Portable Full Channel Sounder for Mobile Robotics by Using Sub-Nanosecond Time Synchronization over Wireless

Óscar Seijo, Iñaki Val
Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA). P.º J.M. Arizmendiarrieta, 2. 20500 Arrasate/Mondragón, Spain.
oseijo@ikerlan.es, ival@ikerlan.es

Jesús A. López-Fernández
Department of Electrical Engineering
Group of Signal Theory and Communications,
University of Oviedo, Gijón 33203, Spain.
jelofer@uniovi.es

Abstract— **Wireless communications have attracted great interest from the industry due to its lower cost and the possibility of enabling new use cases. The new use cases are commonly related to mobile robotics, such as Unmanned Aerial Vehicles. The design of wireless systems for these use cases requires deep knowledge of the channel behavior. However, the weight and size of full channel sounders exceed the payload of most mobile robots. In addition, full channel sounders usually require wired time synchronization. Hence, channel measurements in these scenarios are constrained to use limited channel sounders, which can only measure some specific parameters (frame error rate, channel attenuation, etc.). In this paper, we present the design and implementation of a portable 802.11-based channel sounder combined with a sub-nanosecond wireless time synchronization algorithm. Thanks to the wireless time synchronization, the channel sounder can be used to periodically take complex baseband Channel Impulse Response samples synchronized to absolute time. From these samples, relevant channel parameters can be extracted, including the Power Delay Profile, Doppler spectrum, and channel delay. The verification of the channel sounder through a wireless channel emulator confirms its feasibility for mobile robotics applications.**

Keywords— UAV, Channel Sounder, Wireless Time Synchronization, PTP, mobile robotics, drones, IEEE 802.11, industry 4.0., Wireless channel measurements, channel emulator.

I. INTRODUCTION

In the context of the industry 4.0, wireless communications are an emerging trend due to the reduced deployment and maintenance costs, and free movement of the nodes connected to the wireless network. Free movement is the key enabler of several applications in the context of mobile robotics, which includes for instance: fixed robots with moving elements, small terrestrial robots inside factories and Unmanned Aerial Vehicles (UAVs).

The wireless channel must be accurately characterized to choose the most appropriate wireless solution for a specific industrial scenario. A convenient way to improve the wireless channel knowledge in a specific scenario is to use a channel sounder and perform a channel measurement campaign. Depending on their design philosophy two kinds of channel sounders may be distinguished: limited channel sounders, designed to measure specific parameters, and full channel sounders, which can directly take channel samples.

On the one hand, limited channel sounders [1] are usually designed to measure parameters such as the received power, the Packet Error Rate (PER), or the system throughput. The information provided by limited channel sounders is not raw, and hence the channel knowledge that can be extracted from a measurement campaign is rather limited. They are commonly built with Commercial-Of-The-Shelf (COTS) devices, such as Universal Software Radio Peripheral (USRP) [2]. On the other hand, full channel sounders are designed to periodically measure the complex baseband Channel Impulse Response (CIR). From a collection of CIRs, relevant channel statistics, such as the Power Delay Profile (PDP) or the Doppler spectrum, can be obtained. Full channel sounders based on sequence transmission are suitable for time-variant channels [3]. However, full channel sounders are weighty and require a wired connection to synchronize the master and slave operations [3] [4] [5], which limits their applicability in mobile applications.

In this paper, we present the design and implementation of a portable full channel sounder using an OFDM-based 802.11 [6] modem combined with a high-performance wireless time synchronization algorithm. The use of a high-performance time synchronization algorithm stands as a reasonable trade-off between a compact, but limited channel sounder, and a high-performance, but complex and weighty full channel sounder. In addition, wireless time synchronization provides several appealing advantages for mobile applications, such as simpler operation and free movement. The channel sounder is based on the sub-nanosecond wireless time synchronization algorithm presented in [7]. The wireless data collection has been implemented over a Field Programmable Gate Array (FPGA)-based 802.11 modem and the synchronization has been implemented using a post-processing time synchronization algorithm. The channel sounder has been in-lab verified over a Non-Line-of-Sight channel model with mobile conditions using a wireless channel emulator.

The rest of the paper is organized as follows. First, in Section II, the channel sounder design based on PTP is presented. A post-processing algorithm to simplify the HW implementation is proposed in Section III. The verification of the channel sounder using a wireless channel emulator is presented in Section IV. Finally, Section V summarizes some conclusions of the work.

II. CHANNEL SOUNDER DESIGN

The channel sounder operation is based on a modification of the PTP frame exchange and the 802.11 PHY layer. Full channel sounders commonly use a unidirectional communication, where the master periodically transmits a sequence and the slave detects the sequence. Nonetheless, this channel sounder uses a full-frame exchange and thus a bidirectional communication is required.

From the channel sounding perspective, the most relevant element of the standard is the preamble. The 802.11 legacy preamble (Fig. 1) uses a 20 MHz bandwidth and comprises two fields, the Short Training Field (STF) and the Long

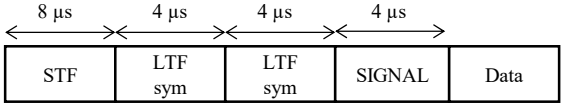

Fig. 1. 802.11 preamble format.

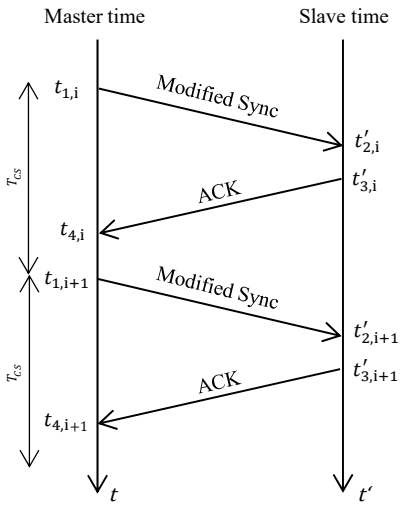

Fig. 2. Modified PTP frame exchange with a reduced number of frames.

Byte offset:

0	18	22	30	38	42	250	253
802.11 header	Frame seq ID (i)	$t_{1,\mathrm{i}}$	$t_{4,\mathrm{i-1}}$	$\tilde{g}_{ACK,\mathrm{i-1}}$	$\tilde{P}'_{ACK,\mathrm{i-1}}[\mathrm{l}]$	CRC	

Fig. 3. Modified Sync frame format.

Training Field (LTF). The STF is mainly used for Automatic Gain Control (AGC), and carrier frequency offset correction. The LTF is used to detect the start of a frame. The LTF is divided into two OFDM symbols with a length of 4 µs each. Each symbol comprises 64 samples plus a cyclic prefix of 16 samples. In this case, we are using the 64 samples of the second LTF symbol as the $s[\mathrm{k}]$ sequence. In the frequency domain, the 802.11 OFDM modulation has 64 subcarriers: 52 effective subcarriers, 1 dc subcarrier, and 11 guard subcarriers. Hence, its effective bandwidth is 16.25 MHz.

We have designed a modified version of the PTP frame exchange with only two frames to perform both the channel measurements and the time synchronization. This frame exchange is very convenient because it minimizes the channel sampling period (T_{cs}). The modified frame exchange is depicted in Fig. 2, being i the index of the frame exchange.

The master transmits every T_{cs} a modified sync frame to the slave and takes the Tx timestamp ($t_{1,i}$). The modified sync frames are received by the slave, which takes the Rx timestamp ($t'_{2,\mathrm{i}}$) and estimates the CIR from the modified Sync. Then, the slave answers to the modified Sync frame with a standard 802.11 ACK and takes the Tx timestamp ($t'_{3,\mathrm{i}}$). Afterward, the master receives the ACK, takes the Rx timestamp ($t_{4,i}$), and estimates the CIR from the ACK. The frame exchange is indefinitely executed until the end of the measurement.

The modified sync frame joints the functionalities of the PTP sync and PTP delay response frames, whereas the ACK transmitted from the slave to the master acts as the PTP delay request. The structure of the sync frame is depicted in Fig. 3. The frame comprises 254 bytes and has 5 data fields. The 802.11 header is used to be compliant with the 802.11 standard. The *Frame seq ID* field is used to identify the current frame, and it contains the channel sample index i. The Most Significant Bit (MSB) of the *Frame seq ID* field indicates if the ACK of the previous exchange was correctly received. The field $t_{1,\mathrm{i}}$ contains the transmission timestamp of the sync frame. $t_{4,\mathrm{i-1}}$, $\tilde{g}_{ACK,\mathrm{i-1}}$, and $\tilde{P}'_{ACK,\mathrm{i-1}}[\mathrm{l}]$ are related to the previous ACK received by the master. $t_{4,\mathrm{i-1}}$ contains the ACK timestamp, $\tilde{g}_{ACK,\mathrm{i-1}}$ is the estimation of the channel gain in dB, which is computed as the estimated Rx power minus the Tx power in dBm, and $\tilde{P}'_{ACK,\mathrm{i-1}}[\mathrm{l}]$ is the estimated CIR of the ACK in the frequency domain normalized to unit energy. The ACK fields are set to 0 if the MSB bit of the *Frame seq ID* field is set to 0. $\tilde{P}'_{ACK,\mathrm{i-1}}[\mathrm{l}]$ is represented by its IQ components in complex format. It uses 4 bytes for each subcarrier and has a length $L = 52$. Finally, the standard 4-byte Cyclic Redundancy Check (CRC) is used to check if there is an error in the frame.

At the end of each frame exchange, the slave computes the estimated CIRs in the time domain as

$$\tilde{p}_{ACK,\mathrm{i}}[\mathrm{n}] = \mathrm{IFFT}\left\{10^{\frac{\tilde{g}_{ACK,\mathrm{i}}}{20}}\tilde{P}'_{ACK,\mathrm{i}}[\mathrm{l}]\right\}, \quad (1)$$

$$\tilde{p}_{Sync,\mathrm{i}}[\mathrm{n}] = \mathrm{IFFT}\left\{10^{\frac{\tilde{g}_{Sync,\mathrm{i}}}{20}}\tilde{P}'_{Sync,\mathrm{i}}[\mathrm{l}]\right\}, \quad (2)$$

being IFFT{·} the inverse Fourier transform, $\tilde{P}'_{Sync,\mathrm{i}}[\mathrm{l}]$ the estimated CIR from the Sync frame normalized to unit energy, and $\tilde{g}_{Sync,\mathrm{i}}$ the estimation of the channel gain, and $\tilde{p}_{Sync,\mathrm{i}}[\mathrm{n}]$ and $\tilde{p}_{ACK,\mathrm{i}}[\mathrm{n}]$ the CIR in the time domain. Then, the slave stores:

- The channel sample index, *Frame seq ID*: i.
- The four timestamps: $t_{1,\mathrm{i}}, t'_{2,\mathrm{i}}, t'_{3,\mathrm{i}}, t_{4,\mathrm{i}}$.
- The estimated CIR from the Sync: $\tilde{p}_{Sync,\mathrm{i}}[\mathrm{n}]$.
- The estimated CIR from the ACK: $\tilde{p}_{ACK,\mathrm{i}}[\mathrm{n}]$.
- A Boolean to indicate if the frame exchange was correctly performed: $exchange_{ok,\mathrm{i}}$.
- A Boolean to indicate if the sync frame was correctly received: $SyncFrame_{ok,\mathrm{i}}$.

Additionally, the slave also stores the number of frame exchanges (I), the carrier frequency (f_c), and T_{cs}.

III. Offline Time Synchronization Algorithm

The offline time synchronization algorithm is used to align $\tilde{p}_{Sync}$ to an absolute time reference and to compensate the

Algorithm 1. Offline time synchronization algorithm.

Input: $\tilde{p}_{ACK}, \tilde{p}_{Sync}, t_1, t'_2, t'_3, t_4, exchange_{ok}, k_p, k_{Int}, I$
Output: $\tilde{t}_{err,i}, t'_{c,2}$
Initialization:
$t_{\mathrm{Int}} = 0$.
$\tilde{t}_{acum\ error} = 0$.

1. for i = 0 to I-1 do:
2. Correct slave clock timestamps:
2.1. $t'_{c,2,\mathrm{i}} = t'_{2,\mathrm{i}} - \tilde{t}_{acum\ err}$.
2.2. $t'_{c,3,\mathrm{i}} = t'_{3,\mathrm{i}} - \tilde{t}_{acum\ err}$.
3. if $exchange_{ok,\mathrm{i}}$ = false do:
3.1. $\tilde{t}_{err,i} = k_{Int} t_{\mathrm{Int}}$.
3.2. $\tilde{t}_{acum\ err} = \tilde{t}_{acum\ err} + \tilde{t}_{err,i}$.
3.3. Go to the next iteration.
4. Take the enhanced timestamps:
4.1 $t'_{e,2,\mathrm{i}} = t'_{c,2,\mathrm{i}} + \mathrm{DS}\{\tilde{p}_{Sync,\mathrm{i}}[\mathrm{n}]\}$.
4.2 $t_{e,4,\mathrm{i}} = t_{4,\mathrm{i}} + \mathrm{DS}\{\tilde{p}_{ACK,\mathrm{i}}[\mathrm{n}]\}$.
5. Compute Time Sync error:
5.1. $\tilde{t}_{ms,i} = \frac{t'_{e,2,\mathrm{i}} - t_{1,\mathrm{i}} + t_{e,4,\mathrm{i}} - t'_{c,3,\mathrm{i}}}{2}$.
5.2. $\tilde{t}_{o,i} = t'_{e,2,\mathrm{i}} - t_{1,\mathrm{i}} - \tilde{t}_{ms,\mathrm{i}}$.
5.3. $t_{\mathrm{Int}} = t_{\mathrm{Int}} + \tilde{t}_{o,\mathrm{i}}$.
5.4. $\tilde{t}_{err,i} = k_p \tilde{t}_{o,\mathrm{i}} + k_{Int} t_{\mathrm{Int}}$.
5.5. $\tilde{t}_{acum\ err} = \tilde{t}_{acum\ err} + \tilde{t}_{err,i}$.
6. end for
7. return

local oscillator drift. The algorithm is based on the time synchronization algorithm that we proposed in [7]. The core of the algorithm is the novel timestamping method, which provides subnanosecond precision, and hence, subnanosecond time synchronization performance, in a vast variety of wireless conditions. The enhanced timestamps are based on a conventional threshold-based timestamping method combined with the delay spread operator.

The post-processing may be divided into three main tasks: compute the enhanced timestamps, apply the PTP correction algorithm, and correct $\tilde{p}_{Sync}$. The first two tasks are summarized in Algorithm 1 and further explained below.

The algorithm output is the time synchronization error ($\tilde{t}_{err,i}$) and the conventional timestamps ($t'_{c2,i}$). The algorithm uses the enhanced timestamps combined with a Proportional Integral (PI) filter with constants (k_p, k_{int}) to minimize the time synchronization error. $t_1, t'_2, t'_3, t_4, exchange_{ok}$ are vectors with length I, and $\tilde{p}_{ACK}$ and $\tilde{p}_{Sync}$ are matrices with I rows and $N = 64$ columns.

In step 2., the slave timestamps $t'_{2,\mathrm{i}}$ and $t'_{3,\mathrm{i}}$ are corrected based on the accumulated time synchronization error ($\tilde{t}_{accum\ err}$). The result is the corrected conventional timestamps.

In step 3., the algorithm checks if the frame exchange was successfully done. In case that $exchange_{ok} = false$, it will mean that one of the frames in the i frame exchange failed and the time synchronization error ($\tilde{t}_{o,\mathrm{i}}$) is assumed to be 0.

In step 4., The enhanced timestamps ($t'_{e,2,\mathrm{i}}, t_{e,4,\mathrm{i}}$) are computed. To do so, the conventional timestamps $t'_{c,2,\mathrm{i}}$ and $t_{4,\mathrm{i}}$ are corrected by the mean delay spread (DS{·}) of $\tilde{p}_{Sync,\mathrm{i}}[\mathrm{n}]$ and $\tilde{p}_{ACK,\mathrm{i}}[\mathrm{n}]$ respectively.

In step 5. the PTP synchronization algorithm is used to compute the time synchronization error. First, $\tilde{t}_{ms,\mathrm{i}}$ and $\tilde{t}_{o,\mathrm{i}}$ are obtained. Then, $\tilde{t}_{o,i}$ is introduced to a PI filter with constants k_p, k_{Int} and the error of this frame exchange ($\tilde{t}_{err}$) is computed. $\tilde{t}_{err}$ is accumulated in $\tilde{t}_{acum\ err}$.

One the time synchronization error is computed, the estimated CIRs obtained from the Sync frames can be realigned. To align $\tilde{p}_{Sync,\mathrm{i}}[\mathrm{n}]$, the CIRs must be shifted by a delay d_{i} using a fractional delay filter. d_{i} can be obtained as the time difference between the detection of the sync frames ($t'_{c,2,\mathrm{i}}$) and its transmission ($t_{1,\mathrm{i}}$)

$$d_{\mathrm{i}} = (t'_{c,2,\mathrm{i}} - t_{1,\mathrm{i}}). \tag{3}$$

On the other hand, the carrier frequency offset in each frame exchange ($\tilde{f}_{o,i}$) can be estimated from $\tilde{t}_{err,i}$

$$\tilde{f}_{o,i} = \frac{\tilde{t}_{err,i}}{T_{CS}} f_c. \tag{4}$$

Then, the relative carrier frequency phase between the master and slave ($\tilde{\phi}_i$) is

$$\tilde{\phi}_i = \tilde{f}_{o,i} \cdot T_{CS} = \tilde{t}_{err,i} \cdot f_c. \tag{5}$$

Given $\tilde{\phi}_i$ and d_{i}, $\tilde{p}_{Sync\ aligned,\mathrm{i}}[\mathrm{k}]$ is computed using a fractional delay filter as

$$\tilde{p}_{Sync\ aligned,\mathrm{i}}[\mathrm{k}] = e^{-j\tilde{\phi}_i} \sum_{n=-\infty}^{+\infty} \tilde{p}_{Sync,\mathrm{i}}[\mathrm{n}]\ \mathrm{sinc}\left(\mathrm{k} - \mathrm{n} - \frac{1}{T_s} d_{\mathrm{i}}\right). \tag{6}$$

being

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{x}. \tag{7}$$

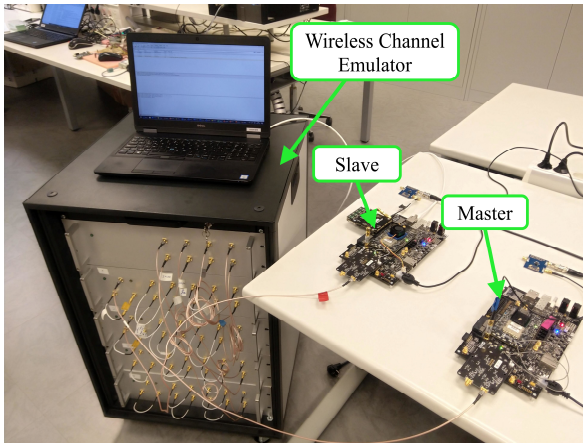

Fig. 4. Channel sounder verification testbed.

TABLE I. EMULATED WIRELESS CHANNEL.

	Tap 1	Tap 2	Tap 3
Gain [dB]	-44	-54	-47
Delay [ns]	910	910 + 195.3	910 + 390.6
Fading	Rayleigh		
Doppler spectrum	Jakes with a maximum Doppler shift of 62 Hz.		

Hence, the channel samples of this channel sounder are equivalent to the operation of a full channel sounder, where the master periodically transmits an IQ sequence and triggers the slave at the start of the IQ sequence to take the channel samples.

IV. CHANNEL SOUNDER VERIFICATION THROUGH THE USE OF A CHANNEL EMULATOR

The channel sounder has been implemented over an 802.11 modem built in a Picozed SDR. The Picozed SDR comprises a Zynq System on Chip (SoC) and an AD9361 radio chip. The Zynq includes a Field Programmable Gate Array (FPGA) and two ARM cortex A9 microcontrollers. The AD9361 chip offers a BW of 56 MHz and supports a carrier frequency from 100 MHz to 6 GHz [8]. The offline time synchronization algorithm has been implemented in Matlab.

The channel sounder has been in-lab verified using a wireless channel emulator. The channel emulator operates in the 2.4 GHz band and has a bandwidth of 100 MHz. It has four bidirectional ports with a delay from port to port of 910 ns, and a fixed attenuation of 44 dB. The emulator can hold a wireless channel model up to 10 taps that can be programmed with different gains and fading profiles. It supports four fading distributions: Rayleigh, Rice, Nakagami and constant; and three Doppler spectrum shapes: Jakes, Bell, and Gaussian.

The verification testbed is shown in Fig. 4. The master RF port is connected to the first port of the channel emulator, and the slave RF port is connected to the second port. The Ethernet port of the slave board is connected to a laptop, which is used to record the data from the frame exchanges. f_c has been set to 2.48 GHz, T_{CS} to 500 µs, and the Tx power to 10 dBm.

The channel emulator is programmed with the wireless channel model described in Table I. The wireless channel model emulates an NLoS scenario with mobile nodes. It comprises three taps with different gains and evenly spaced 195.3 ns. The fading has been set to Rayleigh (NLoS), and the Doppler spectrum has been set to Jakes with a Doppler frequency of 62 Hz.

We took a record of 20 seconds, equivalent to $I = 40.000$ channel samples. The record length is enough to measure the tap position, gain, and fading properties. During the record,

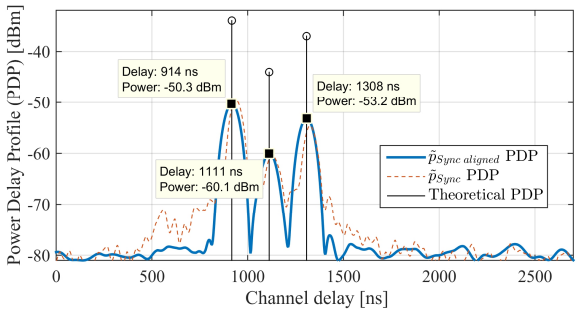

Fig. 5. PDPs obtained from $\tilde{p}_{Sync\ aligned}$ and from $\tilde{p}_{Sync}$.

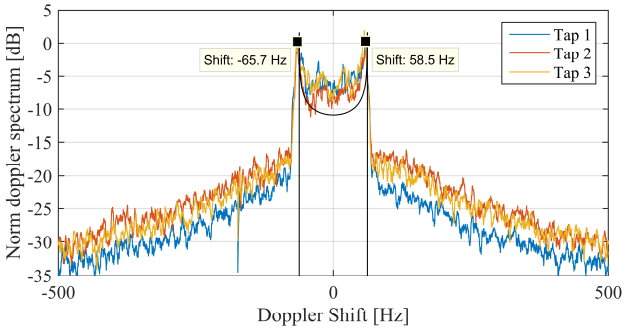

Fig. 6. Measured Doppler spectrum of Taps 1, 2, and 3.

the wireless channel randomly varied according to the programmed configuration.

Fig. 5. depicts the PDPs with synchronization and without synchronization. There are significant differences between the PDPs. Regarding the $\tilde{p}_{Sync}$ PDP, the position of the taps randomly varies around their real position in $\tilde{p}_{Sync}$. Due to this, the PDP have a significant tail around the taps and a considerable error in the relative gain between taps (2 dB). On the contrary, the $\tilde{p}_{Sync\ aligned}$ PDP is smooth without tails, the relative gain between taps have an error below 0.5 dB, and their relative delay is very accurate. In addition, the difference between the peak and the noise floor is around 30 dB, which gives a reasonable PDP dynamic range.

In addition, to verify that the channel sounder was able to measure the channel variations, we have obtained the Doppler spectrum and the fading distributions of each of the taps. On the one hand, the Doppler spectrum of each of the taps is represented in Fig. 6. As can be clearly seen, the Doppler spectrum of the three taps clearly follows a Jakes Doppler distribution with an absolute maximum Doppler shift of 65.7 Hz, which is very close to 62 Hz. On the other hand, we have represented the fading distribution of each tap using an histogram (Fig. 7). The theoretical Rayleigh distribution are also represented. It is noticeable from the plot that the power of each tap approximately fits a Rayleigh distribution at their respective power. These results confirm the feasibility of the channel sounder to measure raw channel samples in mobile conditions.

V. Conclusions

In this paper, we exploit a sub-nanosecond wireless time synchronization scheme to create a compact portable full channel sounder that does not require any wired time synchronization. We have carried out the channel sounder implementation over a HW-based SDR platform, where we have implemented an 802.11 modem in FPGA and a post-

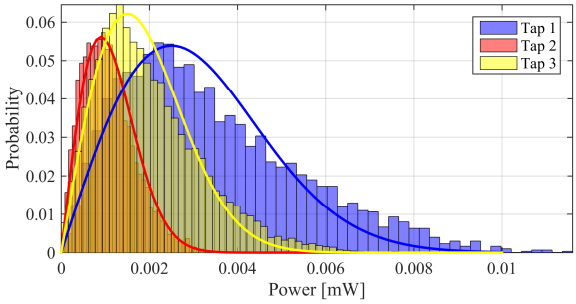

Fig. 7. Measured fading distributions of Taps 1, 2, and 3.

processing algorithm in Matlab to perform the channel sounding.

We have verified the channel sounder using a wireless channel emulator under NLoS and time-variant conditions. The experiment showed that the channel sounder was able to measure the wireless channel delay, the number of taps of the channel, and their gain and relative delay. In addition, we computed the Doppler spectrum and fading distributions of each of the taps. The experiment confirms the feasibility of high-performance wireless time synchronization algorithm in channel sounding.

Acknowledgment

This work has been partly supported by u4INDUSTRY (ELKARTEK) project of the Basque Government (Spain).

References

[1] Y. Mostofi, M. Malmirchegini, and A. Ghaffarkhah, "Estimation of communication signal strength in robotic networks," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1946–1951, 2010.

[2] D. Maas, M. H. Firooz, J. Zhang, N. Patwari, and S. K. Kasera, "Channel sounding for the masses: Low complexity GNU 802.11b channel impulse response estimation," *IEEE Trans. Wirel. Commun.*, vol. 11, no. 1, pp. 1–8, 2012.

[3] B. T. Maharaj, J. W. Wallace, M. A. Jensen, and L. P. Linde, "A low-cost open-hardware wideband multiple-input-multiple-output (MIMO) wireless channel sounder," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 10, pp. 2283–2289, 2008.

[4] J. Vychodil, A. Chandra, T. Mikulasek, A. Prokes, and V. Derbek, "UWB time domain channel sounder," *Proc. 25th Int. Conf. Radioelektronika, RADIOELEKTRONIKA 2015*, pp. 268–271, 2015.

[5] G. R. MacCartney and T. S. Rappaport, "A flexible millimeter-wave channel sounder with absolute timing," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1402–1418, 2017.

[6] *IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE, 2012.

[7] Ó. Seijo, J. A. López-fernández, H.-P. Bernhard, and I. Val, "Enhanced Timestamping Method for Sub-Nanosecond Time Synchronization in IEEE 802.11 over WLAN Standard Conditions," *IEEE Trans. Ind. Informatics*, 2020.

[8] "AD9361 Reference Manual UG-570." [Online]. Available: http://www.farnell.com/datasheets/2007082.pdf.