sensors

MDPI

Article

Direction of Arrival Method for L-Shaped Array with RF Switch: An Embedded Implementation Perspective †

Tiago Troccoli [1,2,*], Juho Pirskanen [2], Jari Nurmi [1], Aleksandr Ometov [1], Jorge Morte [2], Elena Simona Lohan [1] and Ville Kaseva [2]

[1] Faculty of Information Technology and Communication Sciences, Tampere University, 33720 Tampere, Finland
[2] WIREPAS Ltd., 33720 Tampere, Finland
* Correspondence: tiago.troccoli@wirepas.com or tiago.troccolicunha@tuni.fi
† This paper is an extended version of our paper published in Troccoli, T.; Pirskanen, J.; Ometov, A.; Nurmi, J.; Kaseva, V. Implementation of Embedded Multiple Signal Classification 672 Algorithm for Mesh IoT Networks. In Proceedings of the 2022 International Conference on Localization and GNSS (ICL-GNSS), Tampere, Finland, 7–9 June 2022; pp. 1–7.

Abstract: This paper addresses the challenge of implementing Direction of Arrival (DOA) methods for indoor localization using Internet of Things (IoT) devices, particularly with the recent direction-finding capability of Bluetooth. DOA methods are complex numerical methods that require significant computational resources and can quickly deplete the batteries of small embedded systems typically found in IoT networks. To address this challenge, the paper presents a novel Unitary R-D Root MUSIC for L-shaped arrays that is tailor-made for such devices utilizing a switching protocol defined by Bluetooth. The solution exploits the radio communication system design to speed up execution, and its root-finding method circumvents complex arithmetic despite being used for complex polynomials. The paper carries out experiments on energy consumption, memory footprint, accuracy, and execution time in a commercial constrained embedded IoT device series without operating systems and software layers to prove the viability of the implemented solution. The results demonstrate that the solution achieves good accuracy and attains an execution time of a few milliseconds, making it a viable solution for DOA implementation in IoT devices.

Keywords: Direction of Arrival (DOA); Multiple Signal Classification (MUSIC); Internet of Things (IoT); embedded systems; array signal processing

Citation: Troccoli, T.; Pirskanen, J.; Nurmi, J.; Ometov, A.; Morte, J.; Lohan, E.M.; Kaseva, V. Direction of Arrival Method for L-Shaped Array with RF Switch: An Embedded Implementation Perspective. *Sensors* **2023**, *23*, 3356. https://doi.org/10.3390/s23063356

Academic Editor: Chris Rizos

Received: 31 January 2023
Revised: 15 March 2023
Accepted: 17 March 2023
Published: 22 March 2023

1. Introduction

Direction of Arrival (DOA) estimation is an array signal processing application found in radars, navigation, military devices, and medical appliances [1]. Most recently, it has been applied in wireless communication systems such as Bluetooth to localize devices in indoor environments where satellite-based radio-navigation systems fail to do so. In 2015, Bluetooth released its positioning technology based on the Received Signal Strength (RSS) [2]. In that past technology, a Bluetooth Low Energy (BLE) tag transmitted radio frequency (RF) signals to receivers. The receivers could estimate the positions of such tags from RSS measurements. The positions were not precisely estimated, in fact, the tag could be anywhere inside a circular zone if trilateration was used, which is a typical RSS-based positioning method. According to Bluetooth itself, such technology has an accuracy of a few meters (1–10 m) [3], and even if more complex algorithms are employed, there are limitations on the achievable positioning accuracy. In some cases, such accuracy of a few meters is enough, but some occasions need higher accuracy, for example, machine navigation for autonomous mobile robots, drones, industrial automation, or navigation systems that guide people in indoor environments. Higher accuracy is also desirable in asset tracking, where factories track material workflow and hospitals track equipment location.

Most recently, Bluetooth has released a new capability that makes it possible to estimate DOAs [4], unlocking high accuracy as DOA-based positioning is reported to attain sub-meter-level position accuracy [3,5–7]. In the direction of arrival version of that new capability, receivers have an array of antennas to make it possible to estimate the azimuth and elevation angles coming from a signal emitted from a BLE tag. By applying, for example, the triangulation method, which employs the cited angles for each receiver and their positions, it is possible to estimate the location of BLE tags. Besides using for DOA estimation, BLE is broadly used for wireless communication of the Internet of Things (IoT). In the IoT terminology, its networks are composed of many nodes that are small battery-powered resource-constrained embedded systems. Particularly, in IoT networks that deploy DOA-based positioning systems, some nodes, called anchor nodes, have an array of antennas that execute DOA methods to estimate the azimuth and elevation of RF signals emitted from BLE tags which are also nodes. Figure 1 depicts an IoT mesh network with the cited positioning systems.

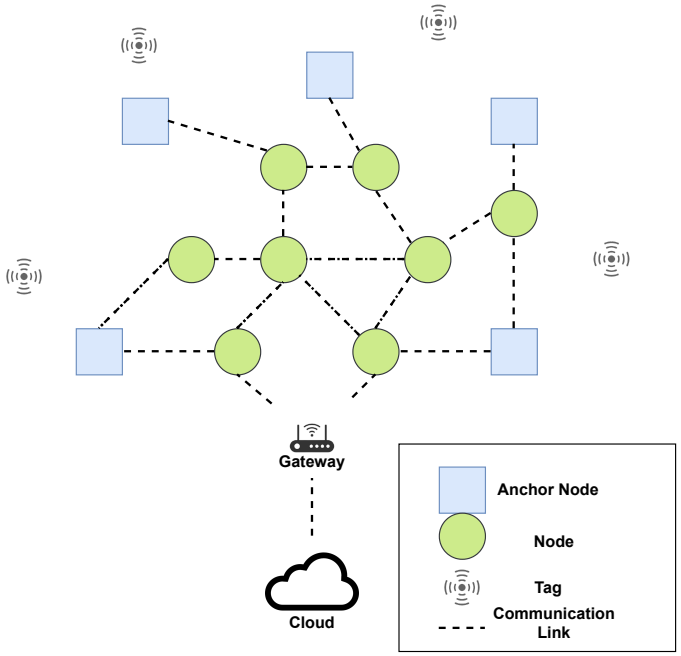

Figure 1. An IoT mesh network with a DOA-based positioning system.

In IoT networks, it is important to note that nodes are often constrained embedded systems. These systems typically consist of three distinct subsystems [8], as illustrated in Figure 2. The microcontroller or computing subsystem is responsible for controlling the node's functionalities, and runs instructions on a low-power processor that commonly operates at a few megahertz. It also has flash memory and RAM whose sizes are in order of kilobytes. Flash memory is a non-volatile memory that stores the program's instructions and constant data values, whereas RAM is a volatile memory that is used as data storage for the program while it is running. Nodes usually have a simple real-time operating system that executes all functionalities, also known as tasks, concurrently. The sensor subsystem collects data from natural phenomena in the form of analog signals via sensor readings and transforms them into digital signals using an Analog-to-Digital Converter (ADC). Finally, the communication subsystem is made up of a transceiver and normally a co-processor device that is responsible for transmitting and receiving data.

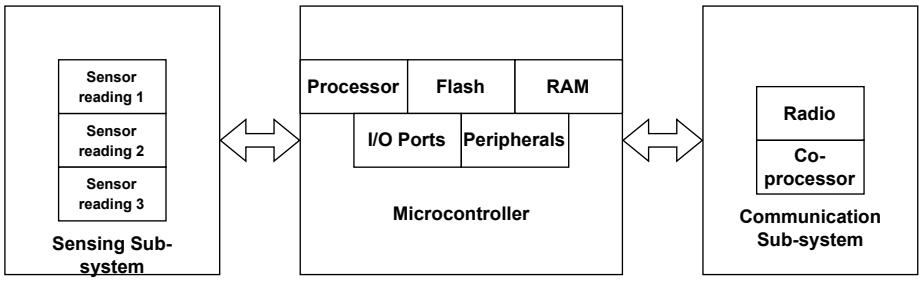

Figure 2. An overview example of a node's hardware architecture.

While Bluetooth specifies a protocol for transmitting a constant tone for DOA purposes, the development of direction of arrival algorithms is left to implementation. However, the implementation of DOA methods in IoT devices presents a significant challenge as these devices are often battery-powered and resource-constrained embedded systems with limited computational resources as previously mentioned. In contrast, DOA methods consist of complex numerical algorithms that are resource-intensive and time-consuming, potentially leading to rapid battery depletion, unacceptable execution times, and resource starvation. Moreover, IoT devices typically run a simple real-time operating system that executes small tasks concurrently, such as data acquisition from sensors and communication with other devices. The execution of DOA methods in such a multi-threaded environment is even more challenging, and computational resource management needs to be carefully thought out.Therefore, developing DOA algorithms for IoT devices demands an innovative approach that considers the balance between resource constraints and DOA accuracy, without compromising battery life or real-time system performance.

2. Literature Review

Typically, research about DOA does not take into account the implementation point of view sufficiently. In many cases, research is carried out using multi-paradigm programming languages such as MATLAB, which already have a range of pre-made general-purpose numerical functions, resulting in little motivation to develop tailor-made numerical algorithms for DOA methods. However, when implementing these methods in constrained embedded systems, numerical algorithms must be developed from scratch, as C programming language libraries offer very limited support. Additionally, widely-used linear algebra libraries such as LAPACK [9] and Armadillo [10] are not suitable for use in constrained embedded systems. Although the Common Microcontroller Software Interface Standard (CMSIS) DSP Software Library is designed for these systems, it lacks some numerical algorithms used in DOA methods. Furthermore, DOA estimations are typically used in radars or large antenna arrays that employ much more powerful processors than those in constrained embedded systems. While accuracy is usually the primary performance criterion of interest, energy consumption, and memory usage are also crucial considerations for battery-powered IoT devices. Therefore, DOA methods should not occupy significant amounts of memory, blocking other IoT tasks. Additionally, complex numerical algorithms require considerable computation, which affects the execution time, an essential performance criterion for real-time applications.

The array of antenna used in this research is the L-shaped array. The L-shaped array has an interesting propriety since it is composed of two orthogonal Uniform Linear Arrays (ULAs). One-dimensional (1D) DOA methods can be applied separately for two ULAs to estimate the azimuth and elevation angles. Other shapes of planar arrays, such as Uniform Rectangular Arrays (URAs) and Uniform Rectangular Frame Arrays (URFAs), rely on two-dimensional (2D) DOA methods which are more complicated than their 1D versions. Moreover, well-known fast algorithms were specifically developed for ULAs. Additionally, some exploit the Vandermonde structure in the signal model found in such an array to speed up their computations. Namely, Root Multiple Signal Classification (Root MUSIC) [11], Estimation of Signal Parameters via Rotational Invariant Techniques (ESPRIT) [12], Fourier

Domain MUSIC [13], and Rank-Reduction Method (RARE) [14]. Most recently, many modified versions of the cited methods have been devised, claiming to be a better version in a certain way. Nevertheless, ESPRIT was later designed for URAs and Uniform Circular Arrays (UCAs) [15], whereas Fourier Domain MUSIC can be applied to non-uniform linear arrays as well.

Among many DOA methods, Multiple Signal Classification (MUSIC) [16], invented in the 80s, is an important algorithm that has been extensively tested in simulation and the real world for many decades as well as comprehensively studied, culminating in some well-known modified versions. Notably, the standard MUSIC detects DOAs by searching for peaks in the spatial spectrum. It also can be extended to find azimuth and elevation angles by searching for peaks in a 2D spatial spectrum of the planar array of antennas. However, that 2D search is a prohibitively expensive operation which motivated the development of Reduced-Dimension (R-D) MUSIC [17] that exploits the structure of an L-shaped array of antennas to do that search in two 1D spatial spectra, one for each ULA. Knowing Root-MUSIC is a search-free method that exploits the Vandermonde structure of ULAs to apply a root-finding method that substantially reduces its execution time, the Reduced-Dimension (R-D) Root MUSIC came naturally as a modified version that executes Root MUSIC two times, one execution for each ULA.

Research about the BLE direction finding is still scarce as it is a recent feature released in 2017. In [18], research was conducted involving real-world experiments using that BLE feature in an indoor environment focused on array calibrations using two ULAs and one URA. It reported that mutual coupling had a minimal influence or even had no clear improvement on the accuracy of a variation of MUSIC. However, Carrier Frequency Offset (CFO) compensation substantially impacted it. In [5], four 4×4 URAs were employed in an indoor environment of 25 m $\times$ 15 m wide composed of pillars, walls, human movements, tables, chairs, devices, and lamps. In total, 8 eight distinct positions were estimated, and 32 different estimations were evaluated using BLE direction finding and the MUSIC method. The average error was 0.7 m, attaining a good result concerning the sub-1-meter accuracy purpose.

Papers about real-world implementations are less common than about in a simulation. In [19], a modified version of MUSIC was developed in a Digital Signal Processor (DSP) for underwater acoustic sources. Notably, it employed a Reduced-Order Root-MUSIC method that reduces the polynomial order of Root MUSIC to speed up computations. In [20,21], researchers conducted a small-scale experiment using a ULA with four elements and one single transmitter. The array of antennas was connected to the NI PXI platform, and an antenna transmitter was connected to another PX platform. After running a single-source MUSIC and a Total Least Squares ESPRIT in LabView NI hardware to estimate two different DOAs, they concluded both methods could be used in real-time applications. Moreover, the development of 2D MUSIC for an L-shaped antenna array to estimate the azimuth and elevation angles based on parallel computing was successfully devised for a Digital Signal Processor (DSP) [22]. More specifically, researchers parallelized the eigenvalue decomposition to construct the signal or noise subspace and the peak searching method to find DOAs by exhaustive search. Despite the parallelization, the execution time of parallel MUSIC was 190.39 ms, which can be seen as slow considering that the experiment used a DSP of 1 GHz.

Most of the research did not focus on the implementation aspect regarding the optimization of the method to be adapted to constrained embedded systems. Except for one cited paper, the rest did not evaluate other important performance criteria such as execution time, energy consumption, and memory footprint. This research takes a step further by adapting the R-D Root MUSIC for a Radio Frequency (RF) switch based on Bluetooth specification. Additionally, the adapted method applies unitary transformations to avoid complex arithmetic, and a real-valued Eigenvalue Decomposition Method (EVD) is employed to find the roots of complex-valued polynomials. The novel method also

exploits the radio communication system design of Bluetooth to speed up its execution based on BLE receivers' capability of estimating one single DOA only.

Notation: by defining the complex number $z = ce^{j\theta}$, the operators Re $(\cdot)$ and Im $(\cdot)$ return the real and imaginary part of z, respectively. The argument of a complex number arg $(\cdot)$ is an operator that returns the phase angle (θ) in the interval $[-\pi, \pi]$ while $\overline{(\cdot)}$ is the complex conjugate. The complex absolute operator is defined as $|\cdot| = \sqrt{\text{Re}(\cdot)^2 + \text{Im}(\cdot)^2}$. The operator diag $(\cdot)$ denotes a diagonal matrix formed by an input vector. The Hermitian transpose is $(\cdot)^H$ which applies the transpose and complex conjugate on a matrix. The operator $\lfloor \cdot \rfloor$ is the floor function, that is, it outputs the greatest integer less than or equal to the input which is a real number. The Hadamard product $(\circ)$ is an operator that takes two matrices, for example, **A** and **B** of the same dimension, and outputs another matrix whose elements are given by $(\mathbf{A} \circ \mathbf{B})_{ij} = (\mathbf{A}_{ij})(\mathbf{B}_{ij})$. $\mathbf{I}_n$ is an identity matrix while $\mathbf{0}_n$ is a zero column vector both of size n.

3. Mathematical Model for L-Shaped Uniform Array

Figure 3 shows the structure of the L-shaped uniform array. It is composed of two orthogonal uniform linear arrays of M antennas in the X-Y plane in which the distance between two adjacent antennas is Δ. All antennas are assumed to be identical, isotropic, and omnidirectional. Suppose there are d $(d < M)$ independent far-field narrowband stationary signals, $s_i(t)$ such that $i = 1, \ldots, d$, incident on the array plane at 2D angle $(\theta_1, \phi_1), (\theta_2, \phi_2), \ldots, (\theta_d, \phi_d)$ in which θ_i is the azimuth and ϕ_i is the elevation angle. Let us also assume the signals propagate in an AWGN channel with linear and isotropic transmission medium. DOA methods compute the broadside angle, which is the signal direction measured relative to the line perpendicular to the array. Since that array is composed of two ULAs, there are two broadside angles, α_i, and β_i, as shown in Figure 3, which correspond to the x-axis and y-axis ULAs, respectively. From geometric properties, the relation between α_i and β_i with azimuth and elevation are shown in Equations (1) [23],

$$\begin{aligned} \cos\alpha_i &= \cos\theta_i \sin\phi_i, \\ \cos\beta_i &= \sin\theta_i \sin\phi_i. \end{aligned} \tag{1}$$

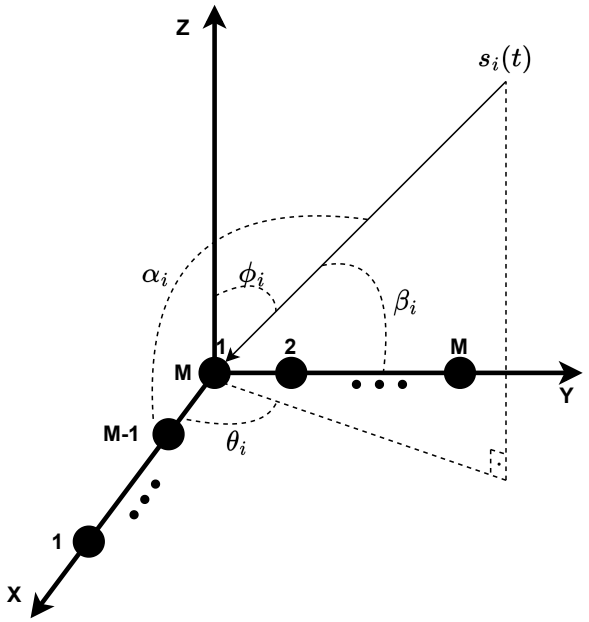

Figure 3. Depiction of L-shaped array with its antennas (black dots), angles, and the signal direction.

Assume that the L-shaped array is not subject to nonidealities such as mutual coupling and cross-polarization effect. Additionally, consider that all antennas are identical and have

omnidirectional gain functions, i.e., $g(\theta, \phi) = 1$. The model of the signal samples received by x-axis and y-axis arrays at a timestamp t can be expressed as Equations (2) and (3) [24],

$$\mathbf{x}(t) = \mathbf{A}_x \mathbf{s}(t) + \mathbf{n}_x(t), \tag{2}$$

$$\mathbf{y}(t) = \mathbf{A}_y \mathbf{s}(t) + \mathbf{n}_y(t), \tag{3}$$

where $\mathbf{x}(t) = \begin{bmatrix} x_1(t) & x_2(t) & \dots & x_M(t) \end{bmatrix}^T$ is the array observation at timestamp t of the x-axis ULA which is a vector of the signal samples for each individual 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 y-axis ULA. Moreover, $\mathbf{s}(t) \in \mathbb{C}^{d \times 1}$ is a vector of signals of d sources, $\mathbf{n}_x(t), \mathbf{n}_y(t) \in \mathbb{C}^{M \times 1}$ are the additive white Gaussian noise (AWGN) and $\mathbf{A}_x, \mathbf{A}_y \in \mathbb{C}^{M \times d}$ are the ideal steering matrices of the x-axis array and y-axis array, respectively, as shown in Equations (4) and (5),

$$\mathbf{A}_x = \begin{bmatrix} \mathbf{a}(\alpha_1) & \mathbf{a}(\alpha_2) & \dots & \mathbf{a}(\alpha_d) \end{bmatrix} \tag{4}$$

$$\mathbf{A}_y = \begin{bmatrix} \mathbf{a}(\beta_1) & \mathbf{a}(\beta_2) & \dots & \mathbf{a}(\beta_d) \end{bmatrix} \tag{5}$$

where the ideal array responses are defined in Equations (6) and (7),

$$\mathbf{a}(\alpha_i)^T = \begin{bmatrix} 1 & e^{j\mu_{\alpha_i}} & e^{j2\mu_{\alpha_i}} & \dots & e^{j(M-1)\mu_{\alpha_i}} \end{bmatrix}, \tag{6}$$

$$\mathbf{a}(\beta_i)^T = \begin{bmatrix} 1 & e^{j\mu_{\beta_i}} & e^{j2\mu_{\beta_i}} & \dots & e^{j(M-1)\mu_{\beta_i}} \end{bmatrix}, \tag{7}$$

in which $\mu_{\alpha_i} = -\frac{2\pi f_c}{c}\Delta \cos\theta_i \sin\phi_i = -\frac{2\pi f_c}{c}\Delta \cos\alpha_i$ and $\mu_{\beta_i} = -\frac{2\pi f_c}{c}\Delta \sin\theta_i \sin\phi_i = -\frac{2\pi f_c}{c}\Delta \cos\beta_i$, and c is the speed of light, f_c is the carrier frequency

Moreover, in our analyses, it would be useful in some cases to consider the model signal samples for the whole L-shaped array instead of two separate ULAs. Let us define $\mathbf{h}(t) = \begin{bmatrix} h_1(t) & h_2(t) & \dots & h_{2M-1}(t) \end{bmatrix}^T$ as the array observation of the L-shaped array at timestamp t which is a vector of the signal samples for each individual antenna in the L-shaped array, such that $h_i(t)$ corresponds a single signal sample received from the antenna i at timestamp t. The signal samples of an L-shaped array are composed of $h_i(t) = x_i(t)$ such that $i = 1, \dots, M-1$, the common antenna $h_M(t) = x_M(t) = y_1(t)$ for both ULAs in addition to $h_{M+j-1}(t) = y_j(t)$ such that $j = 2, \dots, M$. The model of the signal samples received by the L-shaped array at a timestamp t can be expressed as Equation (8),

$$\mathbf{h}(t) = \mathbf{A}_h \mathbf{s}(t) + \mathbf{n}_h(t), \tag{8}$$

where $\mathbf{n}_h(t) \in \mathbb{C}^{(2M-1)\times 1}$ is the additive white Gaussian noise (AWGN) and $\mathbf{A}_h \in \mathbb{C}^{(2M-1)\times d}$ is the ideal steering matrix of the L-shaped array, respectively, shown in Equation (9),

$$\mathbf{A}_h = \begin{bmatrix} \mathbf{a}(\alpha_1, \beta_1) & \mathbf{a}(\alpha_2, \beta_2) & \dots & \mathbf{a}(\alpha_d, \beta_d) \end{bmatrix}, \tag{9}$$

if each element of the ideal L-shaped array response, $\mathbf{a}(\alpha, \beta) \in \mathbb{C}^{(2M-1)\times 1}$, is represented in Equation (10),

$$a_{k_x, k_y} = e^{j(k_x - 1)\mu_\alpha} e^{j(k_y - 1)\mu_\beta}, \tag{10}$$

where $1 \leq k_x \leq M$ and $1 \leq k_y \leq M$, hence the L-shaped array response can be expressed in a compact form in accordance with Equation (11),

$$\mathbf{a}(\alpha, \beta)^T = \begin{bmatrix} a_{1,1} & a_{2,1} & \dots & a_{M,1} & a_{M,2} & a_{M,3} & \dots & a_{M,M} \end{bmatrix}^T. \tag{11}$$

4. Unitary Reduced-Dimension Root MUSIC

In the standard 2D MUSIC, the array observation of both x-axis and y-axis ULAs are combined in such a way that it is possible to estimate the azimuth and elevation angle of each signal source by performing a 2D search on the spatial spectrum. In contrast, the Unitary Reduced-Dimension (R-D) Root MUSIC computes the two ULAs separately by applying a root-finding method that does not require a 2D search on the spatial spectrum. Consequently, that method is substantially faster than standard 2D MUSIC [17]. Moreover, the Unitary transformation turns centro-hermitian matrices into real-valued ones which further decreases the execution time and reduces memory consumption.

For the standard 2D MUSIC, let $\mathbf{z}(t)$ be the combination of the array observation of the x-axis and y-axis in a timestamp t as defined in Equation (12),

$$\mathbf{z}(t) = \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{y}(t) \end{bmatrix}. \tag{12}$$

After collecting N array observations, the method computes its sample covariance matrix expressed in Equation (13),

$$\mathbf{R}_{zz} \approx \hat{\mathbf{R}}_{zz} = (\frac{1}{N})\mathbf{Z}\mathbf{Z}^H, \tag{13}$$

such that $\mathbf{Z} = \begin{bmatrix} \mathbf{z}(t_1) & \mathbf{z}(t_2) & \dots & \mathbf{z}(t_N) \end{bmatrix} \in \mathbb{C}^{2M \times N}$. Under some assumptions [24], the covariance matrix, $\mathbf{R}_{zz}$, is non-singular and its eigenvectors are divided into two subspaces: a noise subspace ($\mathbf{U}_N$) and a signal subspace ($\mathbf{U}_S$). More specifically, the signal subspace ($\mathbf{U}_S$) is composed of eigenvectors corresponding to the d largest eigenvalues. While the noise subspace ($\mathbf{U}_N$) is composed of eigenvectors corresponding to the $N - d$ smallest eigenvalues. Ideally, the $N - d$ smallest eigenvalues are zeros; however, due to AGWN, they are nonzeros. Moreover, the noise subspace is orthogonal to the combined steering vector as indicated by Equation (14),

$$\mathbf{U}_N^H \begin{bmatrix} \mathbf{a}(\alpha_i) \\ \mathbf{a}(\beta_i) \end{bmatrix} = \mathbf{0}, \; i = 1, \dots, d. \tag{14}$$

From Equation (14), the spatial spectrum of 2D MUSIC method can be denoted in Function (15),

$$P_{2D-MUSIC}(\alpha, \beta) = \frac{1}{\begin{bmatrix} \mathbf{a}(\alpha) \\ \mathbf{a}(\beta) \end{bmatrix}^H \mathbf{U}_N \mathbf{U}_N^H \begin{bmatrix} \mathbf{a}(\alpha) \\ \mathbf{a}(\beta) \end{bmatrix}} \tag{15}$$

and the angles (α_i, β_i), $i = 1, \dots, d$ can be found by performing a 2D search of the d peaks on Equation (15). As previously mentioned, since the 2D search peak finding is prohibitively time consuming, the Unitary R-D Root MUSIC overcomes this problem by performing a root-finding method instead. Before describing the Unitary R-D Root MUSIC, we have to define three matrices that are used in the unitary transformation to convert centro-hermitian matrices into real-valued ones. Let $\mathbf{\Pi}_p \in \mathbb{C}^{p \times p}$ be any anti-diagonal identity matrix in keeping with Definition (16),

$$\mathbf{\Pi}_p \triangleq \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}, \tag{16}$$

and $\mathbf{Q}_n \in \mathbb{C}^{n\times n}$ be an unitary transform matrix expressed in Definitions (17) and (18),

$$\mathbf{Q}_{2n} \triangleq \frac{1}{\sqrt{2}} = \begin{bmatrix} \mathbf{I}_n & j\mathbf{I}_n \\ \mathbf{\Pi}_n & -j\mathbf{\Pi}_n \end{bmatrix} \quad \text{or} \tag{17}$$

$$\mathbf{Q}_{2n+1} \triangleq \frac{1}{\sqrt{2}} = \begin{bmatrix} \mathbf{I}_n & 0 & j\mathbf{I}_n \\ \mathbf{0}_n^T & \sqrt{2} & \mathbf{0}_n^T \\ \mathbf{\Pi}_n & 0 & -j\mathbf{\Pi}_n \end{bmatrix}, \tag{18}$$

depending if its size is even or odd. The algorithm is outlined below.

1. Collect N array observations for timestamp $t_1, t_2, \ldots, t_N$. Afterward, we can estimate the covariance matrix of the x-axis and y-axis separately in line with Equations (19) and (20),

$$\mathbf{R}_{xx} \approx \hat{\mathbf{R}}_{xx} = (\frac{1}{N})\mathbf{X}\mathbf{X}^H, \tag{19}$$

$$\mathbf{R}_{yy} \approx \hat{\mathbf{R}}_{yy} = (\frac{1}{N})\mathbf{Y}\mathbf{Y}^H, \tag{20}$$

   where $\mathbf{X} = \begin{bmatrix} \mathbf{x}(t_1) & \ldots & \mathbf{x}(t_N) \end{bmatrix} \in \mathbb{C}^{M\times N}$ and $\mathbf{Y} = \begin{bmatrix} \mathbf{y}(t_1) & \ldots & \mathbf{y}(t_N) \end{bmatrix} \in \mathbb{C}^{M\times N}$ are the N array observations of x-axis and y-axis respectfully.
2. Apply forward-backward averaging on the covariance matrix, that is, $\hat{\mathbf{R}}_{xx}^{fb} = \hat{\mathbf{R}}_{xx} + \mathbf{\Pi}_M\overline{\hat{\mathbf{R}}_{xx}}\mathbf{\Pi}_M$ to deal with coherent signals due to the multipath reflections [25]. Since $\hat{\mathbf{R}}_{xx}^{fb}$ is a centro-hermitian matrix, we apply the unitary transformation to convert that complex-valued matrix into a real-valued one, that is, $\hat{\mathbf{C}}_x = \{\mathbf{Q}_M^H\ \hat{\mathbf{R}}_{xx}^{fb}\ \mathbf{Q}_M\} = \mathrm{Re}\{\mathbf{Q}_M^H\ \hat{\mathbf{R}}_{xx}\ \mathbf{Q}_M\} \in \mathbb{R}^{M\times M}$. The same goes for the covariance matrix of the y-axis, $\hat{\mathbf{C}}_y = \mathrm{Re}\{\mathbf{Q}_M^H\ \hat{\mathbf{R}}_{yy}\ \mathbf{Q}_M\} \in \mathbb{R}^{M\times M}$.
3. Apply the real-valued EVD in $\mathbf{C}_x$ and $\mathbf{C}_y$ to construct the noise subspace $\mathbf{U}_{N,x}$ and $\mathbf{U}_{N,y}$ of the x-axis and y-axis, respectively. It is a time-consuming operation that is optimized in this paper.
4. As previously mentioned, the Unitary R-D Root MUSIC estimates the DOAs based on the roots of two polynomials, so it avoids the exhaustive search of standard MUSIC. Defining $z = e^{-j\frac{2\pi}{\lambda}\Delta\cos\alpha}$ for the x-axis, the array response in Equation (6) can be redefined as a function of z, as shown in Equation (21),

$$\mathbf{a}(\alpha)^T = \mathbf{a}(z)^T = \begin{bmatrix} 1 & z & z^2 & \ldots & z^{M-1} \end{bmatrix}, \tag{21}$$

   likewise for the y-axis if we define $z = e^{-j\frac{2\pi}{\lambda}\Delta\cos\beta}$. In addition, since $a(z)^H = a^T(\frac{1}{z})$, the MUSIC spectrum can be viewed as a polynomial function whose DOA information is contained in some of its roots. The polynomial of the Unitary R-D Root MUSIC for the x-axis and y-axis are defined in Equations (22) and (23) [11],

$$p_x(z) = z^{M-1}\mathbf{a}^T(\frac{1}{z})\mathbf{Q}_M\mathbf{U}_{N,x}(\mathbf{U}_{N,x})^H\mathbf{Q}_M^H\mathbf{a}(z), \tag{22}$$

$$p_y(z) = z^{M-1}\mathbf{a}^T(\frac{1}{z})\mathbf{Q}_M\mathbf{U}_{N,y}^y(\mathbf{U}_{N,y})^H\mathbf{Q}_M^H\mathbf{a}(z), \tag{23}$$

whose degree is $2M-2$. By defining $\mathbf{G} = \mathbf{Q}_M \mathbf{U}_{N,x} (\mathbf{U}_{N,x})^H \mathbf{Q}_M^H$, we can derive Equations (24),

$$\begin{aligned} p_x(z) &= z^{M-1} \begin{bmatrix} 1 & z^{-1} & z^{-2} & \dots & z^{-(M-1)} \end{bmatrix} \mathbf{G} \begin{bmatrix} 1 & z & z^2 & \dots & z^{M-1} \end{bmatrix}^T \\ &= \begin{bmatrix} z^{M-1} & z^{M-2} & z^{M-3} & \dots & 1 \end{bmatrix} \begin{bmatrix} g_{1,1} & \cdots & g_{1,M} \\ \vdots & \dots & \vdots \\ g_{M,1} & \cdots & g_{M,M} \end{bmatrix} \begin{bmatrix} 1 \\ z \\ z^2 \\ \vdots \\ z^{M-1} \end{bmatrix} \\ &= a_0 + a_1 z + \cdots + a_{2M-2} z^{2M-2}, \end{aligned} \tag{24}$$

whose coefficients are defined in the relation (25),

$$a_{k-1} = \begin{cases} \sum_{i=1}^{k} g_{i,M-k+i} & \text{if} \quad k = 1, 2, \dots, M \\ \sum_{i=1}^{(2M-1)-k+1} g_{i,M-k+i} & \text{if} \quad k = M+1, M+2, \dots, 2M-1. \end{cases} \tag{25}$$

The same procedure goes to $p_y(z)$.

5. The d complex roots of $p_x(z)$ and $p_y(z)$ that are inside of a unit circle and closest to it, namely, $\hat{z}_{x,1}, \hat{z}_{x,2}, \dots, \hat{z}_{x,d}$ and $\hat{z}_{y,1}, \hat{z}_{y,2}, \dots, \hat{z}_{y,d}$, respectively, contain information about the d DOAs. The azimuth and elevation angles can be found as indicated by the set of equation in (26),

$$\begin{aligned} &\hat{\theta}_i = \arctan\left(\frac{\arg(\hat{z}_{y,i})}{\arg(\hat{z}_{x,i})}\right), \\ &\hat{\phi}_i = \arcsin\left(\frac{\lambda}{2\pi\Delta}\sqrt{(\arg(\hat{z}_{x,i})^2 + \arg(\hat{z}_{y,i})^2}\right), \\ &\forall i = 1, 2, \cdots, d. \end{aligned} \tag{26}$$

Note that finding all roots of a complex-valued polynomial is a difficult task and time-consuming, thus it needs an efficient and accurate root-finding method. The implemented solution circumvents complex arithmetic and finds them by a real-valued EVD, which is described in Section 6.

5. RF Switch Model

Theoretically, all antennas within an array should sample the signal at the same time at each antenna port. However, this would require each antenna to have its own RF front-end, which includes components such as analog-to-digital converters, filters, mixers, and low-noise amplifiers. Unfortunately, incorporating such analog components for each antenna would lead to increased power consumption, physical size, and higher costs for constrained embedded IoT devices. To address these challenges, it is more appropriate for the array to have a single RF front-end and an RF switch, enabling each antenna to utilize the RF front-end at different times. The Bluetooth protocol considers this radio architecture for its direction-finding capability, and in this research, we opted to utilize the switching protocol outlined in the Bluetooth 5.1 specification [4] with an L-shaped array.

Bluetooth utilizes Gaussian Frequency-Shift Keying (GFSK) where 0s and 1s are modulated into different frequency shifts [26]. The two frequencies are equal to the central frequency (f_c) in addition to a frequency deviation ($\pm f_\Delta$). To comply with the theoretical assumption, the signal should be stationary, that is, a signal with a constant time-frequency is desirable. Thus, the transmitter (signal source) sends a Constant Tone Extension (CTE), composed of a continuous series of digital ones, so the frequency remains the same during

the IQ sampling. Notably, if the signal were non-stationary, DOA methods would be more complex [27].

During the CTE, the first 4 µs is the guard period. The reference period, which takes 8 µs, is the beginning of the IQ sampling operation but only one antenna of the receiver (anchor node) carries out the sampling operation. Only one single IQ sample is performed per 1 µs, totaling eight IQ samples. Afterward, a series of sampling and switching operations begin, which is referred to switch-sample period. The switch and sample slot last 1 µs or 2 µs; in this research we consider 1 µs time slot. For every switch slot, the RF switch device switches from one antenna to another, so that another antenna can acquire a single IQ sample during the next sample slot. Since this research considers a 1 µs time slot, there are 74 sample slots in the switch-sample period. Figure 4 depicts all operations during the CTE.

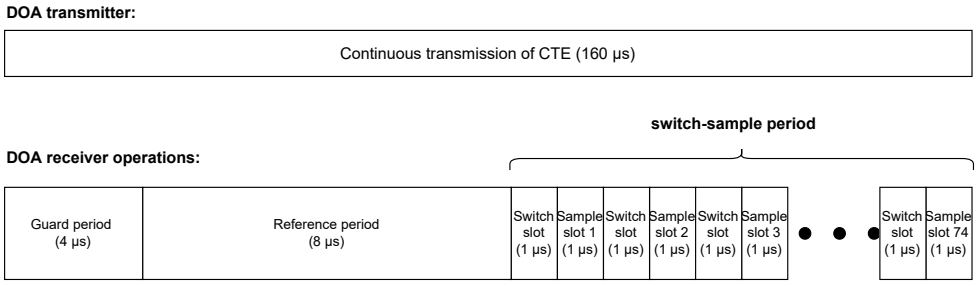

Figure 4. Depiction of the transmitter and receiver operations.

Bluetooth low energy signals can be considered narrowband when using 1 MHz bandwidth in indoor scenarios where the typical delay spread is between 20 ns and 60 ns [28]. Therefore, BLE satisfies the narrowband premise in Section 3, if we take into account all the cited assumptions as well, the mathematical model is the same as Equations (2) and (3) in addition to the phase shift due to the RF switch. Without AWGN, the phase shift between two consecutive samples in the reference period was reported to be about 80°–100° [29]. These numbers double for two consecutive samples in the switch-sample period. As a result, it is imperative to develop a phase compensation to make the DOA method work properly. As previously mentioned, the transmitter sends a continuous signal representing digital ones which is the CTE where its carrier frequency f_c is between 2402 MHz and 2480 MHz depending on the used channel. The narrowband incoming signal can be expressed in a complex format in Equation (27) [18],

$$u(t) = \mathrm{Re}(s(t)e^{j2\pi f_c t}) = I(t)\cos 2\pi f_c t - Q(t)\sin 2\pi f_c t \tag{27}$$

where t is the time, and $s(t) = I(t) + jQ(t)$ is called the complex envelope. However, that frequency in order of gigahertz is too high for the ADC, so the receiver RF front-end performs complex downconversion also known as quadrature demodulation. That operation outputs the in-phase and quadrature (IQ) components of $u(t)$ in the baseband in such a way that the central frequency (f_c) corresponds to a DC [29]. As a result, the IQ components can be expressed in Equation (28),

$$\begin{aligned} s(t) &= Ae^{j(2\pi(f_\Delta + f_o)t + \psi)} \\ &= \underbrace{A\cos(2\pi(f_\Delta + f_o)t + \psi)}_{I(t)} + j\underbrace{A\sin(2\pi(f_\Delta + f_o)t + \psi)}_{Q(t)}, \end{aligned} \tag{28}$$

where A is the amplitude, ψ is the initial phase, f_Δ = 250 kHz is the frequency deviation considering LE 1M physical layer [4], and f_o is the carrier frequency offset (CFO) which is in order of 10 kHz [29]. Without loss of generality, let us consider that IQ samples from the reference period correspond to the first antenna of the L-shaped array. From Equation (28),

the phase shift between two consecutive samples of the reference period without considering AWGN is evidenced by Equation (29),

$$h_1(t + \Delta t) = e^{j2\pi\Delta f \Delta t} h_1(t), \tag{29}$$

where Δ t = 1 µs and $\Delta f = f_\Delta + f_o$. In other words, it is possible to estimate the phase shift over a 1 µs time period by using the samples of the reference period. However, we are interested in Δf since we can use it to estimate the phase shift over other time periods. Assuming the carrier frequency offset is constant during the CTE, we can estimate Δf by calculating the average of the phase difference between two consecutive IQ samples of the reference period, as shown in Equation (30),

$$\widehat{\Delta f} = \frac{1}{7(2\pi\Delta t)} \sum_{i=1}^{7} \arg\Big(\frac{h_1(t + i\Delta t)}{h_1(t + (i-1)\Delta t)}\Big) \tag{30}$$

By adopting the Round Robin switch pattern, each antenna from the L-shaped array carries out IQ sampling sequentially, as shown in Figure 5. Note that the switch pattern begins in the last reference period slot. As a result, the L-shaped array samples 75 IQ samples in total, 74 samples from the switch sample period, and 1 sample from the last reference period slot. Moreover, the array observation, in this case, is defined as one single sequence of the Round Robin pattern, that is, when all antennas in the array complete the IQ sampling. Observe that antenna k performs an IQ sampling $2(k-1)$µs after the array observation starts. It means that the phase shift is $e^{j2\pi\Delta f \Delta t_k}$ without considering AGWN as indicated by Equation (31),

$$h_k(t + \Delta T_k) = e^{j2\pi\Delta f \Delta t_k} h_k(t), \tag{31}$$

where $\Delta T_k = 2(k-1)\mu s$. We can generalize the observation of Equation (31). As a result, let $\mathbf{h}_{ss}(t)$ be an array observation of a Round Robin sequence that begins at timestamp t, the array $\mathbf{h}_{ss}(t)$ relates to $\mathbf{h}(t)$ by the phase shift matrix due to the RF switch labeled as $\mathbf{O}$ in accordance with Equation (32),

$$\mathbf{h}_{ss}(t) = \begin{bmatrix} h_1(t) \\ h_2(t + \Delta T_2) \\ \vdots \\ h_{2M-1}(t + \Delta T_{2M-1}) \end{bmatrix} \begin{bmatrix} h_1(t) \\ h_2(t) e^{j2\pi\Delta f_c \Delta T_2} \\ \vdots \\ h_{2M-1}(t) e^{j2\pi\Delta f_c \Delta T_{2M-1}} \end{bmatrix} = \mathbf{O} \begin{bmatrix} h_1(t) \\ h_2(t) \\ \vdots \\ h_{2M-1}(t) \end{bmatrix} = \mathbf{O}\mathbf{h}(t), \tag{32}$$

such that,

$$\Delta T_k = 2(k-1)T_{slot},\ 1 \leq k \leq 2M-1,$$

where $T_{slot} = 1\mu s$ and the RF switch phase shift matrix is a diagonal matrix defined in Equation (33),

$$\mathbf{O} \triangleq \text{diag}(1, e^{j2\pi\Delta f_c \Delta T_2}, \cdots, e^{j2\pi\Delta f_c \Delta T_{2M-1}}) \in \mathbb{C}^{(2M-1)\times(2M-1)}. \tag{33}$$

DOA methods such as MUSIC can estimate multiple DOAs during their execution, so radar applications sending sounding signals and measuring when their own signal is received from different reflections can take full advantage of that capability by identifying multiple copies of their own reflected signal. However, in IoT radio communication systems where anchors are employed to locate multiple tags, this is not possible in practice with low-cost single receiver anchor nodes operating at a single RF channel at a given time such as in Bluetooth receivers [30]. That is, if more than one BLE tag sends a signal to an anchor node at the same time and frequency resources, the signal to interference and noise ratio would be too low for that radio receiver to detect transmission reliably.

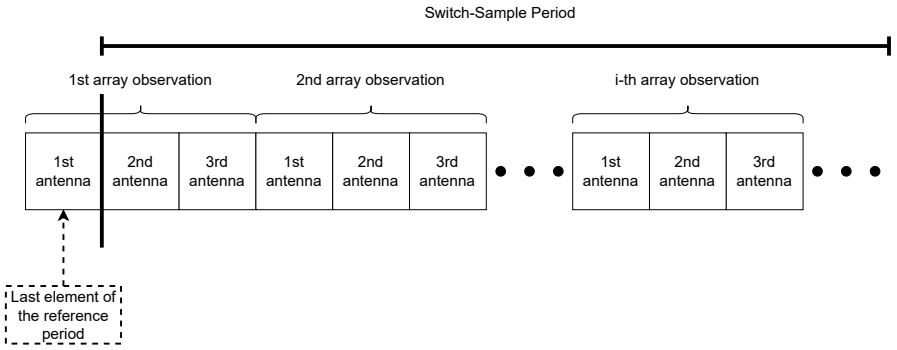

Figure 5. Example of the Round Robin switch pattern of a L-shaped array with three antennas. Only the sample slots are shown.

For example, a receiver could not be able to decode both transmitter IDs of the transmitters reliably as each transmission would interfere with the other. Therefore, in this scenario, DOA methods can only estimate a single DOA only per execution. As result, the L-shaped array observation model of one sequence of the Round Robin pattern is defined in Equation (34),

$$\mathbf{h}_{ss}(t) = \mathbf{O}\mathbf{h}(t) = \mathbf{O}\mathbf{a}(\alpha, \beta)s(t) + \mathbf{n}_h(t). \tag{34}$$

Note, $s(t)$ is a scalar that represents one single signal in opposition to the vector $\mathbf{s}(t)$ found in Equations (2) and (3). Moreover, the number of array observations depends on the number of antennas, that is, $N = \left\lfloor \frac{75}{(2M-1)} \right\rfloor$. Notably, 75 is the number of IQ samples and $2M-1$ is the number of antennas in the L-shaped array, such that $2M-1 \in [3, 75]$. Observe that, if 75 is not divisible by $2M-1$, the last $75 \mod (2M-1)$ IQ samples are not used.

MUSIC was devised considering that all antennas perform IQ sampling at the same time, which is not the case for Bluetooth specification. Thus, without a phase compensation, the accuracy of Unitary R-D Root MUSIC is totally compromised, as shown experimentally in Section 7. The DOA method receives N array observations as the input shown in Equation (35),

$$\mathbf{H}_{ss} = \begin{bmatrix} \mathbf{h}_{ss}(t_1) & \mathbf{h}_{ss}(t_2) & \dots & \mathbf{h}_{ss}(t_N) \end{bmatrix}. \tag{35}$$

From Equation (32), we know that $\mathbf{O}^{-1}\mathbf{h}_{ss}(t) = \mathbf{h}(t)$. Thus, the DOA method needs to apply the RF switch compensation matrix ($\mathbf{O}^{-1}$) into the N array observation matrix ($\mathbf{H}_{ss}$) as expressed in Equation (36),

$$\mathbf{H} = \mathbf{O}^{-1}\mathbf{H}_{ss}, \tag{36}$$

where $\mathbf{H} = \begin{bmatrix} \mathbf{h}(t_1) & \dots & \mathbf{h}(t_N) \end{bmatrix}$. Note that in the real world, the equality in Equation (36) is an approximation, since the RF switch compensation matrix ($\mathbf{O}^{-1}$) takes an estimation of Δf calculated in Equation (30). Moreover, the Unitary R-D Root MUSIC needs to obtain matrices $\mathbf{X}$ and $\mathbf{Y}$, which are the N array observations from x-axis and y-axis ULA, respectively, as defined in step 1 Section 4. To do that, observe that $\mathbf{X}$ is equal to the first M rows of $\mathbf{H}$, and $\mathbf{Y}$ is equal to the last M rows of $\mathbf{H}$. As a result, step 1 needs an additional operation to obtain matrices $\mathbf{X}$ and $\mathbf{Y}$ prior to covariance calculations.

6. Modified Unitary R-D Root MUSIC

The implemented solution optimizes the Unitary R-D Root MUSIC by exploiting Bluetooth's radio communication system design where only a single BLE tag transmits a signal at a time, as discussed in Section 5. It means that the signal subspaces, $\mathbf{U}_{S,x}$ and $\mathbf{U}_{S,y}$, is a column vector. As a result, the implemented solution can void applying the time-consuming EVD and instead apply the Power Method, which is a much simpler algorithm. Experimentally, we found the Power Method converge mostly in four iterations

only in our solution. Moreover, the computation of the RF switch compensation matrix is performed in a linear time complexity instead of executing the inverse of a matrix with a cubic time complexity. The implemented solution utilizes a finding-root method based on EVD that does not require computing complex arithmetic despite the polynomial having complex coefficients and roots. However, the implemented solution employs the ideal array response. The real array response must be empirically found and plugged into the implemented solution. Refer to [18,31,32] to know how to compute the real array response.

The objective of the optimization is to reduce the memory consumption and execution time of Unitary R-D Root MUSIC to attain satisfactory portability to run in constrained embedded systems. Thus, the algorithms were implemented from scratch in the C99 programming language, except the inverse of sine, the inverse of a tangent, and squared root, which are functions from *math.h*. The tailor-made numerical methods include the Power Method, an EVD, which is an adaptation of [33] that consists of the Shifted QR Algorithm, the Balancing technique, Hessenberg decomposition, and auxiliary linear algebra algorithms such as the norm of a vector and multiplication of a matrix with a vector. Since one of the objectives of the implemented solution is to attain a minimal memory footprint as much as possible, it does not use the *complex.h* library from C programming language. Instead, it has a data structure for complex numbers with two variables representing the real and imaginary parts and functions for complex multiplication, addition, division, and conjugation. Notably, the implemented solution only employs *math.h* and *stdint.h* libraries, reassuring its minimal computational resources consumption goal and portability.

Due to the switching protocol of Bluetooth 5.1, more operations are required in step 1 of Section 4. That is, the method collects samples from the reference period and N array observations ($\mathbf{H}_{ss}$) by performing the Round Robin switch pattern. Subsequently, the implemented solution calculates $\widehat{\Delta f}$ from Equation (30) using the samples from the reference period. Afterward, it applies the RF switch phase compensation (Equation (36)) to estimate the matrix $\mathbf{H}$. Then, it separates the IQ samples of the x-axis ULA from the y-axis one. More specifically, $\mathbf{X}$ is composed of the first M rows of the estimated $\mathbf{H}$, while $\mathbf{Y}$ is the last M rows. Finally, it calculates the two covariance matrices, $\mathbf{R}_{xx}$ and $\mathbf{R}_{yy}$ from Equations (19) and (20).

The first and simpler optimization concerns Equation (19). As discussed in [34], the matrix $\mathbf{X}$ is big for constrained embedded devices since it contains many IQ samples that are complex numbers. In fact, if the implemented solution uses all the IQ samples, $\mathbf{X}$ will occupy 600 bytes considering the single-precision floating point. By performing the sample covariance matrix, the code may have to store a temporary matrix $\mathbf{X}^H$ as well, which would double the memory consumption. The implemented solution does not store $\mathbf{X}^H$. To analyze how it is possible, let us define $\mathbf{V}$ as a matrix that stores $\mathbf{X}^H$. Normally, the standard way to multiply two matrices, particularly $\hat{\mathbf{R}}_{xx} = \mathbf{XV}$, is evidenced by Equation (37),

$$\hat{r}_{xx}(i,j) = (\frac{1}{N}) \sum_{k=1}^{N} x(i,k)v(k,j), \forall i,j = 1,\ldots,M. \tag{37}$$

Since $\mathbf{V} = \mathbf{X}^H$, then $v(k,j) = \overline{x}(j,k)$; therefore, Equation (37) could be written as indicated by Equation (38),

$$\hat{r}_{xx}(i,j) = (\frac{1}{N}) \sum_{k=1}^{N} x(i,k)\overline{x}(j,k), \forall i,j = 1,\ldots,M. \tag{38}$$

Moreover, since $\hat{\mathbf{R}}_{xx}$ is Hermitian, which means $\hat{r}_{xx}(j,i) = \overline{\hat{r}_{xx}}(i,j)$, thus the solution applies matrix multiplication only on its upper triangular part; therefore, Equation (38) is remodeled as the set of relations in (39),

$$\begin{aligned} \hat{r}_{xx}(i,j) &= (\frac{1}{N}) \sum_{k=1}^{N} x(i,k) * \overline{x}(j,k), \\ \hat{r}_{xx}(j,i) &= \overline{\hat{r}_{xx}}(i,j), \\ \forall i,j &= i,\ldots,M. \end{aligned} \tag{39}$$

From Equation (39), the implemented solution estimates the covariance matrix using the same matrix $\mathbf{X}$ twice by applying an element-wise conjugate transpose operation, thus it does not need to store $\mathbf{X}^H$. Furthermore, it only computes the upper triangular part of $\hat{\mathbf{R}}_{xx}$. To sum up, that approach saves execution time by half and memory usage in the order of MN. That is an important improvement, since calculating the sample covariance matrix is the second most time-consuming operation, as shown in Section 7. The same optimization is carried out in Equation (20) for the y-axis ULA.

Another minor optimization concerns the computation of the RF switch compensation matrix, which requires the inverse matrix calculation of $\mathbf{O}$ defined in Equation (33). The Gauss–Jordan elimination is a popular algorithm to calculate the inverse of a matrix whose complex is $\mathcal{O}(n^3)$ [35]. However, since $\mathbf{O}$ is a diagonal matrix in which the generic form of its elements is known, we can avoid applying the time-consuming inverse matrix calculation. From complex arithmetic, we know that $(e^{j\theta})e^{-j\theta} = 1$, thus, the inverse of $\mathbf{O}$ is in line with Equation (40),

$$\mathbf{O}^{-1} = \text{diag}(1, e^{-j2\pi\Delta f_c \Delta T_2}, \cdots, e^{-j2\pi\Delta f_c \Delta T_{2M-1}}) \in \mathbb{C}^{(2M-1)\times(2M-1)}. \tag{40}$$

Since $e^{-j2\pi\Delta f_c \Delta T_k} = e^{-j2\pi\Delta f_c (k-1)\Delta T_2}$, then $e^{-j2\pi\Delta f_c \Delta T_k} = (e^{-j2\pi\Delta f_c \Delta T_2})^{k-1}$ for $2 \leq k \leq 2M-1$. By defining, $z = e^{-j2\pi\Delta f_c \Delta T_2}$, the inverse of $\mathbf{O}$ can be redefined in a more compact form as evidenced by Equation (41),

$$\mathbf{O}^{-1} = \text{diag}(z^0, z^1, \ldots, z^{2M-2}) \in \mathbb{C}^{(2M-1)\times(2M-1)}. \tag{41}$$

From the redefinition in Equation (41), we can derive the Relation (42),

$$\mathbf{O}^{-1}(k,k) = \mathbf{O}^{-1}(k-1,k-1)z, \quad 2 \leq k \leq 2M-1. \tag{42}$$

Thus, to calculate $\mathbf{O}^{-1}$, the implemented solution only needs to set $\mathbf{O}^{-1}(1,1) = 1$, compute $z = e^{-j2\pi\Delta f_c \Delta T_2}$, and apply Equation (42) successively for $k = 2 \ldots 2M-1$ to take advantage of the previous computation. As a result, the implemented solution does not need to compute each element of $\mathbf{O}^{-1}$ explicitly, that is $e^{-j2\pi\Delta f_c \Delta T_k}$, which may require computing the finite Maclaurin series $2M-2$ times to calculate all of $2M-2$ elements, except the first, which is 1. Notably, the finite Maclaurin series is a well-known method to evaluate complex numbers by computers, as illustrated in Equation (43),

$$e^{jx} = \underbrace{\left(1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots \frac{(-1)^n}{(2n)!}x^{2n}\right)}_{\cos(x)} + j\underbrace{\left(x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots \frac{(-1)^n}{(2n+1)!}x^{2n+1}\right)}_{\sin(x)}, \tag{43}$$

where n is the number of elements. Moreover, to reduce memory consumption, the implemented solution just needs to store the diagonal of $\mathbf{O}^{-1}$ as a column vector and compute the element-wise (Hadamard product) of that vector with $\mathbf{h}_{ss}(t)$ defined in Equation (32). More specifically, let $\mathbf{d} = \begin{bmatrix} z^0 & z^1 & \ldots & z^{2M-2} \end{bmatrix}^T$ be the cited column vec-

tor, since $\mathbf{h}_{ss}(t) \circ \mathbf{d} = \mathbf{O}^{-1}\mathbf{h}_{ss}(t)$, the implemented solution calculates $\mathbf{H}$ by applying the Hadamard product of $\mathbf{d}$ in each element of $\mathbf{H}_{ss}$ as shown in Equation (44),

$$\mathbf{h}(t_i) = \mathbf{h}_{ss}(t_i) \circ \mathbf{d} \quad i = 1, 2, \ldots, N, \tag{44}$$

replacing Equation (36). Particularly, the computation in Equation (44) is faster than (36), since the RF switch compensation is a matrix in Equation (36), whereas in Equation (44) it is a vector $\mathbf{d}$. Moreover, Algorithm 1 calculates the RF switch compensation ($\mathbf{O}^{-1}$) whose complexity is $\mathcal{O}(n)$.

Algorithm 1: computation of the RF switch compensation ($\mathbf{O}^{-1}$)

Input: the carrier frequency offset (Δf_c) and time slot (ΔT_{slot})
Output: the elements of the diagonal of $\mathbf{O}^{-1}$ stored in $\mathbf{d}$.
1 Define $\mathbf{d} \in \mathbb{C}^{2M-1}$ such that $\mathbf{d}_1 \leftarrow 1$ and calculate $z \leftarrow e^{-j2\pi\Delta f_c \Delta T_2}$.
2 **for** $k = 2, \cdots, 2M-1$ **do**
3 $\quad \mathbf{d}_k \leftarrow \mathbf{d}_{k-1} z$
4 **end**

We can compute the two covariance noise subspaces ($\mathbf{U}_{N,x}$ and $\mathbf{U}_{N,y}$) indirectly via their respective signal subspace ($\mathbf{U}_{S,x}$ and $\mathbf{U}_{S,y}$) by Equations (45) and (46) [36],

$$\mathbf{U}_{N,x}(\mathbf{U}_{N,x})^H = \mathbf{I}_n - \mathbf{U}_{S,x}(\mathbf{U}_{S,x})^H, \tag{45}$$

$$\mathbf{U}_{N,y}(\mathbf{U}_{N,y})^H = \mathbf{I}_n - \mathbf{U}_{S,y}(\mathbf{U}_{S,y})^H. \tag{46}$$

Remember that the implemented solution only estimates one single DOA, which means, $d = 1$. Thus, the signal subspace is composed of only one eigenvector, which means we can apply the Power Method [37] that only estimates one eigenvector, which is the signal subspace, as proved in the next paragraph. The noise subspace is composed of $N-1$ eigenvectors corresponding to the smallest eigenvalues. Therefore, if we calculate the covariance noise subspace directly, we would apply an EVD method that computes all eigenvectors and eigenvalues. In addition, computing all of them requires a very complicated and time-consuming algorithm in addition to more memory footprint.

Notably, the complexity of the QR Algorithm, a typical method for EVD, is $6n^3 + \mathcal{O}(n^2)$ per iteration [38], not to mention the Hessenberg decomposition that could be performed before the QR Algorithm, and an algorithm to create the noise subspace from the probable unsorted pairs of eigenvalue-eigenvector. However, since they could be unsorted, to construct a noise subspace a reasonable approach seems to apply a sorting algorithm that could have an average complexity between $\mathcal{O}(n \log(n))$ to $\mathcal{O}(n^2)$ [39]. While the Power Method has a complexity of $\mathcal{O}(n^2)$ per iteration, it calculates the signal subspace, requires very simple computations, and experimentally we found it converges mostly in four iterations only in our solution. Figure 6 depicts the algorithm overview of the covariance noise subspace computation. The left one computes the covariance directly while its fastest version (the right one) calculates the covariance indirectly via signal subspace.

To guarantee the Power Method will converge, the matrix must be diagonalizable, there must exist only one eigenvalue with the greatest absolute value, and it must be a real number [40]. For example, considering $\lambda_i \in \mathbb{R}, i = 1, \ldots, M$ to be eigenvalues of a diagonalizable matrix, if $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_M|$ then the cited matrix satisfies the convergence requirements. The matrix $\mathbf{C}_x$ is a real covariance matrix, thus it is symmetric [41]; therefore, it is diagonalizable, and its all eigenvalues are real numbers [42].

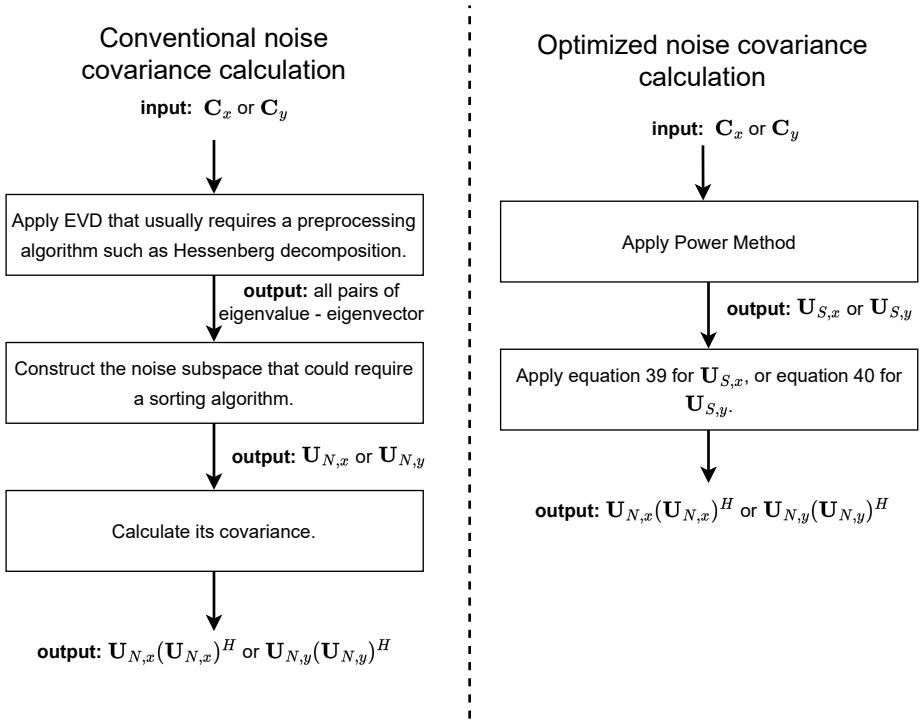

Figure 6. Algorithm overview of the two covariance noise subspace computations. The left method calculates that covariance directly, but the right one calculates indirectly via the signal subspace.

The Power Method outputs the eigenvector and its associated eigenvalue, which is the greatest in magnitude. Here, we prove that the greatest eigenvalue in magnitude is the one of the signal subspace. Note that $\mathbf{C}_x$ is a real covariance matrix, hence, it is positive semi-definite [41], which means all eigenvalues are non-negative. The line-of-sight (LOS) component of the received signal that constitutes the eigenvalue of the signal subspace is greater than the eigenvalues of the noise subspace [24], and since they are all non-negative, it is not possible to have eigenvalues of the noise subspace 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 analysis goes to $\mathbf{C}_y$.

The implemented Power Method (Algorithm 2) does not compute the eigenvalue since the solution only needs the eigenvector. We considered $K = 30$ and $tol = 10^{-6}$. We carried out thousands of experiments and we verified that in most cases the Algorithm 2 takes 4 to 5 iterations to converge, and 30 iterations are much more than enough in all experimental instances, hence for $k > 30$ we assume the algorithm fails to compute the signal subspace.

Finding all roots of a polynomial is a difficult computational task that requires time-consuming methods, and to aggravate the problem, the polynomial in question has complex coefficients with complex roots. Classical algorithms that operate directly on the polynomial function, such as the Newton–Raphson method, Secant method, and Brent's method, may be hard to work in practice. They only estimate one single root, some are guaranteed to converge if only certain conditions are satisfied and might not work on complex-valued polynomials, and the initial point must be chosen wisely since it could impact their convergence [43,44]. Although they can be extended to estimate multiple roots, they are highly sensitive to computing error since they operate directly on the polynomial function. That is the reason practical computer eigenvalue solvers, such as in MATLAB [45], hardly resemble these root-finding algorithms [46]. Instead, they apply EVD on the polynomial's companion matrix to find the roots. Since such a matrix is non-symmetric, the implemented solution estimates the roots of polynomial $p_x(z)$ (or $p_y(z)$) defined in Equation (22) (or (23)) by applying the Shifted QR Algorithm in which its implementation is an adaptation of the algorithm found in [33]. The Shifted QR Algorithm is a well-known method that performs

exceptionally well in practice and works even on non-symmetric matrices. It is an EVD method and an improved version of the standard QR Algorithm by including the shifting technique for rapid convergence. This algorithm calculates the eigenvalues of the companion matrix of $p_x(z)$ (or $p_y(z)$), which are its roots. To simplify, let us consider a polynomial of the form $p(z)$, where it could be either $p_x(z)$ or $p_y(z)$.

Algorithm 2: Power Method

Input: covariance matrix $\mathbf{C} = \mathbf{C}_x$ or $\mathbf{C} = \mathbf{C}_y$.
Output: signal subspace $\mathbf{U}_{S,x}$ (or $\mathbf{U}_{S,y}$).

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 companion matrix of the polynomial $p(z)$ is defined in Equation (47) [43],

$$\mathbf{P} \triangleq \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 & 1 \\ -c_0 & -c_1 & \ldots & -c_{2M-4} & -c_{2M-3} \end{bmatrix} \in \mathbb{C}^{(2M-2)\times(2M-2)}, \tag{47}$$

where $c_i = a_i / a_{2M-2} \forall i = 0, 1, \ldots, 2M-2$. That is, the companion matrix by definition relates to a polynomial in which its highest degree coefficient is one ($a_{2M-2} = 1$), that is the reason we have to divide all coefficients by a_{2M-2} as indicated by Equation (48),

$$\frac{p(z)}{a_{2M-2}} = \frac{a_0}{a_{2M-2}} + \frac{a_1}{a_{2M-2}} z + \cdots + \frac{a_{2M-2}}{a_{2M-2}} z^{2M-2} = c_0 + c_1 z + \cdots + z^{2M-2}. \tag{48}$$

The matrix $\mathbf{P}$ is in a complex domain, but the implemented solution executes the Shifted QR Algorithm for real-valued matrices only. To circumvent this problem, that algorithm solves the complex EVD via equivalent real formulation by converting the complex-valued companion matrix into a real-valued one. That is, the $(2M-2) \times (2M-2)$ complex eigenvalue problem in Equation (49),

$$(\mathrm{Re}(\mathbf{P}) + j\mathrm{Im}(\mathbf{P})) \cdot (\mathbf{u} + j\mathbf{v}) = \lambda(\mathbf{u} + j\mathbf{v}) \tag{49}$$

can be reformulated as $(4M-4) \times (4M-4)$ real matrix problem in accordance with Equation (50) [47],

$$\underbrace{\begin{bmatrix} \mathrm{Re}(\mathbf{P}) & -\mathrm{Im}(\mathbf{P}) \\ \mathrm{Im}(\mathbf{P}) & \mathrm{Re}(\mathbf{P}) \end{bmatrix}}_{\mathbf{P}_R} \cdot \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}. \tag{50}$$

However, the eigenvalue (λ) could still be a complex number since the matrix in problem (50) is non-symmetric ([33], pp. 486–487). That would apparently require complex arithmetic, but the implemented Shifted QR Algorithm circumvents it. Refer to [33] for more detail. The eigenvalue decomposition of $\mathbf{P}_R$ in the reformulated problem (Equation (50)) outputs two times the number of eigenvalues of $\mathbf{P}$ and the complex-valued ones come in pairs of $(\lambda, \overline{\lambda})$ since $\mathbf{P}_R$ is a real matrix [46,48]. The implemented solution can easily detect which one, λ or $\overline{\lambda}$, is the eigenvalue of $\mathbf{P}$ by verifying which is the root of the polynomial $p(z)$. Moreover, the EVD of $\mathbf{P}_R$ will increase the computations since its size is two times greater than $\mathbf{P}$, but since the number of antennas (M) is small for an ULA in IoT devices, we can afford this small overburden, especially because the elimination of complex arithmetic could partly or even totally compensate this computational increment.

However, before executing the Shifted QR Algorithm the implemented solution applies the balancing technique, and afterward, it reduces the matrix to Hessenberg form. Both algorithms are an adaptation of [33] for the implemented solution. The balancing technique is a method to reduce the rounding error sensitivity of eigenvalues during the execution of EVD. Hessenberg decomposition transforms a matrix into a simpler one (Hessenberg form) to speed up the execution time of the Shifted QR Algorithms.

Algorithm 3 outputs $4M-4$ eigenvalues in which $2M-2$ are the roots of the polynomial $p(z)$, and the other $2M-2$ are not roots but the complex conjugate of the cited eigenvalues, as explained previously. Since $d=1$, ideally the implemented solution needs to find only one single root closest to the unit circle and inside it. However, due to AWGN, that root does not need to be inside the unit circle, in mathematical terms,

$$\begin{aligned} \arg\min \quad & (|\lambda_i| - 1)^2 \\ \text{s.t.} \quad & p(\lambda_i) = 0 \\ & \lambda_i \in \{\lambda_1, \lambda_2, \ldots, \lambda_{4M-4}\}. \end{aligned} \tag{51}$$

Algorithm 3: Polynomial Finding Roots Method

Input: The polynomial $p(z)$ in which $p(z) = p_x(z)$ or $p(z) = p_y(z)$.
Output: The roots of $p(z)$ and their conjugate, $\lambda_1, \lambda_2, \ldots, \lambda_{4M-4}$.

1 Define the companion matrix $\mathbf{P} \in \mathbb{C}^{(2M-2)\times(2M-2)}$ of $p(z)$ as described in Equation (47).
2 Solve the complex EVD via equivalent real formulation as explained previously. Therefore, let's define a real matrix from $\mathbf{P}$, that is,

$$\mathbf{P}_R \triangleq \begin{bmatrix} \mathrm{Re}(\mathbf{P}) & -\mathrm{Im}(\mathbf{P}) \\ \mathrm{Im}(\mathbf{P}) & \mathrm{Re}(\mathbf{P}) \end{bmatrix} \in \mathbb{R}^{(4M-4)\times(4M-4)}.$$

3 Apply the Balancing technique in $\mathbf{P}_R$ to reduce the rounding errors sensitivity of eigenvalues.
4 Reduce the balanced $\mathbf{P}_R$ to Hessenberg form to speed up the execution of Shifted QR Algorithm.
5 Apply the Shifted QR Algorithm to the Hessenberg form of the balanced $\mathbf{P}_R$ to get its eigenvalues, $\lambda_1, \lambda_2, \ldots, \lambda_{4M-4}$.

The implemented solution applies Algorithm 4 to solve the optimization problem (51). However, instead of applying the complex absolute operator ($|\cdot|$), the algorithm uses its squared value ($|\cdot|^2$), to avoid calling the square root method in every iteration of line three. The square root method usually is an iterative algorithm and it needs to converge to a point. However, modern processors have a built-in circuit for square roots. Despite that, by avoiding that operation the execution time of Algorithm 4 is slightly decreased. In lines 2–6, observe that the algorithm finds the eigenvalue ($\lambda_{solution}$), which is the closest to the unit circle. However, its conjugate also is the closest to the unit circle since $(|\overline{\lambda}_{solution}|^2 - 1)^2 = (|\lambda_{solution}|^2 - 1)^2$. Thus, in lines 7–9, the algorithm finds which one is the root of the polynomial $p(z)$. That is,

if the eigenvalue $\lambda_{solution}$ is also a root, then $|p(\lambda_{solution})|^2 \ll 1$ which means $|p(\lambda_{solution})|^2 < |p(\overline{\lambda}_{solution})|^2$, otherwise, its conjugate is the root. In that way, the algorithm does not need to check if the eigenvalue is the polynomial root in every iteration (lines 2–6) which saves execution time.

Algorithm 4: Find the eigenvalue which is the root closest to the unit circle

Input: The eigenvalues of $\mathbf{P}$, $\lambda_1, \lambda_2, \cdots, \lambda_{4M-2}$.
Output: $\lambda_{solution}$ which is the root of $p(z)$ that is closest to the unit circle.

```
1 Define λ_solution ← λ_1
2 for i = 2, ··· , 4M − 4 do
3 |   if (|λ_i|^2 − 1)^2 < (|λ_solution|^2 − 1)^2 then
4 |   |   λ_solution ← λ_i
5 |   end
6 end
7 if |p(λ̄_solution)|^2 < |p(λ_solution)|^2 then
8 |   λ_solution ← λ̄_solution
9 end
```

7. Experiments

The objective of the experiment consisted of showing the modified Unitary R-D Root MUSIC (implemented solution) works and is feasible for commercial embedded IoT devices. The experiment comprises two parts. In the first part, to check the effectiveness of the RF switch compensation, we compared the implemented solution with the cited compensation and without it in a MATLAB environment only. The second part is more complex. It is composed of a simulation in MATLAB and the real world. In summary, the baseband signals were artificially generated in MATLAB to be the input of the implemented solution developed in C99 programming language and executed in a constrained embedded IoT device. Therefore, we could measure the memory footprint, execution time, energy consumption, and accuracy. To be as accurate as possible, such a device executed the implemented solution only without any operating systems or software layer, that is, the implemented solution was bare-metal programmed. An overview of the experiment is depicted in Figure 7.

7.1. Experimental Setup

For both parts of the experiments, the artificial baseband signals were generated using the 5G Toolbox, Phased Array System Toolbox, and Communication Toolbox provided by MATLAB. The simulation parameters are shown in Table 1. The Tapped Delay Line TDL-E channel model (corresponding to Line of Sight propagation) was employed to simulate the multipath propagation phenomenon in indoor environments alongside Additive White Gaussian Noise (AWGN). The center carrier frequency and the frequency deviation correspond to the CTE, and the simulation also randomly generated the CFO between [−30 kHz,+30 kHz] using Gaussian distribution. The CFO values were chosen based on the empirical experiment in [49], which estimated that 99% of CFO values in Bluetooth were within such interval. The L-shaped array is composed of seven isotropic antennas, that is, four antennas for each ULA. This number of antennas is small, thus, it is reasonable for IoT devices. The distance between antennas is half of the Bluetooth signal wavelength.

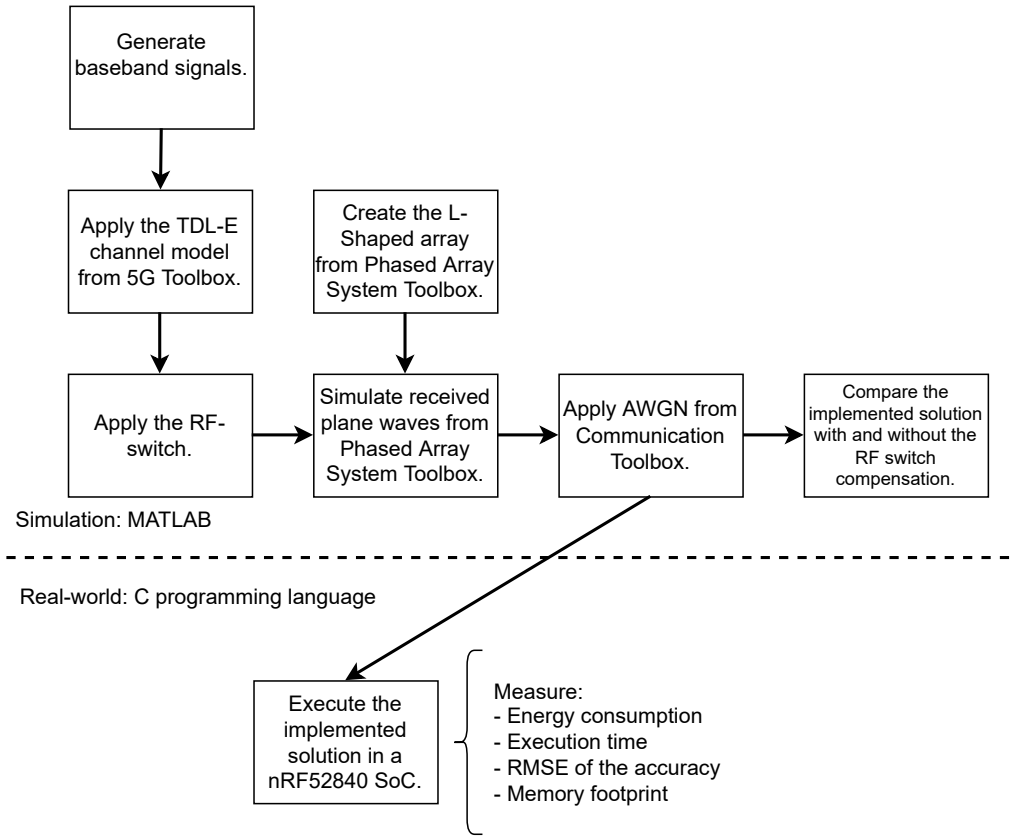

Figure 7. Overview of the experiment.

Table 1. Simulation parameters in MATLAB.

Simulation Parameters	
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 kHz, $+$30 kHz]
Center carrier frequency (f_c)	2.4 GHz
Sampling frequency	1 MHz
Channel model	TDL-E + AWGN
Number of antennas	7

To measure the memory footprint (RAM and Flash), execution time, energy consumption, and accuracy, we employed a PCA10056 development kit that comes with an nRF52840 System-on-Chip (SoC) having an Arm Cortex-M4 of 64 MHz with a Floating-Point Unit (FPU). The SoC did not use an operating system or software layers. The hardware floating-point instructions and hardware floating-point linkage (*-mfloat-abi=hard*) were activated so the processor could fully operate the FPU. All devices of the nRF52 series have support for 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 5.1 or later versions and come with an Arm Cortex-M4 or Arm Cortex-M33 processor. Another popular one

is the EFR32BG22 SoC series developed by Silicon Labs, which has the direction-finding capability, ARM Cortex-M33 with 76.8 MHz, up to 512 kB of flash, and 32 kB of RAM. Table 2 shows some SoCs with Bluetooth Direction Finding capability.

Table 2. SoCs with Bluetooth Direction Finding capability.

SoC	Processor	Flash Memory	RAM	Does It Have 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	Yes
EFR32BG24 Series	ARM Cortex M33 78 MHz	Up to 1536 KB	Up to 256 KB	Yes

The implemented solution used 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 [34], and achieve the same accuracy as the other two floating-point formats. The half-precision floating-point (FP16) is another format employed by ARM processors; however, that format is used as a storage format only for Arm Cortex-M4. When operating in FP16, the processor promotes FP16 into FP32 before and demotes it after every calculation [50]. Those operations create a small overhead that could increase the Flash consumption and the execution time. A slower execution time may translate into more energy consumption. Another supported format is double-precision floating-point (FP64), but the FPU of Arm Cortex-M4 does not have support for FP64 at all. Thus, the C compiler emulates FP64 calculations [51,52], and that emulation creates an excess of computations culminating in a substantial increment of execution time and Flash usage. In fact, the execution time of DOA methods using FP64 was shown to be about 20 times slower than ones using FP32 [34].

To measure the energy consumption, we connected the Otti Arc (power measurement tool) to the nRF52840, and to measure the execution time, we utilized the Saleae logic analyzer. However, in both measurements we needed to activate a General-Purpose Input/Output (GPIO) port, thus a GPIO was set high and low before and after the execution of the algorithm. So, we could check when the method started and finished to properly carry out the two measurements. Thus, energy usage is slightly overestimated.

Additionally, we measured the stack memory consumption and the relative execution time of the principal operations in the implemented solution. There is no dynamic memory usage. We define the relative execution time as the running time of an operation divided by the execution time of the implemented method in percentage.

Moreover, we calculated the Root Mean Squared Error (RMSE) of accuracy considering a 500 azimuth-elevation pair for each SNR. In mathematical terms, the RMSE of accuracy is defined in Equation (52),

$$RMSE = \sqrt{\left(\frac{1}{L}\right)\sum_{i=1}^{L}((\theta_i - \hat{\theta}_i)^2 + (\phi_i - \hat{\phi}_i)^2)}, \quad (52)$$

where $L = 500$ is the number of estimated azimuth-elevation pair, and θ and $\hat{\theta}$ is the actual azimuth and its estimation in degrees, respectively. Similarly for the elevation (ϕ). The SNR values were 5 dB, 10 dB, 15 dB, 20 dB, 25 dB, and 30 dB. In total, we analyzed 6000 different pairs.

7.2. Results and Discussions

Figure 8 shows the comparison between the implemented solution with RF switch compensation (a) and the one without it (b). Both were implemented in MATLAB. The implemented solution without the RF switch compensation is totally inaccurate, whereas the one with it attains much better accuracy as the SNR increases which demonstrates the effectiveness of such compensation. As previously explained, this experiment was the only one carried out totally in a simulation environment. For the next experiments, the implemented solution was run in an nRF52840 SoC.

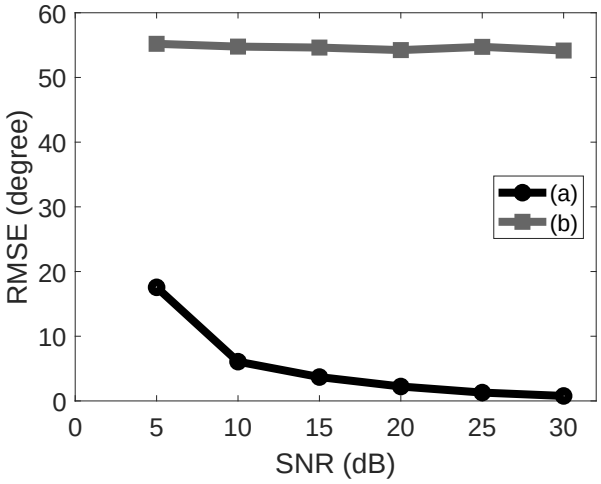

Figure 8. The implemented solution with RF switch compensation (a) and without it (b) run in a simulation (MATLAB).

Accuracy is the most important performance criterion of a DOA method. Some companies [53,54] reported an average accuracy of about 5 degrees with an average position accuracy of around 1 m. Thus, it would be desirable for the implemented solution to achieve such a value. Figure 9 shows the RMSE of the accuracy for each SNR run in an nrf52840 SoC, which have almost the same values as one in MATLAB. We clearly see that as the noise decreases the accuracy improves. However, low accuracy is observed between SNR 5–10 dB. However, it should not be a concern since the minimum SNR value for Bluetooth to operate reliably is between 10–15 dB. With less than 10 dB, it is most likely that the receiver fails to decode and the cyclic redundancy check fails as well [55]. With SNR slightly higher than 10 dB, the RMSE values attain less than 5 degrees, reaching our desired result.

Nevertheless, the experiments did not consider non-idealities of the antenna array and RF front-end; in fact, the antennas have an ideal isotropic characteristic. As mentioned previously, the implemented solution considers an ideal array response. Since those imperfections deteriorate the accuracy, one should compute a real array response and use it in the implemented solution. The real array response should attenuate the problem, but not completely solve it.

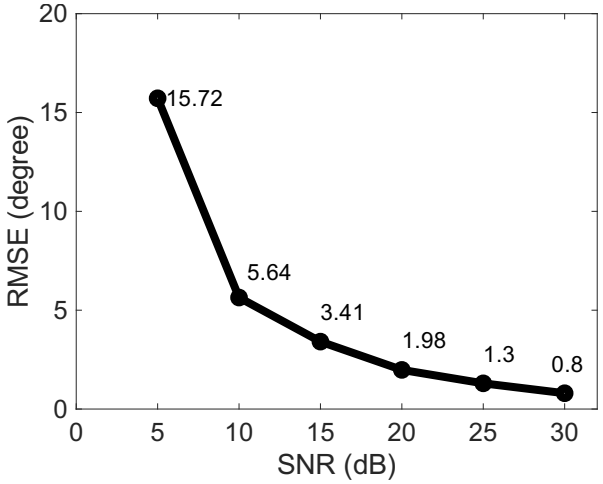

Figure 9. RMSE of the accuracy over the SNR run in a nRF52840 SoC.

Table 3 shows each main operation of the implemented solution with its respective stack memory consumption and relative execution time. All operations have a stack memory consumption of a few hundred bytes; thus, setting maximum stack memory to be 512 kB in the microcontroller is enough since none of the stack memory surpasses that value. The fifth operation concerns the finding-root method, which took up an incredible 83.17% of the method's execution time. It is not difficult to verify that no other operation demands so much computation as the fifth one, since finding polynomial roots composed of complex roots and coefficients is a difficult numerical computing task. Notably, Algorithm 3 tackles this problem by applying the Shifted QR Algorithm, a complicated method that requires two pre-processing methods to speed up its convergence and accuracy. On the other side of the spectrum, computing the companion matrix (four operations) is the least demanding task. It computes the polynomial coefficients defined in Equation (25), and afterward, it constructs the real companion matrix, which is $\mathbf{P}_R$ defined in Equation (50).

Table 3. Stack memory consumption and relative execution time of principal operations.

Operation	Total Stack Memory	Relative Execution Time
Estimate the frequency and apply the RF phase compensation	128 B	2.94%
Calculate the covariance matrices	112 B	7.67%
Compute the noise covariance matrix	440 B	5.27%
Compute the companion matrix	64 B	0.93%
Find the polynomial roots	224 B	83.17%

Moreover, the second operation is the second most time-consuming operation. It involves computing Equation (39) and the conversion of a complex-valued into a real-valued covariance as explained in step 2 Section 4. Notably, even though the implemented solution applies the optimization (Equation 39), which reduces the computation by half and stack memory by 600 B, that operation comes in second, which shows the importance of that optimization in reducing both the execution time and stack memory. Furthermore, the third operation constitutes the Power Method (Algorithm 2), and Equations (45) and (46). As explained in Section 6, the optimization avoids the execution of an EVD method, which could be as computationally demanding as the fifth operation. Finally, the first operation is composed of Algorithm 1 and Equations (44) and (30), which are computationally inexpensive.

Table 4 shows the memory footprint, execution time, and energy consumption. From these values, the implemented solution satisfies the memory requirements for an nRF52, nRF53, and EFR32BG22 SoC series, as can be verified in [56–58]. The execution time is 16.2 ms. By comparison, the standard MUSIC implemented in [59] takes about 18 ms to 133 ms with a median of roughly 31 ms to estimate a single DOA of one ULA in the same SoC (nRF52840). That means that even though the implemented solution estimates DOAs from two ULAs, it is still almost two times faster than the median of the standard MUSIC. A 2D standard MUSIC was implemented in C programming language based on parallel computing using a Digital Signal Processor of 1 GHz and an L-shaped antenna [22]; however, despite the parallelization and a much more powerful processor, it takes about 190 ms. Nonetheless, the implemented solution could be much slower than the modified 2D Unitary TLS ESPRIT in [34]. Its 1D version was developed in the same SoC, and it takes about 0.855 ms using the same precision format (single-precision floating-point). Hence, we can roughly estimate that its 2D version for L-shaped arrays could be two times that value, that is, 1.71 ms if it were developed.

Coin batteries are used for small electronic devices [60], including constrained IoT ones. We found that the capacity of such batteries ranges from 1 mAh to 2000 mAh [61] in a well-established global distributor of semiconductors and electronic components. That means, considering the implemented solution as the only source of energy consumption, the nRF52 series can run from 16,574 to more than 33 million times approximately. Therefore, the implemented solution can be used for battery-powered small embedded devices. However, it is worth mentioning the experiment did not measure the RF front-end, since we did not employ a real array of antennas. Therefore, in practice, we considered that the measurement was evaluated after IQ sampling.

Table 4. Measurement values of the implemented solution executed in nRF52840 SoC.

RAM Consumption	Flash Memory Consumption	Execution Time	Energy Consumption
4.72 KB	20.33 KB	16.2 ms	181 nWh

8. Conclusions

This paper presented a novel Unitary R-D Root MUSIC for L-shaped arrays tailor-made for constrained embedded systems using a switching protocol defined by Bluetooth, and a more insightful implementation perspective that is usually not addressed sufficiently in papers. More precisely, the implemented solution exploits the radio communication system design to speed up its execution, that is, it applies a simple Power Method instead of the time-consuming EVD. It also has a root-finding method that circumvents complex arithmetic despite being used for complex polynomials.

In theory, all antennas in the array sample the signal at each antenna port at the same time; however, Bluetooth specifies that the array has an RF switch, so each antenna samples the signal at a different time. Therefore, the theoretical model was slightly modified to consider the RF switch. We showed that without an RF switch phase compensation, the accuracy of the implemented solution was totally compromised. Therefore, we developed a method of RF switch phase compensation and conceived a linear complexity algorithm to compute the phase compensation matrix.

To prove the solution viability, we carried out experiments on energy consumption, memory footprint, accuracy, and execution time in a commercial constrained embedded IoT device (nRF52080) without operating systems and software layers. Notably, except for accuracy, other performance criterion usually are not carried out in research; however, in ours, they were too important to be neglected. With such measurements, we showed the implemented solution viability for IoT devices verified by us, its few milliseconds execution time, and its good accuracy achievement.

Author Contributions: Conceptualization, T.T. and J.M.; methodology, T.T.; software, T.T.; validation, T.T.; formal analysis, T.T. and J.M.; investigation, T.T.; resources, J.P. and V.K.; data curation, T.T.; writing—original draft preparation, T.T.; writing—review and editing, J.P., J.N., A.O., J.M. and E.S.L.; visualization, T.T.; supervision, J.N., A.O., J.P. and E.S.L.; project administration, T.T. and J.P.; funding acquisition, V.K., J.N., E.S.L. and A.O. All authors have read and agreed to the published version of the manuscript.

Funding: The authors gratefully acknowledge funding from European Union's Horizon 2020 Research and Innovation programme under the Marie Skłodowska Curie Grant Agreement No. 956090 (APROPOS, http://www.apropos-itn.eu/).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Some data and software are available upon request from (T.T.).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zoubir, A.M. Chapter 20—Applications of Array Signal Processing. In *Academic Press Library in Signal Processing. Volume 3: Array and Statistical Signal Processing*; Academic Press: Cambridge, MA, USA, 2014; pp. 859–945.
2. Bluetooth SIG. Indoor Positioning Service 1.0. 2015. Available online: https://www.bluetooth.com/specifications/specs/indoor-positioning-service-1-0/ (accessed on 27 January 2023).
3. Bluetooth SIG. Enhancing Bluetooth Location Services with Direction Finding. 2019. Available online: https://www.bluetooth.com/wp-content/uploads/2019/03/1901_Enhancing-Bluetooth-Location-Service_FINAL.pdf (accessed on 27 January 2023).
4. Bluetooth SIG. Bluetooth Core Specification v5.1. 2019. Available online: https://www.bluetooth.com/specifications/specs/core-specification-5-1/ (accessed on 27 January 2023).
5. Pau, G.; Arena, F.; Gebremariam, Y.E.; You, I. Bluetooth 5.1: An analysis of direction finding capability for high-precision location services. *Sensors* **2021**, *21*, 3589. [CrossRef] [PubMed]
6. Quuppa. Bluetooth Direction Finding: Going Beyond Beacons. 2020. Available online: https://www.quuppa.com/bluetooth-direction-finding-going-beyond-beacons/ (accessed on 27 January 2023).
7. Karlsson, P. Getting Started with Bluetooth for High Precision Indoor Positioning. Available online: https://content.u-blox.com/sites/default/files/Indoor-positioning-Getting-started-u-blox-WhitePaper.pdf (accessed on 27 January 2023).
8. Dargie, W.; Poellabauer, C. *Fundamentals of Wireless Sensor Networks: Theory and Practice*; John Wiley & Sons: Hoboken, NJ, USA, 2010.
9. Anderson, E.; Bai, Z.; Bischof, C.; Blackford, L.S.; Demmel, J.; Dongarra, J.; Du Croz, J.; Greenbaum, A.; Hammarling, S.; McKenney, A.; et al. *LAPACK Users' Guide*; SIAM: Philadelphia, PA, USA, 1999.
10. Sanderson, C.; Curtin, R. Armadillo: A template-based C++ library for linear algebra. *J. Open Source Softw.* **2016**, *1*, 26. [CrossRef]
11. Barabell, A. Improving the resolution performance of eigenstructure-based direction-finding algorithms. In Proceedings of the ICASSP'83, IEEE International Conference on Acoustics, Speech, and Signal Processing, Boston, MA, USA, 14–16 April 1983; IEEE: Piscataway, NJ, USA, 1983; Volume 8, pp. 336–339.
12. Roy, R.; Kailath, T. ESPRIT-estimation of signal parameters via rotational invariance techniques. *IEEE Trans. Acoust. Speech Signal Process.* **1989**, *37*, 984–995. [CrossRef]
13. Rubsamen, M.; Gershman, A.B. Direction-of-arrival estimation for nonuniform sensor arrays: From manifold separation to Fourier domain MUSIC methods. *IEEE Trans. Signal Process.* **2008**, *57*, 588–599. [CrossRef]
14. Pesavento, M.; Gershman, A.B.; Wong, K.M. Direction finding in partly calibrated sensor arrays composed of multiple subarrays. *IEEE Trans. Signal Process.* **2002**, *50*, 2103–2115. [CrossRef]
15. Madisetti, V.; Williams, D. *Digital Signal Processing Handbook on CD-ROM*; CRC Press: Boca Raton, FL, USA, 1999.
16. Schmidt, R. Multiple emitter location and signal parameter estimation. *IEEE Trans. Antennas Propag.* **1986**, *34*, 276–280. [CrossRef]
17. Yang, J.; He, H. A 2-D DOA Estimation Algorithm for L-Shaped Array with Improved Computational Efficiency. *Prog. Electromagn. Res. M* **2022**, *112*, 115–125. [CrossRef]
18. Yao, L. Bluetooth Direction Finding. Ph.D. Thesis, TU Delft Electrical Engineering, Delft, The Netherlands, 2018.
19. Hou, S.Y.; Chang, S.H.; Hung, H.S.; Chen, J.Y. DSP-based implementation of a real-time DOA estimator for underwater acoustic sources. *J. Mar. Sci. Technol.* **2009**, *17*, 320–325. [CrossRef]
20. Tayem, N.; Raza, S.A.; Omer, M.; El-Lakki, M.; Nayfeh, J.F. Hardware Implementation of a Proposed Qr-Tls DOA Estimation Method and Music, ESPRIT Algorithms on Ni-Pxi Platform. *Prog. Electromagn. Res. C* **2013**, *45*, 203–221. [CrossRef]
21. Tayem, N.; Omer, M.; Hussain, A.A. Hardware Implementation of MUSIC and ESPRIT on NI-PXI Platform. In Proceedings of the IEEE Military Communications Conference, Baltimore, MD, USA, 6–8 October 2014; IEEE: Baltimore, MD, USA, 2014; pp. 329–332.

22. Liu, Y.; Cui, H. Antenna Array Signal Direction of Arrival Estimation on Digital Signal Processor (DSP). *Procedia Comput. Sci.* **2015**, *55*, 782–791. [CrossRef]
23. Gao, X.; Hao, X.; Li, P.; Li, G. An improved two-dimensional direction-of-arrival estimation algorithm for L-shaped nested arrays with small sample sizes. *Sensors* **2019**, *19*, 2176. [CrossRef] [PubMed]
24. Chen, Z.; Gokeda, G.; Yu, Y. Chapter 3—Overview of Basic DOA Estimation Algorithms. In *Introduction to Direction-of-Arrival Estimation*; Artech House: Norwood, MA, USA, 2010; pp. 31–63.
25. Tuncer, E.; Friedlander, B. Chapter 4—Narrowband and Wideband DOA Estimation for Uniform and Nonuniform Linear Arrays. In *Classical and Modern Direction-of-Arrival Estimation*; Academic Press: Cambridge, MA, USA, 2009; pp. 125–158.
26. Bluetooth SIG. Bluetooth Core Specification v5.3. 2021. Available online: https://www.bluetooth.com/specifications/specs/core-specification-5-3/ (accessed on 27 January 2023).
27. Zoubir, A.M. Chapter 17—DOA Estimation of Nonstationary Signals. In *Academic Press Library in Signal Processing. Volume 3: Array and Statistical Signal Processing*; Academic Press: Cambridge, MA, USA, 2014; pp. 765–794.
28. European Telecommunications Standards Institute (ETSI). 5G; Study on Channel Model for Frequencies From 0.5 to 100 GHz (3GPP TR 38.901 Version 16.1.0 Release 16). 2020. Available online: https://www.etsi.org/deliver/etsi_tr/138900_138999/138901/16.01.00_60/tr_138901v160100p.pdf (accessed on 15 January 2023).
29. Silicon Labs. AN1297: Custom Direction-Finding Solutions Using the Silicon Labs Bluetooth Stack. 2021. Available online: https://www.silabs.com/documents/public/application-notes/an1297-custom-direction-finding-solutions-silicon-labs-bluetooth.pdf (accessed on 27 January 2023).
30. Nordic Semiconductor. nRF5340 Product Specification 1.2. 2021. Available online: https://infocenter.nordicsemi.com/pdf/nRF5340_PS_v1.2.pdf (accessed on 12 May 2022).
31. Tuncer, E.; Friedlander, B. Chapter 3—Calibration in Array Processing. In *Classical and Modern Direction-of-Arrival Estimation*; Academic Press: Cambridge, MA, USA, 2009; pp. 93–124.
32. Zoubir, A.M. Chapter 19—Array Processing in the Face of Nonidealities. In *Academic Press Library in Signal Processing. Volume 3: Array and Statistical Signal Processing*; Academic Press: Cambridge, MA, USA, 2014; pp. 819–857.
33. Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P. Chapter 11—Eigensystems. In *Numerical Recipes in C. 2*; Cambridge University: Cambridge, UK, 1992; pp. 456–493.
34. Troccoli, T.; Pirskanen, J.; Ometov, A.; Nurmi, J.; Kaseva, V. Fast Real-World Implementation of a Direction of Arrival Method for Constrained Embedded IoT Devices. In Proceedings of the 12th International Conference on the Internet of Things, Delft, The Netherlands, 8–10 November 2022; pp. 1–8.
35. Solomon, J. Chapter 3—Linear Systems and the LU Decomposition. In *Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics*; CRC Press: Boca Raton, FL, USA, 2015; pp. 56–57.
36. Van Trees, H.L. *Optimum Array Processing: Part IV of Detection, Estimation, and Modulation Theory*; John Wiley & Sons: Hoboken, NJ, USA, 2002; pp. 1158–1159.
37. Gilat, A. *Numerical Methods for Engineers and Scientists*; Wiley Global Education: Hoboken, NJ, USA, 2013.
38. Wikipedia contributors. Eigenvalue algorithm—Wikipedia, The Free Encyclopedia. 2022. Available online: https://en.wikipedia.org/wiki/Eigenvalue_algorithm (accessed on 11 June 2022).
39. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; MIT Press: Cambridge, MA, USA, 2022.
40. Ford, W. Chapter 18—The Algebraic Eigenvalue Problem. In *Numerical Linear Algebra with Applications*; Ford, W., Ed.; Academic Press: Boston, MA, USA, 2015; pp. 379–438.
41. Solomon, J. Chapter 4—Designing and Analyzing Linear Systems. In *Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics*; CRC Press: Boca Raton, FL, USA, 2015; pp. 75–76.
42. Ford, W. Chapter 19—The Symmetric Eigenvalue Problem. In *Numerical Linear Algebra with Applications*; Ford, W., Ed.; Academic Press: Boston, MA, USA, 2015; pp. 439–465.
43. Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P. Chapter 9—Root Finding and Nonlinear Sets of Equations. In *Numerical Recipes in C. 2*; Cambridge University: Cambridge, UK, 1992; pp. 347–383.
44. Solomon, J. Chapter 8—Nonlinear Systems. In *Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics*; CRC Press: Boca Raton, FL, USA, 2015; pp. 147–158.
45. MATLAB. Polynomial Roots-MATLAB Roots-MathWorks Nordic. Available online: https://se.mathworks.com/help/matlab/ref/roots.html (accessed on 27 January 2023).
46. Johnson, S.G. Eigenvalue-Polynomials. 2017. Available online: https://nbviewer.org/github/stevengj/1806/blob/fall17/lectures/Eigenvalue-Polynomials.ipynb (accessed on 15 October 2022).
47. Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*; Cambridge University Press: Cambridge, UK, 2007; pp. 590–591.
48. Day, D.; Heroux, M.A. Solving complex-valued linear systems via equivalent real formulations. *SIAM J. Sci. Comput.* **2001**, *23*, 480–498. [CrossRef]
49. Cloudt, S. Bluetooth Low Energy Direction Finding on Embedded Hardware by Mitigating Carrier Frequency Offset and Multipath Fading. Ph.D. Thesis, Master's Thesis, Eindhoven University of Technology: Eindhoven, The Netherlands, 2021.
50. Arm. *Arm Compiler Armclang Reference Guide Version 6.12*; Arm Ltd.: Cambridge, UK, 2019.
51. Yiu, J. *The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors*; Newnes: Cambridge, UK, 2013.

52. Johnson, I. 10 Useful Tips for Using the Floating Point Unit on the Cortex-M4. 2022. Available online: 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 (accessed on 12 May 2022).
53. Silicon Labs. AN1195: Antenna Array Design Guidelines for Direction Finding. 2022. Available online: https://www.silabs.com/documents/public/application-notes/an1195-antenna-array-direction-finding.pdf (accessed on 28 January 2023).
54. u-blox. Product Summary: XPLR-AOA-2. 2022. Available online: https://content.u-blox.com/sites/default/files/XPLR-AOA-2_ProductSummary_UBX-21017999.pdf (accessed on 28 January 2023).
55. Pipino, A.; Liscidini, A.; Wan, K.; Baschirotto, A. Bluetooth low energy receiver system design. In Proceedings of the 2015 IEEE International Symposium on Circuits and Systems (ISCAS), Lisbon, Portugal, 24–27 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 465–468.
56. Silicon Labs. EFR32BG22 Series 2 Bluetooth® Wireless SoC. 2021. Available online: https://www.silabs.com/wireless/bluetooth/efr32bg22-series-2-socs# (accessed on 28 January 2023).
57. Nordic Semiconductors. nRF52 Series. 2022. Available online: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fstruct_nrf52%2Fstruct%2Fnrf52.html (accessed on 12 May 2022).
58. Nordic Semiconductor. nRF5340 Product Specification. 2021. Available online: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fstruct_nrf53%2Fstruct%2Fnrf53.html (accessed on 12 May 2022).
59. Troccoli, T.; Pirskanen, J.; Ometov, A.; Nurmi, J.; Kaseva, V. Implementation of Embedded Multiple Signal Classification Algorithm for Mesh IoT Networks. In Proceedings of the 2022 International Conference on Localization and GNSS (ICL-GNSS), Tampere, Finland, 7–9 June 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–7.
60. Wikipedia contributors. Button Cell—Wikipedia, The Free Encyclopedia. 2022. Available online: https://en.wikipedia.org/w/index.php?title=Button_cell&oldid=1072413928 (accessed on 11 June 2022).
61. Mouser Electronics. Coin Cell Battery. 2022. Available online: https://www.mouser.com/c/power/batteries/coin-cell-battery (accessed on 11 June 2022).