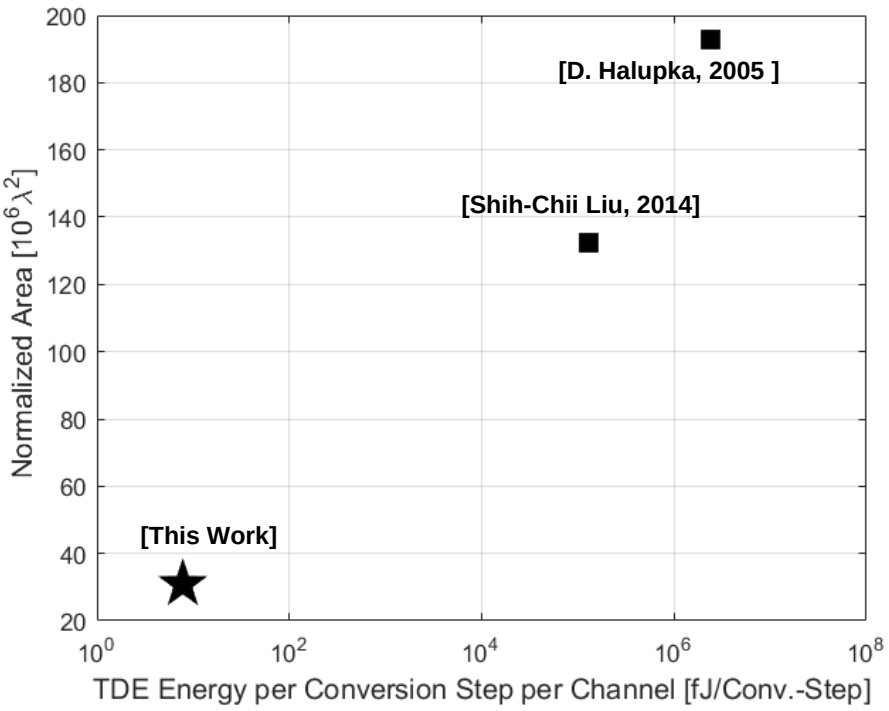

Figure 4.23: Comparison plot of previous sound-source localization solutions. The power FOM of the solutions are plotted versus the prototypes' normalized areas.

4.8 Sound-Source-Localization Experiments

Sound-Source Localization in an Controlled Environment

All previous experiments prove that the PCC-ATDE is able to extract the intersignal time delay from two analog inputs. But, to deploy this technique in a sound-source-localization system, we needed to verify if second-order effects, such as reverberation or mismatches in microphone responses, would affect system operation. We conducted experiments with microphones and speakers and with real-life audio inputs on the PCC-ATDE.

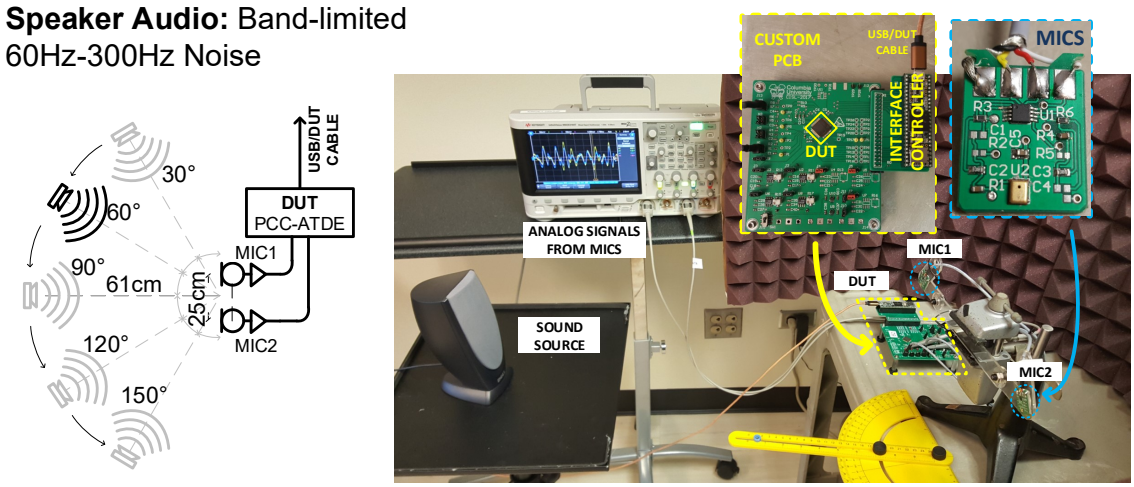

Figure 4.24: Setup used to compare the performance of the PCC-ATDE prototype and traditional time-delay estimation techniques. On the left, the diagram shows how the sound-source rotates around the microphone pair. On the right is a photo of the experimental setup.

TDE Performance Comparison

In this experiment, we compared the results of time-delay estimations using the PCC-ATDE prototype and a standard GCC-PHAT approach. Figure 4.24 shows the setup used in the experiment. A single sound source playing a band-limited white-noise recording is placed near a microphone pair. The TDoA of the sound wave from the speaker to each microphone causes the time delay between the analog inputs. The sound source rotates around the microphone pair, and, for each angular position, the delay estimations from the PCC-ATDE device under test (DUT) and 400ms frames from both analog signals are collected. The experiment was conducted in a 4m-by-6m closed room without acoustic isolation. However, when the microphones were close to the walls, a piece of acoustic foam was used to reduce reverberation.

Figure 4.25 presents the results of the experiment. The delay acquired from the PCC-ATDE DUT and using a GCC-PHAT algorithm with the collected frames are

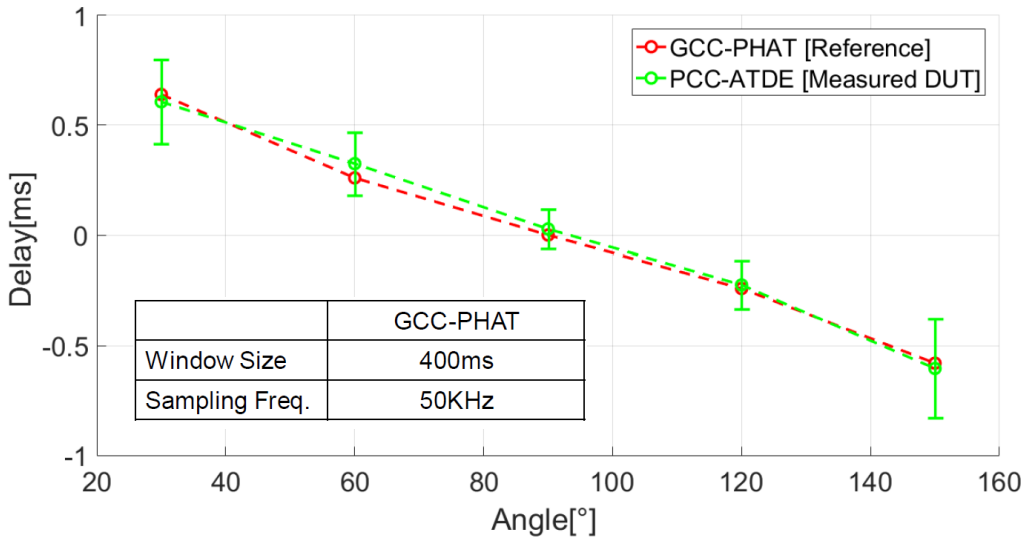

Figure 4.25: Time-delay estimation results from the PCC-ATDE, in green, and from GCC-PHAT, in red. The GCC-PHAT time-delay estimation was performed using $400ms$ sampled at 50kHz by the oscilloscope.

plotted for each incidence angle. The time-delay estimates match closely, with a RMS error of 37.2µs (or 2.3%).

Sensitivity

The next behavior we investigate is the effect of a second sound source in the time-delay estimation. We now place two speakers next to the microphone pair, as shown in Figure 4.26. The speakers are at different positions with distinct TDoAs. The ratio of speaker power is swept form -20dB to 20dB, the resulting time-delay estimation is plotted in Figure 4.27.

D_{A} is the expected time-delay estimation if only Source A was present, and D_{B} the delay with only B. The final result leads to a empirical observation that the resulting time-delay estimation (D_{OUT}) can be expressed as the average of the individual delays

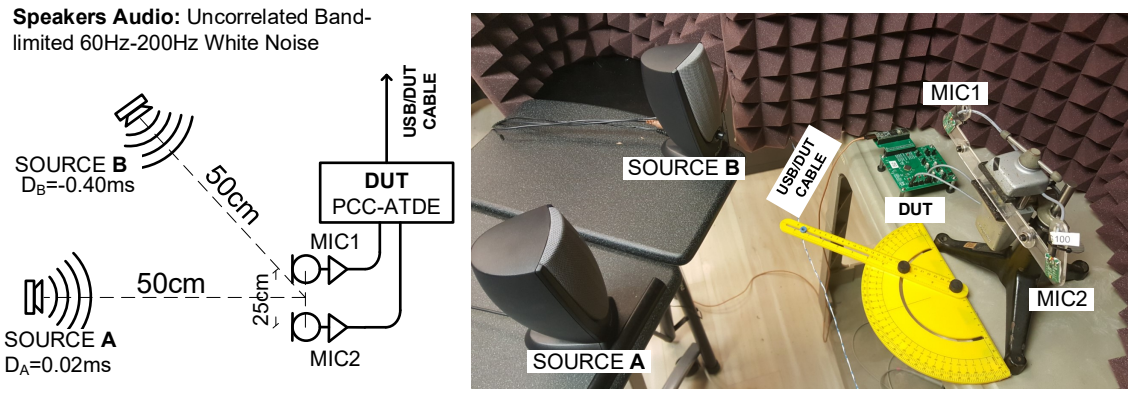

Figure 4.26: Setup used to measure the effect of an interfering sound source in the PCC-ATDE estimation. The diagram at left shows two sound sources with unique TDoAs playing uncorrelated recordings with different power.

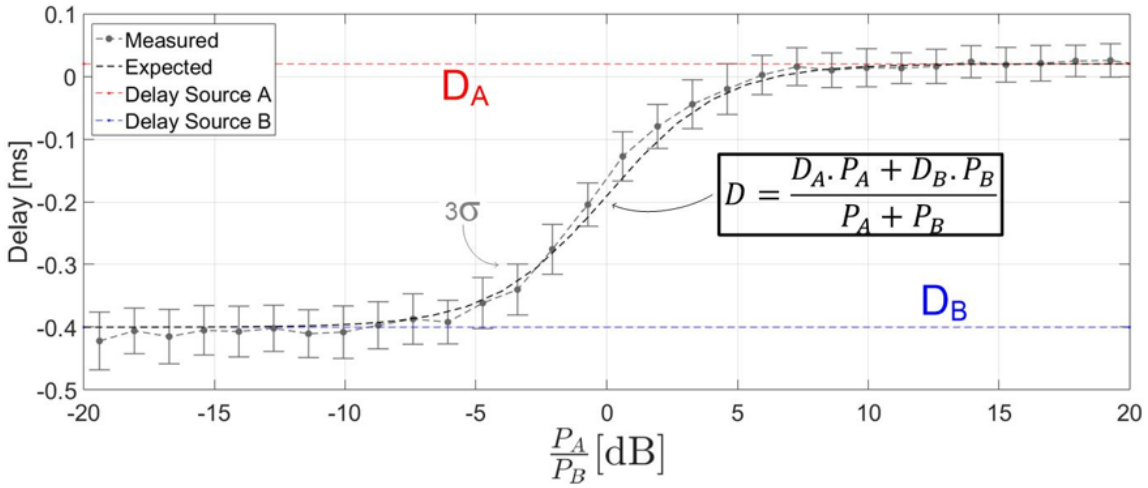

Figure 4.27: Measured time-delay estimation of the PCC-ATDE prototype as function of the relative power of two interfering sound sources. Inside the box is an empirical expression for the resulting delay.

weighted by their relative powers:

$$D_{\text{out}} = \frac{D_{\text{A}} \cdot P_{\text{A}} + D_{\text{B}} \cdot P_{\text{B}}}{P_{\text{A}} + P_{\text{B}}} \tag{4.24}$$

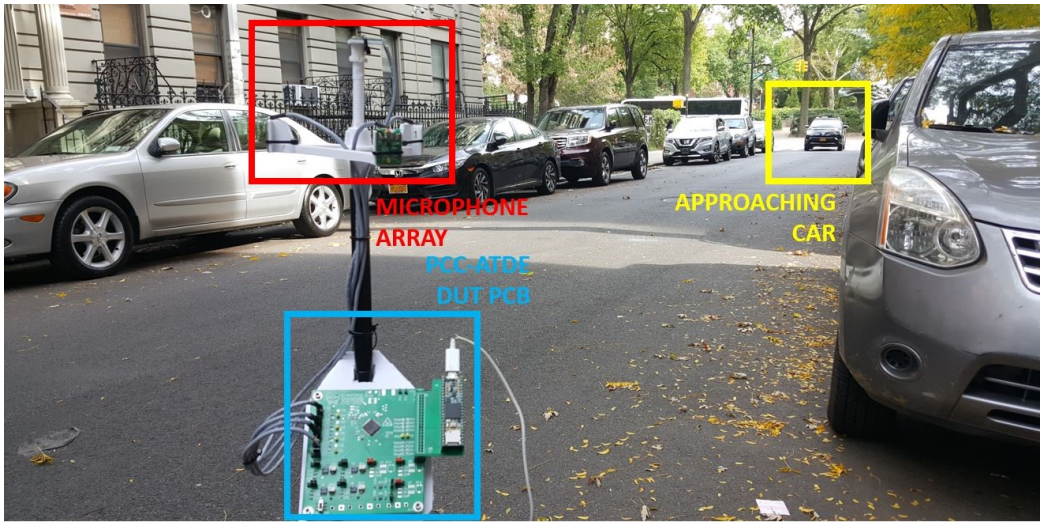

Figure 4.28: Vehicle-bearing experiment. Inside the yellow box is the approaching car the system is detecting. The red box marks the pyramid microphone array; and the blue box shows the PCC-ATDE DUT PCB.

Estimating a Vehicle's Bearing on a City Street

Finally we integrated the PCC-ATDE DUT into an IoT embedded system for vehicle-bearing estimation with the four microphones in a pyramid structure connected to the DUT. We used a microcontroller only to interface the digital output of the time-delay-to-digital converter to a host computer.

Figure 4.28 shows a photo of the setup in the New York street where the experiment was conducted. We placed the system in a one-way street between two busy avenues and measurements were conducted during regular hours of a weekday.

The TDoA of the vehicle's noise to each microphone varies as the car moves from right to left past the array. This is captured by the three extracted intersignal time-delay estimations of the PCC-ATDE plotted in Figure 4.29.

Running on a host computer, a k-nearest neighbors (KNN) [43] machine-learning classifier is used to convert the time-delay data into incidence angles in real time.

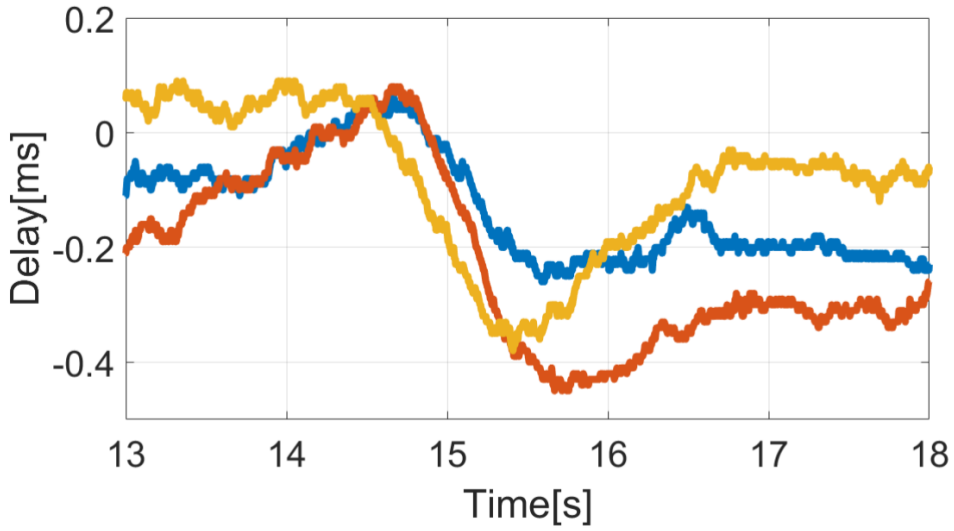

Figure 4.29: Three measured channels of time-delay estimation using the PCC-ATDE embedded setup. The measured delays are caused by an approaching car's noise in the experiment illustrated in Figure 4.28.

Even though the classifier was trained indoors, with car-sound recordings playing at different incidence angles from the array, it was able to track the vehicle accurately. Figure 4.30 shows the recorded real-time output of the KNN classifier changing from 0° when the car is on the right side of the microphone array to 180° after it crosses to the left in with a 30° resolution.

4.9 Conclusions

By condensing the traditional ADC–DSP processing chain in an analog-to-feature converter, we reduce the power consumption to less than 100nW, four orders of magnitude less than conventional techniques. Our analog-to-feature converter, prototyped in 0.18µm CMOS, successfully estimates the time delay under all tested conditions. We carefully analyzed the behavior of the proposed PCC-ATDE, and introduced a

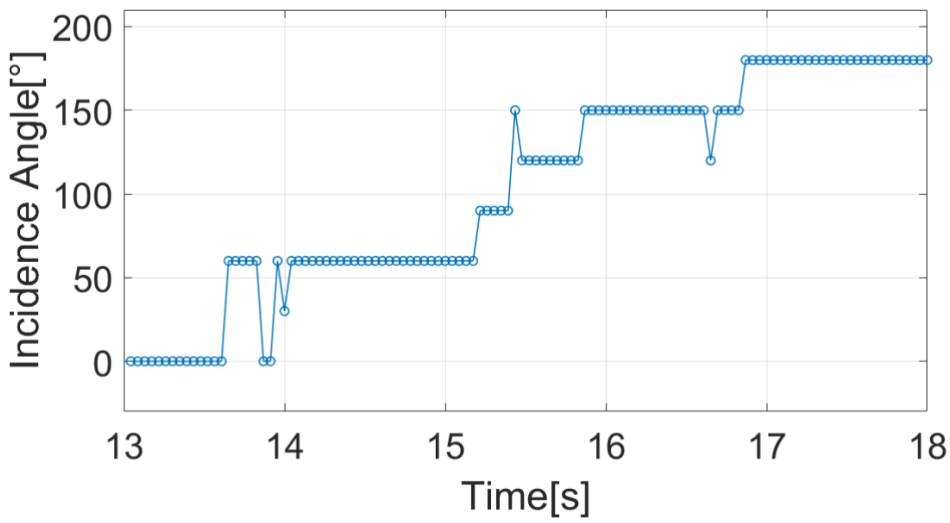

Figure 4.30: Output of the KNN classifier for the vehicle-bearing estimation. With the time delays shown in Figure 4.29, the classifier predicts the incidence angle of the sound waves, 0° for a wave hitting from the right and 180° from the left.

simplified delay-domain model that allows us to accurately predict the behavior of the time-delay estimator.

This mixed-signal approach to obtain features in cyberphysical systems is promising for resource constrained solutions, especially for always-on battery-powered systems. Even though the PCC-ATDE was demonstrated in a sound-source-localization system, the technique can be applied on other systems that use time-delay estimation, and it should especially be considered if the required T_{LSB} is very small and the frequency of the analog input signals is low.

Chapter 5

An Ultra-Low-Power Envelope-to-Digital Converter

5.1 Introduction

After being able to extract the intersignal time delay between the microphones using less than 100nW, the next feature that is investigated in this work is the relative power between the microphones. Just as the intersignal time delay maps to the ITD in the source-localization contest detailed in Chapter 2, the relative power can be used to extract the ILD. For that, an collaborative investigation with Erjia Shi and Prof. Kong Pang Pun was started to the develop an ultra-low-power solution to extract the envelope of an audio signal, and its initial results are presented at [67].

In this chapter, we use the analog-to-feature approach to develop an envelop-to-digital converter (EDC). Unlike conventional solutions that digitize the signal at the Nyquist rate and later extract the envelope amplitude, we isolate the signal envelope and then use an ADC at a much lower sampling rate to convert the result, as shown in Figure 5.1. Other options—diode-based rectifiers [52] or class-AB current conveyors [62]—could extract the envelope effectively, but their integration is challenging due

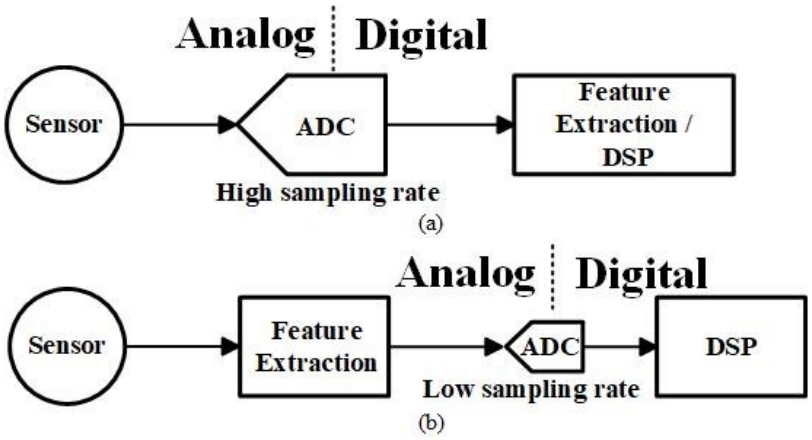

Figure 5.1: (a) Nyquist-rate ADC solution. (b) Analog-to-feature solution. With the envelope isolated from the signal, the ADC's required sample rate drops significantly.

to the large capacitors required for low-frequency envelopes. They are also unfit for low-power, low-voltage designs: They need large bias voltages to turn on their diodes [52] and considerable power to drive their large load capacitors [62, 64].

5.2 Envelope-to-Digital Converter

Figure 5.2 presents the proposed solution for the ultra-low-power EDC. The circuit is divided in two main sections, a switched-capacitor (SC) envelope detector and a low-frequency low-power SAR ADC. Even though the blocks are clearly separated in the diagram, they work inseparably, interchangeably sharing each element to avoid using a buffer.

SC Envelope Detector

The basic blocks required for an analog envelope detector are a nonlinear element, such as a diode or a squarer, and a low-pass filter. The proposed envelope detector,

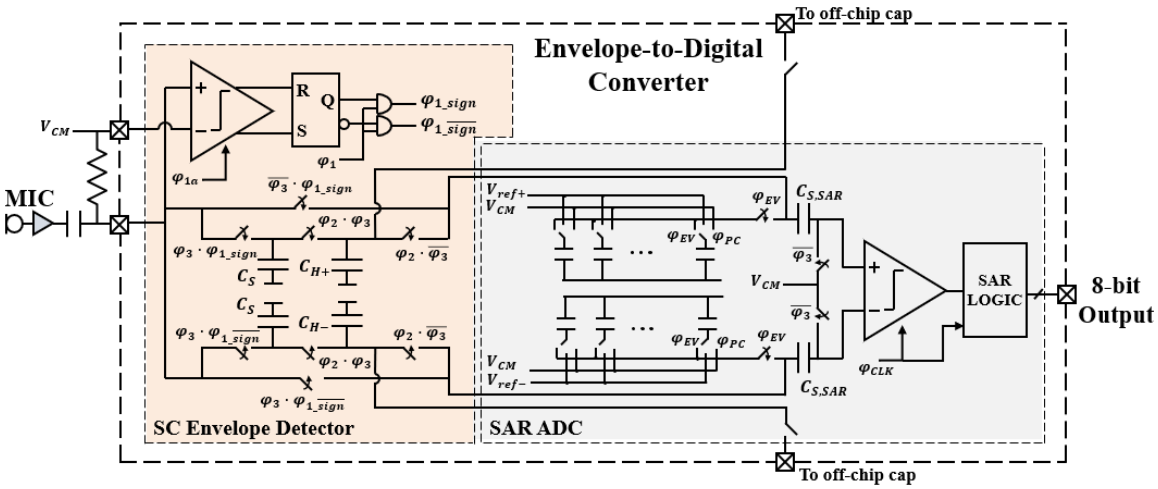

Figure 5.2: Block diagram of the envelope-to-digital converter. The dashed lines mark the boundaries of the integrated circuit. The switched-capacitor envelope detector and the low-frequency SAR ADC sections are marked with different colors.

shown in Figure 5.3, uses a comparator-based polarity detector as the nonlinear element. It rectifies the signal by steering the input of the circuit V_{in} to the upper or lower branch based on its polarity. For the filtering, each branch has a SC first-order low-pass filter. The SC envelope detector must operate at the Nyquist rate or higher, but its power consumption is much less than a complete ADC operating at the same rate. The comparator used in the envelope detector is similar to the comparator presented in Section 4.5. The signal from the microphone V_{in} is AC-coupled and all the analysis is referred to V_{CM}, such that $V_{\text{in}} \cdot \text{sign}(V_{\text{in}}) = |V_{\text{in}}|$.

The timing diagram of the SC envelop detector is shown in Figure 5.4. The clock signal ϕ_{1a}, not shown in the diagram, is slightly advanced from ϕ_1. At ϕ_1, The polarity of the input signal will determine which sampling capacitor C_{S} will be charged. Later, at ϕ_2, the charge of each C_{S} is shared between C_{S} and C_{H}. Note that C_{S} is refreshed every ϕ_1, but C_{H} will hold the contribution of all previous charge sharing. Looking at the voltage hold at C_{H} at a given cycle n at the end of ϕ_2, we have

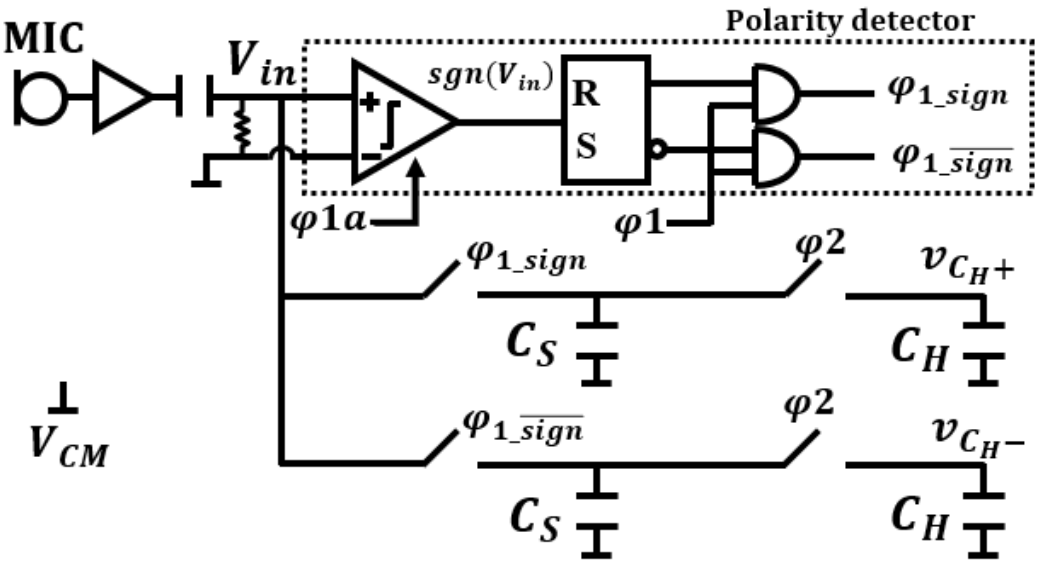

Figure 5.3: Schematic of the switched-capacitor envelope detector circuit. The polarity detector controls the SC circuit's input switch together with ϕ_1. V_{in} will charge the C_{S} of either the top or bottom branch, and the charge sharing with C_{H} will filter out the high frequencies of the signal at ϕ_2.

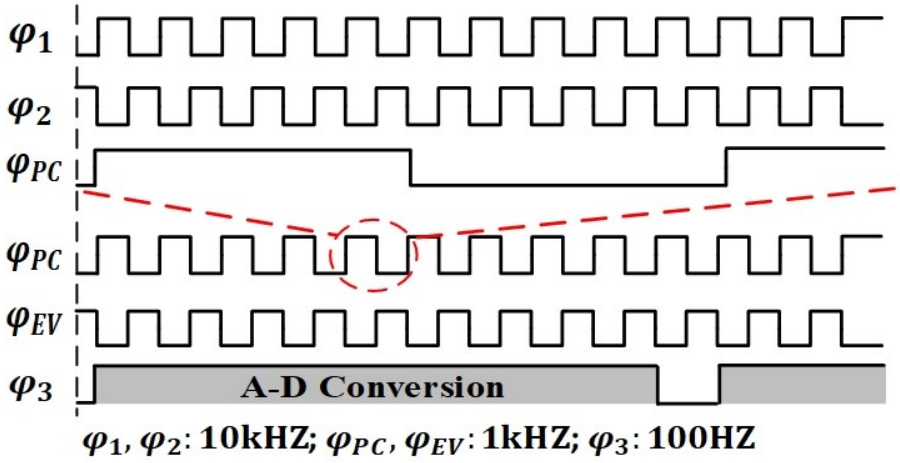

Figure 5.4: Critical clock signal used in the envelope-to-digital converter.

$$V_{\text{CH+}}[n] = \frac{C_{\text{S}} \cdot V_{\text{in}}[n] \cdot (\frac{1}{2} + \frac{1}{2} \cdot \text{sign}(V_{\text{in}}[n])) + C_{\text{H}} \cdot V_{\text{CH+}}[n-1]}{C_{\text{S}} + C_{\text{H}}} + \frac{C_{\text{S}} \cdot V_{\text{CH}}[n-1] \cdot (\frac{1}{2} - \frac{1}{2} \cdot \text{sign}(V_{\text{in}}[n]))}{C_{\text{S}} + C_{\text{H}}} \quad (5.1)$$

$$V_{\mathrm{CH-}}[n] = \frac{C_{\mathrm{S}} \cdot V_{\mathrm{in}}[n] \cdot (\frac{1}{2} - \frac{1}{2} \cdot \mathrm{sign}(V_{\mathrm{in}}[n])) + C_{\mathrm{H}} \cdot V_{\mathrm{CH+}}[n-1]}{C_{\mathrm{S}} + C_{\mathrm{H}}} + \frac{C_{\mathrm{S}} \cdot V_{\mathrm{CH}}[n-1] \cdot (\frac{1}{2} + \frac{1}{2} \cdot \mathrm{sign}(V_{\mathrm{in}}[n]))}{C_{\mathrm{S}} + C_{\mathrm{H}}} \quad (5.2)$$

If we subtract $V_{\mathrm{CH+}}[n]$ and $V_{\mathrm{CH-}}[n]$, the output of the envelope detector $V_{\mathrm{out}}[n]$ is

$$V_{\mathrm{out}}[n] = \frac{C_{\mathrm{S}}|V_{\mathrm{in}}[n]| + C_{\mathrm{H}}V_{\mathrm{out}}[n-1] + \frac{1}{2}C_{\mathrm{S}}V_{\mathrm{out}}[n-1] - V_{\mathrm{out,CM}}[n-1] \cdot \mathrm{sign}(V_{\mathrm{in}}[n])}{C_{\mathrm{S}} + C_{\mathrm{H}}} \quad (5.3)$$

Since the V_{in} is AC coupled, we expect the common mode of the output $V_{\mathrm{out,CM}}[n-1]$ to be close to the bias voltage V_{CM}. If we neglect the term with $V_{\mathrm{out,CM}}$ in (5.3),

$$V_{\mathrm{out}}[n] = \frac{C_{\mathrm{S}} \cdot |V_{\mathrm{in}}[n]| + (C_{\mathrm{H}} + \frac{1}{2} \cdot C_{\mathrm{S}}) \cdot V_{\mathrm{out}}[n-1]}{C_{\mathrm{S}} + C_{\mathrm{H}}} \quad (5.4)$$

From (5.4), we see that V_{out} is the low-pass filtered result of the absolute value of V_{in}, with transfer function

$$\frac{V_{\mathrm{out}}}{|V_{\mathrm{in}}|}(z) = \frac{1}{1 + \frac{C_{\mathrm{H}}}{C_{\mathrm{S}}} - (\frac{1}{2} + \frac{C_{\mathrm{H}}}{C_{\mathrm{S}}})z^{-1}} \quad (5.5)$$

We used a 500fF C_{S} and a 12pF C_{H} in this work. The size and ratio of the capacitors are such that both acceptable thermal noise and sufficient low-pass filtering are achieved. Theoretically, the envelope detector has an conversion gain of 1.2. For an input bandwidth up to 5kHz, the envelope-detector sampling frequency is 10kHz.

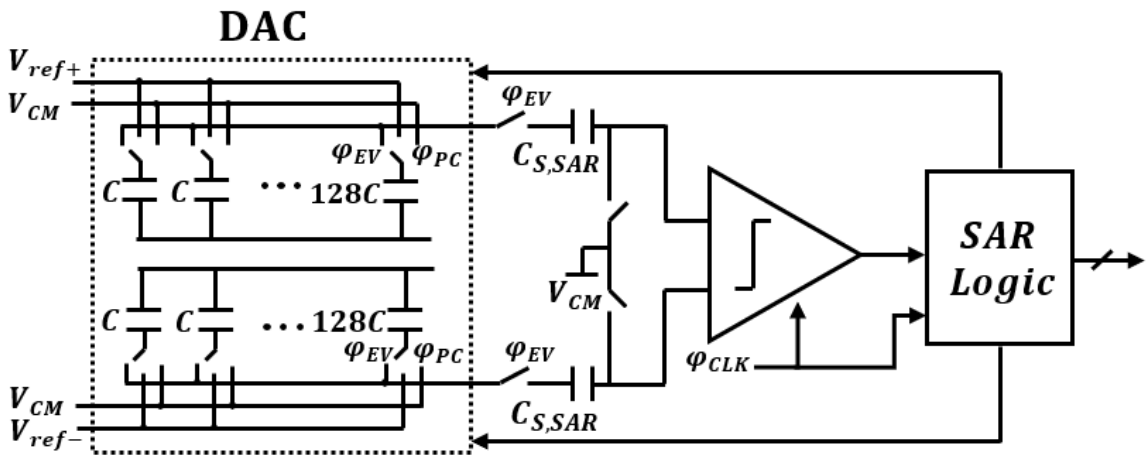

Figure 5.5: Schematic of the low-frequency ultra-low-power SAR ADC.

Low-Frequency Ultra-Low-Power SAR ADC

The next step on the EDC is to digitize the voltages stored in $V_{\mathrm{CH+}}$ and $V_{\mathrm{CH-}}$. As straightforward as this sounds, it is challenging given the sub-10nW power budget of the solution. Remember that the envelope information is stored in the capacitors; they are not able to drive conventional low-power ADCs. Adding buffers between the output of the envelope detector and the ADC input would solve this problem, but the buffer power itself would exceed the 10nW budget. We solve this problem by introducing a ping-pong scheme that uses the charge in C_S directly as the input to the ADC.

Figure 5.5 shows the circuit diagram of the 8-bit SAR ADC [37]. During ϕ_{PC}, the binary-weighted DAC capacitors are precharged to a certain DC voltage controlled by the SAR logic. In ϕ_{EV}, the DAC voltage and the sampled signal are subtracted from the inputs of the comparator, which decides to increase or decrease the SAR register's contents. The ADC's comparator has the same topology as the envelope detector's comparator. By slightly delaying ϕ_{EV}, the phase ϕ_{CLK} is used to control

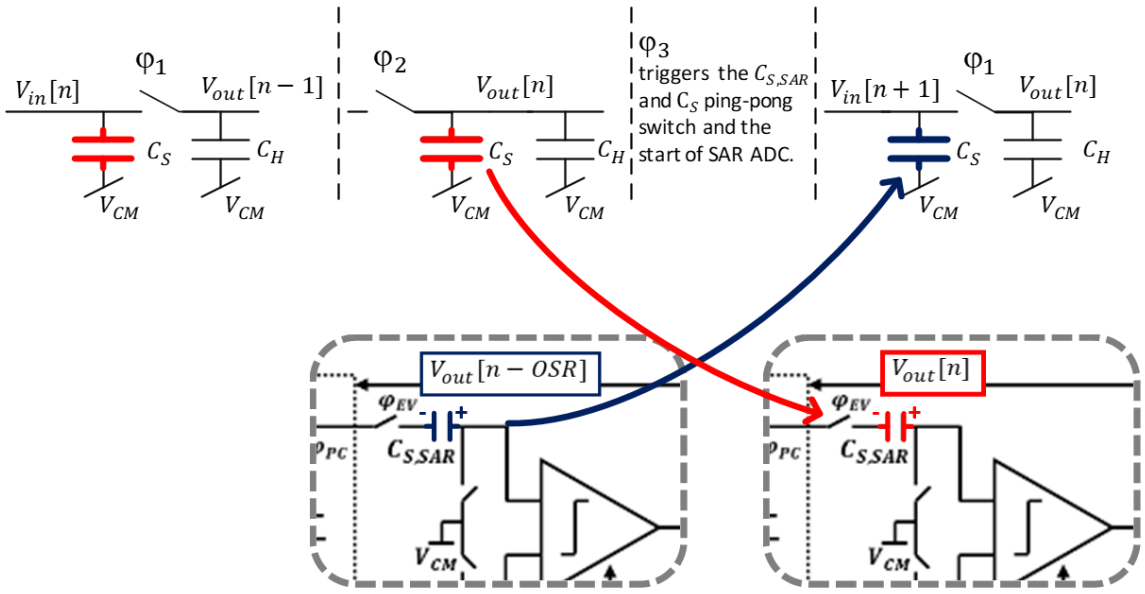

Figure 5.6: Illustration of the ping-pong scheme between C_S and $C_{S,SAR}$. C_S is used in the low-pass filtering of the SC envelope detector and constantly refreshed at the rate of ϕ_1. Triggered by ϕ_3, the C_S that was used in the SC envelope detector and held the value of the last operation is connected as the input to the SAR ADC, and the $C_{S,SAR}$ that was being used previously in the SAR ADC replaces C_S.

the comparator and SAR logic block. Because the voltages in the sampling capacitors are rectified, the DAC needs only a unipolar output and uses an asymmetric reference-voltage scheme.

But, differently from prior works, we do not use the previous stage to charge the capacitor $C_{S,SAR}$. Instead, we directly switch, in a ping-pong manner, $C_{S,SAR}$ and C_S from the SC envelope detector. Figure 5.6 details the step-by-step action of the ping-pong between C_S and $C_{S,SAR}$.

Notice that, as shown by ϕ_3 in Figure 5.4, the SAR ADC sampling is much slower than the SC envelope detector. That can be done because the bandwidth of the envelope detector is much smaller that the raw input signal itself. The oversampling ratio (OSR) between the SC envelope detector and the SAR ADC is set to 100,

determining the overall 100Hz sample rate for the EDC. $C_{\mathrm{S,SAR}}$ needs to have the same value as C_{S} for the ping-pong scheme to be seamless to the SC envelope detector. The DAC's unit capacitor is set to 100fF, four times the minimum available value. The ADC operates at 0.6V except for the SAR logic, which runs at 0.4V to optimize power consumption at low operating frequencies. V_{CM} for both the SC envelope detector and the ADC is 0.3V.

Experimental Results

The EDC prototype was fabricated in a 180nm CMOS process. Figure 5.7 shows the die photograph of the chip, which occupies 0.32mm^2 total area. The EDC is designed to support a wide range of applications with various target frequencies. Hence, a programmable off-chip microcontroller is employed to generate the clocks. The microcontroller is sufficiently fast to provide the synchronized clock. Simulations show that the clock-generation and switch-driving circuits consume a few nanowatts, which will not dominate the power budget if they are implemented on chip. We tested the SC envelope detector alone with amplitude-modulated signals. As shown in Figure 5.8, 100Hz and 1kHz carriers are modulated by a 10Hz signal of varying magnitude. The SC envelope detector tracks the 10Hz sinusoidal envelope and removes the carrier. It attenuates the carrier more with higher frequency, which is in agreement with the transfer function described in (5.5). We conducted single-tone tests with a 1kHz pure sinusoidal signal to examine the accuracy of the SC circuit (Figure 5.9). The SC envelope detector achieves an amplitude RMS error less than 1.55mV: The circuit is

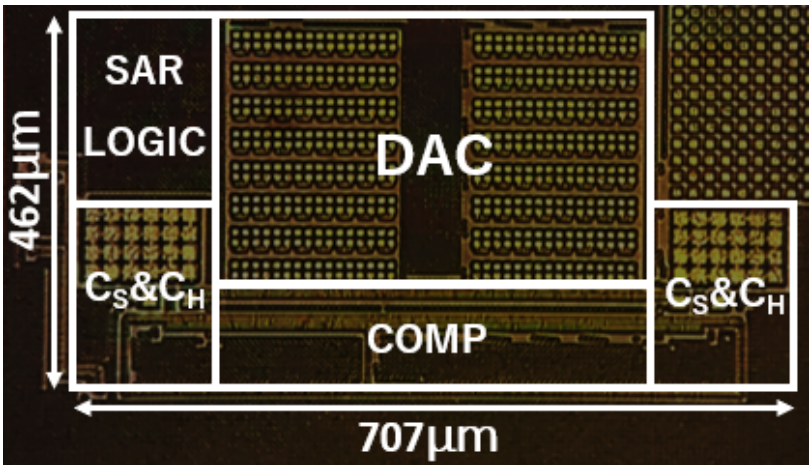

Figure 5.7: Microphotograph of the envelope-to-digital converter CMOS prototype.

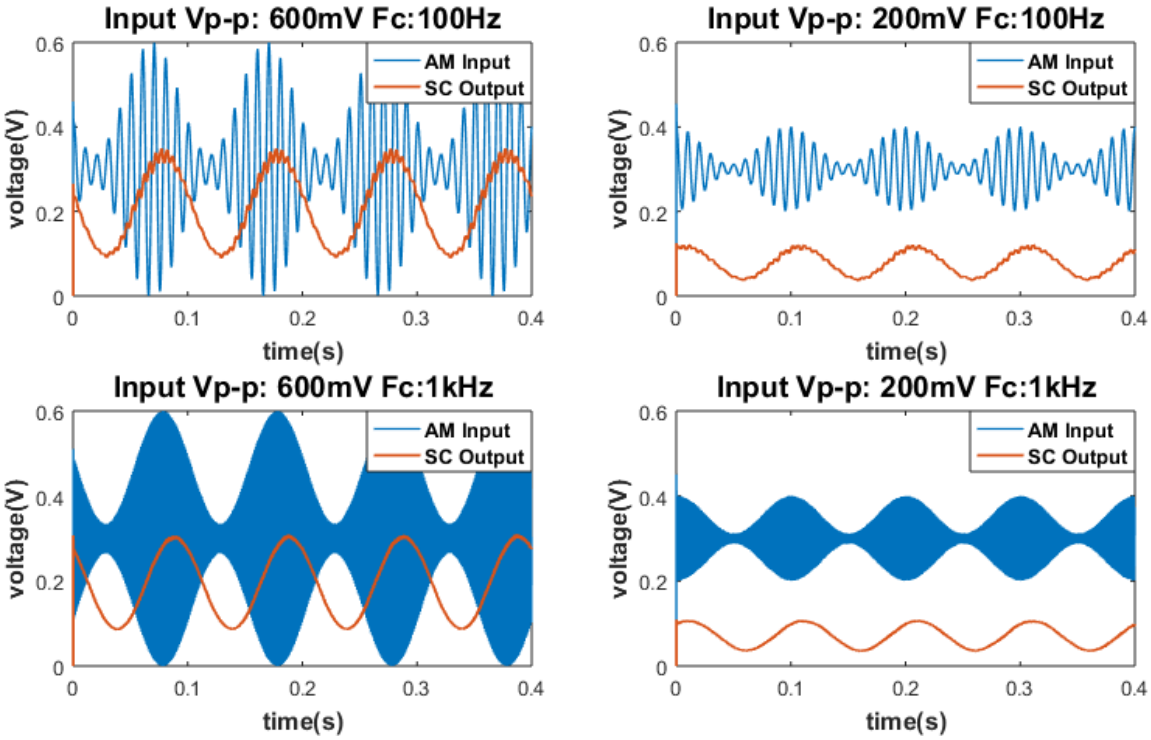

Figure 5.8: SC envelope detector outputs for amplitude-modulated signals with different carrier frequencies F_C (100Hz/1kHz) and amplitudes (0.6Vp-p/0.2Vp-p).

linear. The conversion gain is 1.2, the same as the theoretical value.

The SAR ADC in this chip can be tested alone by applying an input signal across the output pins (pads labeled "To off-chip cap" in Figure 5.2) of the SC envelope

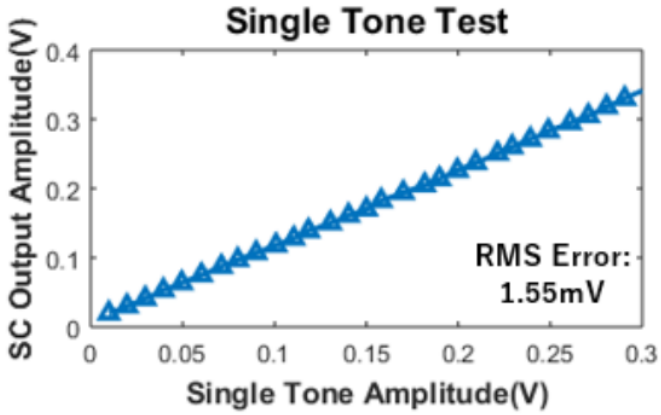

Figure 5.9: Single-tone linearly test of the envelope-to-digital converter.

Table 5.1: Comparison table of low-power envelope extraction solutions.

	This work [67]	**[7]**	**[62]**	**[64]**	**[36]**	**[33]**
Application	Respiratory Monitoring	Wake-up Circuit	Bionic Ear Processor	EMG Readout	ECG Monitoring	Signal Acquisition
Feature	Envelope	Energy	Envelope	Envelope	Raw ECG	Raw Signal
Technology	180nm	90nm	1.5µm	180nm	65nm	65nm
Bandwidth	5kHZ (raw input) 50Hz (envelope)	2kHZ	10kHz	500Hz	250Hz	292Hz
ENOB of ADC	7.1bit	N/A	N/A	9.2bit	N/A	7.14bit
Power	9.6nW *	700nW	875nW	19µW	18.6nW	3nW
Configuration	Envelope detector + ADC **	Wake Up detector	Envelope detector	Preamp + Envelope detector + ADC	Preamp + ADC	Preamp + ADC

* Clock generation is not included. The DAC, SAR logic, and envelope-detector consume 4.8nW, 1.1nW, and 3.6nW, respectively.

** Preamplifier (Preamp) is not implemented in this work. Estimated from [33], the preamp power is 9nW.

detector. The ADC consumes 6nW at 100Hz sampling rate. We used a 5Hz, 0.59VP-P sinusoidal wave to test the ADC's dynamic performance. The ADC achieves a 44dB signal-to-noise-and-distortion ratio (SNDR), corresponding to a 7.1 ENOB. As the SC envelope detector is intrinsically nonlinear due to its rectification operation, we did not characterize the linearity of the entire EDC.

Table 5.1 summarizes the proposed EDC's performance. The entire EDC consumes 9.6nW with 100Hz ADC sampling. Its energy efficiency compares favorably with existing circuits for similar applications.

5.3 Conclusions

The first prototype of the EDC proposed in this work shows promising results. Other work has demonstrated a respiratory-monitoring application using this ultra-low-power EDC [67].

The EDC consumes 9.6nW power with a 100Hz output data rate and supports input bandwidth up to 5kHz. Experimental results show that the EDC is able to extract envelopes from various respiratory sounds and the respiratory rate can be successfully computed from the extracted envelopes.

This demonstrates the concept of the analog-to-feature conversion by removing redundancy in A–D conversion and data transmission and reducing the overall power consumption of the solution. Which is critical to extending the lifetime of these always-on, energy-conscious IoT devices.

There is still work to be done on the proposed EDC. Further characterizing and optimizing the technique may reduce the total power consumption even further. To close the loop with the initial sound-source-localization motivation for the EDC, the circuit still needs to be connected to a microphone array and the digital output used to find the sound source.

Conclusion

This dissertation has discussed ultra-low-power challenges and solutions for feature-extraction front-end blocks applied to sound-source-localization IoT embedded systems. The focus of the research is to take full advantage of the already established machine-learning digital-signal flow to drive the optimization of data converters. Traditional data converters are still bounded by the objective of faithfully representing the analog waveform in the digital domain. But, for machine-learning systems, only specific characteristics, features, of the signal are used. Digitizing the full content of the analog signal, just to have a following digital-signal processing block extract the relevant piece of information, leads to an unnecessary use of power and storage space. Designing data converters that digitize only the features to be used by the machine-learning classifiers can drastically reduce the system's power consumption.

The research started by developing a sound-source-localization end-to-end wearable IoT system using off-the-shelf components and the traditional data-converter approach. The system was designed to tackle a contemporary problem. We developed a warning system embedded in a headset form factor able to detect and localized approaching vehicles and prevent accidents involving distracted pedestrians in busy urban areas. A segmented architecture was designed: We connected multiple

MEMS microphones distributed on the headset to a front-end unit that extracted the features for sound-source localization and transmitted the data to a smartphone via Bluetooth. The machine-learning algorithms running on the smartphone detected the approaching vehicle. To fully develop the motivated system, we also worked on detecting the presence of a vehicle. For that, we used the smartphone microphone, developing new spectral features and machine-learning classifiers the detection. Overall, the system reliably detected and localized approaching vehicles and provided the feedback to the user in time.

After developing a sound-source-localization system using traditional feature-extraction techniques, we choose to investigate the analog-to-feature data converter approach to extract the intersignal time delay. We presented a sub-100nW, three-channel time-delay-to-digital converter that combines the low-complexity of the adaptive time-delay estimator with the theoretical analysis and robustness of cross-correlation-based approaches. The time-delay-to-digital converter used one-bit quantizers in a negative-feedback architecture to search for the peak of the polarity-coincidence correlation function. We extensively analyzed the system to understand its detailed operation, resulting in a multidomain behavioral model that can be used to quickly predict the solution's performance. The time-delay-to-digital converter was built in ASIC and used to implement a much more power efficient sound-source-localization system, lowering the feature-extraction power consumption from milliwatt to nanowatt, as illustrated in Fig. 5.10.

After the successful demonstration of the analog-to-feature converter in the ex-

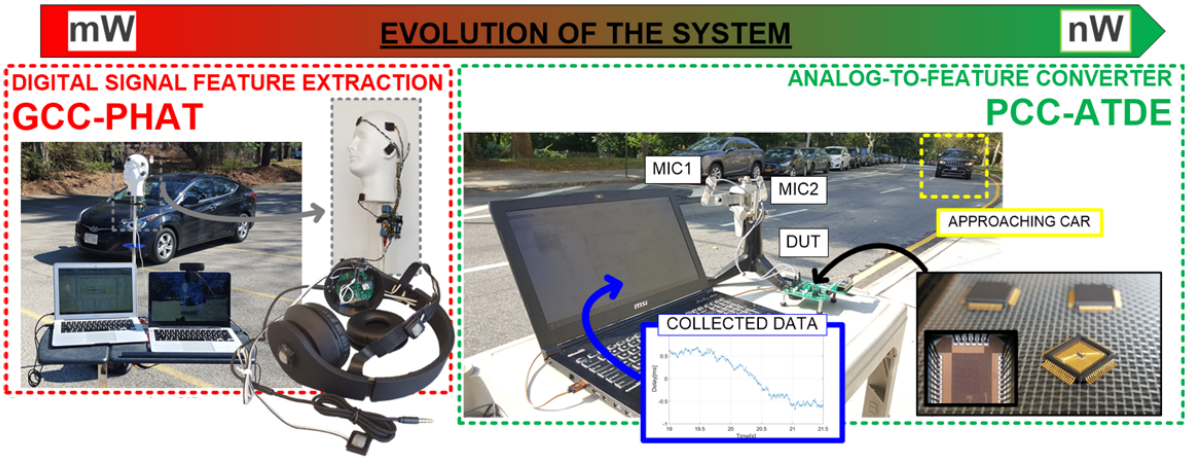

Figure 5.10: Evolution of the sound-source-localization systems presented in this work. Chapter 3 is on the left, with an off-the-shelf components and traditional digital-signal processing solution and a milliwatt-range power consumption; On the right, is the analog-to-feature solution that uses an ultra-low-power PCC-ATDE ASIC to drop the power consumption to nanowatts.

traction of the intersignal time delay, the research focused on detecting the envelope of the audio signal. For that, we presented a sub-10nW one-channel EDC. The EDC combines a SC envelope detector and a low-frequency, ultra-low-power analog-to-digital converter. To cascade these blocks efficiently, we presented a ping-pong scheme where the capacitor is shared between the envelope detector and the ADC. The scheme removed the need for a buffer and allowed the ultra-low-power implementation. This work is still ongoing, more characterization is still needed, but the prototype's initial results are encouraging.

Future work to extend this research can be done to fully integrate the time-delay-to-digital converter and the EDC to the machine-learning processing unit in a single ASIC. Most of the challenge in applying the presented ASIC in real system is in interfacing the signals across multiple platforms. The power consumption on these translations can reduce the benefits of the approach. Having all the signal-processing

units coexisting on a single ASIC could significantly reduce the power and size of the system. Another research vector from this work is to use the same analog-to-feature logic at the transducer driver level. In this work, we used off-the-shelf MEMS microphones and preamplifers for easy integration, but those are overkill for the time-delay-to-digital converter for instance, where we only cared about the polarity of the sound wave. Acoustic transducers with much poorer specifications could have been designed for in the circuit, saving even more power. This work ultimately pushes the understanding of the complete signal-processing flow before setting the specifications of the front-end blocks as the way to break power barriers and enable IoT applications.

Bibliography

[1] Joshua Adkins and Prabal Dutta. "Monoxalyze: Verifying smoking cessation with a keychain-sized carbon-monoxide breathalyzer." In: *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM. 2016, pp. 190–201.

[2] Massimo Alioto. "Ultra-low power VLSI circuit design demystified and explained: A tutorial." In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 59.1 (2012), pp. 3–29.

[3] P. Allen. "A model for slew-induced distortion in single-amplifier active filters." In: *IEEE Transactions on Circuits and Systems* 25.8 (1978), pp. 565–572.

[4] Jithendar Anumula et al. "An event-driven probabilistic model of sound source localization using cochlea spikes." In: *IEEE International Symposium on Circuits and Systems*. IEEE. 2018, pp. 1–5.

[5] Pradeep K. Atrey, Namunu C. Maddage, and Mohan S. Kankanhalli. "Audio based event detection for multimedia surveillance." In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2006, pp. V-813–V-816.

[6] Dimitra Azariadi et al. "ECG signal analysis and arrhythmia detection on IoT wearable medical devices." In: *5th International Conference on Modern Circuits and Systems Technologies*. IEEE. 2016, pp. 1–4.

[7] Komail M.H. Badami et al. "A 90 nm CMOS, 6µW power-proportional acoustic sensing frontend for voice activity detection." In: *IEEE Journal of Solid-State Circuits* 51.1 (2015), pp. 291–302.

[8] Carolina Tripp Barba et al. "Smart city for VANETs using warning messages, traffic statistics and intelligent traffic lights." In: *Intelligent Vehicles Symposium*. IEEE. 2012, pp. 902–907.

[9] Billur Barshan and Roman Kuc. "A bat-like sonar system for obstacle localization." In: *IEEE Transactions on Systems, Man and Cybernetics* 22.4 (1992), pp. 636–646.

[10] Durand R. Begault and Leonard J. Trejo. *3-D Sound for Virtual Reality and Multimedia*. San Diego, CA: Academic Press Professional, 2000.

[11] Massimo Bertozzi et al. "Stereo vision-based vehicle detection." In: *IEEE Intelligent Vehicles Symposium*. 2000, pp. 39–44.

[12] Nikhil Bhave and Preeti Rao. "Vehicle engine sound analysis applied to traffic congestion estimation." In: *Proceedings of the International Symposium on Computer Music Modeling and Retrieval and Frontiers of Research on Speech and Music*. 2011, pp. 59–63.

[13] Michael Brandstein and Darren Ward. *Microphone arrays: Signal processing techniques and applications*. New York: Springer Science & Business Media, 2013.

[14] Leo Breiman. "Random Forests." In: *Machine Learning* 45.1 (2001), pp. 5–32.

[15] Nam Bui et al. "PhO2: Smartphone based blood oxygen level measurement systems using near-IR and RED wave-guided light." In: *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 2017, Article 26.

[16] Jin Cao et al. "Fast authentication and data transfer scheme for massive NB-IoT devices in 3GPP 5G network." In: *IEEE Internet of Things Journal* (2018), pp. 1561–1575.

[17] Z.J. Chong et al. "Synthetic 2D LIDAR for precise vehicle localization in 3D urban environment." In: *IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 1554–1559.

[18] Chloé Clavel, Thibaut Ehrette, and Gaël Richard. "Events detection for an audio-based surveillance system." In: *IEEE International Conference on Multimedia and Expo*. IEEE. 2005, pp. 1306–1309.

[19] Corinna Cortes and Vladimir Vapnik. "Support-vector networks." In: *Machine Learning* 20.3 (1995), pp. 273–297.

[20] Jan Craninckx and Geert Van der Plas. "A 65fJ/conversion-step 0-to-50MS/s 0-to-0.7 mW 9b charge-sharing SAR ADC in 90nm digital CMOS." In: *IEEE International Conference on Solid-State Circuits, Digest of Technical Papers*. IEEE. 2007, pp. 246–600.

[21] Chacko John Deepu, Chun-Huat Heng, and Yong Lian. "A hybrid data compression scheme for power reduction in wireless sensors for IoT." In: *IEEE Transactions on Biomedical Circuits and Systems* 11.2 (2016), pp. 245–254.

[22] Joseph H. DiBiase, Harvey F. Silverman, and Michael S. Brandstein. "Robust localization in reverberant rooms." In: *Microphone Arrays*. New York: Springer, 2001, pp. 157–180.

[23] Hani Esmaeelzadeh and Sudhakar Pamarti. "18.4 A 0.55 nW/0.5 V 32 kHz crystal oscillator based on a DC-only sustaining amplifier for IoT." In: *IEEE International Solid-State Circuits Conference*. IEEE. 2019, pp. 300–301.

[24] Antoni Fertner and Anders Sjolund. "Comparison of various time delay estimation methods by computer simulation." In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34.5 (1986), pp. 1329–1330.

[25] Daniel de Godoy, Ji Jia, and Xiaofan Jiang. "Demo abstract: RIO-40C-A low-cost wearable sunlight exposure monitor for skincare." In: *IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation*. IEEE. 2017, pp. 295–296.

[26] Daniel de Godoy, Xiaofan Jiang, and Peter R Kinget. "A 78.2 nW 3-channel time-delay-to-digital converter using polarity coincidence for audio-based object localization." In: *IEEE Custom Integrated Circuits Conference*. IEEE. 2018, pp. 1–5.

[27] Daniel de Godoy et al. "PAWS: A wearable acoustic system for pedestrian safety." In: *IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation*. IEEE. 2018, pp. 237–248.

[28] Mayank Goel et al. "SpiroCall: Measuring lung function over a phone call." In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI '16. New York, NY, USA: ACM, 2016, pp. 5675–5685. DOI: 10.1145/2858036.2858401.

[29] David Halupka et al. "Robust sound localization in 0.18µm CMOS." In: *IEEE Transactions on Signal Processing* 53.6 (2005), pp. 2243–2250.

[30] Gerhard P. Hancke, Bruno de Carvalho Silva, and Gerhard P. Hancke Jr. "The role of advanced sensing in smart cities." In: *Sensors* 13.1 (2012), pp. 393–425.

[31] Amir A. Handzel et al. "A biomimetic apparatus for sound-source localization." In: *Proceedings of the 42nd IEEE Conference on Decision and Control*. Vol. 6. IEEE. 2003, pp. 5879–5884.

[32] Aki Harma, Martin F. McKinney, and Janto Skowronek. "Automatic surveillance of the acoustic activity in our living environment." In: *International Conference on Multimedia and Expo*. IEEE. 2005, 4–pp.

[33] Pieter Harpe et al. "A 0.20mm^2 3nW signal acquisition IC for miniature sensor nodes in 65nm CMOS." In: *IEEE Journal of Solid-State Circuits* 51.1 (2015), pp. 240–248.

[34] Syed Husain et al. "Mobile edge computing with network resource slicing for Internet-of-Things." In: *IEEE 4th World Forum on Internet of Things*. IEEE. 2018, pp. 1–6.

[35] Shyr-Long Jeng, Wei-Hua Chieng, and Hsiang-Pin Lu. "Estimating speed using a side-looking single-radar vehicle detector." In: *IEEE Transactions on Intelligent Transportation Systems* 15.2 (2014), pp. 607–614.

[36] Dongsuk Jeon et al. "An implantable 64nW ECG-monitoring mixed-signal SoC for arrhythmia diagnosis." In: *IEEE International Solid-State Circuits Conference Digest of Technical Papers*. IEEE. 2014, pp. 416–417.

[37] Seokhyeon Jeong et al. "A 12nW always-on acoustic sensing and object recognition microsystem using frequency-domain feature extraction and SVM classification." In: *IEEE International Solid-State Circuits Conference*. IEEE. 2017, pp. 362–363.

[38] Zhenhua Jia et al. "Continuous low-power ammonia monitoring using long short-term memory neural networks." In: *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2018, pp. 224–236.

[39] Jinkwon Kim et al. "Robust algorithm for arrhythmia classification in ECG using extreme learning machine." In: *Biomedical Engineering Online* 8.1 (2009), p. 31.

[40] Peter R. Kinget. *The World Is Analog*. 2014. URL: `http://circuitcellar.com/tech-the-future/kinget-the-world-is-analog/`.

[41] Tomi Kinnunen et al. "Voice activity detection using MFCC features and support vector machine." In: *International Conference on Speech and Computer*. 2007, pp. 556–561.

[42] Shashidhar G. Koolagudi and K. Sreenivasa Rao. "Emotion recognition from speech: A review." In: *International Journal of Speech Technology* 15.2 (2012), pp. 99–117.

[43] Sotiris B. Kotsiantis, I. Zaharakis, and P. Pintelas. "Supervised machine learning: A review of classification techniques." In: *Emerging Artificial Intelligence Applications in Computer Engineering* 160 (2007), pp. 3–24.

[44] Hae-Seung Lee and Charles G. Sodini. "Analog-to-digital converters: Digitizing the analog world." In: *Proceedings of the IEEE* 96.2 (2008), pp. 323–334.

[45] Jeong-Mook Lim et al. "An audio-haptic feedbacks for enhancing user experience in mobile devices." In: *IEEE International Conference on Consumer Electronics*. IEEE. 2013, pp. 49–50.

[46] Hanli Liu et al. "A 0.98 mW fractional-N ADPLL using 10b isolated constant-slope DTC with FOM of 246dB for IoT applications in 65nm CMOS." In: *IEEE International Solid-State Circuits Conference*. IEEE. 2018, pp. 246–248.

[47] Shih-Chii Liu et al. "Asynchronous binaural spatial audition sensor with $2 \times 64 \times 4$ channel output." In: *IEEE Transactions on Biomedical Circuits and Systems* 8.4 (2013), pp. 453–464.

[48] Vivek Mangal and Peter R. Kinget. "A 0.42 nW 434MHz 79.1dBm wake-up receiver with a time-domain integrator." In: *IEEE International Solid-State Circuits Conference*. IEEE. 2019, pp. 438–440.

[49] Arnaud Martin, Delphine Charlet, and Laurent Mauuary. "Robust speech/non-speech detection using LDA applied to MFCC." In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE. 2001, pp. 237–240.

[50] Gregor J. McDonald et al. "Real-time vehicle identification performance using FPGA correlator hardware." In: *IEEE Transactions on Intelligent Transportation Systems* 13.4 (2012), pp. 1891–1895.

[51] Annamaria Mesaros et al. "Acoustic event detection in real life recordings." In: *18th European Signal Processing Conference*. IEEE. 2010, pp. 1267–1271.

[52] Robert G. Meyer. "Low-power monolithic RF peak detector analysis." In: *IEEE Journal of Solid-State Circuits* 30.1 (1995), pp. 65–67.

[53] S. Molau et al. "Computing Mel-frequency cepstral coefficients on the power spectrum." In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE. 2001, pp. 73–76.

[54] Bruce Moore. "Principal component analysis in linear systems: Controllability, observability, and model reduction." In: *IEEE Transactions on Automatic Control* 26.1 (1981), pp. 17–32.

[55] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. "Voice recognition algorithms using Mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques." In: *Journal of Computing* 2 (2010), pp. 138–143.

[56] Anh Nguyen et al. "A lightweight and inexpensive in-ear sensing system for automatic whole-night sleep stage monitoring." In: *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM. 2016, pp. 230–244.

[57] Madison Park. *Injuries while walking with headphones tripled, study finds*. Ed. by CNN. [Online]. Jan. 2012. URL: `http://thechart.blogs.cnn.com/2012/01/16/injuries-while-walking-with-headphones-triple-study-finds/`.

[58] Christopher J. Plack. *The Sense of Hearing*. London: Lawrence Erlbaum Associates, 2005.

[59] Jose Portelo et al. "Non-speech audio event detection." In: *Acoustics, Speech and Signal Processing*. IEEE. 2009, pp. 1973–1976.

[60] Jennifer Rowley. "The wisdom hierarchy: Representations of the DIKW hierarchy." In: *Journal of Information Science* 33.2 (2007), pp. 163–180.

[61] Rahul Sarpeshkar. "Analog versus digital: Extrapolating from electronics to neurobiology." In: *Neural Computation* 10.7 (1998), pp. 1601–1638.

[62] Rahul Sarpeshkar et al. "An ultra-low-power programmable analog bionic ear processor." In: *IEEE Transactions on Biomedical Engineering* 52.4 (2005), pp. 711–727.

[63] Mingoo Seok, Dennis Sylvester, and David Blaauw. "Optimal technology selection for minimizing energy and variability in low voltage applications." In: *Proceedings of the 2008 International Symposium on Low Power Electronics & Design*. ACM. 2008, pp. 9–14.

[64] Hyeon-Cheon Seol et al. "An EMG readout front-end with automatic gain controller for human-computer interface." In: *IEEE Biomedical Circuits and Systems Conference*. IEEE. 2013, pp. 170–173.

[65] Katherine Shaver. "Safety experts to pedestrians: Put the smartphones down and pay attention." In: *Washington Post* (Sept. 2014). URL: https://www.washingtonpost.com/local/trafficandcommuting/safety-experts-to-pedestrians-put-the-smartphones-down-and-pay-attention/2014/09/19/278352d0-3f3a-11e4-9587-5dafd96295f0_story.html?.

[66] Amit Sheth. "Internet of Things to smart IoT through semantic, cognitive, and perceptual computing." In: *IEEE Intelligent Systems* 31.2 (2016), pp. 108–112.

[67] E. Shi et al. "A 9.6nW, 8-bit, 100S/s envelope-to-digital converter for respiratory monitoring." In: *IEEE Transactions on Circuits and Systems II: Express Briefs* (2019), Advance online publication. DOI: 10.1109/TCSII.2019.2922661.

[68] Kang G. Shin and Yu-Chih Tung. *Real-Time Warning for Distracted Pedestrians with Smartphones*. US Patent App. 14/865,262. Sept. 2015.

[69] H.C. So and P.C. Ching. "Comparative study of five LMS-based adaptive time delay estimators." In: *IEE Proceedings-Radar, Sonar and Navigation* 148.1 (2001), pp. 9–15.

[70] John A. Stankovic. "Research directions for the Internet of Things." In: *IEEE Internet of Things Journal* 1.1 (2014), pp. 3–9.

[71] Zehang Sun, George Bebis, and Ronald Miller. "On-road vehicle detection: A review." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.5 (2006), pp. 694–711.

[72] Bahareh Taji et al. "Impact of skin–electrode interface on electrocardiogram measurements using conductive textile electrodes." In: *IEEE Transactions on Instrumentation and Measurement* 63.6 (2013), pp. 1412–1422.

[73] Andrey Temko et al. "CLEAR evaluation of acoustic event detection and classification systems." In: *International Evaluation Workshop on Classification of Events, Activities and Relationships*. Springer. 2006, pp. 311–322.

[74] Stefan Tertinek, James P. Gleeson, and Orla Feely. "Statistical analysis of first-order bang-bang phase-locked loops using sign-dependent random-walk theory." In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 57.9 (2010), pp. 2367–2380.

[75] Bert Van den Broeck et al. "Time-domain generalized cross correlation phase transform sound source localization for small microphone arrays." In: *5th European Education and Research Conference*. IEEE. 2012, pp. 76–80.

[76] J. Hf. Van Vleck and David Middleton. "The spectrum of clipped noise." In: *Proceedings of the IEEE* 54.1 (1966), pp. 2–19.

[77] Marian Verhelst and Ahmad Bahai. "Where analog meets digital: Analog-to-information conversion and beyond." In: *IEEE Solid-state circuits magazine* 7.3 (2015), pp. 67–80.

[78] Weihua Wang. "Reach on Sobel operator for vehicle recognition." In: *Artificial Intelligence, International Joint Conference on*. IEEE. 2009, pp. 448–451.

[79] Stuart N. Wooters, Benton H. Calhoun, and Travis N. Blalock. "An energy-efficient subthreshold level converter in 130nm CMOS." In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 57.4 (2010), pp. 290–294.

[80] Lin Yang, Wei Wang, and Qian Zhang. "Secret from muscle: Enabling secure pairing with electromyography." In: *SenSys*. 2016, pp. 28–41.

[81] Minhao Yang et al. "A 0.5V 55uW 64 × 2-channel binaural silicon cochlea for event-driven stereo–audio sensing." In: *IEEE Journal of Solid-State Circuits* 51.11 (2016), pp. 2554–2569.

[82] Dai Zhang, Ameya Bhide, and Atila Alvandpour. "A 53-nW 9.1-ENOB 1-kS/s SAR ADC in 0.13-µm CMOS for medical implant devices." In: *IEEE Journal of Solid-State Circuits* 47.7 (2012), pp. 1585–1593.

[83] Shengyan Zhou et al. "Road detection using support vector machine based on online learning and evaluation." In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2010, pp. 256–261.

[84] Xiaodan Zou et al. "A 1-V 450-nW fully integrated programmable biomedical sensor interface chip." In: *IEEE Journal of Solid-State Circuits* 44.4 (2009), pp. 1067–1077.