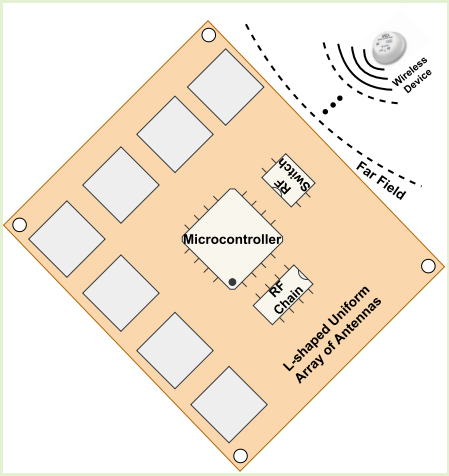
Efficient Embedded Fixed-Point Direction of Arrival Method

Tiago Troccoli, Juho Pirskanen, Jorge Morte, Aleksandr Ometov, *Senior Member, IEEE*, Elena Simona Lohan, *Senior Member, IEEE*, Ville Kaseva, and Jari Nurmi, *Senior Member, IEEE*

***Abstract*—Radio direction finding, traditionally used for localizing radio signal sources, has been adapted for Bluetooth to enable indoor localization of wireless devices. This adaptation is particularly relevant for achieving accurate indoor localization within Internet-of-Things (IoT) networks, especially in battery-powered and resource-limited embedded systems. However, the intricacies of implementing direction of arrival (DOA) methods in such systems, notably those lacking a floating-point unit (FPU), present significant computational challenges. This article addresses these challenges by introducing an innovative fixed-point DOA method, rooted in the estimation of signal parameters via rotation invariance techniques (ESPRIT). Diverging from traditional complex eigenvalue decomposition, our approach employs a simpler power method for DOA estimation and phase offset compensation, utilizing a straightforward trigonometric equation. It also integrates an improved carrier frequency estimator, also based on ESPRIT, which is tens of times more accurate than the conventional method of averaging phase differences. We conducted bare-metal level experiments on an nRF52840 system on chip to evaluate execution time, memory footprint, angle accuracy, and energy consumption. The fixed-point implementation demonstrated an execution time of 2.3 ms and an energy consumption of just 0.348 nWh. These figures represent a 5.9-fold increase in energy efficiency and a 4.4-fold improvement in speed compared to the conventional software-based floating-point approach while maintaining an angle accuracy ranging from nearly 2° to under 0.5°, depending on the signal-to-noise ratio. However, in IoT devices equipped with an FPU, the hardware-based floating-point technique still edges out, being 0.8 ms faster and slightly more energy efficient at 0.319 nWh.**



***Index Terms*— Array signal processing, direction of arrival (DOA), embedded systems, sensor arrays, sensor data processing.**

Nomenclature

$\mathbf{X}$	Matrix (uppercase boldface).
$\mathbf{x}$	Vector (lowercase boldface).
x	Scalar (italic).
$\mathbf{I}_N$	Identity matrix of size N.
$\mathbf{0}_N$	Zero column vector of size N.
$(\bullet)^{-1}$	Inverse.
$\overline{(\bullet)}$	Complex conjugate.
$(\bullet)^T$	Transpose.
$(\bullet)^H$	Conjugate transpose.
$\text{Im}\{\bullet\}$	Imaginary part.
$\text{Re}\{\bullet\}$	Real part.
$\mathbb{E}\{\bullet\}$	Expectation.
$(\bullet) \ll (\bullet)$	Arithmetic left shift.
$(\bullet) \gg (\bullet)$	Arithmetic right shift.
$j = \sqrt{-1}$	Imaginary unit.
$\lvert\bullet\rvert$	Absolute value.
$\text{diag}[\bullet]$	Diag operator.
$\lfloor\bullet\rfloor$	Floor function.
$\lVert\bullet\rVert$	Euclidean norm.
$\text{arctan2}(\bullet)$	Two-argument arctangent.

Manuscript received 8 January 2024; accepted 22 January 2024. Date of publication 6 February 2024; date of current version 14 March 2024. This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under Marie Skłodowska Curie Grant Agreement 956090 (Approximate Computing for Power and Energy Optimisation (APROPOS), http://www.apropos-itn.eu/). The associate editor coordinating the review of this article and approving it for publication was Dr. Waltenegus Dargie. *(Corresponding author: Tiago Troccoli.)*

Tiago Troccoli is with the Faculty of Information Technology and Communication Sciences, Tampere University, 33720 Tampere, Finland, and also with WIREPAS Ltd., 33720 Tampere, Finland (e-mail: tiago.troccolicunha@tuni.fi).

Juho Pirskanen, Jorge Morte, and Ville Kaseva are with WIREPAS Ltd., 33720 Tampere, Finland.

Aleksandr Ometov, Elena Simona Lohan, and Jari Nurmi are with the Faculty of Information Technology and Communication Sciences, Tampere University, 33720 Tampere, Finland.

Digital Object Identifier 10.1109/JSEN.2024.3361658

I. Introduction

A. Overview of Radio Direction Finding

RADIO direction finding (RDF) represents a mature subfield within array signal processing, finding applications across various domains. Its relevance spans from traditional

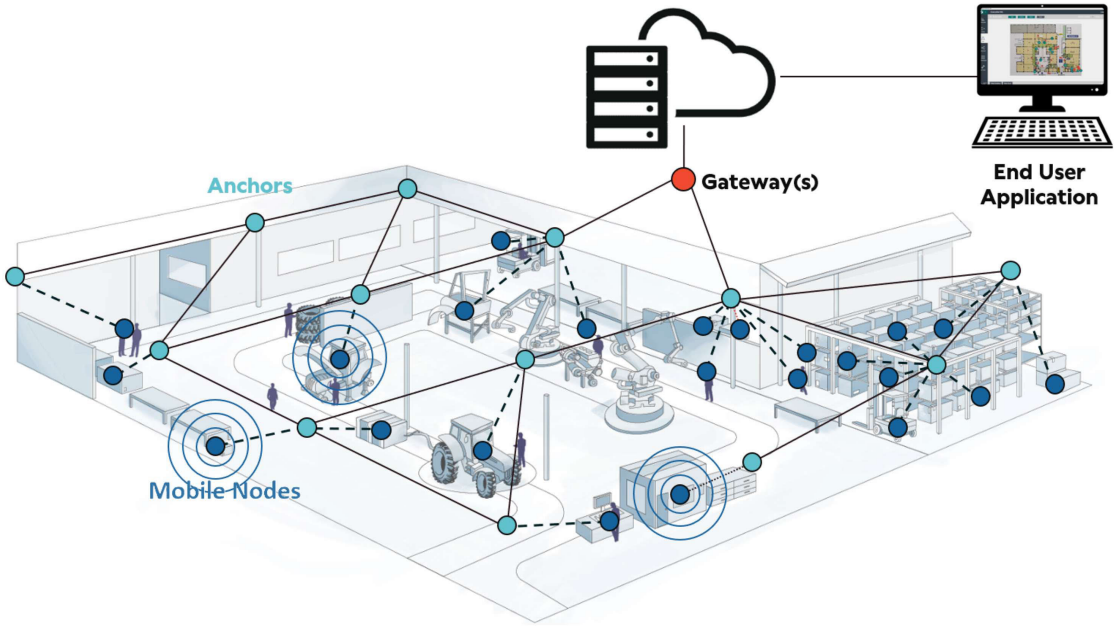


Fig. 1. Illustrative example of an IoT mesh network with an indoor positioning system in a factory.

uses in marine and aircraft navigation systems to emerging paradigms, such as joint sensing and communication in future wireless systems, automotive radar, and drone surveillance [1]. This increasing importance is driven by technological advancements and the ready availability of powerful and cost-effective multiantenna hardware platforms. In recent years, RDF has found application in wireless communication systems of the Internet of Things (IoT), particularly in Bluetooth, to locate wireless devices in indoor environments where satellite-based radio-navigation systems are inadequate. This advancement [2] aims to enhance accuracy beyond the previous indoor localization technology that relied on received signal strength (RSS).

In earlier approaches, Bluetooth Low Energy (BLE) tags transmitted radio frequency (RF) signals, and receivers estimated the distances between tags and themselves based on RSS to calculate the tags' positions. However, RSS-based localization relies on the mapping between RSS and distance, which is often inaccurate and highly sensitive to indoor environmental changes, not to mention multipath fading, signal obstruction, and interference from other devices [3]. These fluctuations make it difficult to establish a consistent mapping between RSS and distance. Even if RSS-based localization systems employ fingerprinting techniques involving a database of RSS measurements at various locations, managing and updating this fingerprint database due to the dynamic nature of indoor environments can be complex and resource-intensive. While the accuracy of such localization systems is acceptable in some cases, certain situations demand higher precision.

Notably, high-accuracy indoor localization plays a crucial role in autonomous robots and automation systems, enabling them to navigate, avoid obstacles, and perform tasks with accuracy and efficiency. Industries such as manufacturing, healthcare, and hospitality benefit from this technology to streamline operations and improve productivity, while warehousing and logistics utilize it to track and manage assets within their facilities.

The radio direction-finding feature in Bluetooth technology enables the estimation of directions of arrivals (DOAs) [4], thereby facilitating precise positioning, potentially with submeter accuracy as reported by some companies [5], [6], [7], [8]. This advancement leverages devices equipped with antenna arrays to estimate the azimuth and elevation angles of radio signals emitted from transmitters. This process aids in pinpointing the location of transmitters through techniques such as triangulation. In this context, the literature defines "anchor nodes" as devices with known locations, equipped with antenna arrays, while "mobile nodes" are those whose locations are not predetermined.

Furthermore, BLE plays a pivotal role in wireless communication within IoT networks. Here, DOA-based indoor positioning systems can be integrated into networks comprising small, battery-operated embedded systems that have limited computational power. Fig. 1 illustrates an example of an IoT mesh network incorporating a DOA-based positioning system.

B. Challenges in RDF

The inherent complexity of the DOA estimation naturally suggests cloud computing as a viable solution for indoor localization within IoT networks. However, this approach is often impractical, especially in certain structures such as mesh networks that incorporate numerous devices. In these setups, anchor nodes would be required to continuously transmit measured signal data—often hundreds of bytes—from one device to another until reaching the cloud. This process would not only rapidly drain their batteries but also contribute to increased network traffic. Another alternative could be the installation of internet cables directly to anchor nodes. While this might alleviate the cited problems, it introduces a notable

increment in infrastructure costs, making it a less favorable option. A more efficient and cost-effective solution lies in implementing the DOA method in anchor nodes. This localized processing means that only the estimated DOAs, which typically comprise a few bytes, need to be transmitted.

However, incorporating DOA methods into IoT devices poses a significant challenge due to their limited computational resources and reliance on battery power [9]. Conversely, DOA methods involve complex numerical algorithms that require substantial resources and are time-consuming, leading to rapid battery drain, long execution time, and resource starvation. In addition, IoT devices typically operate on simple real-time operating systems (RTOSs), where they concurrently handle small tasks such as sensor data acquisition and communication with other devices [10]. The execution of DOA methods within such a multithreaded environment becomes even more challenging, necessitating careful consideration of computational resource management. Consequently, the development of DOA algorithms for IoT devices requires an innovative approach that strikes a balance between resource limitations and DOA accuracy, all while mitigating its impact on battery life and maintaining a reasonable real-time system performance.

Typically, the research on DOA lacks sufficient practical consideration from the computer implementation perspective. Often, studies are conducted using multiparadigm programming languages, particularly MATLAB, which already provides a variety of prebuilt numerical functions, leading to less incentive to develop custom numerical algorithms specifically for DOA techniques. However, when these methods must be implemented on limited embedded systems, numerical algorithms must be created from scratch due to the limited or nonexistent support offered by C programming language libraries and even third-party libraries. Moreover, widely used linear algebra external libraries such as LAPACK [11] and Armadillo [12] are unsuitable for constrained embedded systems. While the Common Microcontroller Software Interface Standard (CMSIS) Software Library [13] is designed for such devices, it lacks some numerical algorithms required for DOA methods.

In addition, it is worth noting that low-cost and low-power embedded processors commonly found in IoT devices often lack a floating-point unit (FPU) [14], which is responsible for executing arithmetic operations involving floating-point numbers. Notably, only one out of four systems on chip (SoCs) manufactured by Nordic Semiconductor, featuring Bluetooth Direction Finding capability, includes an FPU [15], as shown in Table I. SoCs without an FPU tend to be more cost-effective than those equipped with one, potentially improving the feasibility of indoor localization solutions. Even in cases where an FPU is present, disabling it could serve as a means to mitigate power consumption. In both scenarios, the utilization of a DOA method that employs floating-point numbers would result in the C/C++ compiler performing floating-point arithmetic via software instead of utilizing hardware [16], [17], consequently leading to a significant increase in execution time, easily exceeding many folds. For this reason, employing fixed-point arithmetic, which uses binary integer numbers to represent fractional values, emerges as a solution. This technique circumvents the necessity for an FPU, as the processor executes integer arithmetic operations while maintaining the fractional nature of numbers.

TABLE I
SoCs With Bluetooth Direction Finding Capability

SoC	Processor	Flash Memory	RAM	Presence of FPU
nRF52833	ARM Cortex M4 64 MHz	512 KB	128 KB	Yes
nRF52811	ARM Cortex M4 64 MHz	192 KB	24 KB	No
nRF52820	ARM Cortex M4 64 MHz	256 KB	32 KB	No
nRF5340	ARM Cortex-M33 128/64 MHz	1 MB	512 KB	No

C. Motivations and Contributions

The primary motivations of our research are focused on extending battery life and reducing computational latency in constrained embedded systems that operate Bluetooth Direction Finding within IoT networks.

This article provides the following novel contributions:

1) development of an optimized DOA method named estimation of signal parameters via rotation invariance techniques (ESPRIT) specifically for Bluetooth Direction Finding using an L-shaped array of antennas;
2) adaptation of an ESPRIT-type frequency estimator to accurately estimate the carrier frequency offset (CFO);
3) pioneering work on a fixed-point DOA method;
4) practical experiments at the bare-metal level for four key performance criteria: angle accuracy, energy consumption, memory footprint, and execution time.

This article is organized as follows. Section II explains related research in the domain of DOA employing Bluetooth and important manuscripts about real DOA implementations. Section III outlines the ideal model that governs RDF taking into account L-shaped uniform arrays. Notably, Nomenclature lists all mathematical operations utilized in that section and subsequent parts of the document. Section IV gives an algorithmic-level overview of the DOA method upon which this research is based. Section V describes the working principles of the Bluetooth Direction Finding feature and its mathematical model. Section VII delves into the mathematical details of the novel fixed-point DOA method. Section VIII describes the experiment setups and discusses its results. Conclusions are drawn in Section IX.

II. Related Works

In the study [18], researchers explored the application of Bluetooth Direction Finding in an outdoor environment to estimate the position of a single drone. While this real-world application is both interesting and valuable, we identified certain limitations in their implemented solution. Initially, they employed a simple beamformer technique that relied on finding the peak of the pseudospectrum during the first estimation

of DOA. Although peak searching can be time-inefficient, given its one-time usage, we acknowledge its practicality. Subsequently, recognizing that the direction between two consecutive measurements does not change significantly, they utilized the Nelder–Mead method, an optimization technique used for finding the maximum or minimum of functions without derivatives. However, it is important to note that the Nelder–Mead algorithm does not guarantee convergence to a stationary point since it is a heuristic approach. Moreover, considering that the pseudospectrum might not even be convex, the method could potentially converge to a local maximum instead of the global maximum where the peak resides leading to inaccurate direction estimation. In addition, we observed that the CFO estimation process was also time-inefficient as it required peak searching.

In the master thesis [19], the author presented a novel algorithm based on the multiple signal classification (MUSIC) method, which utilized the Bluetooth Direction Finding feature. He implemented this algorithm using a custom linear and rectangular array of antennas connected to an embedded device. To enhance the accuracy of CFO estimation, he employed a return-to-first switching pattern. However, this approach resulted in a reduction of half the number of samples available for estimating the DOA, potentially impacting its accuracy. To mitigate this issue and address the multipath component at the receiver, he incorporated the multitone technique. This technique involved commanding the transmitting device to consecutively send a burst of packets carried by different frequencies within a short time interval.

In addition, to further reduce the impact of multipath effects, he applied the CLEAN algorithm, which resulted in an increased execution time but improved accuracy. As his method was based on MUSIC, one of the major drawbacks remained the exhaustive peak searching operation, which added computational complexity. Furthermore, as MUSIC requires an accurate array response model, the employment of array calibration techniques such as mutual coupling, switch leakage, and path imbalance was crucial for the method's success. The results highlighted the importance of CFO estimation in effectively applying phase compensation and demonstrated the effectiveness of the multitone technique in addressing multipath effects.

Wan et al. [20] investigated the application of the improved signal subtraction subspace (ISSS) algorithm in Bluetooth Direction Finding feature for a uniform linear array (ULA) and a uniform rectangular array (URA) of antennas in a BG22 Bluetooth Dual Polarized Antenna Array Pro Kit. The ISSS algorithm incorporates a spatial smoothing technique along with a median filter for phase compensation, aiming to reduce the impact of the RF switch. The research findings indicated an average positioning error of 0.92 m for single receivers and 0.30 m for dual receivers in a 3×3 m^2 room. In addition, the mean absolute error (MAE) in angle estimation varied across three channels, recorded at approximately 4°, 4.8°, and 3.9°. However, the median filter utilized only the reference period, which is composed of only eight samples. Such a small number of data can easily undermine the phase offset estimation decreasing the angle accuracy considerably.

In the study presented in [21], experiments on BLE Direction Finding were conducted both in an anechoic chamber and an open, unobstructed outdoor environment with a concrete surface. The experiments utilized an nRF52811, equipped with a uniform circular array (UCA), as the receiver. The DOAs were estimated by calculating the average phase difference from sequentially retrieved samples across every two adjacent antennas. With the antenna switching period set at 4 μs, the authors claimed that this setup allowed for a full 360° phase rotation, thus avoiding the need for phase compensation.

Despite these claims, the study overlooked the impact of CFO, a significant factor in BLE devices using low-cost oscillators. In addition, the research methodology employed a basic DOA estimator that relied on the average phase difference. While straightforward, this technique is less comprehensive compared to more advanced DOA methods that account for communication noise and leverage the configurations of antenna arrays. This article notably lacked metrics such as average DOA error estimation, opting instead to present data through histograms and graphs that illustrated the phase differences observed between antennas.

As outlined in the research [22], Hajiakhondi-Meybodi et al. explored the estimation of BLE signals using phase differences in addition to a nonlinear least squares (NLS) method to mitigate the multipath propagation effect. They applied the Kalman filter to deal with the phase shift caused by the RF switch. In addition, they experimented using an angle-of-arrival development kit (DK) composed of two ULA and a CC2640R2F microcontroller. The angle estimation process considered eight channels, yielding an angle error of less than 10° for azimuth angles within ±60°. However, errors increased significantly for angles beyond this range. Notably, this study focused exclusively on the azimuth angles using ULAs, which omitted elevation angles that a planar array could have provided.

As specified in the study [23], Ye et al. conducted a series of experiments on BLE Direction Finding using three different channels. They utilized both a URA and a ULA, paired with an nRF52833 SoC, to estimate azimuth and elevation angles and the position of transmitters. The study employed the propagator direct data acquisition (PDDA) for angle estimation and implemented a least squares (LS) method to compensate for CFO effects. The ULA's average angle estimation errors were reported as approximately 4.8°, 5.1°, and 2.3° for channel 37, 38, and 39, respectively, across a range of 20°–160°. Meanwhile, the URA was specifically used for positional estimations. In this setup, the URA was stationary, while the transmitter was placed at various locations. The research team observed average positional errors of 0.59, 0.74, and 0.77 m for the respective channels.

In our previous research [24], we explored a modified DOA method based on MUSIC, which was implemented in floating-point numbers, using an L-shaped array for the Bluetooth Direction Finding feature. However, in that study, we utilized a CFO estimation approach that, upon reflection, was found to have limitations as it relied solely on samples from the reference period leading to poor estimation results. The phase compensation technique involved computing the

TABLE II
COMPARATIVE OVERVIEW OF RELATED PAPERS

Reference	DOA Method	Array Type	Microcontroller	Frequency Estimator	Digital Number Representation	Measurements
This paper	ESPRIT type	L-shape	nRF52840	ESPRIT type	Floating-point Fixed-point	Angle Accuracy Energy consumption Execution time Memory footprint
[19]	MUSIC type	ULA, URA	Arduino family	Maximum Likelihood	Floating-point	Angle Accuracy
[18]	Bartlett beamformer	URA	nRF52833	Bartlett beamformer	Floating-point	Angle Accuracy
[20]	SSS	ULA, URA	EFR32BG1	Median Filter	Floating-point	Angle Accuracy Position Accuracy
[23]	PDDA	URA, ULA	nRF52833	Least Square Method	Floating-point	Angle Accuracy Position Accuracy
[21]	Phase Differences Average	UCA	nRF52811	None	Floating-point	Angle Accuracy
[22]	Phase Differences Average + NLS	ULA	CC2640R2F	Kalman Filter	Floating-point	Angle Accuracy
[24]	MUSIC type	L-shape	nRF52840	Phase Differences Average	Floating-point	Angle Accuracy Energy consumption Execution time Memory footprint

inverse of a matrix and directly applying it to the samples. While this technique improved the calculation of the inverse of the phase compensation matrix, it still suffered from time inefficiency compared to the solution we implemented in this current research. Furthermore, this modified version of MUSIC required finding polynomial roots, leading to the time-consuming polynomial root-finding method based on QR decomposition. In addition, as MUSIC relies on the direct application of array responses, achieving a highly accurate array calibration becomes crucial in that approach.

Table II provides a comparative overview of previously cited related studies. Our research contributes to the advancement of the field by refining a well-established DOA method, termed ESPRIT, specifically for Bluetooth Direction Finding applications. Diverging from traditional approaches that rely solely on floating-point operations, our method incorporates fixed-point arithmetic. This adaptation addresses the limitations of embedded systems that lack a dedicated FPU, making our solution more practical for a wider range of applications.

In addition, accurate estimation of the CFO is crucial for effective phase compensation caused by the RF switch. The suggested approach from [6], [25], and [26] involves averaging the phase difference between two consecutive samples of the reference period, as detailed in [24]. However, this method does yield poor estimations as demonstrated in the experimental findings in Section VIII. Another option is to employ MUSIC as a frequency estimator [27], but it requires a time-consuming exhaustive search to identify the peak frequency.

To address these limitations, we customized an ESPRIT-type frequency estimator. This approach avoids the shortcomings of the aforementioned methods while providing reliable frequency estimation. Unlike the previous frequency estimators, our method is designed to account for modeling inaccuracies and noise factors, and it utilizes a greater number of samples than what the reference period typically allows. Moreover, this research offers valuable insights by analyzing energy consumption, memory footprint, and execution time, which are frequently neglected in most investigations, despite their importance in the realm of resource-constrained embedded systems.

III. IDEAL MODEL FOR L-SHAPED UNIFORM ARRAY

The L-shaped uniform array consists of two orthogonal ULAs, each containing M antennas, arranged in the xy plane with a separation of Δ meters between adjacent antennas, as depicted in Fig. 2. This distinctive configuration facilitates the implementation of DOA methods by treating it as two separate ULAs, which are characterized by their simple structure, thus enabling the use of less intricate DOA techniques. Notably, despite the cited simplicity, their arrangement exhibits interesting properties, as elaborated on in Sections IV and VII, which make them amenable to the application of well-established DOA techniques.

In the ideal model, all antennas are assumed to be identical and isotropic. Suppose that there are d ($d < M$) independent far-field narrowband stationary radio signals, $s_i(t)$, such that $i = 1, \ldots, d$, incident on the array plane at 2-D angle $(\theta_1, \phi_1), (\theta_2, \phi_2), \ldots, (\theta_d, \phi_d)$ in which $\theta_i \in [-\pi, \pi]$ is the azimuth and $\phi_i \in [-(\pi/2), (\pi/2)]$ is the elevation angle. Note that the azimuth angle is measured counterclockwise relative to the positive x-axis, and the elevation angle is defined relative to the xy plane. Let us also assume the signals propagate in an additive white Gaussian noise (AWGN) channel and in a linear and nondispersive transmission medium, in directions $\mathbf{k}(\theta_i, \phi_i)$ that can be expressed in spherical coordinates as

$$\mathbf{k}(\theta_i, \phi_i) \triangleq -k \begin{bmatrix} \cos(\theta_i)\cos(\phi_i) \\ \sin(\theta_i)\cos(\phi_i) \\ \sin(\phi_i) \end{bmatrix} \tag{1}$$

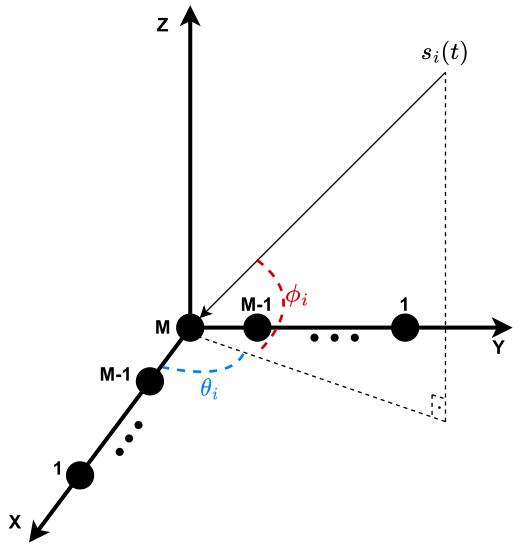


Fig. 2. Depiction of a receiver equipped with an L-shaped array with its antennas (black dots), angles, and the incoming signal.

where $k = (2\pi/\lambda)$ is the wavenumber and λ is the wavelength, while the minus sign indicates that the wave, which carries the signal, is traveling away from the source.

Assume that the L-shaped array is not subject to nonidealities, such as mutual coupling and cross-polarization effect. In addition, consider that all antennas are identical, having isotropic gain functions, i.e., $g(\theta, \phi) = 1$, and are not affected by perturbations and imperfections. By defining $\gamma_i = (\theta_i, \phi_i)$, the model of the signal samples received by the x-axis and y-axis arrays at a timestamp t can be expressed as in [28]

$$\mathbf{x}(t) = \mathbf{A}_x(\boldsymbol{\gamma})\mathbf{s}(t) + \mathbf{n}_x(t) \tag{2}$$
$$\mathbf{y}(t) = \mathbf{A}_y(\boldsymbol{\gamma})\mathbf{s}(t) + \mathbf{n}_y(t) \tag{3}$$

where $\boldsymbol{\gamma} = \begin{bmatrix}\gamma_1 & \gamma_2 & \dots & \gamma_M\end{bmatrix}^T$ is the column vector of azimuth and elevation angle for each signal source and $\mathbf{x}(t) = \begin{bmatrix}x_1(t) & x_2(t) & \dots & x_M(t)\end{bmatrix}^T$ is the array observation at timestamp t of the x-axis ULA, which is a vector of the signal samples for each antenna in the x-axis ULA, such that $x_i(t)$ corresponds to a single signal sample received from the antenna i at timestamp t, likewise for $\mathbf{y}(t) = \begin{bmatrix}y_1(t) & y_2(t) & \dots & y_M(t)\end{bmatrix}^T$ in the case of the y-axis ULA. Moreover, $\mathbf{s}(t) \in \mathbb{C}^{d\times 1}$ is a vector of signals of d transmitters, $\mathbf{n}_x(t), \mathbf{n}_y(t) \in \mathbb{C}^{M\times 1}$ are the AWGN, and $\mathbf{A}_x(\boldsymbol{\gamma}), \mathbf{A}_y(\boldsymbol{\gamma}) \in \mathbb{C}^{M\times d}$ are the ideal steering matrices of the x-axis array and the y-axis array, respectively, as

$$\mathbf{A}_x(\boldsymbol{\gamma}) = \begin{bmatrix}\mathbf{a}_x(\gamma_1) & \mathbf{a}_x(\gamma_2), & \dots, & \mathbf{a}_x(\gamma_d)\end{bmatrix} \tag{4}$$
$$\mathbf{A}_y(\boldsymbol{\gamma}) = \begin{bmatrix}\mathbf{a}_y(\gamma_1) & \mathbf{a}_y(\gamma_2), & \dots, & \mathbf{a}_y(\gamma_d)\end{bmatrix} \tag{5}$$

and ideal array responses are defined as

$$\mathbf{a}_x(\gamma_i)^T = \begin{bmatrix}1 & e^{-j\mathbf{k}(\gamma_2)^T\mathbf{r}_{1,0}}, & \dots, & e^{-j\mathbf{k}(\gamma_d)^T\mathbf{r}_{d,0}}\end{bmatrix} \tag{6}$$
$$\mathbf{a}_y(\gamma_i)^T = \begin{bmatrix}1 & e^{-j\mathbf{k}(\gamma_2)^T\mathbf{r}_{0,1}}, & \dots, & e^{-j\mathbf{k}(\gamma_d)^T\mathbf{r}_{0,d}}\end{bmatrix} \tag{7}$$

where the vector $\mathbf{r}_{p,q} = \begin{bmatrix}p\Delta & q\Delta & 0\end{bmatrix}^T$ represents the position of an antenna relative to the origin in the Cartesian coordinate system. This means that the generic form of the dot product is

$$\mathbf{k}(\gamma_i)^T\mathbf{r}_{p,q} = -k\begin{bmatrix}p\Delta u_i & q\Delta v_i & 0\end{bmatrix}^T \tag{8}$$

where

$$u_i \triangleq \cos(\theta_i)\cos(\phi_i) \tag{9}$$
$$v_i \triangleq \sin(\theta_i)\cos(\phi_i). \tag{10}$$

Noting that the x-axis ULA consists of antennas positioned only along the x-axis, and similarly for the y-axis array along the y-axis, (6) and (7) can be simplified to

$$\mathbf{a}_x(\gamma_i)^T \triangleq \begin{bmatrix}1 & e^{jk\Delta u_i}, & \dots, & e^{jk(M-1)\Delta u_i}\end{bmatrix} \tag{11}$$
$$\mathbf{a}_y(\gamma_i)^T \triangleq \begin{bmatrix}1 & e^{jk\Delta v_i}, & \dots, & e^{jk(M-1)\Delta v_i}\end{bmatrix} \tag{12}$$

which better describes array responses.

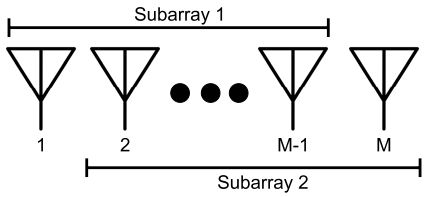


Fig. 3. Implemented ESPRIT divides the ULA into two subarrays of size $m = M - 1$ with $M - 2$ overlapping antennas.

IV. Unitary TLS ESPRIT

ESPRIT [29] is a highly regarded subspace-based technique used for estimating the DOA by leveraging the shift-invariance property of specific planar arrays. Unlike other commonly used methods, such as MUSIC involving time-consuming peak searching operations, ESPRIT offers a more efficient alternative by eliminating this step. In addition, ESPRIT does not rely on precise array responses and its direction estimation accounts for modeling errors to some extent, thereby eliminating the need for full array calibration. The shift-invariance property of the uniform array of antennas allows for the application of ESPRIT to estimate both azimuth and elevation angles in the L-shaped array, using both the x-axis and y-axis arrays separately. While the implementation discussed here focuses on the x-axis array, the procedure remains the same for the y-axis array.

Subspace-based techniques rely on proven properties of the matrix space defined by the covariance matrix in the following equation:

$$\mathbf{R}_{xx} = \mathbb{E}\left\{x(t)x(t)^H\right\} \in \mathbb{R}^{M\times M} \tag{13}$$

where the eigenvectors of $\mathbf{R}_{xx}$ can be categorized into two orthogonal subspaces: the signal subspace ($\mathbf{U}_s$) and the noise subspace ($\mathbf{U}_n$). The eigenvectors related to the d largest eigenvalues span the signal subspace, whereas the others $M-d$ eigenvectors span the noise subspace.

The standard ESPRIT method divides the x-axis ULA into two subarrays. In the implemented solution, the two subarrays are composed of $m = M - 1$ consecutive antennas and $M - 2$ overlapping ones, as shown in Fig. 3. It is possible

to create those subarrays by multiplying the steering matrix $\mathbf{A}$ conforming to

$$\begin{aligned}\mathbf{J_1} &\triangleq [\mathbf{I}_m \quad \mathbf{0}_m] \in \mathbb{R}^{m\times M}\\ \mathbf{J_2} &\triangleq [\mathbf{0}_m \quad \mathbf{I}_m] \in \mathbb{R}^{m\times M}\end{aligned} \tag{14}$$

where $\mathbf{I}_m$ is the identity matrix and $\mathbf{0}_m$ is a column vector of zeroes, both with size m. It can be shown that the shift-invariance propriety of the d array responses is expressed in line with

$$\underbrace{\mathbf{J_1}\mathbf{A}_x}_{\substack{\text{1st}\\ \text{subarray}}}\Psi = \underbrace{\mathbf{J_2}\mathbf{A}_x}_{\substack{\text{2nd}\\ \text{subarray}}} \tag{15}$$

where

$$\Psi \triangleq \text{diag}\left[e^{jk\Delta u_1}\ e^{jk\Delta u_2},\ \ldots,\ e^{jk\Delta u_d}\right]. \tag{16}$$

However, the steering matrix is unknown a priori. Since the range of the signal subspace ($\mathbf{U}_s$) spans the same space as the range of $\mathbf{A}$ for incoherent signals [30], the standard ESPRIT estimates the subspace rotational operation (Ψ) from the signal subspace ($\mathbf{U}_s$) compliant with

$$\mathbf{J_1}\mathbf{U}_s\Psi = \mathbf{J_2}\mathbf{U}_s. \tag{17}$$

Equation (17) represents an overdetermined system, and in real-world scenarios, the equality does not hold precisely due to errors present on both sides of the system. These errors stem from factors such as estimating the signal subspace and modeling inaccuracies in the array response. To address this, the standard ESPRIT estimates Ψ using the LS method, which considers the error on the system's right-hand side. However, for a more accurate solution of Ψ, the total LS (TLS) approach is preferred as it takes into account errors on both sides of the equation [30], [31], [32], [33].

In this research, the implemented solution utilizes a customized version of the Unitary TLS ESPRIT algorithm, a modified variant of the standard ESPRIT algorithm. It is important to note that the Unitary TLS ESPRIT algorithm is specifically designed for planar arrays that fulfill two essential conditions [34]:

1) dual shift-invariant property;
2) centrosymmetric property.

Since ULAs possess centrosymmetric and shift-invariant structures, as a result, the uniform L-shaped array satisfies the cited conditions as it estimates DOAs separately in two ULAs. Unitary transformations ensure that all centrohermitian matrices are converted into real ones offering several advantages over the standard ESPRIT. It reduces the memory footprint and computational burden by avoiding complex matrices, which requires more memory and computational resources than real matrices. Moreover, our Unitary TLS ESPRIT incorporates a built-in forward–backward averaging method to mitigate the impact of multipath propagation on the estimation of DOA, particularly in indoor environments where signal reflections are intensified.

This article does not delve into the mathematical details underpinning the ESPRIT algorithm's workings. Instead, it provides an algorithmic-level overview to better explain the modifications on the Unitary TLS ESPRIT undertaken by this research. For a comprehensive mathematical analysis, please refer to [32], [33], and [35].

Let $\Pi_p \in \mathbb{C}^{p\times p}$ be any antidiagonal identity matrix, that is,

$$\Pi_p \triangleq \begin{bmatrix} 0 & 0 & \ldots & 0 & 1\\ 0 & 0 & \ldots & 1 & 0\\ \ldots & \ldots & \ldots & \ldots & \ldots\\ 0 & 1 & \ldots & 0 & 0\\ 1 & 0 & \ldots & 0 & 0\end{bmatrix}$$

and $\mathbf{Q}_n \in \mathbb{C}^{n\times n}$ be an unitary transform matrix defined as

$$\mathbf{Q}_{2n} \triangleq \frac{1}{\sqrt{2}} = \begin{bmatrix}\mathbf{I}_n & j\mathbf{I}_n\\ \Pi_n & -j\Pi_n\end{bmatrix} \quad \text{or}$$

$$\mathbf{Q}_{2n+1} \triangleq \frac{1}{\sqrt{2}} = \begin{bmatrix}\mathbf{I}_n & 0 & j\mathbf{I}_n\\ \mathbf{0}_n^T & \sqrt{2} & \mathbf{0}_n^T\\ \Pi_n & 0 & -j\Pi_n\end{bmatrix}$$

depending on whether its size is even or odd. The algorithm is outlined in the following.

1) Collect N array observations for timestamp $t_1, t_2, \ldots, t_N$ to estimate the covariance matrix

$$\mathbf{R}_{xx} \approx \widehat{\mathbf{R}}_{xx} = \left(\frac{1}{N}\right)\mathbf{X}\mathbf{X}^H \tag{18}$$

where

$$\mathbf{X} \triangleq \left[\mathbf{x}(t_1)\ \mathbf{x}(t_2),\ \ldots,\ \mathbf{x}(t_N)\right] \in \mathbb{C}^{M\times N}. \tag{19}$$

2) Apply forward–backward averaging on the covariance matrix, that is,

$$\widehat{\mathbf{R}}_{xx}^{fb} \triangleq \widehat{\mathbf{R}}_{xx} + \Pi_M\overline{\widehat{\mathbf{R}}_{xx}}\Pi_M \tag{20}$$

to mitigate coherent signals' effect due to the multipath reflections [36]. Since $\widehat{\mathbf{R}}_{xx}^{fb}$ is a center-hermitian matrix, we apply the unitary transformation to convert that complex-valued matrix into a real-valued one in accordance with

$$\begin{aligned}\mathbf{C} &\triangleq \mathbf{Q}_M^H\,\widehat{\mathbf{R}}_{xx}^{fb}\,\mathbf{Q}_M\\ &= \text{Re}\{\mathbf{Q}_M^H\,\widehat{\mathbf{R}}_{xx}\,\mathbf{Q}_M\} \in \mathbb{R}^{M\times M}.\end{aligned} \tag{21}$$

3) Apply eigendecomposition (EVD) to find the signal subspace $\mathbf{U}_s$ of the real covariance matrix $\mathbf{C}$. The signal subspace is composed of eigenvectors corresponding to the d largest eigenvalues.
4) Thereafter, let us define

$$\begin{aligned}\mathbf{K_1} &\triangleq 2\,\text{Re}\{\mathbf{Q}_m^H\,\mathbf{J}_1\,\mathbf{Q}_M\}\\ \mathbf{K_2} &\triangleq 2\,\text{Im}\{\mathbf{Q}_m^H\,\mathbf{J}_1\,\mathbf{Q}_M\}\end{aligned} \tag{22}$$

as unitary transformation of $\mathbf{J}_1$ and $\mathbf{J}_2$, respectively.
5) Estimate the matrix $\Upsilon \in \mathbb{R}^{d\times d}$ from the following equation:

$$\mathbf{K_1}\mathbf{U}_s\Upsilon \approx \mathbf{K_2}\mathbf{U}_s \tag{23}$$

by means of TLS. To do that, first, compute the matrix

$$\mathbf{E} \triangleq \begin{bmatrix}(\mathbf{K_1}\mathbf{U}_s)^T\\ (\mathbf{K_2}\mathbf{U}_s)^T\end{bmatrix}\left[(\mathbf{K_1}\mathbf{U}_s)\ (\mathbf{K_2}\mathbf{U}_s)\right]. \tag{24}$$

Since $\mathbf{E} \in \mathbb{R}^{2d\times 2d}$ is a real symmetric matrix, then it is diagonalizable [37]; thus, we can apply EVD resulting in $\mathbf{E} = \mathbf{V}\Sigma\mathbf{V}^T$, in which $\Sigma = \text{diag}[\sigma_1, \sigma_2, \ldots, \sigma_{2d}]$ is the matrix of its eigenvalues in such a way that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{2d}$. Let us partition these two matrices into four ones

$$\mathbf{V} \triangleq \begin{bmatrix} \mathbf{V}_{11} & \mathbf{V}_{12} \\ \mathbf{V}_{21} & \mathbf{V}_{22} \end{bmatrix} \quad \text{and} \quad \Sigma \triangleq \begin{bmatrix} \Sigma_1 & \mathbf{0} \\ \mathbf{0} & \Sigma_2. \end{bmatrix}$$

Thus, the right submatrix $\begin{bmatrix}\mathbf{V}_{12} & \mathbf{V}_{22}\end{bmatrix}^T$ is made up of eigenvectors associated with the d smallest eigenvalues. If $\mathbf{V}_{22}$ is nonsingular, then $\Upsilon = -\mathbf{V}_{12}\mathbf{V}_{22}^{-1}$. Otherwise, there is no solution for the TLS [38].

6) The eigenvalues of Υ contain information about DOAs. Thus, the next step is to compute them as

$$\Omega \triangleq \text{diag}\begin{bmatrix}\omega_1 & \omega_2, & \ldots, & \omega_d\end{bmatrix} \tag{25}$$

such that $w_i \triangleq \tan\left(\frac{k\Delta u_i}{2}\right)$ and, therefore, extract the direction for each signal source by the following operations:

$$u_i = \frac{2\arctan(\omega_i)}{k\Delta}, \qquad 1 \leq i \leq d. \tag{26}$$

Algorithm 1 provides a concise summary of the 1-D TLS ESPRIT. For y-axis ULA, the only distinction lies in the input, denoted as $\mathbf{Y}$, and the output, represented by v_i.

Algorithm 1 Summary of 1-D Unitary TLS ESPRIT Algorithm

Input: matrix $\mathbf{X}$ composed of N array observations and the number of signals (d).

Output: DOA information u_i, $i = 1, \ldots, d$.

1) Compute the real-valued signal subspace, $\mathbf{U}_s$, as the d dominant eigenvectors of

$$\mathbf{C} = \text{Re}\{\mathbf{Q}_M^H\, \widehat{\mathbf{R}}_{xx}\, \mathbf{Q}_M\}$$

by applying an EVD algorithm on $\mathbf{C}$.

2) Solve the real-valued invariance equation below for Υ,

$$\mathbf{K_1}\mathbf{U}_s\Upsilon \approx \mathbf{K_2}\mathbf{U}_s,$$

by means of TLS using an EVD algorithm.

3) Calculate the eigenvalues of Υ, which are $\Omega = \text{diag}\begin{bmatrix}\omega_1 & \omega_2 & \ldots & \omega_d\end{bmatrix}$, thereafter, extract the DOA information, u_i, that is,

$$u_i = \frac{2\arctan(\omega_i)}{\left(\frac{2\pi\Delta}{\lambda}\right)}, \; i = 1, \ldots, d.$$

A. Frequency Estimation

In addition, ESPRIT can also serve as a frequency estimator. Consider a signal composed of d harmonic components

$$z(t) = \sum_{i=1}^{d} \alpha_i e^{j2\pi f_i t}. \tag{27}$$

Suppose that the signal is uniformly sampled by one single antenna with a sampling period T satisfying the Nyquist criterion, resulting in N_s samples $z(1), z(2), \ldots, z(N_s)$, where $z(n)$ represents the discrete-time signal, which is equivalent to its continuous counterpart $z(nT)$. In this scenario, we can define a sample matrix $\mathbf{Z}$ with $m < N_s$ rows as follows:

$$\mathbf{Z} \triangleq \begin{bmatrix} z(1) & z(2) & z(3) & \cdots & \cdots \\ z(2) & z(3) & z(4) & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z(m) & z(m+1) & z(m+2) & \cdots & z(N_s) \end{bmatrix}. \tag{28}$$

We can decompose $\mathbf{Z} \in \mathbb{C}^{m\times N_s}$ into two matrices as expressed in

$$\begin{aligned}\mathbf{Z} &= \mathbf{A}_z\mathbf{S} \\ &= \begin{bmatrix} 1 & \ldots & 1 \\ \nu_1 & \ldots & \nu_d \\ \nu_1^2 & \ldots & \nu_d^2 \\ \vdots & \cdots & \vdots \\ \nu_1^{m-1} & \ldots & \nu_d^{m-1} \end{bmatrix} \begin{bmatrix} \alpha_1\nu_1 & \alpha_1\nu_1^2 & \ldots \\ \vdots & \vdots & \vdots \\ \alpha_d\nu_d & \alpha_d\nu_d^2 & \ldots \end{bmatrix}\end{aligned} \tag{29}$$

where $\nu_i = e^{j2\pi f_i T}$. The matrix $\mathbf{A}_z$ could be viewed as a steering matrix of a ULA as both have a Vandermonde structure where each column could be treated as the array response corresponding to a single signal transmitter. The matrix $\mathbf{S}$ could be considered a matrix whose columns contain samples from d transmitters the same way as $\mathbf{s}(t)$ in (2) and (3) for $t = T, 2T, \ldots, N_sT$. As a result, it is possible to utilize the Unitary TLS ESPRIT algorithm to estimate the unique frequencies f_i by exploiting the shift-invariant and centrosymmetric properties of ULAs. The key distinction from the standard Unitary TLS ESPRIT lies in (26), which is replaced with

$$f_i = \frac{2\arctan(w_i)}{2\pi T} \tag{30}$$

since $w_i \triangleq \tan((2\pi f_i T)/2)$. For a detailed mathematical analysis, refer to [39].

V. RF Switch Model

Theoretically, all antennas in an array should sample signals simultaneously from each antenna port. However, achieving this would require each antenna to have its own RF front end, consisting of analog-to-digital converters, filters, mixers, and low-noise amplifiers. Manufacturing every antenna with such components would lead to increased power consumption, physical size, and overall cost, particularly for constrained embedded IoT devices. A practical solution to overcome these limitations is to use a single RF front end and an RF switch that allows each antenna to operate with the front end at different times, as depicted in Fig. 4. By doing so, the array can still sample signals from all the antennas in the array, albeit not simultaneously. As Bluetooth protocol supports this radio architecture for its direction-finding capability, in this study, we employed the switching protocol described in the Bluetooth v5.1 specification [4] with an L-shaped array.

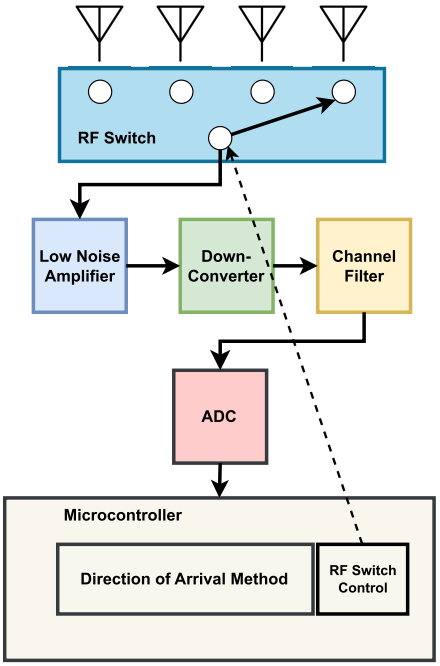


Fig. 4. Overview of a simplified RF front end connected with a microcontroller.

Bluetooth applies Gaussian frequency-shift keying (GFSK) to modulate 0s and 1s into different frequency shifts [40]. These frequency shifts are centered around the carrier frequency (f_c) and include a frequency deviation ($\pm f_\Delta$). Bluetooth operates within the Industrial Scientific and Medical (ISM) band from 2.40 to 2.41 GHz. This band is divided into 40 channels, each spaced 2 MHz apart. During a Bluetooth connection, devices dynamically utilize a subset of available channels for communication in which the channel selection is controlled by an adaptive frequency hopping algorithm. As a result, Bluetooth communication involves the use of multiple frequencies rather than a single fixed frequency. It is important to note that changing the frequency also affects the wavelength, which is a crucial factor in the DOA calculations. To maintain a constant frequency during in-phase and quadrature (IQ) sampling, the transmitter emits a constant tone extension (CTE), which consists of a continuous stream of digital ones, extending for a duration of 160 µs.

The Bluetooth Core Specification v5.1 sets specific timing regulations for switching and sampling in the context of CTEs, as illustrated in Fig. 5. It outlines a structure where CTE processing time is segmented into three parts: a 4-µs initial guard period, followed by an 8-µs reference period, and then an alternating series of slots designated for either switching or sampling. During this alternating series, denoted as the switch-sample period, sampling occurs in the sampling slots and antenna switching occurs in the switch slots. Sample and switch slots may either be 1 µs or 2 µs long. Since this research considers 1-µs slot duration, Bluetooth defines 74 slots for sampling and switching. As a result, the IQ sample interval is $T_s = 2$ µs. During the reference period, the system gathers eight IQ samples every 1 µs from the first antenna, without any antenna switching. These eight reference samples might enable the receiver to estimate the phase difference due to the RF switch.

Standard models commonly assume narrowband signals to simplify the calculation of DOA techniques as it approximates the propagation delays between antennas using phase shifts of the complex envelope. We can consider that assumption if the bandwidth of the signal is small compared to the inverse of the propagation time over the array aperture. Mathematically, the criterion for narrowband approximation is given by $B\tau \ll 1$ [30], [41], where B represents the bandwidth of the signal (in hertz) and τ is the propagation time (in seconds) for the signal to travel across the antenna array. In our case, τ corresponds to the time it takes for the signal to traverse a single ULA of antennas, as the implemented solution processes two ULAs separately. We can express this relationship as

$$B\tau = 10^6 \left(\frac{M\Delta}{c}\right) = 10^6 \left(\frac{M\lambda}{2c}\right) \tag{31}$$

where $\Delta = \lambda/2$, $B = 1$ MHz, and M denotes the number of antennas in the ULA. In the context of constrained IoT devices, arrays of antennas typically consist of a small number of elements. We found that many indoor localization solutions employ small URAs of 4×4 or 3×3 antennas; nonetheless, in our case, $M = 4$. Since $\lambda = c/f_c$ where c is the speed of light, (31) simplifies to

$$B\tau = 10^6 \left(\frac{M}{2f_c}\right) \approx 0.000208 \times M \ll 1. \tag{32}$$

Therefore, BLE satisfies the narrowband condition, and if we take into account all the cited assumptions given in Section III as well, the mathematical model is the same as (2) and (3) in addition to the phase shift due to the RF switch. As a result, it is imperative to develop a phase compensation to make the DOA method work properly. In (33), the received bandpass signal is represented in a complex format

$$\begin{aligned} u(t) &= \mathrm{Re}(s(t)e^{j2\pi f_c t}) \\ &= I(t)\cos 2\pi f_c t - Q(t)\sin 2\pi f_c t \end{aligned} \tag{33}$$

where t is the time and $s(t) = I(t) + jQ(t)$ is called the complex envelope. The receiver extracts the IQ components of the baseband signal, where the central frequency (f_c) is translated to dc [25]. Consequently, the IQ components can be represented by

$$\begin{aligned} s(t) &= Ae^{j(2\pi f_T t + \psi)} \\ &= \underbrace{A\cos(2\pi f_T t + \psi)}_{I(t)} + j\underbrace{A\sin(2\pi f_T t + \psi)}_{Q(t)} \end{aligned} \tag{34}$$

where A is the amplitude, $f_T = f_\Delta + f_o$, $f_\Delta = 250$ kHz considering LE 1M physical layer [4], and f_o is the CFO. The mathematical model must account for the CFO as the crystal oscillator of low-cost BLE devices possesses significant inaccuracy. According to Core Specification v5.1, the deviation of the center frequency can reach up to ±150 kHz. Moreover, the phase shift due to the RF switch of the kth sample is given by

$$s(t + kT_s) = s(t)e^{j2\pi f_T k T_s}. \tag{35}$$

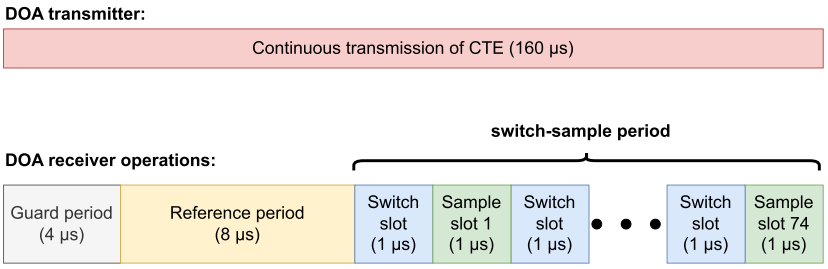


Fig. 5. Depiction of the transmitter and receiver operations.

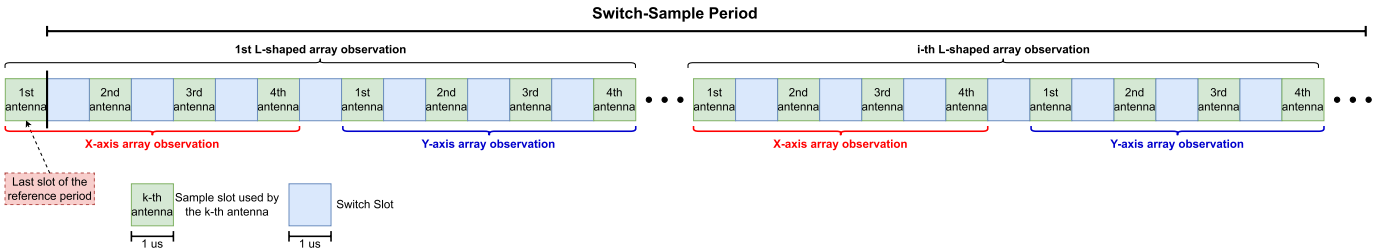


Fig. 6. Example of the Round-Robin switch pattern of an L-shaped array with seven antennas. Note that both ULAs have the fourth antenna in common.

Therefore, to determine the phase shift caused solely by the RF switch, it is necessary to estimate the frequency f_T. We adopted a Round-Robin switching pattern in our research to sequence each antenna's signal sampling in an orderly and fair manner, as depicted in Fig. 6. It starts with the x-axis ULA, where each antenna captures the signal once, following a sequence from the first to the Mth antenna. The process then continues in the same sequential manner along the y-axis ULA. It is important to note that the Mth antenna, at the junction of both ULAs, samples twice. Moreover, the switching sequence is initiated at the final slot of the previous reference period. Therefore, the Round-Robin pattern captures 75 samples, 74 from the switch-sample period, and one from the last reference period slot.

The L-shaped array observation, in this case, is defined as one single sequence of the Round-Robin pattern, that is, when all antennas in the x-axis and y-axis ULAs complete the IQ sampling operation, while the x-axis array observation (likewise for the y-axis) is expressed as

$$\mathbf{x}_s(t) = \begin{bmatrix} x_1(t) \\ x_2(t+T_s) \\ \vdots \\ x_M(t+(M-1)T_s) \end{bmatrix}. \tag{36}$$

Based on (35), (36) amounts to

$$\mathbf{x}_s(t) = \begin{bmatrix} x_1(t) \\ x_2(t)e^{j2\pi f_T T_s} \\ \vdots \\ x_M(t)e^{j2\pi f_T (M-1)T_s} \end{bmatrix} = \mathbf{O}\mathbf{a}_x(\gamma)s(t) \tag{37}$$

where $\mathbf{O} \in \mathbb{C}^{M\times M}$ is the phase shift matrix caused by the RF switch, which is a diagonal matrix defined in (38)

$$\mathbf{O} \triangleq \text{diag}\left[1 \; e^{j2\pi f_T T_s}, \; \ldots, \; e^{j2\pi f_T (M-1)T_s}\right]. \tag{38}$$

Bluetooth Direction Finding can only track the DOA of a single transmitter [4]. Consequently, the L-shaped array observation model is defined as

$$\begin{bmatrix} \mathbf{x}_s(t) \\ \mathbf{y}_s(t) \end{bmatrix} = \begin{bmatrix} \mathbf{O}\mathbf{x}(t) \\ \mathbf{O}\mathbf{y}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{O}\mathbf{a}_x(\gamma) \\ \mathbf{O}\mathbf{a}_y(\gamma) \end{bmatrix} s(t) + \begin{bmatrix} \mathbf{n}_x \\ \mathbf{n}_y \end{bmatrix} \tag{39}$$

where $s(t)$ is a scalar that represents a signal from a single transmitter in opposition to the multiple signals denoted in vector $\mathbf{s}(t)$ found in (2) and (3). Moreover, the number of L-shaped array observations in the CTE is equal to $N = \lfloor 75/2M \rfloor$. Notably, 75 is the total number of samples and $2M$ is the number of samples in the L-shaped array observation. Observe that since 75 is not divisible by $2M$, the last 75 mod $2M$ samples are not used. Therefore, we can define the matrix $\mathbf{X}_s$ similar to (19)

$$\mathbf{X}_s \triangleq \left[\mathbf{x}_s(t_1) \; \mathbf{x}_s(t_2), \; \ldots, \; \mathbf{x}_s(t_N)\right] \tag{40}$$

such that $t_n = 2(n-1)MT_s$.

VI. Signed Fixed-Point Arithmetic

Fixed-point arithmetic is a valuable technique employed to represent fractional numbers using integers, serving as an alternative to floating-point calculations. This approach proves particularly advantageous for microcontrollers lacking a dedicated FPU, as it enhances energy efficiency and computational speed compared to the reliance on software-based floating-point arithmetic. However, even in situations where microcontrollers have an FPU, utilizing fixed-point arithmetic can still provide benefits in certain applications, particularly when it comes to energy consumption, as the FPU can be disabled.

More specifically, fixed-point numbers, as their name suggests, are integer numbers with a fixed number of bits to represent the integer and fractional components of a fractional number. In the processing of fixed-point numbers, the

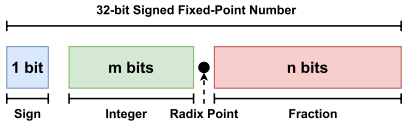


Fig. 7. Illustrative building blocks representation of signed fixed-point numbers.

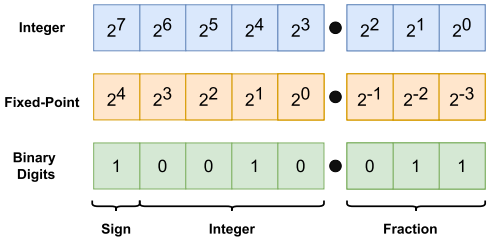


Fig. 8. Example of a fixed-point number with its integer and binary digit representations.

processor treats them as regular binary integers, utilizing integer arithmetic. However, from a programmer's perspective, a virtual radix point is conceptually placed at a fixed position. While fixed-point numbers can be designed with varying lengths, this research specifically focuses on the 32-bit signed fixed-point number, as depicted in Fig. 7 since constrained embedded processors, such as the Arm Cortex M4 and M33 found in SoCs with direction-finding capabilities, operate exclusively in 32-bit mode.

The signed fixed-point representation is often used in the $Qm.n$ format in which m and n are the number of bits of the integers and fractional part, respectively. Let us also define f as the fixed-point number and I as its represented integer number in two's complement. Fig. 8 shows an example of a $Q4.3$ fixed-point number, $f = 10010.011_2$, represented as an integer number, $I = 10010011_2$. As we can clearly see, its value in decimal notation is

$$\begin{aligned} 10010.011_2 &= \frac{-1 \times 2^7 + 1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0}{2^3} \\ &= -13.625_{10}. \end{aligned}$$

In general terms, signed integer numbers in two complements format are represented by

$$I = -2^{m+n} b_{m+n} + \sum_{i=0}^{m+n-1} (2^i b_i) \tag{41}$$

where $b_{m+n}, \ldots, b_2 b_1 b_0$ are binary digits that represent f in $Qm.n$ format. Therefore, the value of a signed fixed-point number can be calculated according to $f = I/2^n$.

Consider two arbitrary $Qm.n$ numbers, denoted as f_a and f_b. From a computer's perspective, it can be demonstrated [16] that the four fundamental arithmetic operations can be executed solely using the signed integer representations of these numbers in two's complement format, denoted as I_a and I_b, as shown in Table III. It is important to note that we are utilizing integer division that outputs the integer part of the quotient only. Its equation can be represented as $q = \lfloor a/b \rfloor$, where a, $b \neq 0$ and q are signed integer numbers. Notably, Arm Cortex M4 and M33 processors, which are widely employed in low-cost IoT devices, perform the integer division utilizing SDIV assembly instruction [17], thereby avoiding the need for an FPU.

TABLE III
FIXED-POINT ARITHMETIC OPERATIONS AND THEIR EQUIVALENT SIGNED INTEGER OPERATIONS

Fixed-point Operation	Signed Integer Operation
$f_a + f_b$	$I_a + I_b$
$f_a - f_b$	$I_a - I_b$
$f_a \times f_b$	$(I_a \times I_b) >> n$
$\frac{f_a}{f_b}$	$\left\lfloor \frac{(I_a << n)}{I_b} \right\rfloor$

When implementing a solution using fixed-point representation, it is crucial to consider two important parameters: resolution and range. During execution, if a number is less than the resolution or falls outside the range permitted by the fixed-point format, the method may encounter issues such as crashes or produce erroneous results. Furthermore, the overall accuracy of the implemented solution is tightly linked to the accuracy of the fixed-point representation. The following paragraphs define these three parameters.

1) *Resolution:* It refers to the smallest possible nonzero real number that can be represented in a digital system. In the context of fixed-point format, it corresponds to the gap between two consecutive numbers that can be represented. For a signed fixed-point format denoted as $Qm.n$, the resolution is 2^{-n}.
2) *Accuracy:* It denotes the closeness of a value represented in a fixed-point format with its true value. That is, it measures the representation error that arises when approximating a real number using a fixed-point format. In other words, accuracy quantifies how well a fixed-point representation captures the exact value of the original real number.
3) *Range:* The range of signed fixed-point numbers can be determined by considering the range of signed integer numbers in two's complement representation with a total of $m+n+1$ bits. This range is given by $[-2^{m+n}, 2^{m+n} - 1]$. Therefore, the range of signed fixed-point numbers is given by $[-2^{m+n}, 2^{m+n} - 1] \times 2^{-n}$, which simplifies to $[-2^m, 2^m - 2^{-n}]$.

VII. IMPLEMENTED SOLUTION

The implemented solution builds upon our previous method, as described in [42], which utilizes a 1-D Unitary TLS ESPRIT algorithm optimized for estimating a single DOA. In this research, we have expanded our method to incorporate the RF switch and estimation of both azimuth and elevation angles and the CFO. To achieve this, we employ an L-shaped uniform array and ensure accurate estimation of the CFO, which is crucial for precise phase compensation in the presence of the RF switch and for accurate angle estimation. Furthermore, the implemented solution employs the fixed-point representation as many constrained embedded systems are absent from an FPU, thereby relying on

software-based floating-point operations, which are time- and energy-inefficient. To address this properly, we have developed efficient fixed-point numerical computations specifically tailored for this purpose, as standard numerical methods may not be suitable for fixed-point numbers. Moreover, we replaced the inverse power method employed in [42] with an even simpler analytical algorithm.

The optimization principles guiding the development of the implemented solution not only prioritize time efficiency but also aim to mitigate computational resource consumption to minimize energy usage. In addition, special consideration is given to ensure the feasibility of the solution in a multithread environment, as many constrained IoT devices utilize basic RTOS. The solution is designed at the bare-metal level, developed in C99 programming language, and adopts the Q15.16 fixed-point format. To achieve the goal of minimal computational resource consumption and enhance portability, all numerical methods were built from scratch, eliminating the reliance on external libraries. This approach ensures greater control over resource utilization and enables the solution to be efficiently implemented on various platforms. The developed solution includes tailor-made numerical methods such as the power method, a frequency estimator, fixed-point implementations of an inverse tangent, tangent, and inverse sine functions, as well as auxiliary linear algebra algorithms, for example, vector norm calculation and matrix–vector multiplication. These custom numerical methods are specifically designed to meet stringent resource constraints and enhance the efficiency of the solution.

A. Power Method

In our research [42], we introduced the power method as a more efficient alternative to computationally intensive eigenvalue decomposition for single DOA estimation. This technique capitalizes on the fact that the signal subspace consists of only one eigenvector. In such a scenario, instead of resorting to eigenvalue decomposition algorithms that unnecessarily compute all eigenvectors and eigenvalues, the power method proves to be a more efficient choice. This method accurately estimates the eigenvector associated with the largest absolute eigenvalue [43], which represents the signal subspace as demonstrated in the next paragraphs.

By employing the power method, our implemented solution avoids the complexities associated with EVD algorithms, such as their time-consuming nature and high memory usage. For instance, the QR algorithm, a well-established EVD method, has a complexity of $\mathcal{O}(n^3)$ per iteration [44], and it often requires additional steps, such as performing a Hessenberg decomposition and searching for the eigenvector related to the largest eigenvalue in this specific case. In contrast, the power method has a simpler complexity of $\mathcal{O}(n^2)$ per iteration, in our case, and involves straightforward computations. In our solution, we have observed that the power method typically converges within just four iterations, making it highly efficient for single-source DOA estimation.

To guarantee the convergence of the power method, the input matrix must be diagonalizable, there must exist one single eigenvalue with the greatest absolute value, and it must be a real number [45]. For instance, if the eigenvalues of a diagonalizable matrix are denoted as $\lambda_i \in \mathbb{R}$ for $i = 1, \ldots, M$, and if $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_M|$, then the matrix satisfies the convergence requirements. The matrix $\mathbf{C}$, defined in (21), is a real covariance matrix; thus, it is symmetric [46]. Therefore, it is diagonalizable, and its eigenvalues are real numbers [47]. For $\mathbf{C}$ to have one single largest absolute eigenvalue, there must exist one single line-of-sight (LOS) signal source in the absence of coherent signals. The latter requirement is caused by multipath propagation effects and is also a mandatory requisite for subspace DOA methods. Nevertheless, the forward–backward averaging technique applied in the Unitary TLS ESPRIT aims to mitigate coherent signals' effect.

Algorithm 2 Power Method

Input: covariance matrix $\mathbf{C}$.
Output: signal subspace $\mathbf{U}_s$.
Define $\mathbf{v}_1 = [1, 1, \ldots, 1]^T \in \mathbb{R}^M$, $\mathbf{v}_0 = \mathbf{0}_M$, $k = 1$, $tol \ll 1 \in \mathbb{R}_{>0}$, and $K \in \mathbb{Z}_{>0}$.
while $k \leq K$ ***and*** $\|\mathbf{v}_k - \mathbf{v}_{k-1}\|^2 > tol$ **do**
  $\mathbf{v}_{k+1} \leftarrow \mathbf{C}\mathbf{v}_k$
  $\mathbf{v}_{k+1} \leftarrow \frac{\mathbf{v}_{k+1}}{\|\mathbf{v}_{k+1}\|}$
  $k \leftarrow k + 1$
end
if $k > K$ **then**
  `/* Convergence failed */`
  return $NULL$
end
else
  `/* Convergence succeeded */`
  return $\mathbf{v}_k$
end

The power method computes the eigenvector related to the largest absolute eigenvalue, which is the signal subspace for single-source DOA estimation. More precisely, since $\mathbf{C}$ is a real covariance matrix, it is positive semidefinite [46], meaning that all eigenvalues are nonnegative. The LOS component of the received signal that constitutes the eigenvalue of the signal subspace is greater than the eigenvalues of the noise subspace [28], [33], and since they are all nonnegative, eigenvalues of the noise subspace cannot be equal to or greater than the one of signal subspace in magnitude. Therefore, the eigenvalue of the signal subspace is the greatest in magnitude; hence, its corresponding eigenvector is the signal subspace. The same reasoning applies to the y-axis ULA.

In the implemented power method (see Algorithm 2), the eigenvalue is not computed since only its eigenvector is required. The chosen parameters were $K = 30$ and $tol = 10^{-6}$. Through numerous experiments, it was observed that Algorithm 2 typically converges within four to five iterations, and in all experimental instances, 30 iterations were more than sufficient. Therefore, for $k > 30$, it is assumed that the algorithm fails to compute the signal subspace.

B. Solving the Total LS

The total LS procedure can be analytically computed in the implemented solution as it estimates a single DOA. In this scenario, the matrix Υ from (23) reduces to a scalar value, denoted as Υ, and no longer possesses eigenvalues. The matrix $\mathbf{E}$ from (24) becomes a 2×2 matrix, resulting in a corresponding eigenvector matrix $\mathbf{V}$ of the same size. Therefore, the DOA information can be extracted using

$$\Upsilon = -\frac{V_{12}}{V_{22}} \tag{42}$$

where V_{12} and $V_{22} \neq 0$ can be calculated analytically by solving a 2×2 eigenvector problem, commonly found in introductory linear algebra textbooks, such as [48]. With the DOA information contained in the scalar Υ, (26) is replaced with

$$u_1 = \frac{2\arctan(\Upsilon)}{k\Delta}. \tag{43}$$

C. BLE ESPRIT

Algorithm 3 introduces a novel Unitary TLS ESPRIT method termed BLE ESPRIT. This method incorporates the power method and leverages the shift-invariant property to simultaneously extract the DOA information and perform phase compensation through a simple trigonometric equation. By employing this approach, it avoids the computationally expensive operations of calculating the phase compensation matrix ($\mathbf{O}$), computing its inverse ($\mathbf{O}^{-1}$), and performing matrix multiplication.

Specifically, to extract the samples $\mathbf{x}(t)$ and $\mathbf{y}(t)$ from the Round-Robin switch pattern samples $\mathbf{x}_s(t)$ and $\mathbf{y}_s(t)$, the conventional approach involves computing the phase compensation matrix $\mathbf{O}$ and finding its inverse, as exemplified in (44) and applied in our previous research [24]

$$\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{y}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{O}^{-1}\mathbf{x}_s(t)\\ \mathbf{O}^{-1}\mathbf{y}_s(t)\end{bmatrix}. \tag{44}$$

However, the implemented solution adopts a faster approach based on a simple trigonometric equation, bypassing the time-consuming steps of matrix inversion and multiplication.

To derive (52), consider that by splitting the x-axis ULA into two subarrays, as illustrated in Fig. 3, the shift-invariant property and Vandermonde structure still hold even if the samples are performed sequentially in a Round-Robin switch pattern, that is,

$$\underbrace{\mathbf{J}_1\mathbf{O}\mathbf{a}_x(\gamma)}_{\substack{1\text{st}\\ \text{subarray}}} e^{j2\pi f_T T_s} e^{jk\Delta u_1} = \underbrace{\mathbf{J}_2\mathbf{O}\mathbf{a}_x(\gamma)}_{\substack{2\text{nd}\\ \text{subarray}}}. \tag{45}$$

More precisely,

$$\mathbf{J}_1\mathbf{O}\mathbf{a}_x(\gamma)e^{j(2\pi f_T T_s + k\Delta u_1)} = \mathbf{J}_2\mathbf{O}\mathbf{a}_x(\gamma). \tag{46}$$

Therefore, it is possible to apply ESPRIT. Observe that the shift-invariant equation incorporates an additional phase shift, $2\pi f_T T_s$, caused by sampling the signal sequentially. In step 2, as described in Algorithm 3, the method estimates Υ that contains f_T, T_s, and u_1. The sample period (T_s) is known and f_T is provided as a prior estimation in Algorithm 4; thus, the method requires calculating u_1 from Υ, which is equal to

$$\Upsilon = \tan\left(\frac{2\pi f_T T_s + k\Delta u_1}{2}\right). \tag{47}$$

The first attempt could involve extracting the angle of (47) by applying the inverse of tangent; however, we found experimentally that the angle may exceed π, which is the periodicity of the tangent function. Even if such an angle does not exceed π in theory, (47) is considerably sensitive to numerical errors and the total LS estimation, which could overestimate the angle beyond π. Thus, the implemented solution applies a trigonometric identity to isolate u_1 before calculating it. By defining $a \triangleq k\Delta u_1/2$ and $b \triangleq 2\pi f_T T_s/2$, from the trigonometric identity in (48)

$$\underbrace{\tan(a+b)}_{\Upsilon} = \frac{\tan(a)+\tan(b)}{1-\tan(a)\tan(b)} \tag{48}$$

we can isolate $\tan(a)$, as long as $1 - \tan(a)\tan(b) \neq 0$, in accordance with (49)

$$\tan(a) = \frac{\Upsilon - \tan(b)}{1 + \Upsilon\tan(b)} \tag{49}$$

thereby allowing the recovery of u_1, as demonstrated in (52). The y-axis ULA of antennas follows the same procedure to calculate v_i.

Algorithm 3 Summary of BLE ESPRIT

Input: matrix $\mathbf{X}_s$ composed of N array observations, f_T and the number of signals $d = 1$.

Output: DOA information u_1.

1) Compute the real-valued signal subspace, as the dominant eigenvector of

$$\mathbf{C} = \mathrm{Re}\{\mathbf{Q}_M^H\, \widehat{\mathbf{R}}_{xx}\, \mathbf{Q}_M\} \tag{50}$$

by applying the Power Method.

2) Solve analytically the real-valued invariance equation below for Υ,

$$\mathbf{K_1}\mathbf{U}_s\Upsilon \approx \mathbf{K_2}\mathbf{U}_s, \tag{51}$$

by applying TLS.

3) Extract u_1 by simultaneously applying the phase compensation, that is,

$$u_1 = \frac{2\arctan\left(\dfrac{\Upsilon - \tan\left(\dfrac{f_T}{2}\right)}{1 + \Upsilon\tan\left(\dfrac{f_T}{2}\right)}\right)}{k\Delta}. \tag{52}$$

D. BLE Frequency Estimator

Algorithm 4 presents an overview of the frequency estimator. It leverages the dual sampling of the Mth antenna,

adhering to the principle that a higher number of samples leads to greater accuracy in estimation. Define the sequence

$$z(n) = \{x_s(T), y_s(T), x_s(2T), y_s(2T) \ldots, x_s(N_sT), y_s(N_sT)\} \tag{53}$$

where $z(n)$ represents the samples of the Mth antenna for all L-shaped array observations and $T = MT_s$ is the sampling period of the cited antenna. Therefore, it is possible to construct the matrix $\mathbf{Z}$, as defined in (28). The frequency estimator, as described in Section IV-A, evaluates f_i from $v_i = e^{j2\pi f_i T}$. In our case, $f_i = f_T = f_\Delta + f_o$ such that $f_\Delta = 250$ kHz and f_o is unknown; therefore, the phase of v_i is

$$\xi = 2\pi(250000 + f_o)\underbrace{MT_s}_{T}. \tag{54}$$

After some calculation, (54) yields

$$\xi = M\pi + 2\pi T f_o. \tag{55}$$

In Algorithm 4, it is important to note that step 3 involves extracting f_o from $\Upsilon = \tan(\xi/2)$; therefore,

$$\Upsilon = \tan\left(\frac{M\pi}{2} + \pi T f_o\right). \tag{56}$$

If M is even, $M = 2n$, and then

$$\Upsilon = \tan(n\pi + \pi T f_o) = \tan(\pi T f_o). \tag{57}$$

The last equality comes from the tangent function periodicity, specifically $\tan(\alpha) = \tan(k\pi + \alpha)$ for any $k \in \mathbb{Z}$ and $\alpha \in \mathbb{R}$. However, if $M = 2n + 1$, thus, (56) becomes

$$\begin{aligned}\Upsilon &= \tan\left(n\pi + \frac{\pi}{2} + \pi T f_o\right) \\ &= \tan\left(\frac{\pi}{2} + \pi T f_o\right) = -\cot(\pi T f_o) \\ &= -\frac{1}{\tan(\pi T f_o)}.\end{aligned} \tag{58}$$

From the result of (58), we can derive (59)

$$\tan(\pi T f_o) = -\frac{1}{\Upsilon}. \tag{59}$$

As a result, the CFO (f_o) estimation depends on whether M is even or odd; thus, the total frequency is computed in line with (63). In addition, since the estimation of f_o depends on the inverse of tangent, the calculation is bounded by its range, which is $|\pi/2|$, in accordance with

$$|\pi T f_o| < \frac{\pi}{2} \implies |f_o| < \frac{1}{2T}. \tag{60}$$

Consequently, in order for Algorithm 4 to effectively estimate f_o, it is crucial that its absolute value be less than half of the sampling frequency of the Mth antenna.

E. Fixed-Point Numerical Methods

Accurate computation of trigonometric functions relies on the Taylor series [49], which is a mathematical technique used to approximate a function by utilizing an infinite sum of expressions derived from the function's derivatives at a single point. However, in practical computation, the Taylor series is truncated and represented with a finite number of components. For instance, (64) expresses such series for the inverse of a tangent when zero is the point where the derivatives are considered

$$\arctan(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}. \tag{64}$$

Such a series is known as the Maclaurin series, a special and well-established practical case of the Taylor series that employs successive derivatives of the function at point zero. However, we have discovered that implementing directly such a series using fixed-point representation is not feasible without any supporting methods. The reason is that the terms of the series converge to zero after just a few iterations when $|x| < 1$, leading to a highly inaccurate approximation as $|x|$ increases. This issue mainly arises from the power component (x^{2n+1}), which causes the terms to become very small. In the fixed-point representation of the implemented solution ($Q15.16$), the resolution, which is about 1.5×10^{-5}, is insufficient to accurately represent these diminishing terms. Furthermore, for large x, the power component (x^{2n+1}) rapidly becomes greater than the upper bound of Q15.16. Other trigonometric functions suffer from a similar effect.

To address these limitations, certain fixed-point libraries constrain the domain range of the function and incorporate techniques such as lookup tables or a combination of the Taylor series and lookup tables [50], [51]. In cases where values fall outside the truncated domain, computations are performed using trigonometric identities. Such technique is implemented in well-established fixed-point libraries such

Algorithm 4 Summary of BLE Frequency Estimator

Input: matrix $\mathbf{Z}$ composed of samples from the M-th antenna.

Output: Estimated frequency $\widehat{f_T}$.

1) Compute the real-valued signal subspace, as the dominant eigenvector of

$$\mathbf{C} = \mathrm{Re}\{\mathbf{Q}_M^H\, \widehat{\mathbf{R}}_{zz}\, \mathbf{Q}_M\}. \tag{61}$$

by applying the Power Method.

2) Solve analytically the real-valued invariance equation below for Υ,

$$\mathbf{K_1}\mathbf{U}_s\Upsilon \approx \mathbf{K_2}\mathbf{U}_s, \tag{62}$$

by applying TLS.

3) Estimate the frequency in Hertz, that is,

$$\widehat{f_T} = f_\Delta + \begin{cases} \dfrac{\arctan(\Upsilon)}{\pi T}, & \text{if } M \text{ is even} \\ \dfrac{\arctan\left(-\dfrac{1}{\Upsilon}\right)}{\pi T}, & \text{if } M \text{ is odd.} \end{cases} \tag{63}$$

TABLE IV
FIXED-POINT APPROXIMATION OF TRIGONOMETRIC FUNCTIONS AND THEIR PERFORMANCE PARAMETERS

Function	Approximation	RMSE (radians)	Maximum Absolute Numerical Error (radian)
$\arctan(x)$	$\begin{cases} \text{sign}(x)\left(\frac{\pi}{2}\right) - \arctan\left(\frac{1}{x}\right), & \text{if} \quad \lvert x\rvert > 1 \\ \frac{x\pi}{4} - x(\lvert x\rvert - 1)(0.2447 + 0.0663\lvert x\rvert), & \text{otherwise} \end{cases}$	0.0011	0.0015
$\tan(x)$	$\begin{cases} \text{sign}(x)\left(\frac{1}{\tan\left(\frac{\pi}{2} - \lvert x\rvert\right)}\right), & \text{if} \quad \lvert x\rvert > \frac{\pi}{4} \\ \text{sign}(x)\left(\frac{\lvert x\rvert^3 - 15\lvert x\rvert}{6x^2 - 15}\right), & \text{otherwise} \end{cases}$	6.16e-05	0.0002
$\arcsin(x)$	$\arctan\left(\frac{x}{\sqrt{1-x^2}}\right)$	0.0011	0.0015

as CMSIS DSP Software Library, which was developed by Arm, IQ—Math Library, which was implemented by Texas Instruments, and Fix32 [13], [52], [53]. However, the lookup table technique alone lacks our desired accuracy and requires substantial memory to reduce numerical errors, even when merged with the Taylor series.

In our implemented solution, we have opted for the Padé approximant [54] to compute the tangent and Lagrange interpolation with minimax optimization [49] to compute the inverse of tangent and the inverse of sine since these techniques attain sufficient accuracy and are more time-efficient than the Taylor series as they employ noniterative simple equations. Notably, Padé approximant of order [3/2] was generated using MATLAB to estimate the tangent function in the range of $|x| < \pi/4$, while a trigonometric identity is computed for values falling outside that range. Table IV shows the trigonometric functions with their approximations employed by the implemented solution.

In situations where embedded processors lack an FPU or have a disabled FPU, the built-in hardware square root operation is unavailable. Thus, the implemented solution must resort to a software implementation of the square root tailor-made for fixed-point representation. Interestingly, certain processors may offer an integer square root operation that can be utilized without requiring an FPU, which may allow for the application of fixed-point square root [55]. However, this is not the case for embedded processors widely used in constrained IoT devices, namely, Arm Cortex M4 or Arm Cortex M33, where such built-in operation is unavailable as well. The implemented solution incorporates the Babylonian method for finding square roots [56]. This iterative method exhibits quadratic convergence and has been found to consistently converge in our fixed-point format. Given a real number $a \in \mathbb{R}$, the Babylonian method calculates the square root of a by iteratively computing

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right). \tag{65}$$

F. L-Shaped BLE ESPRIT

In conclusion, Algorithm 5 provides an overview of the implemented solution. Essentially, it comprises three modified Unitary TLS ESPRIT algorithms. One algorithm is dedicated to estimating the frequency using samples from the Mth antenna, while the other two are utilized to compute the u_1 and v_1 values from the x-axis and y-axis ULAs. Note that arctan2$(\cdot)$ is the four-quadrant inverse tangent.

Algorithm 5 Summary of L-Shaped BLE ESPRIT

Input: matrix $\begin{bmatrix}\mathbf{X}_s \\ \mathbf{Y}_s\end{bmatrix}$ composed of N array observations and the number of signals $d = 1$.

Output: Azimuth (θ) and elevation (ϕ) angles.

1) Estimate f_T through Algorithm 4.
2) Estimate u_1 and v_1 by applying Algorithm 3 two times, one on $\mathbf{X}_s$ and another on $\mathbf{Y}_s$, respectively.
3) Estimate the azimuth and elevation angles

$$\theta = \text{arctan2}(u_1 + jv_1),$$
$$\phi = \arccos\left(\sqrt{u_1^2 + v_1^2}\right).$$

VIII. EXPERIMENTS

The objective of this experiment consisted of demonstrating the feasibility of the implemented solution for commercial embedded IoT devices and validating it experimentally in a simulated indoor environment encompassing multipath propagation, fading, noises, and CFO. The experiment involved comparing the performance of the fixed-point implementation alongside the floating-point implementation, both with and without the support of an FPU. By conducting this comparison, we sought to evaluate the impact of using fixed-point arithmetic versus floating-point arithmetic and the influence of the FPU on the overall performance of the solution. The results obtained from this analysis not only serve as evidence of the

TABLE V
SIMULATION PARAMETERS IN MATLAB

Parameter	Characteristic
Antenna array type	L-shaped array
Antenna type and frequency ranges	Isotropic of 2 GHz–3 GHz
Distance between antennas (Δ)	$\lambda/2$
Frequency deviation (f_Δ)	250 kHz
Carrier frequency offset (f_o)	$[-30, +30]$ kHz
Center carrier frequency (f_c)	2.4 GHz
Sampling frequency	1 MHz
Channel model	TDL-E + AWGN
Number of antennas	7

solution's effectiveness but also provide valuable insights for making informed decisions about the implementation approach in real-world scenarios.

In summary, in our experiment, we generated artificial baseband signals in MATLAB, which served as the input for our solution implemented in the C99 programming language. The solution was executed on a resource-constrained embedded IoT device. Therefore, we could measure its memory footprint, execution time, energy consumption, and accuracy. To ensure maximum reliability, the device ran the implemented solution without any operating systems or software layers; it was programmed at the bare-metal level. Fig. 9 illustrates an overview of the experiment.

A. Experimental Setup

The simulation generates artificial baseband signals employing a combination of MATLAB's 5G Toolbox, Phased Array System Toolbox, and Communication Toolbox. Table V shows the parameters of the simulation. To accurately represent indoor environments, we employed the tapped delay line (TDL-E) channel model that accounts for LOS propagation and simulates the multipath propagation phenomenon in addition to the AWGN. The simulation randomly generates a CFO based on a Gaussian distribution within the range of ± 30 kHz. These CFO values were based on empirical experiments conducted in [27], which estimated that 99% of CFO values in the CTE fell within this interval. For the antenna array configuration, we utilized an L-shaped array composed of seven isotropic antennas, with five antennas forming each ULA. This configuration is suitable for IoT devices, considering its small size. The distance between antennas was set to half the wavelength of the BLE carrier frequency.

To measure the memory footprint (RAM and flash), execution time, energy consumption, and accuracy, we employed an nRF52840 DK that comes with an nRF52840 SoC having an Arm Cortex-M4 of 64 MHz with an FPU. The SoC did not use an operating system or software layers. Table VI summarizes the implementation parameters. For fixed-point implementation, the floating-point coprocessor was disabled in the Co-Processor Access Control Register (CPACR) following the reference manual [57] and illustrated in Fig. 10. For the

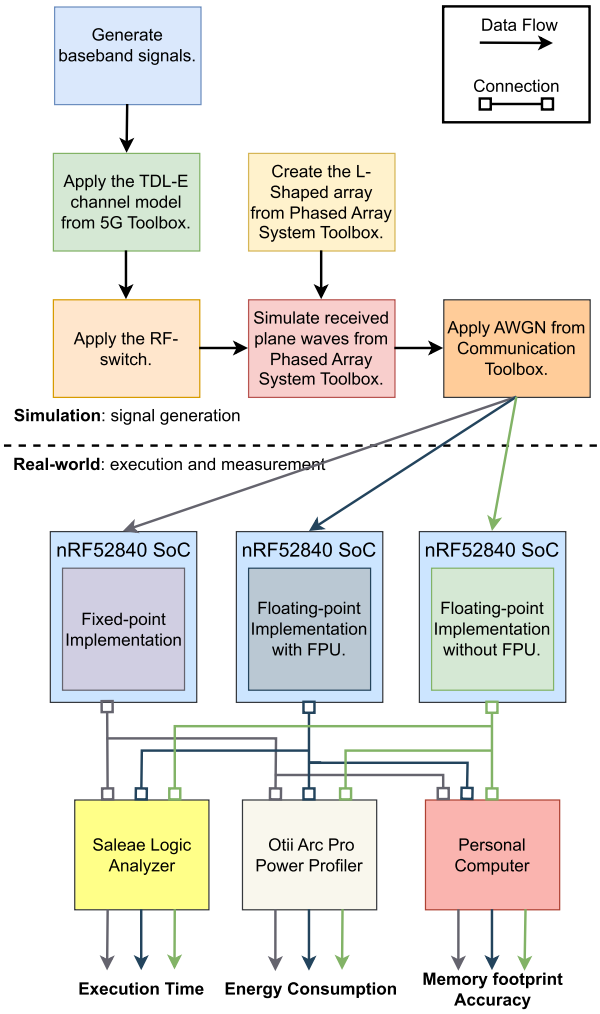


Fig. 9. Overview of the experiment.

TABLE VI
IMPLEMENTATION PARAMETERS

Parameter	Characteristic
Programming Language	C99
Compiler Optimization	-O1
Device	nRF52840 DK
Operating System	None
Fixed-Point Format	Q15.16
Floating-Point Format	Single Precision (FP32)

floating-point implementation, the floating-point coprocessor was activated, and the hardware floating-point instructions and hardware floating-point linkage (*-mfloat-abi=hard*) were activated as well, so the processor could fully operate the FPU. All devices of the nRF52 series support BLE, and although nRF52840 does not have Bluetooth Direction Finding capability, it is almost identical to other nRF52 and nRF53 devices that do have it. Notably, the nRF52 and nRF53 devices are a well-known series of constrained IoT devices developed by Nordic Semiconductor with a radio module of Bluetooth

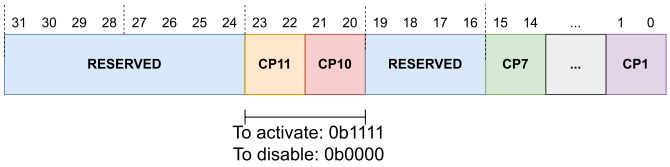


Fig. 10. Depiction of the CPACR that activates and disables the FPU.

5.1 or later versions and come with an Arm Cortex-M4 or Arm Cortex-M33 embedded processor.

The implemented solution used the 32-bit signed-fixed point in Q15.16 format since both embedded processors, namely, Arm Cortex M4 and Arm Cortex M33, operate natively in 32-bit format only. For its floating-point implementation, it employed the single-precision floating-point (FP32), under IEEE 754-2008 specification since the FPU of Arm Cortex-M4 does have support for FP32 only. Hence, floating-point operations with FP32 attain the fastest execution time as empirically shown in [42] and achieve the same accuracy as the other two floating-point formats. Although the cited embedded processors also support half-precision floating-point (FP16), it is utilized solely as a storage format for Arm Cortex-M4.

When operating in FP16, the processor promotes the data to FP32 before performing calculations and demotes it back to FP16 afterward [58]. This process introduces a small overhead that can increase both flash consumption and execution time. Slower execution times can result in higher energy consumption as well. Furthermore, the FPU of both embedded processors lacks support for double-precision floating-point (FP64) calculations. Therefore, the C compiler emulates FP64 computations [17], [59], leading to additional computations and a significant increase in execution time and flash usage. In fact, DOA methods using FP64 were found to be approximately 20 times slower than those using FP32, as reported in [42].

To accurately measure energy consumption, we employed the Otti Arc, a power measurement tool connected to the nRF52840 DK. For measuring the execution time, we utilized the Saleae logic analyzer. However, in both measurements, it was necessary to activate a general-purpose input/output (GPIO) port. Therefore, a GPIO signal was set to a high state when the implemented solution started to execute and set to a low state when it finished. This allowed us to determine the start and end points of the method and facilitate the accurate measurement of both energy consumption and execution time. However, due to the activation of the GPIO port, energy usage may be slightly overestimated.

While the clock cycles for arithmetic operations supported by the FPU are readily available in [60], the execution times for fixed-point operations are intricately tied to the custom assembly code either manually implemented or generated by the C compiler. Consequently, we conducted measurements to determine the clock cycles required for each fixed-point arithmetic operation. Table VII shows the results. Fixed-point addition and subtraction demonstrate efficiency, necessitating just a single clock cycle each, as they utilize native Arm Cortex M4 integer addition and subtraction instructions.

TABLE VII
CLOCK CYCLES FOR EACH OPERATION

Operation	Floating-Point Unit (clock cycles)	Fixed-Point (clock cycles)
Addition	1	1
Subtraction	1	1
Multiplication	1 – 3	11
Division	14	>21

In contrast, fixed-point multiplication and division involve a more intricate set of instructions, elucidated in the subsequent subsection. The clock cycle measurements for these operations were obtained through a detailed manual analysis of the assembly code, generated by the C compiler, cross-referenced with their corresponding clock cycle for each assembly instruction reported in the Cortex-M4 Technical Reference Manual [61]. Fixed-point division stands out as a particularly intricate operation. It invokes two functions from the ARM Embedded Application Binary Interface, specifically __aeabi_ldivmod.S [62] and udivmoddi4.c [63]. Unfortunately, due to the complexity and extensive code size of the latter function, manual measurement of its clock cycles proved unfeasible. Nonetheless, we estimated that the remaining portion of the fixed-point division operation consumes approximately 21 clock cycles.

Moreover, we calculated the root mean square error (RMSE) of accuracy considering 500 azimuth–elevation pairs for each signal-to-noise ratio (SNR). In mathematical terms, the RMSE of accuracy is defined as

$$\mathrm{RMSE}(\theta,\phi)=\sqrt{\left(\frac{1}{L}\right)\sum_{i=1}^{L}((\theta_i-\widehat{\theta}_i)^2+(\phi_i-\widehat{\phi}_i)^2)} \tag{66}$$

where $L=500$ is the number of estimated azimuth–elevation pairs, and θ and $\widehat{\theta}$ are the actual azimuth and its estimation in degrees, respectively, similarly for the elevation (ϕ) angle. We evaluated the performance of different pairs under various SNR conditions, specifically at 10, 15, 20, 25, and 30 dB. In total, we analyzed a dataset comprising 2500 pairs. Note that Bluetooth typically requires a minimum SNR of 10–15 dB for reliable operation. SNR values below 10 dB are likely to result in receiver failures during decoding and subsequent cyclic redundancy check failures as well [64]. Note that the RMSE of frequency follows a similar equation, namely:

$$\mathrm{RMSE}(f_T)=\sqrt{\left(\frac{1}{L}\right)\sum_{i=1}^{L}(f_{T,i}-\widehat{f}_{T,i})^2} \tag{67}$$

where $f_{T,i}$ and $\widehat{f}_{T,i}$ are the actual frequency and its estimation in Hz, respectively.

B. Results and Discussion

Fig. 11 depicts the angle accuracy comparison between the floating-point implementation [see Fig. 11(a)] and the fixed-point implementation [see Fig. 11(b)]. Initially, we observe

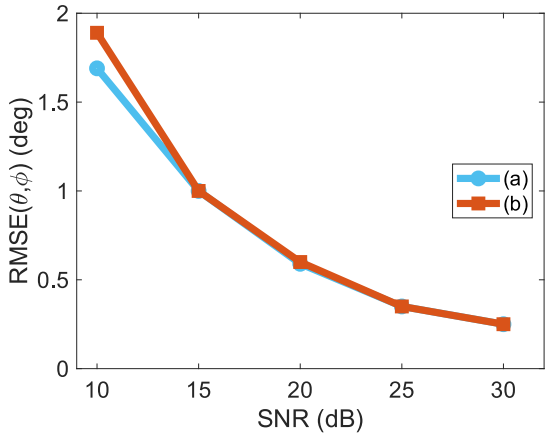


Fig. 11. Visualization of (a) accuracy of the floating-point implementation and (b) fixed-point implementation.

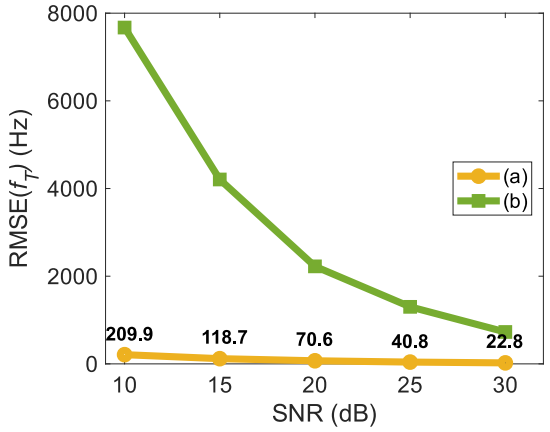


Fig. 12. Visualization of (a) accuracy of the frequency estimator implemented in this research and (b) phase differences average.

a marginal difference in accuracy between the two implementations. However, as the SNR increases, the fixed-point implementation quickly catches up and demonstrates comparable accuracy to its floating-point counterpart. Note that plot Fig. 11(a) refers to software- and hardware-based floating-point methods since they have the same accuracy. Fig. 12 presents the frequency accuracy comparison between the phase differences average approach [see Fig. 12(b)] suggested by Silicon Labs and Bluetooth, and the implemented solution [see Fig. 12(a)] in this research. This suggested that the solution utilizes only eight samples that constitute the reference period. On the other hand, our implemented solution is not limited to the reference period and utilizes 20 samples along with an improved estimator (see Algorithm 4), as clearly demonstrated in the experiment.

Table VIII presents the performance metrics of the implemented solution using different numerical representations: fixed-point, floating-point with FPU support, and floating-point without FPU support. The floating-point implementation without FPU relies totally on software-based floating-point computations, resulting in significantly slower execution times compared to both the FPU-supported floating-point and fixed-point implementations, which benefit from hardware-based calculations. Interestingly, the floating-point implementation

TABLE VIII
PERFORMANCE METRICS

Implementation	Execution Time (ms)	Energy Usage (nWh)	Flash (kB)	RAM (kB)
Floating-point without FPU	10.1	2.05	16.2	5.9
Floating-point with FPU	1.5	0.319	15.5	5.9
Fixed-point format	2.3	0.348	19.9	5.9

with FPU support showcases the fastest performance, even surpassing the fixed-point solution.

This can be attributed to the fact that while fixed-point addition and subtraction operations require only one clock cycle, fixed-point multiplication needs 11 cycles, and division operations take a long time surpassing 21 cycles, exceeding the time required for floating-point multiplication and division computations, as indicated in Table VII. It is a consequence of the 32-bit fixed-point multiplication and division computations as they require the utilization of 64-bit integers as an intermediate step [16]. However, because the Arm Cortex M4 and M33 architectures lack native support for 64-bit calculations, these operations introduce a slight overhead as they are executed in software. Section VIII-C provides a more detailed explanation.

From Table VIII, it can be observed that the fixed-point implementation utilizes slightly more flash memory compared to its two counterparts. This is because the hardware-based floating-point implementation requires only a single line of assembly code to perform multiplication and division operations, while the fixed-point implementation necessitates multiple lines of code, resulting in a small increase in flash memory usage, where the programming instructions are stored. Surprisingly, this increment is even higher than that of the software-based floating-point solution. In terms of RAM consumption, since all implementations utilize four bytes of data (FP32 and Q16.15), they exhibit the same RAM usage, as RAM stores the programming data. Notably, those memory consumption satisfies the requirements for devices in Table I.

We observed a clear linear relationship between energy usage and execution time. Specifically, the software-based floating-point implementation exhibited a 6.73 times slower execution time compared to its hardware-based counterpart, resulting in approximately 6.42 times higher energy consumption. In line with this trend, the fixed-point implementation, being 1.53 times slower than the hardware-based approach, would be expected to consume around 1.53 times more energy. However, interestingly, this was not the case. The fixed-point implementation only consumed energy 1.09 times higher than the hardware-based alternative. We attribute this energy-saving effect to the absence of the FPU, which was disabled in the fixed-point solution.

Moreover, coin batteries, commonly utilized in small electronic devices, including constrained IoT ones, exhibit a

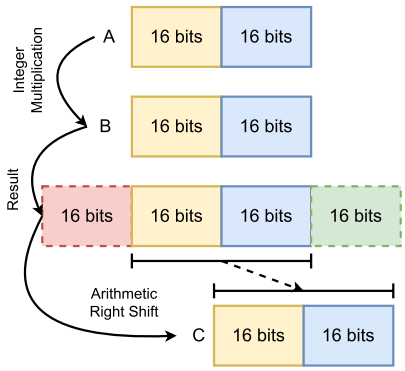


Fig. 13. Depiction of fixed-point multiplication, $C = A \times B$, in Q15.16 format.

capacity range of 1 to 2000 mAh.[1] Considering the fixed-point implemented solution as the sole source of energy consumption, the nRF52 series can operate approximately from about 9.5 million to 6.2 trillion cycles. Thus, the implemented solution is well-suited for battery-powered small embedded devices. However, it is important to note that the experiment did not measure the energy usage of the RF front end. This limitation stems from our setup's absence of a real array of antennas as we artificially generated the signals. Consequently, in practice, we assume that the measurement was assessed after IQ sampling.

C. Fixed-Point Shortcomings

When performing fixed-point multiplication, the intermediate result of the operation in parentheses, $I_a \times I_b$ (in Table III), requires storage as a 64-bit signed integer. This is because a 32-bit integer is insufficient to represent the resulting value. Let us consider an example to illustrate this. Suppose that we have two fixed-point numbers in Q15.16 format, $f_a = 40$ and $f_b = 3$, which are viewed from the computer perspective as signed integers $I_a = 2621440_{10}$ and $I_b = 196608_{10}$, respectively. To calculate the fixed-point multiplication, we perform a signed integer multiplication of I_a and I_b, resulting in 515396075520_{10} as an intermediate step. This value exceeds the maximum positive number that can be represented by a 32-bit signed integer, which is $2^{31}-1$.

Consequently, to accommodate the result, a 64-bit signed integer is required. In the next step, we perform a right shift operation of 515396075520_{10} by 16 bits, denoted as $515396075520_{10} \gg 16$, yielding 7864320_{10}. This value, represented in Q15.16 format, corresponds to 120, and it fits within a 32-bit signed integer, effectively restoring the result to the original format. Fig. 13 illustrates the fixed-point multiplication process. A similar analysis can be applied to the fixed-point division operation as well.

To reduce the overhead associated with fixed-point multiplication and division, some developers implement their operations in assembly code instead of relying on the compiler's implementation in C. While fixed-point multiplication is generally efficient requiring a few lines of assembly

[1]According to Mouser Electronics, see: https://www.mouser.com/c/power/batteries/coin-cell-battery.

```
; r0,r1,r4 = registers for fixed-point number A,
; B and A x B respectively.
; Multiply r0 x r1, and place the lower 32 bits
; in r2 and the higher 32 bits in r3
SMULL r2, r3, r0, r1
; Logical Shift Left Signed
; Equivalent to r3 = r3 << 16 in C
LSLS r3, r3, #16
; Logical shift right Signed.
; Equivalent to r2 = r2 >> 16 in C
LSRS r2, r2, #16
; Combine two halfwords into a word
; Equivalent to r4 = r2 | r3 in C
ORR r4, r2, r3
```

Code. 1: Example of a Q15.16 Fixed-Point Multiplication

code, as shown in the provided example (see Code 1), division can be time-consuming and complicated. To address this issue, the division operation often employs the Newton–Raphson method to estimate the reciprocal of the divisor (b) in the fraction a/b, resulting in $c = 1/b$. This transforms the division operation into a multiplication operation, namely, $a \times c$. The application of the Newton–Raphson method for reciprocal estimation is feasible because it only requires multiplication and addition operations to find the reciprocal. The Newton–Raphson method is an iterative approach that utilizes linear approximation to find increasingly accurate approximations to the roots of a real-valued function. In the case of finding the reciprocal of a real number b, the reciprocal can be obtained as a zero of the function defined as

$$f(x) = \frac{1}{b} - x. \tag{68}$$

By repeatedly applying (69) in each iteration, the method calculates the root, which represents the reciprocal

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i(2 - bx_i). \tag{69}$$

However, it is crucial to select an appropriate initial point when applying the Newton–Raphson method. Choosing an improper initial point can result in the method either failing to converge or producing an incorrect estimation. In addition, it is important to set a maximum number of iterations to ensure the termination of the method. The specific value for the maximum number of iterations can be determined through empirical analysis and fine-tuning. Furthermore, to expedite fixed-point division, certain implementations employ a combination of a lookup table and the Newton–Raphson method [52], [65]. This approach can enhance the efficiency of the division operation by leveraging precomputed values stored in a table and utilizing the Newton–Raphson method to refine the approximation further. However, it is important to note that this research did not include the development of custom fixed-point multiplication and division operations. Instead, it relied on the C compiler's run-time library.

Nevertheless, there are approaches available to mitigate the impact of fixed-point division operations and reduce their frequency in firmware. One such approach involves leveraging the properties of fixed-point numbers, particularly when the divisor is equal to 2^m, where m is a positive integer. In such cases, an arithmetic shift operation can be employed, which

can be executed in an assembly instruction that takes one single clock cycle. This operation involves right-shifting the dividend a by m bits, so the division operations become $a \gg m$. The use of arithmetic shift is possible because fixed-point numbers are essentially integer representations from the computer's perspective.

Furthermore, if multiple divisions with the same divisor b are performed successively in a loop, it is time-efficient to precalculate the reciprocal of b, denoted as $c = 1/b$, before entering the loop. This allows the successive divisions to be transformed into simple multiplications, which are faster operations. Nevertheless, it is worth noting that modern C compilers are often equipped with intelligent optimizations that may automatically apply such approaches to improve code performance.

D. Limitations

Our implemented solution encounters three primary limitations, each presenting an opportunity for enhancement. The first involves the use of the C compiler's runtime library for fixed-point multiplication and division, which is potentially inefficient for our specific needs. A custom assembly code implementation of these operations could improve the execution time of our implemented solution leading to reduced energy consumption. Notably, integrating the Newton–Raphson method with lookup tables seems a promising strategy for improving fixed-point division, as outlined in Section VIII-C.

The second limitation comprises the matrix multiplication. Although our implemented solution bypasses the time-consuming eigenvalue decomposition by employing the power method, it still depends on the brute-force approach for matrix multiplication in (61). This approach entails multiplying each row of one matrix with each column of the other. Although there are fast matrix multiplication algorithms available, such as the Strassen or Coppersmith–Winograd methods, their practical implementation presents considerable challenges. More specifically, these cited advanced algorithms, while theoretically offering quicker asymptotic complexity, are impeded in real-world applications by their high constant factors rendering them impractical for matrices of a size manageable by constrained embedded systems [66]. In addition, these algorithms increase memory consumption and are complex to implement, posing further limitations to their usability in many practical scenarios.

The third limitation concerns the array response. Implementing a DOA method for practical antenna array systems typically benefits from array calibration to capture the actual array response and hardware imperfections. Although our implemented solution does not require a precise array response such as MUSIC, incorporating an array calibration technique could enhance its accuracy.

IX. Conclusion

This research confirmed the effectiveness of the implemented solution for Bluetooth-enabled IoT devices. Our study's cornerstone comprised the introduction of a novel fixed-point DOA method. This method, deeply rooted in ESPRIT, diverges from conventional practices by employing a simpler power method for DOA estimation. It incorporates an enhanced carrier frequency estimator, which is tens of times more accurate than the conventional method of averaging phase differences. In addition, the research highlighted the need for developing new numerical methods, specifically designed for fixed-point arithmetic, as prevalent floating-point techniques may prove inadequate or inefficient when directly translated to a fixed-point format.

Our research demonstrated that the fixed-point approach substantially reduced execution time, clocking in at a swift 2.3 ms, alongside a low energy consumption of just 0.348 nWh. In direct comparison, this method significantly surpassed the software-based floating-point alternative, exhibiting a substantial 5.9-fold boost in energy efficiency and a 4.4-fold acceleration in processing speed. Notably, this was achieved with virtually no compromise on accuracy. The empirical findings of this evaluation definitively conclude that software-based floating-point computations lag in efficiency when juxtaposed with their fixed-point counterparts.

Our research also revealed that for IoT devices equipped with an FPU, the hardware-based floating-point algorithm still holds a slight edge in terms of speed and energy efficiency compared to its fixed-point alternative. This is because the hardware-based floating-point approach, benefiting from direct hardware support, accomplishes faster mathematical operations. In conclusion, while the fixed-point method significantly enhances the execution in systems lacking an FPU, it is the hardware-based floating-point technique that prevails in devices where such a computational resource is present.

References

[1] M. Pesavento, M. Trinh-Hoang, and M. Viberg, "Three more decades in array signal processing research: An optimization and structure exploitation perspective," *IEEE Signal Process. Mag.*, vol. 40, no. 4, pp. 92–106, Jun. 2023.

[2] Bluetooth SIG. (2015). *Indoor Positioning Service 1.0*. Accessed: Jan. 27, 2023. [Online]. Available: https://www.bluetooth.com/specifications/specs/indoor-positioning-service-1-0/

[3] S. R. Jondhale, R. S. Deshpande, S. M. Walke, and A. S. Jondhale, "Issues and challenges in RSSI based target localization and tracking in wireless sensor networks," in *Proc. Int. Conf. Autom. Control Dyn. Optim. Techn. (ICACDOT)*, Sep. 2016, pp. 594–598.

[4] Bluetooth SIG. (2019). *Bluetooth Core Specification V5.1*. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-5-1/

[5] Bluetooth SIG. (2019). *Enhancing Bluetooth Location Services With Direction Finding*. Accessed: Jan. 27, 2023. [Online]. Available: https://www.bluetooth.com/wp-content/uploads/2019/03/1901_Enhancing-Bluetooth-Location-Service_FINAL.pdf

[6] G. Pau, F. Arena, Y. E. Gebremariam, and I. You, "Bluetooth 5.1: An analysis of direction finding capability for high-precision location services," *Sensors*, vol. 21, no. 11, p. 3589, May 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/11/3589

[7] Quuppa. (2020). *Bluetooth Direction Finding: Going Beyond Beacons*. Accessed: Jan. 27, 2023. [Online]. Available: https://www.quuppa.com/bluetooth-direction-finding-going-beyond-beacons/

[8] P. Karlsson. *Getting Started With Bluetooth for High Precision Indoor Positioning*. Accessed: Jan. 27, 2023. [Online]. Available: https://content.u-blox.com/sites/default/files/Indoor-positioning-Getting-started-u-blox-WhitePaper.pdf

[9] Y. B. Zikria, H. Yu, M. K. Afzal, M. H. Rehmani, and O. Hahm, "Internet of Things (IoT): Operating system, applications and protocols design, and validation techniques," *Future Gener. Comput. Syst.*, vol. 88, pp. 699–706, Nov. 2018.

[10] Y. B. Zikria, S. W. Kim, O. Hahm, M. K. Afzal, and M. Y. Aalsalem, "Internet of Things (IoT) operating systems management: Opportunities, challenges, and solution," *Sensors*, vol. 19, no. 8, p. 1793, Apr. 2019.

[11] E. Anderson et al., *LAPACK Users' Guide*. Philadelphia, PA, USA: SIAM, 1999.

[12] C. Sanderson and R. Curtin, "Armadillo: A template-based C++ library for linear algebra," *J. Open Source Softw.*, vol. 1, no. 2, p. 26, Jun. 2016.

[13] Arm. *Common Microcontroller Software Interface Standard*. Accessed: Jul. 18, 2023. [Online]. Available: https://arm-software.github.io/CMSIS_5/General/html/index.html

[14] W. Elmenreich, A. Wolf, and M. Rosenblattl, "Providing standardized fixed-point arithmetics for embedded c programs," in *Intelligent Technical Systems*. Dordrecht, The Netherlands: Springer, 2009, pp. 101–114.

[15] Nordic Semiconductor. (2019). *Bluetooth Direction Finding*. Accessed: Jul. 28, 2023. [Online]. Available: https://www.nordicsemi.com/Products/Bluetooth-Direction-Finding

[16] Y. Zhu, "Fixed-point and floating-point arithmetic," in *Embedded Systems With ARM Cortex-M Microcontrollers in Assembly Language and C*. New York, NY, USA: E-Man Press, 2015, ch. 12, pp. 269–322.

[17] J. Yiu, *The Definitive Guide To ARM® Cortex®-M3 and Cortex®-M4 Processors*. Cambridge, U.K.: Newnes, 2013.

[18] M. L. Sollie, K. Gryte, T. H. Bryne, and T. A. Johansen, "Outdoor navigation using Bluetooth angle-of-arrival measurements," *IEEE Access*, vol. 10, pp. 88012–88033, 2022.

[19] L. Yao, "Bluetooth direction finding," M.S. thesis, Dept. Elect. Eng., Math. Comput. Sci., TU Delft Electr. Eng., Delft, The Netherlands, 2018.

[20] Q. Wan et al., "A high precision indoor positioning system of BLE AOA based on ISSS algorithm," *Measurement*, vol. 224, Jan. 2024, Art. no. 113801.

[21] N. Paulino, L. M. Pessoa, A. Branquinho, and E. Gonçalves, "Design and experimental evaluation of a Bluetooth 5.1 antenna array for angle-of-arrival estimation," in *Proc. 13th Int. Symp. Commun. Syst., Netw. Digit. Signal Process. (CSNDSP)*, Jul. 2022, pp. 625–630.

[22] Z. Hajiakhondi-Meybodi, M. Salimibeni, K. N. Plataniotis, and A. Mohammadi, "Bluetooth low energy-based angle of arrival estimation via switch antenna array for indoor localization," in *Proc. IEEE 23rd Int. Conf. Inf. Fusion (FUSION)*, Jul. 2020, pp. 1–6.

[23] H. Ye, B. Yang, Z. Long, and C. Dai, "A method of indoor positioning by signal fitting and PDDA algorithm using BLE AOA device," *IEEE Sensors J.*, vol. 22, no. 8, pp. 7877–7887, Apr. 2022.

[24] T. Troccoli et al., "Direction of arrival method for L-shaped array with RF switch: An embedded implementation perspective," *Sensors*, vol. 23, no. 6, p. 3356, Mar. 2023.

[25] Silicon Labs. (2021). *AN1297: Custom Direction-Finding Solutions Using the Silicon Labs Bluetooth Stack*. Accessed: Jan. 27, 2023. [Online]. Available: https://www.silabs.com/documents/public/application-notes/an1297-custom-direction-finding-solutions-silicon-labs-bluetooth.pdf

[26] M. Woolley. (2021). *Bluetooth Direction Finding A Technical Overview*. Accessed: Jul. 18, 2023. [Online]. Available: https://www.bluetooth.com/wp-content/uploads/Files/developer/RDF_Technical_Overview.pdf

[27] S. Cloudt, "Bluetooth low energy direction finding on embedded hardware by mitigating carrier frequency offset and multipath fading," Ph.D. thesis, Dept. Elect. Eng., Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2021.

[28] Z. Chen, G. Gokeda, and Y. Yu, "Overview of basic DOA estimation algorithms," in *Introduction to Direction-of-Arrival Estimation*. Norwood, MA, USA: Artech House, 2010, ch. 3, pp. 31–63.

[29] R. Roy and T. Kailath, "ESPRIT-estimation of signal parameters via rotational invariance techniques," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 7, pp. 984–995, Jul. 1989.

[30] E. Tuncer and B. Friedlander, *Classical and Modern Direction-of-Arrival Estimation*. New York, NY, USA: Academic, 2009.

[31] B. Ottersten, M. Viberg, and T. Kailath, "Performance analysis of the total least squares ESPRIT algorithm," *IEEE Trans. Signal Process.*, vol. 39, no. 5, pp. 1122–1135, May 1991.

[32] Z. Chen, G. Gokeda, and Y. Yu, "DOA estimations with ESPRIT algorithms," in *Introduction to Direction-of-Arrival Estimation*. Norwood, MA, USA: Artech House, 2010, ch. 5, pp. 81–122.

[33] A. M. Zoubir, "Subspace methods and exploitation of special array structures," in *Academic Press Library in Signal Processing* (Array and Statistical Signal Processing), vol. 3. New York, NY, USA: Academic, 2014, ch. 15, pp. 651–717.

[34] V. Madisetti and D. Williams, "ESPRIT and closed-form 2-D angle estimation with planar arrays," in *Digital Signal Processing Handbook on CD-ROM*. Boca Raton, FL, USA: CRC Press, 1999, ch. 63.

[35] M. Haardt and J. A. Nossek, "Unitary ESPRIT: How to obtain increased estimation accuracy with a reduced computational burden," *IEEE Trans. Signal Process.*, vol. 43, no. 5, pp. 1232–1242, May 1995.

[36] E. Tuncer and B. Friedlander, "Narrowband and wideband DOA estimation for uniform and nonuniform linear arrays," in *Classical and Modern Direction-of-Arrival Estimation*. New York, NY, USA: Academic, 2009, ch. 4, pp. 125–160.

[37] C. R. Johnson and R. A. Horn, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1985.

[38] I. Markovsky and S. V. Huffel, "Overview of total least-squares methods," *Signal Process.*, vol. 87, no. 10, pp. 2283–2302, 2007.

[39] A.-J. Van Der Veen, C. M. Vanderveen, and A. Paulraj, "Joint angle and delay estimation using shift-invariance techniques," *IEEE Trans. Signal Process.*, vol. 46, no. 2, pp. 405–418, Feb. 1998.

[40] Bluetooth SIG. (2021). *Bluetooth Core Specification V5.3*. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-5-3/

[41] A. M. Zoubir, "DOA estimation methods and algorithms," in *Academic Press Library in Signal Processing* (Array and Statistical Signal Processing), vol. 3. New York, NY, USA: Academic, 2014, ch. 14, pp. 599–650.

[42] T. Troccoli, J. Pirskanen, A. Ometov, J. Nurmi, and V. Kaseva, "Fast real-world implementation of a direction of arrival method for constrained embedded IoT devices," in *Proc. 12th Int. Conf. Internet Things*, Nov. 2022, pp. 1–8.

[43] A. Gilat, *Numerical Methods for Engineers Scientists*. Hoboken, NJ, USA: Wiley Global Education, 2013.

[44] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[45] W. Ford, "The algebraic eigenvalue problem," in *Numerical Linear Algebra With Applications*. Boston, MA, USA: Academic, 2015, ch. 18, pp. 379–438.

[46] J. Solomon, "Designing and analyzing linear systems," in *Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics*. Boca Raton, FL, USA: CRC Press, 2015, ch. 4, pp. 75–76.

[47] W. Ford, "The symmetric eigenvalue problem," in *Numerical Linear Algebra With Applications*. Boston, MA, USA: Academic, 2015, ch. 19, pp. 439–465.

[48] G. Strang, *Linear Algebra and Its Applications*. Boston, MA, USA: Cengage Learning, 2006.

[49] S. Rajan, S. Wang, R. Inkol, and A. Joyal, "Efficient approximations for the arctangent function," *IEEE Signal Process. Mag.*, vol. 23, no. 3, pp. 108–111, May 2006.

[50] M. R. D. Rodrigues, J. H. P. Zurawski, and J. B. Gosling, "Hardware evaluation of mathematical functions," *IEE Proc. E Comput. Digit. Techn.*, vol. 128, no. 4, p. 155, 1981.

[51] R. Lyons, "Another contender in the arctangent race," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 109–110, Jan. 2004.

[52] Warpco. (Mar. 2019). *Highly Optimized 32-Bit Fixed-Point Math Library for Embedded Systems*. [Online]. Available: https://github.com/warpco/fix32

[53] Texas Instruments. (2015). *IQmathLib Users Guid*. Accessed: Jul. 19, 2023. [Online]. Available: https://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/IQmathLib/latest/exports/MSP430-IQmathLib-UsersGuide.pdf

[54] S. Wu and G. Bercu, "Padé approximants for inverse trigonometric functions and their applications," *J. Inequal. Appl.*, vol. 2017, no. 1, pp. 1–12, Dec. 2017.

[55] Arm Ltd. (1996). *Fixed Point Arithmetic on the Arm*. Accessed: Aug. 2, 2023. [Online]. Available: https://developer.arm.com/documentation/dai0033/latest/

[56] O. Kosheleva, "Babylonian method of computing the square root: Justifications based on fuzzy techniques and on computational complexity," in *Proc. Annu. Meeting North Amer. Fuzzy Inf. Process. Soc. (NAFIPS)*, Jun. 2009, pp. 1–6.

[57] *ARMv7-M Architecture Reference Manual*, Arm Ltd., Cambridge, U.K., 2021.

[58] *Arm Compiler Armclang Reference Guide Version 6.12*, Arm Ltd., Cambridge, U.K., 2019.

[59] I. Johnson. (2022). *10 Useful Tips for Using the Floating Point Unit on the Cortex-M4*. Accessed: May 12, 2022. [Online]. Available: https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/10-useful-tips-to-using-the-floating-point-unit-on-the-arm-cortex–m4-processor
[60] *Floating Point Unit: Cortex-M4 Technical Reference Manual*, Arm Ltd., Cambridge, U.K., 2010, ch. 7, pp. 66–74.
[61] *Programmers Model: Cortex-M4 Technical Reference Manual*, Arm Ltd., Cambridge, U.K., 2010, ch. 3, pp. 26–49.
[62] Chandlerc. (2012). *Aeabi_uldivmod.s—EABI Uldivmod Implementation*. [Online]. Available: https://github.com/llvm/llvm-project/blob/main/compiler-rt/lib/builtins/arm/aeabi_uldivmod.S
[63] D. Boulby. (May 2018). *Udivmoddi4.c—Implement__udivmoddi4*. [Online]. Available: https://github.com/ARM-software/arm-trusted-firmware/blob/master/lib/compiler-rt/builtins/udivmoddi4.c
[64] A. Pipino, A. Liscidini, K. Wan, and A. Baschirotto, "Bluetooth low energy receiver system design," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 465–468.
[65] P. Curtis. (2021). *Algorithms for Division—Part 4—Using Newton's Method*. Accessed: Jul. 19, 2023. [Online]. Available: https://blog.segger.com/algorithms-for-division-part-4-using-newtons-method/
[66] I. McNeil, *Matrices Trigonometry*, 1st ed. Waltham Abbey, U.K.: EdTech, 2020.

Tiago Troccoli received the B.Sc. degree in computer science from the University of São Paulo (USP), São Paulo, Brazil, in 2017, and the M.Sc. degree in computer science from the State University of Campinas (Unicamp), Campinas, Brazil, in 2020.

After a short period as an embedded system engineer, he is currently an early stage researcher in Finland, pursuing a doctorate program focused on developing a novel indoor localization system based on radio direction finding. This research project addresses the challenges of implementing complex numerical methods in Internet-of-Things networks, particularly in constrained embedded systems. It involves a collaborative effort between the industry (WIREPAS Ltd., Tampere, Finland) and academia (Tampere University, Tampere), supported by the EU H2020 MSCA funding.

Juho Pirskanen received the M.Sc. degree in engineering (telecommunications) from the Tampere University of Technology, Tampere, Finland, in 2000.

He has over 20 years of experience in research and technology development on wireless radio technologies, such as 3G, high-speed packet access (HSPA), long term evolution (LTE), and wireless local-area network (WLAN), and the latest on different 5G technologies. In addition, he has participated actively for several years in different standardization forums, including 3GPP, IEEE802.11, and ETSI. Since 2017, after joining WIREPAS Ltd., Tampere, his research interests have been in wireless Internet-of-Things (IoT) systems and wireless mesh networks. He currently acts as an Editor and a Rapporteur of ETSI TS 103.636-4 DECT-2020 New Radio (NR), Part-4, MAC Layer Technical Specification.

Jorge Morte received the B.Sc. degree in telecommunication engineering from the University of Zaragoza, Zaragoza, Spain, in 2017, and the M.Sc. degree in telecommunication engineering from the Technical University of Madrid (UPM), Madrid, Spain, in 2019.

He has worked as a researcher in wireless communications, with a specific focus on physical and MAC layers in wireless systems. He is currently a Researcher of layer 2 and 3 protocols with WIREPAS Ltd., Tampere, Finland. His main research interests include wireless mesh networks and DECT NR+.

Aleksandr Ometov (Senior Member, IEEE) received the M.Sc. degree in information technology and the D.Sc. (Tech.) degree in telecommunications from the Tampere University of Technology (TUT), Tampere, Finland, in 2016 and 2018, respectively, and the Specialist degree in information security from the Saint Petersburg State University of Aerospace Instrumentation (SUAI), Saint Petersburg, Russia, in 2013.

He is a Senior Research Fellow with Tampere University (TAU), Tampere. He is working on EU H2020 MSCA A-WEAR and APROPOS projects and coordinating the CONVERGENCE of Humans and Machines research field. His research interests include wireless communications, information security, computing paradigms, distributed systems, and wearable applications.

Elena Simona Lohan (Senior Member, IEEE) the M.Sc. degree in electrical engineering from the Politehnica University of Bucharest, Bucharest, Romania, in 1997, the DEA degree (French equivalent of master) in econometrics from École Polytechnique, Paris, France, in 1998, and the Ph.D. degree in telecommunications from the Tampere University of Technology, Tampere, Finland, in 2003.

She is a Professor with the Electrical Engineering Unit, Tampere University, Tampere. Her current research interests include global navigation satellite system (GNSS), low earth orbit positioning, navigation and timing (LEO-PNT), indoor location techniques, and privacy-aware positioning solutions.

Ville Kaseva received the D.Sc. (Tech.) degree from the Tampere University of Technology (TUT), Tampere, Finland, in 2013.

He boasts a wealth of experience spanning both academic exploration and industrial application after nearly two decades of expertise in the field of low-power wireless mesh networks. He has been the CTO, a Co-Founder, and a Partner of WIREPAS Ltd., Tampere, since 2010, in which his primary areas of engagement involve the strategic development of the Wirepas mesh technology roadmap and the landscape of system architecture. His journey started with extensive research into mesh networks, a path that he pursued for several years at TUT.

Jari Nurmi (Senior Member, IEEE) received the D.Sc. (Tech) degree from the Tampere University of Technology, Tampere, Finland, in 1994.

He has been a Professor with the Electrical Engineering Unit, Tampere University [TAU; formerly the Tampere University of Technology (TUT)], Tampere, Finland, since 1999. He is working on embedded computing systems, system-on-chip, approximate computing, software-defined radio and networks, wireless localization, and positioning receiver implementations. He held various research, education, and management positions at TUT from 1987 to 1994 and was the Vice-President of the SME VLSI Solution Oy, Tampere, from 1995 to 1998. He is the Director of the National DELTA Doctoral Training Network of 200 Ph.D. students, a coordinator of the APROPOS European Doctoral Training Network, and the Head of the A-WEAR European Joint PhD Program at TAU.

Dr. Nurmi is a member of the Technical Committee on VLSI Systems and Applications of IEEE Circuits and Systems Society (CASS) and an associate editor of three international journals.