Fine Time Measurement for the Internet of Things: A Practical Approach Using ESP32

Valentín Barral Vales, Omar Campos Fernández, Tomás Domínguez-Bolaño, Carlos J. Escudero, *Senior Member, IEEE*, and José A. García-Naya, *Senior Member, IEEE*

***Abstract*—In the world of Internet of Things (IoT), obtaining the physical location of devices has always been a task of great interest for developing increasingly complex location-based services (LBS). That is why in recent years wireless communication standards have been incorporating new additions focused on providing localization mechanisms to technologies widely used in the IoT world, such as Wi-Fi or Bluetooth. In particular, the IEEE 802.11-2016 Wi-Fi standard introduced ranging estimation between two devices through the so-called fine time measurement (FTM) protocol, defined by the IEEE 802.11mc. FTM is not yet widespread in the IoT field, but commercial modules capable of offering this functionality at a reasonable price are starting to appear. In early 2021, the most widespread system on a chip (SOC) family among IoT devices, the ESP32-XX series, added support for this Wi-Fi standard, enabling, for the first time, the use of a standard designed for location-based systems. This article analyzes the performance of this FTM implementation by carrying out and studying several measurement campaigns in different indoor and outdoor scenarios. Additionally, this work proposes an alternative real-time implementation for distance estimation inside the ESP32 using an approach based on machine learning. Such an implementation is successfully validated in a scenario totally different than those considered for the training and test sets. Finally, both the measurement sets and the developed software are available to the scientific community.**

***Index Terms*—Location management, low-cost sensors and devices, other sensors and devices, other services and applications topics.**

I. Introduction

ACCURATE location of sensors in the Internet of Things (IoT) world is important in multiple areas of interest, such as smart cities and buildings, healthcare environments, industry, or agriculture [1]. All of them demand location technologies capable of positioning the sensors to deliver more and better location-based services (LBS). Recently, the global pandemic of COVID-19 demonstrated the important need not only to know all kinds of data, such as CO_2 rates or occupancy levels but also to know the physical location of the sensors that produce them [2]. The location of sensors within the aforementioned scenarios often becomes challenging due to the following reasons [3].

1) In many occasions, global navigation satellite system (GNSSs) are not an option because the sensors are located in scenarios (inside buildings, urban canyons, etc.) where the performance of such systems is very poor or they simply do not work. And even in open areas where these systems could work, the cost of adding such capabilities to the sensors would be too high to be viable for large-scale deployments.
2) The complexity of the scenarios in which IoT sensors are placed often makes their localization very difficult. This is especially true when radio frequency (RF) technologies are employed. Phenomena such as interference or non-line of sight (NLOS) propagation yield significant errors in position estimates.
3) Typical deployments in the IoT world require the installation of many sensors, which means that the individual cost of each unit must also be very low in order not to exceed the project budget. Therefore, in certain cases, some localization technologies are excluded in advance for budget reasons, even if they perform satisfactorily.

There are, therefore, numerous technological alternatives to achieve position estimation within the IoT world, each with its own particular characteristics in terms of accuracy level, deployment and maintenance costs, reliability, or operational requirements.

Among the various localization systems available [4]–[6], those based on RF are some of the most popular. Their basic operation consists of emitting and receiving some type of waveform with certain characteristics and extracting from this transmission some physical parameters that allow for estimating the position of the emitter or receiver.

Based on these RF-based technologies, there are numerous implementations, including radio frequency identification (RFID) [7], Wi-Fi (IEEE 802.11) [8], Bluetooth/Bluetooth low energy (BLE) [9], ZigBee [10], and ultra wideband (UWB) [11].

However, when applications require very accurate and precise localization, the range of technologies that can be considered is restricted [4]. Traditional technologies exclusively

Manuscript received 29 September 2021; revised 20 December 2021 and 9 February 2022; accepted 26 February 2022. Date of publication 11 March 2022; date of current version 23 September 2022. This work was supported in part by the Galician Innovation Agency (GAIN) and the Regional Ministry of Economy, Employment, and Industry, Xunta de Galicia through ERDF under Grant COV20/00604; in part by the Xunta de Galicia under Grant ED431C 2020/15 and Grant ED431G 2019/01 to support the Centro de Investigación de Galicia "CITIC"; in part by the Agencia Estatal de Investigación of Spain under Grant RED2018-102668-T and Grant PID2019-104958RB-C42; and in part by the European Regional Development Fund (ERDF) Funds of the EU (FEDER Galicia 2014–2020 and AEI/FEDER Programs, UE). *(Corresponding author: Valentín Barral Vales.)*

The authors are with the CITIC Research Center, Department of Computer Engineering, University of A Coruña, 15071 A Coruña, Spain (e-mail: valentin.barral@udc.es; omar.campos@udc.es; tomas.bolano@udc.es; escudero@udc.es; jagarcia@udc.es).

Digital Object Identifier 10.1109/JIOT.2022.3158701

provide received signal strength (RSS) estimates, which can be heavily affected by the signal propagation conditions, and thus they may be highly variable and unpredictable, especially in indoor environments. Therefore, in recent years, newer technologies providing parameters, such as time of arrival (TOA), time difference of arrival (TDOA), angle of arrival (AOA), or angle of departure (AOD), are considered since they exhibit a much better accuracy and precision than those based on RSS. Many of the latest versions of RF-based wireless communication standards offer these new alternatives since they are evolving with the aim of providing reliable references for application in location systems. In particular, Wi-Fi (with amendments IEEE 802.11az/bd/mc) [12]–[14] and 5G [15], [16] provide TOA and TDOA, whereas Bluetooth 5.1 with direction finding [17] provides AOA and AOD.

Wi-Fi networks are one of the widely used mechanisms for communication between IoT nodes (e.g., in home automation). In previous years, the Task Group mc (TGmc) of the IEEE 802.11 Working Group (IEEE 802.11mc), incorporated several amendments into the IEEE 802.11 standards, resulting in the publication of the IEEE 801.11-2016 standard in 2016. This standard incorporates a new protocol for estimating the propagation time between devices, the so-called fine time measurement (FTM) protocol. The advantage of this standard is that it has been introduced in recent years into consumer devices and, for example, is supported by the Android operating system since version 9.0 (Android Pie with API level 28) [18]. In fact, some tests with commercial smartphones using this standard are already available in [19]–[21].

The same is not true for the implementation of this technology in devices employed in the IoT world. In this case, there are very few choices offering Wi-Fi FTM off-the-shelf. One of the main reasons is that most Wi-Fi modules on the market are not yet compatible with the 802.11-2016 standard, and only a few recent models, such as the 88W8987 family from NXP [22], offer this functionality. It is important to mention that these modules do not offer an autonomous solution since they need an external microcontroller to act as a host and communicate with them. However, there is another alternative, which are the system on a chip (SOC) devices that already incorporate a microcontroller (typically a low-power model), and a series of additional communication modules such as Wi-Fi or Bluetooth. Several chips of this type are available off-the-shelf, such as the 88MW32X from NXP [23], or the CYW43907, CYW43903, and CYW54907 from Cypress [24]. In all these models, however, the Wi-Fi module built into the SOC does not support the 802.11-2016 standard. Currently, to the best of our knowledge, the only option available on the market with FTM support is the well-known ESP32-XX family of chips from the Chinese manufacturer Espressif Systems. Specifically, its ESP32-S2 model released in late 2019 was the first to support FTM, with the release in February 2021 of its firmware version 4.3-Beta1 [25].

It is more than likely that this will lead to an explosion of LBS in the IoT world due to the high popularity and market penetration of these SOC models. It is important to know that the SOC market is projected to grow from U.S. $471 millions in 2020 to U.S. $1656 millions in 2025 with a compound annual growth rate (CAGR) of 28.6% [26].

In addition, we should keep in mind that the consumer device industry and communities are focused on the development of devices based on the ESP32-XX family for multiple reasons: low cost (around U.S. $5 each), low energy consumption, compatibility with popular development environments (Arduino, MicroPython, etc.), and the availability of high-quality documentation. Thus, a wide variety of applications have emerged in various fields (industry, home automation, monitoring, security, wearables, surveillance, location, traceability, etc.). Another example of its success is that there is a wide variety of firmware versions that allow these SOCs to be integrated in various automation projects, for example, ESPHome [27], Tasmota [28], or ESP easy [29].

This article focuses on assessing whether the existing FTM implementation in a module, such as the ESP32-S2, is mature enough in terms of accuracy and robustness to build large-scale LBS in the IoT world. For this purpose, several measurement campaigns were carried out in realistic and heterogeneous environments, including indoors and outdoors, recording the distance values estimated by the chip and comparing them with the actual values. The data set of such measurements is publicly available in [30].

During this performance study, a large difference in the distance estimation from the raw time values was observed depending on the measuring scenario: indoors or outdoors. Therefore, taking advantage of the ease of programming the ESP32, this article presents another contribution: the proposal of an alternative method to estimate the distance from the raw time values returned by the Wi-Fi module. Such a proposal is based on the use of machine learning (ML) algorithms trained with part of the acquired data during the measurements. The performance of this alternative is tested not only with offline measurements but also with a real-time implementation running on the ESP32-S2 chip. Moreover, such a real-time implementation is validated in a scenario totally different than those considered for the training and test sets. Finally, the code of such an implementation was also publicly released under an open-source license (see [31]–[34]).

The remainder of this article is organized as follows. Section II includes an analysis of the state of the art related to Wi-Fi FTM measurements. Section III describes the ESP32-S2 and the measurement scenarios. The obtained results are analyzed in Section IV, whereas Section V presents an approach based on ML to obtain distance estimates considering both round trip time (RTT) and received signal strength indicator (RSSI). A description of a new testing scenario to check the robustness of the estimation using ML is included in Section VI together with the results obtained after the experiments. Section VII details the implementation of the estimator built within the ESP32-S2, as well as an analysis of the current consumption when performing FTM operations. Finally, Section VIII is devoted to the conclusions and future work.

II. Related Work

Since the definition of the FTM protocol in the IEEE 802.11-2016, several works have focused on analyzing its performance in distance and location estimation tasks.

In [21], a study is presented on the performance of several FTM-capable Wi-Fi chips in different indoor and outdoor scenarios, with and without Line-of-Sight (LoS) propagation. The study considered 20 and 40 MHz bandwidths at the 2.4-GHz frequency band, and a bandwidth of 80 MHz at the 5-GHz frequency band. The results showed an accuracy about 1 m in open space (after correction for a constant offset), and about 5 m in indoors. This work also showed little difference between using 20 or 40 MHz bandwidths at the 2.4-GHz band. However, there was a clear improvement when using the 80-MHz bandwidth in the 5-GHz band.

Dvorecki *et al.* [35] considered ML to train an artificial neural network (ANN) capable of predicting the TOA of the first path in a Wi-Fi signal with the FTM protocol. In this case, channel impulse response (CIR) samples are employed, achieving an estimator that yielded an error below 4 m for 90% of the samples analyzed.

In [36], different neural networks (NNs) were applied to perform position estimation using RSS and FTM on different smartphones. Considering the 5-GHz band and a bandwidth of 40 MHz, measurements in an indoor environment showed that only 20% of the samples were below 4 m of error. These values increased to almost 90% of the samples below 4 m of error when an ANN-based correction factor was applied to the raw samples. A comparison between position accuracy using RSSI or FTM was performed in [19]. The measurements were carried out in the 2.4-GHz band and with a bandwidth of 20 MHz. The findings indicated that the RSSI estimates outperform those obtained from the FTM sensors, unless the sensors are calibrated individually, in which case the FTM localization is the best. The error values, in this case, yielded a mean error of 3.52 m for calibrated FTM and a mean error of 4.47 m using RSSI. Horn [20] used a smartphone to carry out Wi-Fi FTM measurements to obtain a positioning estimator. To improve its accuracy, the author integrated the measurements from two different frequency bands, 2.4 and 5 GHz, showing that there is no correlation between them. The work also included a study of the possible reasons why errors appear with this technology.

In general, all the aforementioned works showed that the Wi-Fi FTM technology can provide good results in terms of accuracy in outdoor environments without obstacles, but many problems arise in indoor environments. In these complex scenarios, the studies showed that, without a minimum calibration or optimization of each device, the results may lead to errors of tens of meters. In all the studies, Wi-Fi cards or smartphones were used as initiating devices, hence none of them addressed the problem of energy consumption or cost when describing the technology. In this article, we will analyze whether the ESP32-S2 offers results with similar accuracy to those obtained in previous works, but also take into account its characteristics as a device to be used in IoT environments.

III. Experimental Setup

A. Fine Time Measurements and the ESP32

The ESP32-S2 [37] is an evolution of the famous ESP8266, massively used in automation projects, which introduces new Bluetooth and Wi-Fi capabilities. Both ESP32-S2 and ESP8266 SOCs are part of a family of low-cost SOC microcontrollers from Espressif Systems, a Chinese company from Shanghai.

In particular, the ESP32-S2 and the ESP32-C3 series have recently added support for the IEEE 802.11-2016 standard, which introduces the FTM protocol to provide ranging estimates using Wi-Fi links. Basically, this protocol measures the RTT between two devices and, based on this time, estimates the distance between them. These devices can be of any type (access points (APs), mobile terminals, SOCs, etc.), provided that they have a Wi-Fi chipset compatible with the 802.11-2016 standard and a firmware that supports FTM. Due to its enormous market penetration, this work focuses on the ESP32-S2, although the results should be similar for the ESP32-C3.

The ESP32-S2 [37] has integrated support for the IEEE 802.11 b/g/n and a variety of peripherals. It has 43 general-purpose input/output (GPIO) pins, a 240-MHz Xtensa 32-bit LX7 single-core processor with 320 kB of static random-access memory (SRAM), and 128 kB of read-only memory (ROM). Compared to its predecessor, the ESP32-S2 SOC comes with enhanced encryption capabilities and improved radio performance. There are also versions of the SOC in the form of modules, such as the ESP32-S2-WROOM and the ESP32-S2-WROVER; and development kits such as the ESP32-S2-DevKitM-1 or the ESP32-S2-Saola-1. Using the 43 GPIO pins, the device can be configured to provide different peripheral interfaces, including four serial peripheral interfaces (SPIs), two inter-IC sound (I^2S) and interintegrated circuit (I^2C) interfaces, three universal asynchronous receiver–transmitter (UART) interfaces, 16 pulse-width modulation (PWM) channels for light-emitting diodes (LEDs), a liquid-crystal display (LCD) interface, a camera interface, 18 analog-to-digital converter (ADC) channels, two 8-bit digital-to-analog converter (DAC) channels, and 14 capacitive touch interfaces.

The most important characteristic of the ESP32-S2 for this work is that it supports the IEEE 802.11-2016 standard with the FTM protocol, which enables localization based on distance estimates between two Wi-Fi devices. However, it should be noted that only the 2.4-GHz Wi-Fi band is available, thus the bandwidth is restricted to 20 or 40 MHz, which limits the accuracy that can be achieved with the FTM protocol [18]. More specifically, according to [18], the 90th percentile of the error is expected to be 8 m for 40 MHz, 4 m for 20 MHz, and 2 m for 80 MHz.

B. Measurement Scenarios

To analyze the performance of the ESP32-S2 module using Wi-Fi RTT, two different measurement campaigns were planned. One of them was carried out in an outdoor environment, away from buildings and possible interferences. The

Fig. 1. Indoor scenario.

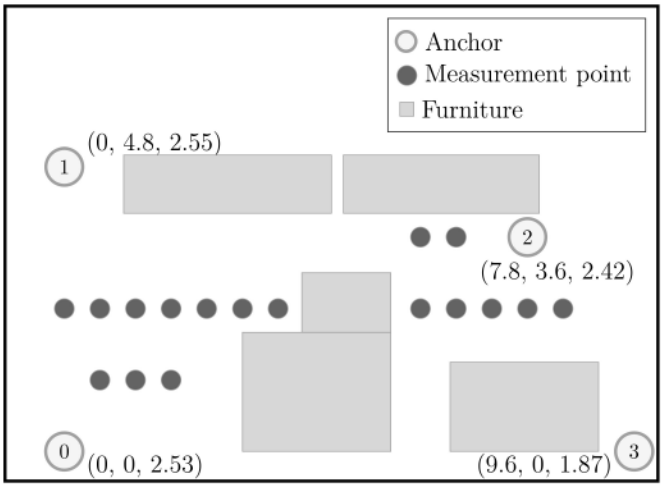

Fig. 2. Indoor measurement setup.

Fig. 3. Outdoor scenario.

second one was carried out in an indoor scenario, in a showroom of the Centre for Information and Communications Technology Research (CITIC), at the University of A Coruña, Spain. The ESP32-S2-Saola-1 development boards from Espressif Systems were used in both cases. Such modules were mounted on tripods and then placed in a series of known positions in order to compare the distance estimates with the actual values. Some of the modules were used as Wi-Fi FTM responders (acting as beacons) and others as communication initiators (acting as tags).

1) Indoor Scenario: The CITIC showroom (see Fig. 1) at the University of A Coruña, Spain, was selected as an indoor environment. This is a space dedicated to the dissemination of the research activities carried out in the center. It is a room with many obstacles inside and surrounded by sources of interference such as other Wi-Fi networks (more than 7 were detected with a high signal level from inside the room) coexisting with ZigBee sensor networks and Bluetooth devices. The dimensions of the room are 12 m long and 6 m wide, with a height of 3.2 m.

To have a greater diversity of measurements, reduce data collection time, and have a data set applicable to future positioning algorithms, four ESP32-S2 devices were placed at the corners of the room acting as beacons, while the communication initiating tag was moved to different points along a straight line. This tag was placed at a height of 2 m. A schematic of the anchors and measurement positions is shown in Fig. 2. Due to some of the obstacles present in the room, measurements could not be taken in certain areas. At each point, measurements were taken for 180 s and the ESP32-S2 devices were configured to work with a 40-MHz bandwidth, which is the maximum bandwidth allowed at the 2.4-GHz band. Notice that, due to the relative placement between tag and beacons in this scenario, measurements were obtained at different distances and with different relative angles between transmitter and receiver.

2) Outdoor Scenario: In order to obtain more information about the capabilities of the ESP32-S2 module for ranging operations with the Wi-Fi FTM protocol, another measurement campaign was performed in an outdoor environment. In this way, we tried to minimize the possible effect of interference and multipath typically found in indoor scenarios. In fact, in this scenario, not a single nearby Wi-Fi network was detected that could be a source of interferences. To carry out the measurements, we selected an area of the campus at the University of A Coruña away from the buildings. Only two modules were used in the experiment, one acting as a tag and the other as a beacon. They were mounted on tripods and placed one in front of the other at the same height (1.75 m). The measurements were taken during 120 s at distances varying from 1 to 20 m, at 1 m intervals. Two different measurement campaigns were performed, one with the modules configured with a bandwidth of 20 MHz, and the other with a bandwidth of 40 MHz. Fig. 3 shows a picture of the outdoor measurement scenario.

C. Software Development

In order to collect the measurements, several software modules were implemented. On the one hand, a new firmware was programmed for the ESP32-S2s acting as beacons. In this case, the application configured the module as a Wi-Fi AP with the

FTM responder features enabled. On the other hand, another firmware was programmed for the module acting as a tag. In this case, the functionality of this firmware was divided into three main parts: 1) a scan of APs capable of responding to an FTM request was performed; 2) once this list was obtained, the module entered in a loop of FTM requests with each of the APs detected in the previous phase; and 3) the measurements obtained were written to the serial port of the development board.

To assess the collected data from a PC, a Robot Operating System (ROS) environment was deployed on a Raspberry Pi 3B. ROS [38] is a software commonly used in robotics and allows for decoupling communications between hardware devices (typically sensors) and the software elements in charge of processing their data. Within ROS, the fundamental working element is the node. A ROS node can publish information in a given topic (a text string similar to a file path or a URL) and can also subscribe to the data available from topics published by other nodes. In our case, we implemented a node capable of collecting the FTM measurements captured from the ESP32-S2 (connected via a USB port) and publishing them in the ROS ecosystem. On the PC we used *rosbag*, an application included by default in ROS that allows for the subscription and subsequent storage of measurements in different logs for later analysis. Those measurements were sent from the Raspberry Pi to the PC via an Ethernet link. Note that the choice of ROS in this work has been solely to take advantage of its communication system and its measurement collection tools, and not because the nature of what is described here is related to the field of robotics.

As part of the contributions of this article, the code of all these software modules is publicly available as open source.[1]

IV. Measurements Analysis

This section shows a descriptive analysis of the measurements obtained in the different campaigns. Each measurement includes the following parameters.

1) `anchorId:` Identifier of the module that acted as beacon in the measurement.
2) `rtt_raw:` RTT value averaged among the different frames sent and expressed in nanoseconds.
3) `rtt_est:` RTT estimation provided by the ESP32-S2 firmware in nanoseconds.
4) `dist_est:` Distance estimation in meters. Internally, `rtt_est` is used to calculate this value.
5) `num_frames:` The number of frames successfully sent during the RTT communication.
6) `frames:` A list of all successfully sent frames.

Each individual frame includes the following information.

1) `rssi:` RSS in dBm.
2) `rtt:` RTT value for that frame in nanoseconds.
3) `t1:` Outgoing timestamp of the first packet from the sender in picoseconds.
4) `t2:` Timestamp of the reception of the ranging request at the receiver expressed in picoseconds.
5) `t3:` Timestamp (in picoseconds) of the response message at the receiver.
6) `t4:` Timestamp (in picoseconds) of the reception of the response message from the receiver at the sender.

The results corresponding to indoor and outdoor campaigns are analyzed in Sections IV-A and IV-B, respectively.

[1]The source code can be downloaded from *https://github.com/valentinbarral/**, where * is one of these project identifiers: *esp32s2-ftm-tag*, *esp32s2-ftm-anchor*, *rosftm*, or *rosmsgs* (see [31]–[34]).

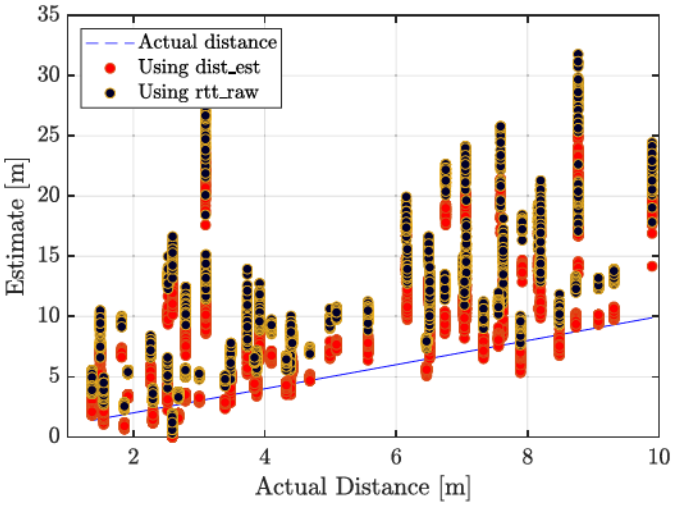

Fig. 4. Actual versus estimated distance in the indoor scenario.

A. Analysis of the Indoor Measurements

The first information we extract from the measurements captured in the indoor environment is the distance estimate. The ESP32-S2 provides two RTT parameters: 1) `rtt_raw`, which corresponds to the average of the values obtained in the different frames and 2) `rtt_est`, which is estimated by the chip. How the `rtt_est` estimate is obtained from the raw value is not documented by the manufacturer. These RTT parameters can be easily translated into a distance estimate d using the formula

$$d = \frac{\texttt{rtt} \cdot c}{2} \tag{1}$$

where `rtt` is the RTT value, and c is the speed of light. Note that depending on the value of `rtt` used, `rtt_est` or `rtt_raw`, two different distance estimates can be obtained. The ESP32-S2 firmware uses `rtt_est` to provide an estimate of the distance between the devices available as `dist_est`.

Fig. 4 shows the captured samples, their actual position, and the two estimates: the one provided by the chip itself, `dist_est`, and the value obtained from `rtt_raw` using (1). The values obtained in this indoor environment are clearly very inaccurate. There are large differences between actual and estimated values. It can be seen that the estimates based on the `rtt_raw` value lead to the worst results, whereas those based on the `dist_est` achieve a significant improvement in terms of accuracy. The error level obtained can be better seen in Fig. 5, where the empirical cumulative distribution function (ECDF) of the absolute error is shown for both cases. Besides observing the general level of inaccuracy of the measurements,

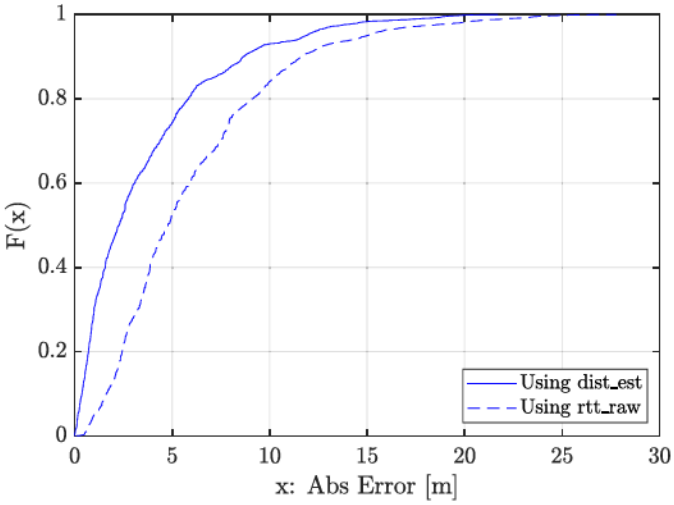

Fig. 5. ECDF of the absolute error in the indoor scenario.

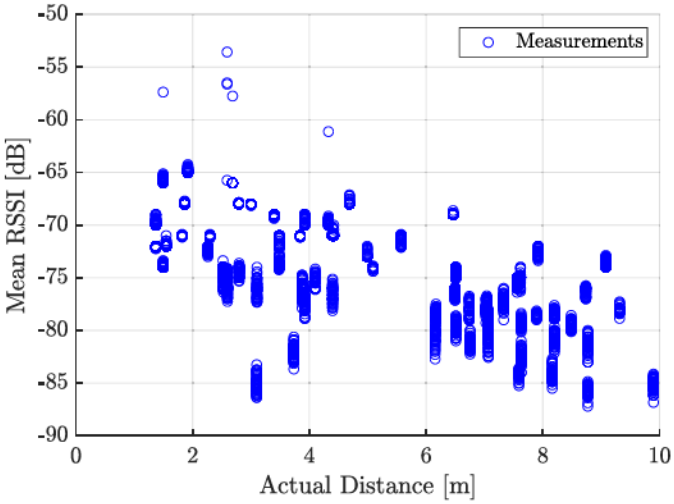

Fig. 6. Mean RSSI for each measurement versus actual distance for the indoor scenario.

with errors up to 20 m in some cases, we can see that the distance estimates obtained by the chip are able to improve those from the raw RTT samples. As an example, by means of the ESP32-S2 estimation, about 75% of the measurements are below 5 m of error, whereas using the `rtt_raw` parameter only 50% are below such a threshold.

Another parameter of interest is the RSSI value. Fig. 6 shows the different measurements in the indoor scenario according to their RSSI value and the actual distance at which they were taken. Note that the ESP32-S2 does not generate an RSSI value for each FTM sample, but rather for each frame transmitted within an FTM communication. In this case, the RSSI values shown in Fig. 6 correspond to the average of all the frames in each FTM sample. It can be observed that there is a slight decay of the energy as the distance between the devices increases. However, the strong multipath propagation in the indoor environment leads to a severe small-scale fading, and some samples are obtained with high RSSI values at long distances and low RSSI values at short distances.

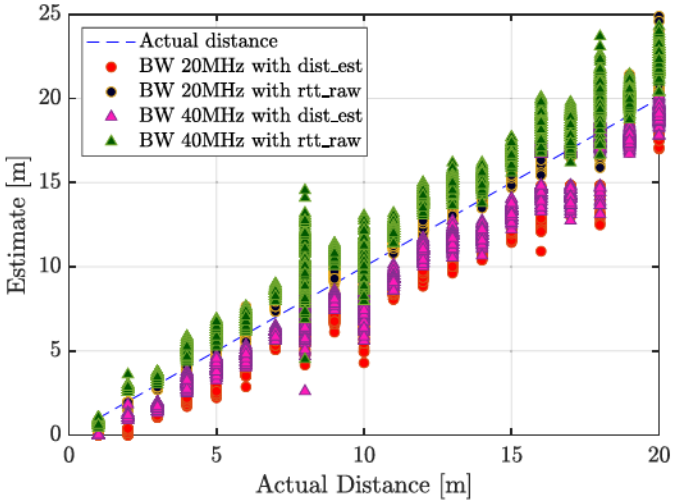

Fig. 7. Comparison between actual and estimated distances for both outdoor data sets, and for bandwidth values of 20 and 40 MHz.

B. Analysis of the Outdoor Measurements

Fig. 7 shows the distance estimates obtained for the two bandwidth values considered (20 and 40 MHz) versus the actual distance. As in the indoor case, both the distance values provided directly by the chip (`dist_est`) and those generated from the temporal `rtt_raw` values from (1) are shown. In Fig. 7, it can be seen that how the estimated values are much closer to the actual ones than in the indoor case (shown in Fig. 4) for both the 20 and 40 MHz bandwidth values. However, in this scenario, a counter-intuitive phenomenon can be seen: the estimates generated with the raw RTT values (`rtt_raw`) are closer to the actual values than the estimates provided by the chip. Such a phenomenon can be clearly observed in Fig. 8, where the estimates from the ESP32-S2 are noticeably worse than those obtained directly from `rtt_raw`. This effect is clearly visible in the measurements corresponding to the 20-MHz data set, whereas in the 40-MHz data sets, both values are more similar. This apparent inconsistency is discussed in more detail in Section IV-C. Fig. 8 also shows that the raw values from the 20-MHz configuration (`rtt_raw`) yield better results than those obtained with 40 MHz, whereas with the values estimated by the chip (`dist_est`), this effect is reversed, and the 40-MHz results are slightly better.

Finally, Fig. 9 shows the RSSI values corresponding to the two bandwidths obtained in the outdoor scenario. We can see that the RSSI values exhibit less small-scale fading than in the indoor case (see Fig. 6), and they are similar regardless of the bandwidth. The energy clearly decreases with distance, although there are points where multipath effects are observed, probably caused by the signal bouncing off the ground or other obstacles close to the measurement area.

C. Estimated RTT Versus Raw RTT

After the analysis of the data captured in the different measurement campaigns, one of the most surprising aspects detected was the difference in accuracy between the distance

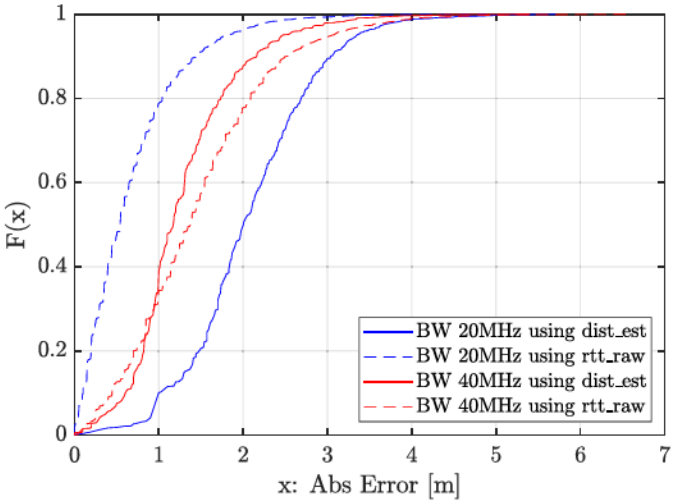

Fig. 8. ECDF of the absolute error for both outdoor data sets and for both bandwidth values (20 and 40 MHz).

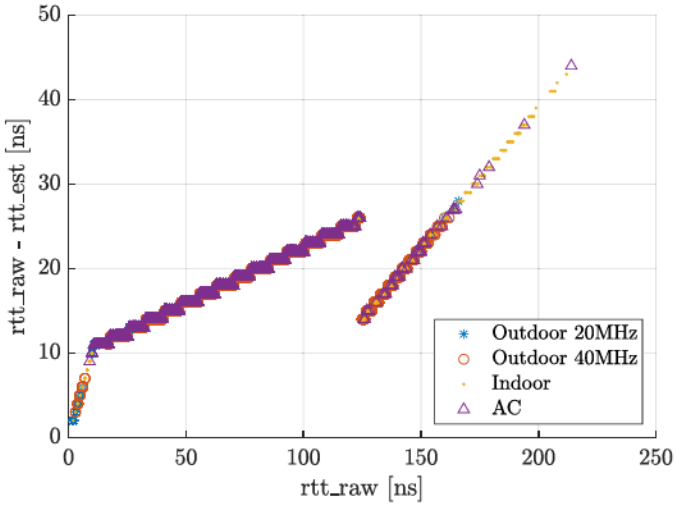

Fig. 10. `rtt_raw` − `rtt_est` for all data sets.

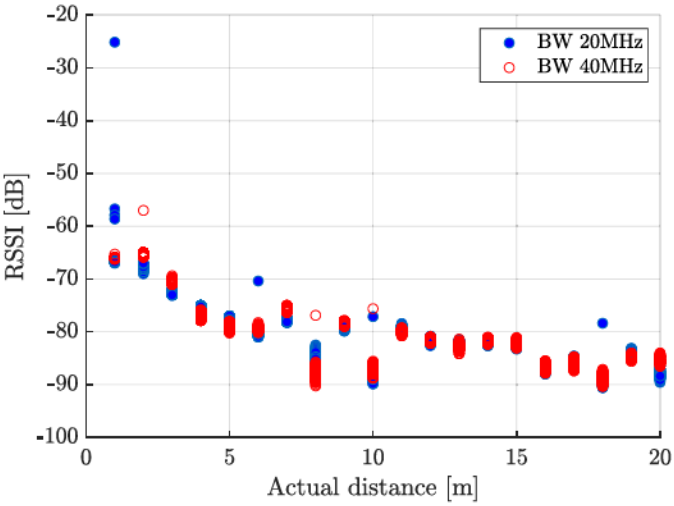

Fig. 9. RSSI versus actual distance for both outdoor data sets, with bandwidth values of 20 and 40 MHz.

estimated by the ESP32-S2 and the value computed directly from the raw RTT values. While in the indoor case, the estimation improved the raw accuracy, this was not the case in the outdoor measurements, especially for the case with a bandwidth of 20 MHz. In the latter case, the ESP32-S2 estimates were clearly worse than those obtained using the `rtt_raw` value directly.

To try to detect the reason for this behavior, and since there is no documentation from the manufacturer about how the distance estimate is generated by the device, a study of the differences between the `rtt_raw` and `rtt_est` values was carried out. Fig. 10 shows the difference between these two values for all captured measurements (indoors and outdoors) with respect to the `rtt_raw` value, in which three linear correction models that are directly related to the `rtt_raw` value itself can be identified. That is, in view of these values, it seems clear that, to generate the `rtt_est` value, the ESP32-S2 simply applies a correction factor to `rtt_raw` that depends linearly on its value. Such a linear relationship has a different gradient depending on whether the values are below 10 ns, between 10 and 124 ns, or above 124 ns. Probably, due to the still recent implementation of the FTM support on these chips, the chosen correction is not the best. Although it is difficult to say for sure, it seems that an attempt was made to improve the accuracy in indoor environments (a typical scenario to use a Wi-Fi module) at the cost of worsening the results in better conditions, with a good LoS, and in the absence of obstacles or significant interference between the devices. What does seem clear is that these thresholds, located at 10 and 124 ns, are consistent across all measurement scenarios, so they must be hard-coded in the ESP32-S2 firmware.

V. Proposed Estimator Based on Machine Learning

After observing the limitations of the current ESP32-S2 estimation algorithm described in Section IV-C, an opportunity appears to propose a better alternative. This new method should be able to operate within the computational limitations of the ESP32-S2, so that it must compute the distance estimate from the `rtt_raw` in real time and for each FTM communication, providing a more accurate result. Having hundreds of data records after the measurement campaigns, one of the direct approaches to the problem would be using ML techniques to generate a regression model of the desired correction. This approach is very common to deal with this type of problems [35], [39] and was also used previously by the authors in previous works for the UWB technology [40].

As training features, we selected the `rtt_raw` values and the average RSSI values, discarding other parameters that did not provide any improvement in a preliminary study, such as the number of frames successfully transmitted or the variance of `rtt_raw` for each sample. To create the training set, we randomly selected 70% of the samples from each of the three data sets (the one corresponding to the indoor measurements and the two outdoors) and merged them into a single one. The

remaining samples were reserved as test sets. At the time of training, cross-validation was used with fivefold, an adequate number for the not excessively large size of the training set (around 7000 samples). The Bayesian optimization [41] was used to search for the best hyperparameters of each algorithm.

The considered ML algorithms were regression trees, support vector machines (SVMs), Gaussian process (GP) regression, and shallow NNs. Their details can be found in Section V-A. More complex techniques, such as those based on deep learning, were discarded because of their computational needs, out of the capabilities of the ESP32-S2, and because the number of samples available was too low to apply these techniques with confidence.

A. Description of the ML Algorithms

The considered ML algorithms are described below. All of them are classical algorithms in the literature, and they are widely used in regression problems.

1) *Regression Trees:* This is an intuitive technique based on binary search trees applied to the regression problem [42]. The idea of these algorithms is to associate homogeneous input sets with the same output. The minimum leaf size is a fundamental parameter in this type of algorithms since, depending on its value, the trees can easily tend to overfitting. In the case presented in this work, the regression tree had a minimum leaf size equal to four.
2) *SVM:* This is another classic ML algorithm originally described in [43] that can be used in both classification and regression problems [44]. Among the hyperparameters that can be configured in this algorithm, the main one is the type of kernel used to map the input elements in the n-dimensional feature space in which the regression process is applied. In the implementation considered in this work, a Gaussian kernel was chosen with the form
$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) \tag{2}$$
being $\mathbf{x}_i, \mathbf{x}_j \in R^n$ with $n = 2$ since the input vectors consider two features in our case.
3) *GP:* This is a generalization of the Gaussian probability distribution in which the behavior of a function is described. Using this idea, regression and classification models are built with high accuracy and performance [45]. In the implementation considered in this work, we considered an exponential kernel
$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp(-r/\sigma_l) \tag{3}$$
being $\mathbf{x}_i, \mathbf{x}_j \in R^n$, σ_f is the signal standard deviation, and σ_l is the characteristic length scale. Finally, r is defined as the Euclidean distance between the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$
$$r = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)}.$$
After the validation phase, the kernel parameters for the best case were set to $\sigma_f = 4.6873$ and $\sigma_l = 0.7051$.
4) *Neural Network:* The last of the considered classical algorithms was a fully connected feedforward NN. This is one of the most classic ML algorithms, already referenced in the middle of the last century, and a pillar of the most complex networks used nowadays in deep learning [46]. In this work, we implemented a simple configuration with a single hidden layer consisting of 100 neurons with a rectified linear unit (ReLU) activation function, and a linear activation function in the output layer. All these parameters, as in the rest of the algorithms, were obtained using the Bayesian optimization mechanism limited to 50 iterations.

Table I shows a summary of the parameters of each algorithm.

TABLE I
MAIN PARAMETERS OF TESTED ML ALGORITHMS

Algorithm	Main Parameters
Regression Trees	Minimum leaf size = 4
SVM	Kernel: Gaussian
GP	$\sigma_f = 4.6873$ $\sigma_l = 0.7051$
NN	Number of hidden layers = 1 Number of neurons = 100 Activation function = ReLU Activation function output = linear

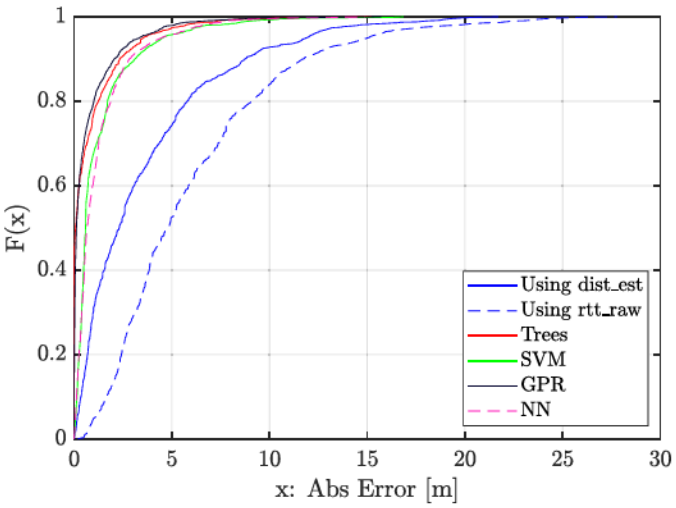

Fig. 11. ECDF of the absolute error. Test set: indoors with a bandwidth of 20 MHz.

B. Results of the ML Algorithms

This section shows the results obtained after applying the trained algorithms to the different test sets, each one coming from a different data set. Fig. 11 shows the ECDF of the absolute error in the indoor scenario in which all the trained ML algorithms exhibit a clear improvement over the estimates from the ESP32-S2. Even with a simple algorithm, such as the regression trees, a noticeable improvement is achieved for most of the samples, being able to place 80% of them below 1.5-m error, whereas with the original estimates, the corresponding error increases up to 6 m.

Fig. 12 shows the ECDF of the absolute error in the outdoor scenario considering a bandwidth of 20 MHz. This was the case in which the ESP32-S2 estimates were clearly worse than

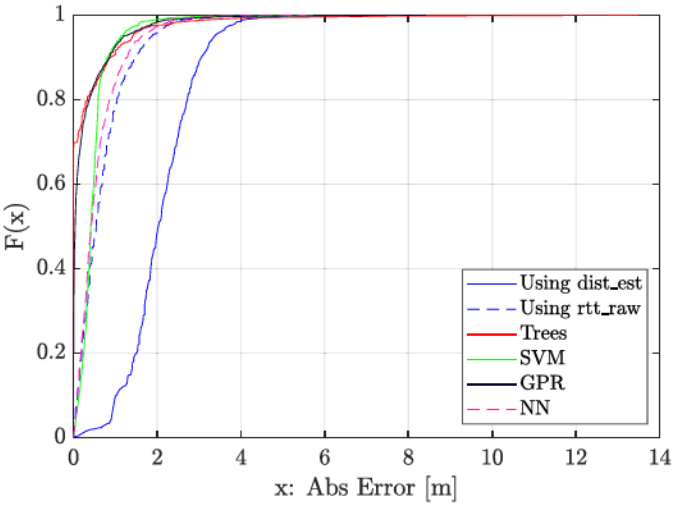

Fig. 12. ECDF of the absolute error. Test set: outdoors with a bandwidth of 20 MHz.

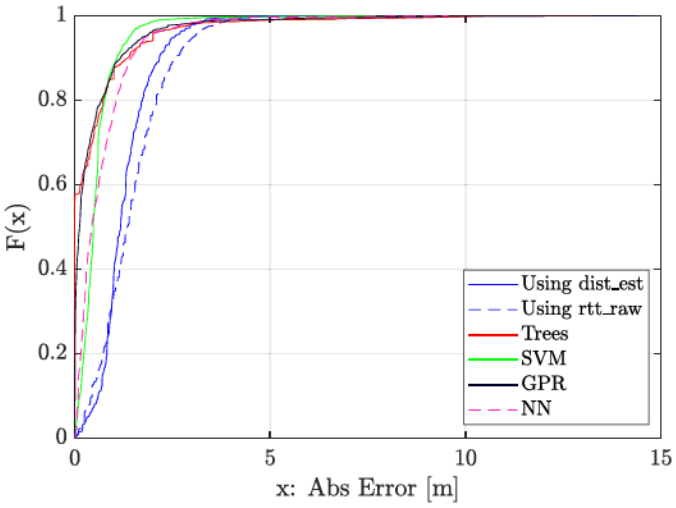

Fig. 13. ECDF of the absolute error. Test set: outdoors with a bandwidth of 40 MHz.

those computed using the `rtt_raw` parameter directly. In view of Fig. 12, all the ML algorithms outperform both the ESP32-S2 estimates and the values computed from `rtt_raw`.

Finally, Fig. 13 shows the ECDF of the absolute error also in the outdoor scenario, but in this case using the 40-MHz bandwidth configuration. With this configuration, and according to the ECDF in Fig. 8, there was originally very little difference between the ESP32-S2 estimates and the values computed from the `rtt_raw` parameter (although the estimates yielded the best result, as shown in Fig. 8). However, in view of Fig. 13, a significant improvement is observed when the ML algorithms are employed.

In a conclusion, in view of the results shown in Figs. 11–13, ML-based estimators improve the accuracy of the measurements to some degree. However, it should be noted that, although the test set did not include any samples in common with the training set, it is true that the general distribution is similar because they all come from the same data set. For this reason and to assess the robustness of the models, a new

Fig. 14. Scenario in the Scientific Area building.

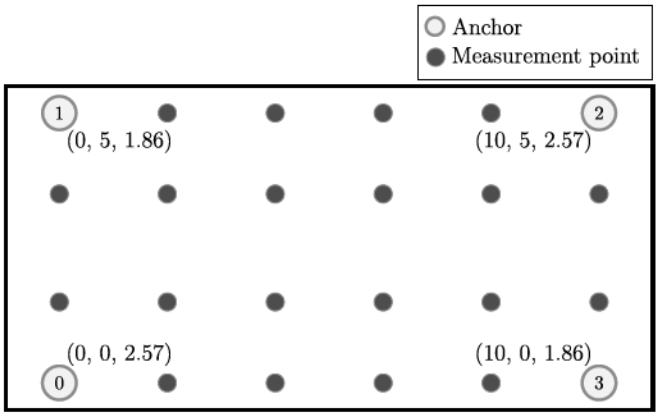

Fig. 15. Measurement setup of the scenario in the Scientific Area building. Anchors 0 and 1 use a bandwidth of 20 MHz, whereas anchors 2 and 3 employ 40 MHz.

experiment was carried out in a completely new measurement scenario. The details and results obtained from this experiment are shown in Section VI.

VI. Assessment of the ML Trained Models in Different Environment

A proposal was presented in Section V to use classical ML algorithms to estimate the actual distance from the `rtt_raw` value and the signal level. These algorithms were trained with part of the samples collected in the two proposed scenarios, the outdoor and the indoor one. The results of this training were tested with the samples from these data sets that had not been included in the training set. In a further step, to validate this approach and check if it can be used in other different environments, a new measurement campaign was carried out in a different scenario. This new environment is described in Section VI-A.

A. Test Scenario

To test the trained models, we looked for a scenario different from the outdoor and indoor scenarios already considered. The place chosen (see Fig. 14) was the ground floor of the Scientific Area building, a research building located in the Campus of Elviña at the University of A Coruña, Spain. From the point of view of radio propagation, it is a complicated scenario due to its low ceiling, glazed areas, and metal structures

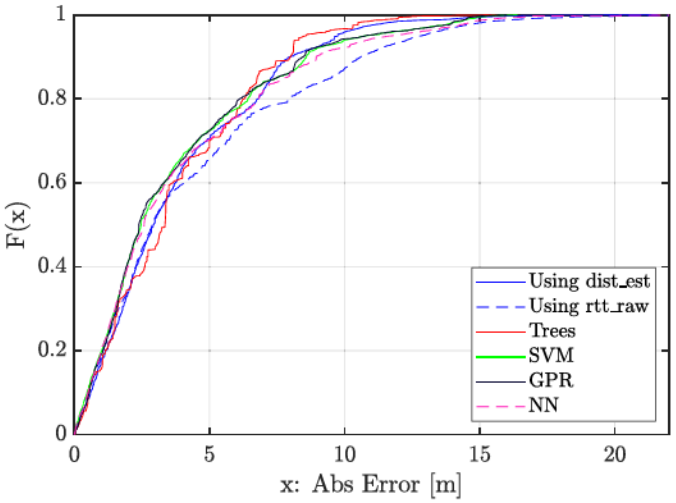

Fig. 16. ECDF of the absolute error. Test set: Scientific Area building with bandwidth values of 20 and 40 MHz.

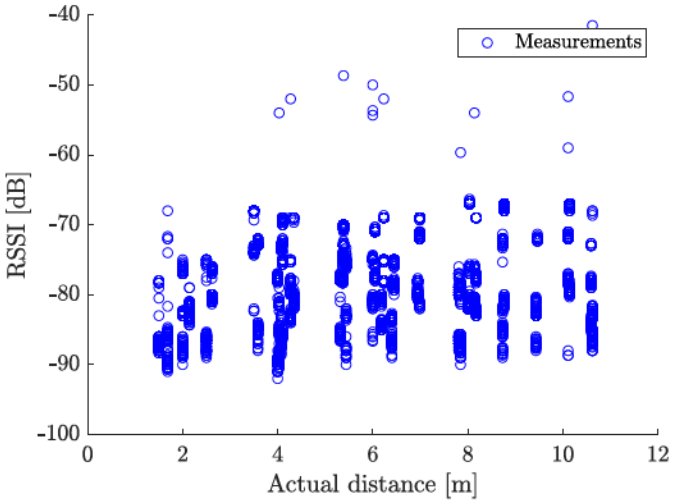

Fig. 17. Mean RSSI on the measurements captured in the Scientific Area building with bandwidth values of 20 and 40 MHz.

located at different points. In addition, being a research building, there are several nearby devices radiating in the 2.4-GHz band that could affect to some degree the communications with the ESP32-S2. Specifically, after scanning inside the area, 5–7 APs with a medium-low signal level were detected.

To perform the measurements, four tripods were placed at the corners of a 10×5 m rectangle, mounting an ESP32-S2, configured as an anchor, on each of them (at different heights), as shown in Fig. 15. Anchors labeled with 0 and 1 employed a bandwidth of 20 MHz, whereas anchors 2 and 3 used 40 MHz. A series of points were marked within a grid as shown in Fig. 15 and measurements were taken with a fifth ESP32-S2 device mounted on another tripod and moved over each of the measurement points. FTM measurements were captured for 120 s at each point using the same software and hardware configuration as in the previous data sets. That is, a Raspberry Pi 3B connected to the ESP32-S2, which acted as a tag, was in charge of reading the measured values through the serial port and publishing them within a ROS environment to be later saved in a remote PC. All the samples collected, as well as those from the other data sets, are publicly available to the research community in [30].

B. Results

After performing the measurement campaign in the Scientific Area building, the algorithms trained with the other data sets were run on the new samples. All the results from this measurement scenario consider simultaneously values corresponding to 20 and 40 MHz bandwidth values. Fig. 16 shows the ECDF of the absolute error and several details of interest can be observed. First, we see that the error is slightly lower than in the indoor scenario (see Fig. 11), but much higher than in the outdoor scenario (see Figs. 12 and 13). On the other hand, we see that the ESP32-S2 estimation improves the value computed directly from `rtt_raw`, although the differences are smaller than in the indoor scenario. These differences are found in the values with the highest level of error, whereas the number of values with errors below 4 m is practically the same using both approaches.

As for the predictions of the ML algorithms, we see in Fig. 16 that we do not achieve a performance improvement similar to that obtained with the other data sets. The values obtained are very close to the ESP32-S2 estimates, slightly better in some cases. The reason for this difference with respect to the results is due to the nature of the chosen scenario and its propagation problems, as well as by the ESP32-S2 antenna and its radiation pattern. Fig. 17 shows the RSSI-level distribution of the samples with respect to the distance. There are hardly any differences between the values obtained at a distance of 1.5 m with respect to those captured at distances of more than 10 m. If we compare these values with those obtained from the other scenarios (see Figs. 6 and 9), we can see that, in this case, the values are strongly affected by multipath, and there is not a clear energy drop as the distance between the emitter and the receiver increases. This explains, at least in part, why the estimation algorithms in this scenario, which were trained with the `rtt_raw` features and the RSSI, do not perform as well as in the other scenarios. However, it must be said that the estimations are never worse than those provided by the ESP32-S2, hence it is realistic to think that, in less aggressive scenarios, the results will be better and more similar to those obtained in the initial test scenarios. Therefore, ML algorithms provide stability and robustness against any kind of scenario, being a more adequate solution than the one included with the ESP32-S2 firmware.

VII. Deployment in the ESP32-S2

Taking advantage of the ESP32-S2 capabilities, one of the previously trained algorithms was implemented inside the chip. In particular, due to its simplicity, we implemented the regression trees algorithm. This algorithm, once trained, has very low computational requirements to generate a new estimate. In this way, real-time estimates could be obtained at the same time as the ESP32-S2's own estimates were

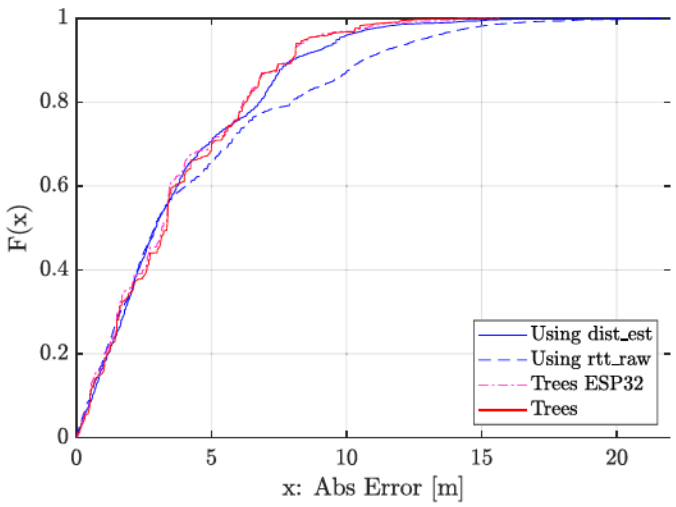

Fig. 18. ECDF of the absolute error. Test set: Scientific Area building. Offline versus real-time ESP32-S2 implementation considering bandwidth values of 20 and 40 MHz.

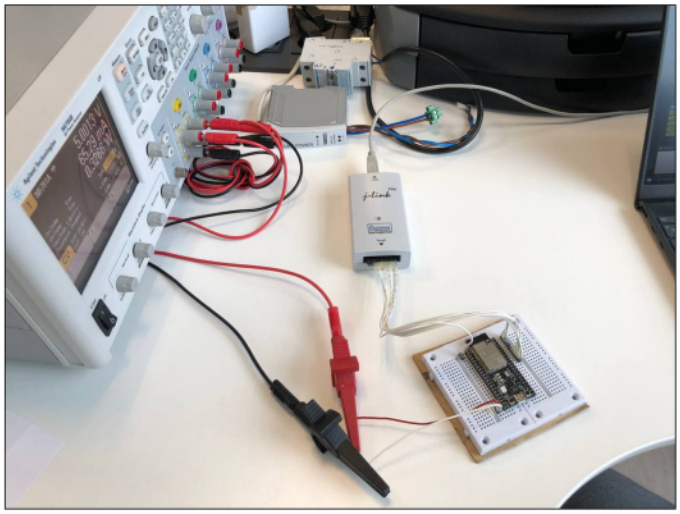

Fig. 19. ESP32-S2 current consumption measurement setup.

calculated. Thus, the measurements obtained during the measurement campaign in the Scientific Area building already include an additional `own_est` parameter that corresponds to the real-time estimation provided by this algorithm.

Hence, the ESP32-S2 firmware was modified to include a new function whose workflow was:

1) get the `rtt_raw` value of the last FTM measurement;
2) generate the average RSSI of all frames correctly sent within the measurement;
3) normalize these values with the same normalization parameters used during the training phase;
4) run the regression tree algorithm to estimate a distance value;
5) add this new value as the `own_est` parameter within the measurements written to the serial port.

For the implementation, we started from the code generated using the MATLAB Coder utility. Once this base was obtained, it was modified and adapted to the needs of the ESP32-S2, both in terms of memory management and the use of libraries. This implementation is also publicly available as open source [31]. The final size of the executable is only 403 kB, out of a total of 4 MB available on the ESP32-S2.

Fig. 18 shows the ECDF of the absolute value of the two versions: 1) the offline algorithm and 2) the real-time algorithm implemented within the chip. It can be seen how the results are practically identical, the small differences being due to possible inaccuracies in some values at the time of averaging the RSSI or the normalization process.

A. ESP32 Current Consumption

Within the IoT world, sensor energy consumption is often a factor of utmost importance. This section shows the current consumption details of the ESP32-S2 when performing positioning tasks using FTM. These measurements were obtained using an N6705 Keysight DC Power Analyzer (see Fig. 19). Notice that the voltage applied to the power supply was set to 5 V, hence the conversion from current to power consumption is direct.

In order to get the current consumption values, the board was switched to the deep-sleep mode, and the red LED was removed to save energy and to obtain more realistic current measurements. The current consumption obtained was 560.6 μA, which basically includes the consumption of the ESP32-S2, the USB-UART bridge, and the low-dropout regulator (LDO). To break down the consumption of each of these components, the ESP32-S2 and the USB-UART bridge were disassembled, and the consumption was measured again. In this way, the LDO consumption was obtained, which was 337.9 μA. Taking into account the USB-UART bridge, and the ESP32-S2 datasheets [37], [47], the typical current consumption, when they are not in use, ranges between 195 and 200 μA in the case of the USB-UART bridge, and between 20 and 25 μA in the case of the ESP32-S2. The difference between 560.6 and 337.9 μA results in 222.7 μA, which corresponds to the current consumption specified for the ESP32-S2 and the USB-UART bridge. Once these calculations were available, the current consumption of the ESP32-S2 module was estimated, which varies between $222.7 - 200 = 22.7$ μA and $222.7 - 195 = 27.7$ μA.

Finally, after assembling the components that were removed from the board, the current consumption was measured while performing FTM with a resolution of 128 000 points/s. The setup can be seen in Fig. 19, whereas Fig. 20 shows the obtained results. When the ESP32-S2 is transmitting, high consumption peaks appear with a maximum of 454 mA, and when it is not transmitting, the consumption is almost constant around 73 mA. Since an FTM operation takes 636 ms to complete, during that time the average consumption is 75.6 mA. This is a large current value since the microcontroller unit (MCU) cannot be switched to the deep-sleep mode when it is transmitting data. If the chip were switched to the deep-sleep mode when it is not transmitting, the average consumption could be reduced significantly.

This large difference in power consumption between the idle mode and the FTM transmission mode can be seen more

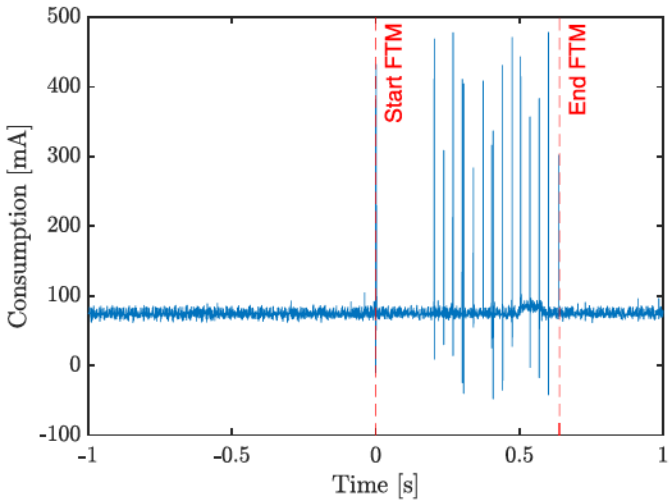

Fig. 20. ESP32-S2 current consumption while FTM protocol.

clearly using the concept of the energy budget. This concept describes the energy consumption of the chip depending on the different working modes in which it is configured throughout the day. This idea is similar to the one presented in other works, such as [48] and [49]. Thus, we define the daily energy budget as

$$E_{(\text{daily budget})} = E_{(\text{idle})} + E_{(\text{FTM})} \tag{4}$$

where $E_{(\text{daily budget})}$ is the energy consumed by the chip in one day, $E_{(\text{idle})}$ is the energy consumed when the chip is in the idle mode, and $E_{(\text{FTM})}$ is the energy consumed when FTM protocol messages are being transmitted. Obviously, this energy consumption will be different depending on the number of FTM measurements taken during the day. Thus, as an example, Fig. 21 shows the daily energy budget considering two different intervals. In the case of Fig. 21(a), the energy values are considered when performing a single FTM measurement per minute, while in Fig. 21(b), the values correspond to a measurement periodicity of 10 min. It can be seen that how the impact of the FTM measures is very large in the total, even though the chip spends most of the day in idle mode (98.8% of the time when the FTM periodicity is 1 min, and 99.89% of the time when the FTM periodicity is 10 min).

In order to verify the impact on the current consumption caused by the proposed solution based on ML, new measurements were taken with and without this functionality activated. Table II shows an estimation of the average current consumption if the number of FTM frames were 1 every 10 s, 1 every min, etc., considering a 2000-mAh battery to estimate the battery lifetime. The estimation was made taking into account that the chip was in the deep-sleep mode when it was not transmitting. As it can be seen in Table II, the proposal hardly has an impact on the current consumption.

VIII. Conclusion and Future Work

In this work, we have analyzed the performance of the ESP32-S2 module as a device capable of providing distance estimation using FTM Wi-Fi technology implemented according to the IEEE 802.11-2016 standard. For this purpose, we

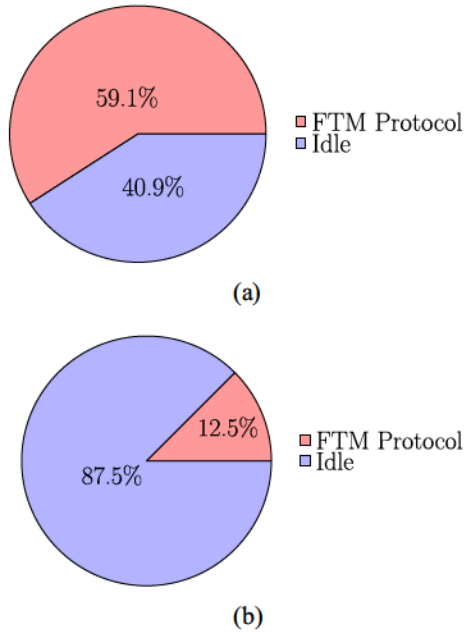

Fig. 21. Energy budget for one day performing a different number of FTM transmissions. (a) Performing a single FTM transmission per minute. (b) Performing a single FTM transmission per 10 min.

TABLE II
FTM Current Consumption Estimation and Estimated Battery Lifetime With a 2000-mAh Battery

Algorithm	FTM measurement period	Current consumption	Estimated battery lifetime
FTM regression trees	10 s	5.33 mA	15 days
	1 min	1.36 mA	61 days
	10 min	0.64 mA	130 days
	30 min	0.59 mA	142 days
	1 h	0.57 mA	145 days
FTM Espressif	10 s	5.29 mA	15 days
	1 min	1.35 mA	61 days
	10 min	0.64 mA	130 days
	30 min	0.59 mA	142 days
	1 h	0.57 mA	145 days

have performed several measurement campaigns in different scenarios, using several ESP32-S2 devices with a specific firmware. Subsequently, we have evaluated the estimates obtained, resulting in high error values in indoor environments (up to 5 m for the 75% of the measurements with a bandwidth of 20 MHz) and lower errors in outdoor environments (up to around 1.5 m for the 75% of the measurements with a bandwidth of 40 MHz and up to 2.5 m when the bandwidth is 20 MHz). This difference in the error levels is due to several factors. One of them is the utilization of an algorithm based on an energy threshold to perform the temporal marking of the packets during the FTM protocol. In this case, multipath and interference problems in indoor environments cause that the first path of the signal be missed, and a bounce is taken instead. In this case, the time of flight (TOF) of the signal becomes longer than the actual one, and therefore the distance estimation is also longer.

During the study, it has also been found that the ESP32-S2 employs RTT distance estimates different than the ones observed in the frames during FTM communications. Although this is not detailed in the ESP32-S2 documentation, we have experimentally verified that a linear correction is applied to the raw RTT values depending only on the values themselves. We have found that this approach improves the accuracy in some cases but produces large errors in other cases.

We have proposed to use classical ML algorithms to perform the final distance estimation, taking as training features the raw RTT values and the signal level indicator of each sample. We have trained and tested the algorithms, showing an improvement in the absolute error obtained. Additionally, we have implemented one of these trained algorithms inside the ESP32-S2, so that we have been able to generate the estimates in real time and simultaneously with those provided by the chip. An additional measurement campaign was carried out, in a totally different scenario from those considered to obtain the training samples, with the objective of validating the ML algorithms and the real-time implementation. The results obtained, despite the difficulties of the environment, demonstrated that our proposal based on ML is suitable to address the problem of generating the final estimates from the RTT values.

Overall, the FTM implementation in the ESP32-S2 is still far from being usable in complex indoor environments. While its performance in outdoor environments exhibited good results, with almost 90% of the samples below 2 m of absolute error for a bandwidth value of 40 MHz, in indoor environments, only 40% of the samples fall below such a threshold (for a bandwidth value of 20 MHz). In addition, in the indoor scenario, there were very large estimation errors for some of the samples, with almost 10% of the measurements having more than 8 m of error. Thus, with these results, it would be very difficult to generate 2-D or 3-D position estimates with an error smaller than the size of a medium-sized room.

In future work, new experiments will be designed to assess the accuracy obtained in a 3-D environment, considering not only the location in a plane but also the height.

REFERENCES

[1] F. Khelifi, A. Bradai, A. Benslimane, P. Rawat, and M. Atri, "A survey of localization systems in Internet of Things," *Mobile Netw. Appl.*, vol. 24, pp. 761–785, Jun. 2019.

[2] P. Tedeschi, S. Bakiras, and R. Di Pietro, "IoTrace: A flexible, efficient, and privacy-preserving IoT-enabled architecture for contact tracing," *IEEE Commun. Mag.*, vol. 59, no. 6, pp. 82–88, Jun. 2021.

[3] Y. Li *et al.*, "Toward location-enabled IoT (LE-IoT): IoT positioning techniques, error sources, and error mitigation," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4035–4062, Mar. 2021.

[4] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodriguez, C. Vargas-Rosales, and J. Fangmeyer, "Evolution of indoor positioning technologies: A survey," *J. Sens.*, vol. 2017, Mar. 2017, Art. no. 2630413.

[5] W. Sakpere, M. Adeyeye-Oshin, and N. B. Mlitwa, "A state-of-the-art survey of indoor positioning and navigation systems and technologies," *South African Comput. J.*, vol. 29, no. 3, pp. 145–197, 2017.

[6] C. Laoudias, A. Moreira, S. Kim, S. Lee, L. Wirola, and C. Fischione, "A survey of enabling technologies for network localization, tracking, and navigation," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3607–3644, 4th Quart., 2018.

[7] F. Seco and A. R. Jiménez, "Smartphone-based cooperative indoor localization with RFID technology," *Sensors*, vol. 18, no. 1, p. 266, 2018.

[8] C. Yang and H.-R. Shao, "WiFi-based indoor positioning," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 150–157, Mar. 2015.

[9] M. Terán, J. Aranda, H. Carrillo, D. Mendez, and C. Parra, "IoT-based system for indoor location using Bluetooth low energy," in *Proc. IEEE Colombian Conf. Commun. Comput. (COLCOM)*, 2017, pp. 1–6.

[10] C.-N. Huang and C.-T. Chan, "ZigBee-based indoor location system by k-nearest neighbor algorithm with weighted RSSI," *Procedia Comput. Sci.*, vol. 5, pp. 58–65, Jan. 2011.

[11] Z. Sahinoglu, S. Gezici, and I. Guvenc, *Ultra-Wideband Positioning Systems*. New York, NY, USA: Cambridge Univ. Press, 2008.

[12] E. Au, "The latest progress on IEEE 802.11mc and IEEE 802.11ai [standards]," *IEEE Veh. Technol. Mag.*, vol. 11, no. 3, pp. 19–21, Sep. 2016.

[13] O. Hashem, K. A. Harras, and M. Youssef, "Accurate indoor positioning using IEEE 802.11mc round trip time," *Pervasive Mobile Comput.*, vol. 75, Aug. 2021, Art. no. 101416.

[14] C. Gentner, M. Ulmschneider, I. Kuehner, and A. Dammann, "WiFi-RTT indoor positioning," in *Proc. IEEE/ION Position Location Navig. Symp. (PLANS)*, 2020, pp. 1029–1035.

[15] A. Dammann, R. Raulefs, and S. Zhang, "On prospects of positioning in 5G," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, 2015, pp. 1207–1213.

[16] R. M. Buehrer, H. Wymeersch, and R. M. Vaghefi, "Collaborative sensor network localization: Algorithms and practical issues," *Proc. IEEE*, vol. 106, no. 6, pp. 1089–1114, Jun. 2018.

[17] N. Suryavanshi, K. V. Reddy, and V. Chandrika, "Direction finding capability in Bluetooth 5.1 standard," in *Proc. Ubiquitous Commun. Netw. Comput.*, Bangalore, India, Feb. 2019, pp. 53–65.

[18] "Wi-Fi RTT (IEEE 802.11mc)." 2021. [Online]. Available: https://source.android.com/devices/tech/connect/wifi-rtt

[19] M. Bullmann, T. Fetzer, F. Ebner, M. Ebner, F. Deinzer, and M. Grzegorzek, "Comparison of 2.4 GHz WiFi FTM- and RSSI-based indoor positioning methods in realistic scenarios," *Sensors*, vol. 20, no. 16, p. 4515, 2020.

[20] B. K. P. Horn, "Doubling the accuracy of indoor positioning: Frequency diversity," *Sensors*, vol. 20, no. 5, p. 1489, 2020.

[21] M. Ibrahim *et al.*, "Verification: Accuracy evaluation of WiFi fine time measurements on an open platform," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 417–427. [Online]. Available: https://doi.org/10.1145/3241539.3241555

[22] "88W8987 DB 1x1 802.11ac, Bluetooth 5.2." [Online]. Available: https://www.nxp.com/products/wireless/wi-fi-plus-bluetooth/2-4-5-ghz-dual-band-1x1-wi-fi-5-802-11ac-plus-bluetooth-5-2-solution:88W8987 (Accessed: Sep. 22, 2021).

[23] "88MW32X Wi-Fi® Microcontroller SoC." [Online]. Available: https://www.nxp.com/products/wireless/wi-fi-plus-bluetooth/88mw32x-802-11n-wi-fi-microcontroller-soc:88MW32X (Accessed: Sep. 22, 2021).

[24] "Wireless MCUs." [Online]. Available: https://www.cypress.com/products/wireless-mcus (Accessed: Sep. 22, 2021).

[25] "Release ESP-IDF Pre-Release v4.3-Beta1." espressif/esp-idf. 2021. [Online]. Available: https://github.com/espressif/esp-idf/releases/tag/v4.3-beta1

[26] "Global SOC as a Service Market Size, and Forecast to 2028." 2021. [Online]. Available: https://www.quincemarketinsights.com/industry-analysis/soc-as-a-service-market

[27] "ESPHome." 2021.[Online]. Available: https://esphome.io/index.html

[28] "Tasmota." 2021. [Online]. Available: https://tasmota.github.io/docs/About/

[29] "ESP Easy." 2021. [Online]. Available: https://espeasy.readthedocs.io/en/latest/ESPEasy/AboutUs.html

[30] V. Barral, O. Campos, T. Domínguez-Bolao, C. J. Escudero, and J. A García-Naya, 2021. "ESP32S2 FTM Measurements," tex.referencetype: data. [Online]. Available: https://dx.doi.org/10.21227/2pv8-ze59

[31] V. Barral. "ESP32 FTM Tag Source Code." 2021. [Online]. Available: https://github.com/valentinbarral/esp32s2-ftm-tag

[32] V. Barral. "ESP32 FTM Anchor Source Code." 2021. [Online]. Available: https://github.com/valentinbarral/esp32s2-ftm-anchor

[33] V. Barral. "ROS FTM Reader Source Code." 2021. [Online]. Available: https://github.com/valentinbarral/rosftm

[34] V. Barral. "ROS Messages Source Code." 2021. [Online]. Available: https://github.com/valentinbarral/rosmsgs

[35] N. Dvorecki, O. Bar-Shalom, L. Banin, and Y. Amizur, "A machine learning approach for Wi-Fi RTT ranging," in *Proc. Int. Techn. Meeting Inst. Navig.*, Reston, VA, USA, Jan. 2019, pp. 435–444.
[36] J. Choi, Y.-S. Choi, and S. Talwar, "Unsupervised learning techniques for trilateration: From theory to android APP implementation," *IEEE Access*, vol. 7, pp. 134525–134538, 2019.
[37] "ESP32-S2." Espresssif Systems. 2021. [Online]. Available: https://www.espressif.com/en/products/socs/esp32-s2
[38] "ROS Main Site." ROS. 2021. [Online]. Available: http://www.ros.org
[39] H. Wymeersch, S. Marano, W. M. Gifford, and M. Z. Win, "A machine learning approach to ranging error mitigation for UWB localization," *IEEE Trans. Commun.*, vol. 60, no. 6, pp. 1719–1728, Jun. 2012.
[40] V. Barral, C. J. Escudero, J. A. García-Naya, and P. Suárez-Casal, "Environmental cross-validation of NLOS machine learning classification/mitigation with low-cost UWB positioning systems," *Sensors*, vol. 19, no. 24, p. 5438, 2019.
[41] M. Pelikan and D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. 1st Annu. Conf. Genet. Evol. Comput.*, 1999, pp. 525–532.
[42] G. G. Moisen, "Classification and regression trees," in *Encyclopedia of Ecology*, vol. 1, S. E. Jørgensen and B. D. Fath, Eds. Oxford, U.K.: Elsevier, 2008, pp. 582–588. [Online]. Available: https://www.fs.usda.gov/treesearch/pubs/30645
[43] V. Vapnik, *The Nature of Statistical Learning Theory* (Information Science and Statistics), 2nd ed. New York, NY, USA: Springer-Verlag, 2000. [Online]. Available: https://www.springer.com/gp/book/9780387987804
[44] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Int. Conf. Adv. Neural Inform. Process. Syst.*, vol. 28, Jan. 1997, pp. 779–784.
[45] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning: ML Summer Schools*, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds. Heidelberg, Germany: Springer, 2004, pp. 63–71. [Online]. Available: https://doi.org/10.1007/978-3-540-28650-9_4
[46] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015. [Online]. Available: http://arxiv.org/abs/1404.7828
[47] "CP2102N-GQFN28 USBXpress USB Bridges." Silicon Labs. 2021. [Online]. Available: https://www.silabs.com/interface/usb-bridges/usbxpress/device.cp2102n-gqfn28
[48] M. Pincheira, M. Vecchio, R. Giaffreda, and S. S. Kanhere, "Cost-effective IoT devices as trustworthy data sources for a blockchain-based water management system in precision agriculture," *Comput. Electron. Agr.*, vol. 180, Jan. 2021, Art. no. 105889.
[49] T. Bouguera, J.-F. Diouris, J.-J. Chaillout, R. Jaouadi, and G. Andrieux, "Energy consumption model for sensor nodes based on LoRa and LoRaWAN," *Sensors*, vol. 18, no. 7, p. 2104, Jul. 2018.

Valentín Barral Vales received the Ph.D. degree in computer engineering with a thesis titled *Ultra Wideband Location in Scenarios Without Clear Line of Sight* from the University of A Coruña, A Coruña, Spain, in 2020.

He works as an Associate Researcher with the University of A Coruña. His research field is indoor localization using radio technologies, a field in which he has published several works since 2012. He has also participated in many national and European projects, always related to indoor localization in complex environments.

Omar Campos Fernández received the B.S. degree in electronics engineering from the University of A Coruña, A Coruña, Spain, in 2019, and the M.Sc. degree in electronics systems engineering from the University Carlos III of Madrid, Madrid, Spain, in 2020.

His research interests include indoor localization systems and wireless sensor networks.

Tomás Domínguez-Bolaño received the B.S. degree in computer engineering and the Ph.D. degree in computer engineering (with the Distinction "Doctor with European Mention") from the University of A Coruña, A Coruña, Spain, in 2014 and 2018, respectively.

Since 2014, he has been with the Group of Electronics Technology and Communications, University of A Coruña. In 2018, he was a Visiting Scholar with Tongji University, Shanghai, China. He is an author of more than 15 papers in peer-reviewed international journals and conferences. He was awarded with a predoctoral grant and two research-stay grants. His research interests include channel measurements, parameter estimation and modeling, and experimental evaluation of wireless communication systems.

Carlos J. Escudero (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of A Coruña (UDC), A Coruña, Spain, in 1998.

He is a Full Professor with UDC in the area of signal theory and communications. Since 1992, he has been a Faculty Member with the Department of Computer Engineering, UDC, where he became an Associate Professor in 1998. He was the Secretary of the Department, the Vice-Dean of the Faculty of Informatics, and the Head of the Department of Electronics and Systems, UDC, where he has been the Deputy Rector of Information and Communication Technologies (University CIO) since January 2016. His research focuses on the applications of signal processing in communications, wireless sensor networks, and indoor location systems. Proof of this are some of the merits alleged in the curriculum: publications in the most prestigious journals and books, many national and international collaborations with renowned researchers, dozens of papers/publications in national and international conferences, highlighting the most significant in the curriculum, participation in more than 25 competitive research projects in which he has been the principal investigator on 7, responsible for 20 contracts with companies, author of two patents, and the founding member of two spinoff companies.

José A. García-Naya (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer engineering from the University of A Coruña (UDC), A Coruña, Spain, in 2005 and 2011, respectively.

He is with the CITIC Research Center and the Group of Electronics Technology and Communications, UDC, where he is an Associate Professor. He is the coauthor of more than 120 peer-reviewed papers in journals and conferences. He was a member of the research team in more than 40 research projects funded by public organizations and private companies, being principal investigator in two of them. His research interests focus on wireless engineering, with special emphasis on experimental evaluation, including wireless channel characterization, high-mobility vehicular transportation, time-modulated arrays, location systems, and IoT.