Design and Implementation of Acoustic Source Localization on a Low-Cost IoT Edge Platform

Germán Fabregat, Jose A. Belloch*, *Member, IEEE,* José M. Badía, and Maximo Cobos, *Senior Member, IEEE*

***Abstract*—The implementation of algorithms for acoustic source localization on edge platforms for the Internet of Things (IoT) is gaining momentum. Applications based on acoustic monitoring can greatly benefit from efficient implementations of such algorithms, enabling novel services for smart homes and buildings or ambient-assisted living. In this context, this work proposes extreme low-cost sound source localization system composed of two microphones and the low power microcontroller module ESP32. A Direction-Of-Arrival (DOA) algorithm has been implemented taking into account the specific features of this board, showing excellent performance despite the memory constraints imposed by the platform. We have also adapted off-the-shelf low-cost microphone boards to the input requirements of the ESP32 Analog-to-Digital Converter. The processing has been optimized by leveraging in parallel both cores of the microcontroller to capture and process the audio in real time. Our experiments expose that we can perform real-time localization, with a processing time below 3.3 ms.**

***Index Terms*—ESP32, Embedded Systems, Sound Source Localization, DOA Estimation.**

I. Introduction

Sound in general, and speech in particular, constitutes a natural and intuitive modality for the development of human-machine interfaces. The use of location information and its potential for the development of ambient intelligence applications has significantly promoted the design of local positioning systems during the last decade [1]. As a representative example, in Ambient-Assisted Living (AAL) applications, location information is really valuable, since knowing the position of the user enables the implementation of services that may make the living environment easier, safer or more comfortable [2]. Moreover, the use of sound in AAL systems is significantly beneficial, since it has been shown that microphones can be easily integrated into existing living environments and they tend to be smaller, lightweight, and less intrusive than video cameras [2].

From the hardware perspective, audio capturing and processing is considered a challenging problem, which involves a trade-off between the complexity of the audio processing task and the hardware resources [3]. The audio monitoring process introduces some specific requirements on hardware platforms. On the one side, audio signals are normally sampled at relatively high rates, demanding large memories and high computational capabilities. On the other side, signal processing tasks should be programmed carefully to deal with the audio sampling process and to optimize the system resources properly.

Estimating the Direction-Of-Arrival (DOA) of multiple sound sources in a real scenario is a difficult task [4]. Cross-correlation-based methods have been widely applied in DOA estimation [5]. However, for small devices with close microphones, these methods usually offer poor angular resolution, motivating the use of subspace or time-frequency-based approaches [6]. When only two microphones are used, DOA estimation is usually performed by estimating interchannel amplitude and time differences, as sound arrives slightly earlier in time at the microphone that is physically closer to the source, and with somewhat greater energy [7]. Moreover, sound sources can be usually assumed to be disjoint in the time-frequency domain [7] and, thus, the analysis of temporal (phase) and amplitude differences at each time-frequency bin facilitates the identification of several simultaneously active sources. Such analysis implies an intensive use of Fast Fourier Transform (FFT) operations, which may be appropriately handled by the hardware and accommodated within its general operation without affecting other concurrent tasks.

Multiple embedded platforms have been used in Internet-Of-Things-based applications. In fact, most of them are analysed in [8] for carrying out a personalized stress detection. In [9], a wearable Internet of Things (IoT) Monitoring device uses an Arduino Due for improving battery life and false alarms at photoplethysmogram analysis. An audio-assisted system for understanding Braille symbols is developed in [10]. Most of these works make use of a Raspberry Pi for data processing. However, the energy consumption and the economic cost of the Raspberry is higher than what is expected from a low-cost and low-power system in comparison to other alternatives such as the ESP32. Especially, the study carried out in [11] points out the difficulties of implementing a real-time sound source localization system that works properly in a general-purpose platform. The authors of this work propose an Application-Specific Integrated Circuit (ASIC), which, spite of fulfilling real-time constraints, limits possible improvements related both in circuit design and in the localization algorithm.

The ESP32 is a general-purpose low-cost, low-power system-on-chip series of microcontrollers with Wi-Fi and Bluetooth capabilities and a highly integrated structure powered by a dual-core Tensilica Xtensa LX6 microprocessor. The main goal of this paper is to design and evaluate a DOA-based

J. A. Belloch is with the Depto. de Tecnología Electrónica, Universidad Carlos III de Madrid, Spain.

* Corresponding author. E-mail: jbelloc@ing.uc3m.es

J. M. Badía and G. Fabregat are with the Depto. de Ingeniería y Ciencia de Computadores, Universitat Jaume I, 12071, Castellón, Spain.

M. Cobos is with the Computer Science Department, Universitat de València, Spain, Valencia, Spain.

This work has been supported by the Spanish Government through TIN2017-82972-R, ESP2015-68245-C4-1-P and RTI2018-097045-B-C21, and by the Valencian Regional Government through PROMETEO/2019/109.

sound-source localization system using two microphones implemented on an ESP32 platform, which is considered to be an excellent option for data processing due to its performance, low-power consumption and price, as shown in [12], [13].

This manuscript presents hardware design and development details of a real-time sound source localization system using low-cost off-the-shelf components providing low-power consumption features. Specifically, the system has been implemented over a widely employed general-purpose microcontroller, facilitating its integration into IoT-based acoustic applications. To this end, the edge computing paradigm is followed, where all the processing is carefully accommodated within the system itself, avoiding any unnecessary communication with other components or with the cloud. Therefore, the system can be successfully exploited without incurring problems related to data protection regulations. Moreover, the localization system, besides running in real-time, only makes use of a small part of the computational power of the processors, leaving room for other simultaneous processing tasks. Finally, it should be emphasized that sound-source localization systems for industrial applications have traditionally made use of dedicated hardware [14]. Thus, the proposed system may be seamlessly incorporated with little cost into a wide range of existing applications.

The contributions of the paper are: 1) a detailed system design and implementation considering the limited hardware and processing resources of the ESP32 and the microphone subsystem; 2) a comparative evaluation of a time-frequency-based DOA estimation system implemented on a PC and in ESP32, considering different programming languages and resources; and 3) an evaluation of the localization accuracy of the final system in a real office scenario. Besides, we present a solution to several constraints of the ESP32 in order to process two audio channels concurrently.

II. DOA Estimation Model

This section presents the signal model and localization method that motivates our proposed implementation.

A. Signal Model

Consider two microphones, $m = 1, 2$ and N sound sources located on the azimuth plane, each one emitting from an angle $\theta_n \in [0, \pi]$ with respect to the center of the two microphones (see Figure 1). The discrete-time signals captured by the sensors can be expressed as

$$x_m[t] = \sum_{n=1}^{N} s_n[t] * h_{mn}[t], \quad m = 1, 2, \tag{1}$$

where $s_n[t]$ is the signal emitted by the n-th source and $h_{mn}[t]$ is the impulse response of the acoustic channel from the n-th source to the m-th microphone. Since location information is contained on the direct-path delays between the two microphones, we omit possible acoustic reflections and consider a simplified free-field model where the impulse responses are given by $h_{mn} = a_{mn}\delta(t - \tau_{mn})$, where a_{mn} is a scalar amplitude decay factor and τ_{mn} is a direct-path delay. The microphone signals are therefore given by

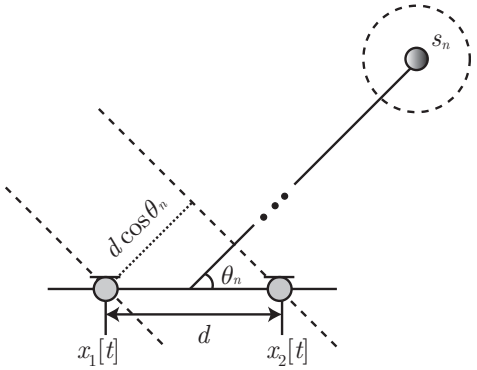

Fig. 1. DOA estimation geometry.

$$x_m(t) = \sum_{n=1}^{N} a_{mn} s_n[t - \tau_{mn}], \quad m = 1, 2. \tag{2}$$

Equivalently, in the Short-Time Fourier Transform (STFT) domain, the microphone signals are

$$X_m[k, l] = \sum_{n=1}^{N} a_{mn} S_n[k, l] e^{-j\omega_k \tau_{mn}}, \quad m = 1, 2. \tag{3}$$

where $S_n(k, l)$ denotes the coefficient of the n-th source at frequency bin k and time frame l, and $j = \sqrt{-1}$. The term $\omega_k = k\frac{2\pi}{K}$, $k = 0, \ldots, K-1$ is the normalized digital angular frequency considering K FFT points.

B. DOA Estimation

Assuming plane-wave incidence from an angle θ_n and an inter-microphone distance d, the time delay in seconds between the two microphones is given by

$$\Delta\tau_n = \frac{1}{f_s}(\tau_{1n} - \tau_{2n}) = \frac{d}{c}\cos(\theta_n), \tag{4}$$

where c is the sound propagation speed ($\approx$ 343 m/s) and f_s is the sampling frequency of the acquired signals. The corresponding phase difference between the two microphones at a frequency ω_k is therefore given by

$$\angle\left(\frac{X_1(k, l)}{X_2(k, l)}\right) = \omega_k \frac{f_s d}{c}\cos(\theta_n), \tag{5}$$

where $\angle(\cdot)$ denotes the phase of a complex number.

As a result, the cosine of the DOA angle can be estimated for each time-frequency point (k, l) as

$$\widehat{\cos(\theta)}(k, l) = \frac{1}{f_s d}\frac{c}{\omega_k}\angle\left(\frac{X_1(k, l)}{X_2(k, l)}\right) \tag{6}$$

For phase differences having a magnitude higher than π, the resultant wrapping makes such difference ambiguous. Therefore estimates corresponding to $\omega_k > \pi\frac{c}{df_s}$ are discarded, i.e. the maximum non-ambiguous FFT coefficient is

$$k_{\max} = \left\lfloor \frac{K}{2}\frac{c}{d \cdot f_s} \right\rfloor, \tag{7}$$

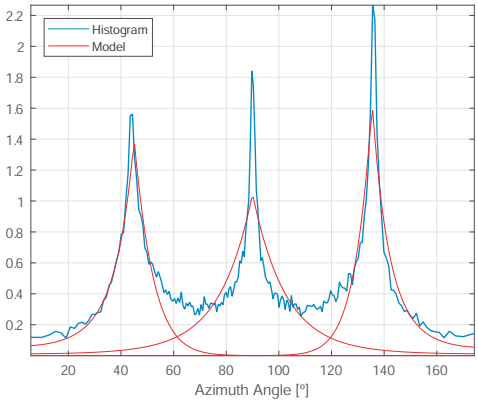

Fig. 2. An example histogram of DOA estimates with three sources.

where $\lfloor \cdot \rceil$ denotes the rounding operator. Note that small microphone distances are needed in order to have a wider range of valid frequencies available. The DOA of the multiple sources can be estimated by picking the peaks of a histogram of DOA estimates over all frequency bins. Alternatively, more sophisticated approaches based on model fitting, such as the described in [6] using a Laplacian mixture model, can also be used for improved accuracy. An example histogram for an example of the sources and its corresponding fitted model is shown in Figure 2. All our tests have been performed using only one sound source and taking its position as the maximum peak of the histogram obtained with the DOA algorithm.

III. Suitability and Challenges of the Platform

The processor of the ESP32 is a 32-bit dual-core Tensilica Xtensa LX6, able of executing up to 600 Mega Instructions Per Second (MIPS). The ESP32 modules include 448 KB of Read-Only Memory (ROM), 520 KB of Static Random Access Memory (SRAM), and up to 16 MB of external Quad Serial Peripheral Interface (QSPI) flash that is cached for increased performance. The system is clocked at up to 240 MHz and includes a low frequency, ultra-low-power coprocessor able to process certain events and wake the system from the deep-sleep mode. The microcontroller features of the ESP32 are completed with a flexible General-Purpose Input/Output (GPIO) subsystem and an important list of on-chip peripherals and sensors, including a Digital-to-Analog Converter (DAC) and two Analog-to-Digital Converter (ADC) with up to 18 channels [15]:

Espressif provides the IoT Development Framework (IDF) Integrated Development Environment (IDE) to easily program ESP32 applications as FreeRTOS-based applications. Two of the characteristics of the ESP32 make it an adequate target for implementing the DOA algorithm. Two ADC converters are required, as audio signals from two microphones have to be captured and used by the algorithm in real-time. Note that having several channels with a unique ADC is not possible because the signals would be captured at different instants, delaying the conversion time. The second is the possibility to use one core to perform the conversion task, as Direct Memory Access (DMA) transport is not possible for both channels, while the other core implements all the processing required by the DOA algorithm.

Several important design challenges had to be solved in order to implement the DOA algorithm on low-cost components. Firstly, the DOA is a computation-intensive and memory-consuming algorithm. The ESP32 is a micro-controller with a small amount of RAM and a limited computing power. Secondly, our audio acquisition subsystem uses two low-cost electret microphones and the dual ADC of the ESP32. The quality of the signal provided by the former and the sample rate and accuracy of the latter are a serious issue that will prove or deny the robustness of our implementation. To state this challenge suffice to say that the LyraT audio board series, built by Espressif, the company that developed the ESP32 micro-controller, does not rely on the ESP32 ADC but include external hardware codecs to sample the input audio.

IV. Implementation Issues

To evaluate the performance of the ESP32 we focused first on the FFT, the most time-consuming step of the DOA algorithm. We modified the FastFFT algorithm by Reliable Software used on its frequency analyzer application to obtain a C implementation that worked on a fixed-size input buffer. This implementation was first compiled with MinGW Studio using the gcc compiler and tested for correctness on a PC. Once the FFT algorithm was proved correct it was ported to the ESP32 and compiled using the IDF Software Development Kit (SDK). The resultant algorithm offers meaningful performance since it precomputes once the exponential coefficient vectors and the butterfly reordering indexes, and then it uses them as required to perform the actual FFT call. The drawback of this approach is the large amount of memory used. Our first tests shown that the maximum size of the buffer on which the transform can be performed is 1024 single precision elements. With this size and taking into account that the Floating Point Units (FPUs) on the ESP32 work with single precision, 32 bit floats, the amount of RAM consumed by the coefficient array is 88 KB. The ESP32 SDK linker allocates 256 KB for static data, so only 168 KB remain available for data structures and buffers of the DOA code. Then we measured the performance of our basic implementation of the FFT using synthetic data. Our results show that the time to compute a 1024-points FFT is around 1 ms, and thus, there is enough margin to perform computation between successive buffer acquisitions: around 50 ms at 22 Kilo samples per second (Ksps) or 25 ms at 44 Ksps.

The following tests were performed to test ADC speed and quality, as it is not documented on the technical manuals of the ESP32. They showed that both ADCs can capture more than 40 Ksps but the conversion time is not stable and does not always reach 44 Ksps. Unfortunately, the ESP32 is not able to use DMA to carry data directly from both ADCs to the in-memory buffers and end of conversion interrupts are not implemented. Hence, to use the ADCs one of the cores has to be in charge of ADC management and synchronize the conversion using the timers of the system. Given that the required sps is less than the maximum of 40 Ksps, the sample rate was set to 22 Ksps. During these tests we had

to modify the Integrated Development Environment (IDE) to provide non-blocking start of conversion functions and separated data collecting ones to allow for simultaneous data acquisition in both ADC. When these tests were performed, we put all the pieces together to program a simple system that uses both cores in parallel. One core reads from the two ADCs while the other performs FFT and other computations. The system duplicated buffers and a ping-pong scheme for continuous flow of conversion and processing and no memory issues arose. These tests demonstrated enough quality on the signals captured by the ADCs to port the DOA algorithm with confidence.

A. Porting the DOA algorithm

Once the basic audio processing system was ported and tested, the next step was to port the DOA algorithm introduced in [6]. The original algorithm was programmed in Octave and all the input audio data was taken from a wav file and processed as a matrix. So we firstly modified the original DOA as follows:

1) The program produces the histogram used to obtain the most likely DOA angle, without applying the subsequent Laplacian fitting.
2) Matrix calculation is not possible, as the system must work with real-time audio data. Therefore we transformed it to vector calculations, on-the-fly processing both captured buffers, one for each audio channel. The coherence-based selection phase of the algorithm is then computed using a circular buffer of vectors.

We implemented a C version of the code and tested it on a PC using different buffer sizes, overlapping factors and sampling rates obtaining similar results to those observed with the Octave version. Finally, we tested the code using single precision floating point elements on the ESP32. The tests shown that the time consumed by the DOA algorithm is less than 3.3 ms, while filling a buffer of 1024 samples at 22 Ksps is around 50 ms, therefore the ESP32 can effectively be used to perform these tasks in real time and possibly add more computation to improve the results.

The final step was to evaluate the system in real time with real audio data on the ESP32. The development just required to use the two ADCs audio capture task that was tested in our first experiments, that ran on a core, and use the fully tested DOA algorithm on the other core. Simple synchronization variables were used to correctly implement the ping-pong structure of the system that guaranteed no concurrency problems. The first results obtained with the complete ESP32 system showed a random behavior in DOA estimation. The cause of the problems was found to be in the microphones and the adapting circuitry, as discussed in the following section.

B. The microphone subsystem

In order to keep low the cost and complexity of the system, off-the-shelf, low-cost amplified electret microphones were used. The basic boards, one per channel, included a microphone, a MAX9812 amplifier, a voltage regulator an a few discrete components (see Figure 3). The output of the boards was an AC signal centered on 0 volts. As the ADCs of the ESP32 work in DC the output capacitors of the boards were bypassed using thus the output of the amplifier. The ESP32 can configure the ADC attenuation rate to be 0dB (range 0 to 1.1V), 2.5dB (0 to 1.5V), 6dB (0 to 2.2V) or 11dB (0 to 3.9V). When bypassing the capacitor, the signal given by the MAX9812 is centered at 1.5V, so attenuation of 6 or 11 dB had to be used. If the amplitude of the signal is low, the final resolution of the converted data is not sufficiently good, leading to poor results. Then an external output capacitor and an adjustable high-impedance potentiometer were added to center the output level to 0.5V (see Figure 3).

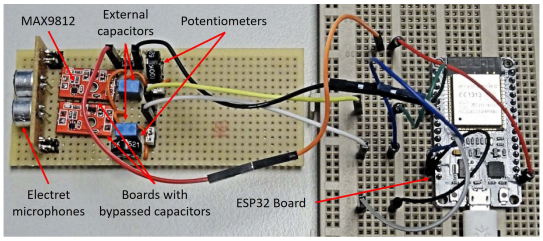

Fig. 3. ESP32 SoC and the electret microphones connection.

V. Tests and Results

For validation purposes we will compare first the result of four different versions of the DOA algorithm, all of them using single precision floating point variables and overlapping half of the samples: Two Octave scripts using 4096 samples with 44 Ksps and 1024 samples with 22 Ksps; and two C versions with the same configurations, one running on a PC and the last on the ESP32 platform. To perform the comparison, we considered one-minute of real sound recorded from three different angles. To this end, we used the previously described simple audio system connected to the line-in input of a PC. All the experiments were conducted by using the same stored audio recording, bypassing the ADCs of the different subsystems to properly validate the DOA algorithm over the different implementations and platforms. These audio streams were sliced into 12 chunks of approximately 5 seconds each, and fed into the different versions of the algorithm on the PC and on the actual ESP32 version. Figure 4 shows the average results obtained with the 12 chunks for each implementation. As expected, the results with the large-size buffers do not depend on the programming language, providing the same results in the three positions of the source. When we decrease the buffer size and the sample rate, the results are not exactly the same. Nonetheless, despite the observed slight differences, the results agree with the actual source direction, validating the ESP32 implementation.

After this initial experiment, we present the results obtained on the ESP32 board implementation using real-time audio on a defined grid of azimuth angles with a separation of 10 degrees. All the tests were performed using a buffer size of 1024 samples with 50% overlap, and a sample rate of 22 Ksps. The experiments were conducted in a laboratory without any special acoustic treatment, placing a small loudspeaker at a

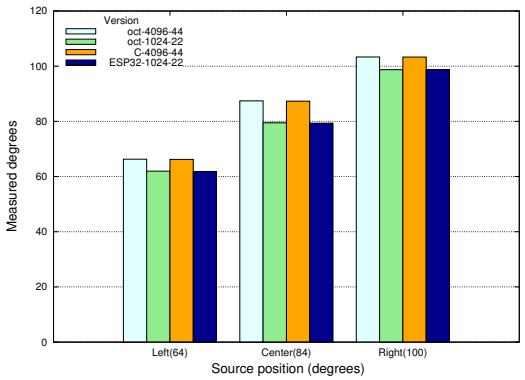

Fig. 4. Comparison of different versions of the algorithm using pre-recorded sounds on three source positions.

distance of 2m from the microphones, free of obstacles. The test consisted in reproducing segments of a vocal song with broadband spectral characteristics from each angle. Figure 5 shows the average results of 10 measurements of 5 seconds in each audio source position. Although the results were consistent with the change in direction, extreme angles were underestimated, motivating a first-order regression-based correction: $\hat{\theta} = 2\theta - 122.82$, where θ is the algorithm output before correction and $\hat{\theta}$ the final estimate after correction. These results confirm that relatively good sound localization features can be obtained from a low-cost and low-power system based on the ESP32, even when low-quality hardware limits the attainable accuracy. This outcome enables the development of innovative IoT-based applications making use of sound analysis and localization.

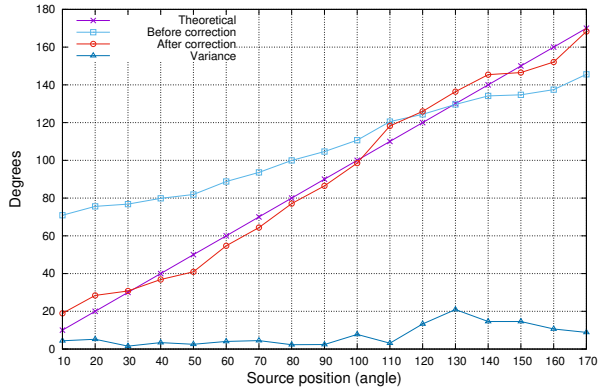

Fig. 5. Localization results using real audio varying the source position in increments of 10 degrees.

VI. Conclusions

This paper presented a real-time sound-source localization system implemented on the ESP32 microcontroller. The system was carefully designed to provide a low-cost and low-power solution for IoT-based applications, showing the potential of emerging platforms and their processing capabilities for developing context-aware services based on sound analysis. On the one hand, as we used cheap microphones and amplifiers, we had to apply slight modifications to the hardware that allowed the full exploitation of the platform. On the other hand, the processing was optimized to leverage both cores of the ESP32 in parallel: one to capture the two ADCs audio streams and the other to run the localization algorithm. Moreover, in order to overcome the problems imposed by the scarce memory of the platform, we modified the algorithm using a circular buffer of vectors to process the two audio buffers on-the-fly. The results demonstrated that, despite the hardware constraints, the system was able to estimate with relatively good accuracy the DOA of a broadband acoustic source. Future work will focus on combining multiple ESP32 devices in a distributed system to estimate the absolute positions of acoustic sources in the monitored space. Finally, it is worth to remark that the ideas and adaptations described in this paper can be extended to other problems in the IoT field.

References

[1] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 829–846, April 2018.

[2] M. Cobos, J. Perez-Solano, and L. Berger, "Acoustic-based technologies for ambient assisted living," in *Introduction to Smarte eHealth and eCcare Technologies*, S. Merilampi and A. Sirkka, Eds. Boca Raton, FL, USA: Taylor & Francis Group, 2016, ch. 9, pp. 159–180.

[3] J. A. Belloch, A. González, A. M. Vidal, and M. Cobos, "On the performance of multi-GPU-based expert systems for acoustic localization involving massive microphone arrays," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5607–5620, 2015.

[4] D. Pavlidi, A. Griffin, M. Puigt, and A. Mouchtaris, "Real-time multiple sound source localization and counting using a circular microphone array," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2193–2206, 2013.

[5] C. Knapp and G. Carter, "The generalized correlation method for estimation of time delay," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 4, pp. 320–327, August 1976.

[6] M. Cobos, J. J. Lopez, and D. Martinez, "Two-microphone multi-speaker localization based on a Laplacian Mixture Model," *Digital Signal Processing*, vol. 21, no. 1, pp. 66 – 76, 2011.

[7] O. Yilmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *IEEE Transactions on Signal Processing*, vol. 52, no. 7, pp. 1830–1847, July 2004.

[8] N. Attaran, A. Puranik, J. Brooks, and T. Mohsenin, "Embedded Low-Power Processor for Personalized Stress Detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 12, pp. 2032–2036, Dec 2018.

[9] S. Vadrevu and M. S. Manikandan, "Real-Time PPG Signal Quality Assessment System for Improving Battery Life and False Alarms," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 11, pp. 1910–1914, Nov 2019.

[10] M. M. Garcillanosa et al., "Audio-assisted standalone microcontroller-based braille system tutor for grade 1 braille symbols," in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2016, pp. 439–442.

[11] J. Jungdong et al., "Real-time sound localization using generalized cross correlation based on 0.13 um cmos process," *Journal of Semiconductor Technology and Science*, vol. 14, no. 2, pp. 175–183, 2014.

[12] A. Maier, A. Sharp, and Y. Vagapov, "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things," in *2017 Internet Technologies and Applications (ITA)*, Sep. 2017, pp. 143–148.

[13] I. Allafi and T. Iqbal, "Design and implementation of a low cost web server using ESP32 for real-time photovoltaic system monitoring," in *2017 IEEE Electrical Power and Energy Conference (EPEC)*, Oct 2017, pp. 1–5.

[14] C. Rascon and I. Meza, "Localization of sound sources in robotics: A review," *Robotics and Autonomous Systems*, vol. 96, pp. 184 – 210, 2017.

[15] Expressif Systems, "ESP32 Technical Reference Manual. Version 4.0," *Expressif Inc.*, 2018.