Low-cost test measurement setup for real IoT BLE sensor device characterization

David Perez-Diaz-de-Cerio, Ángela Hernández-Solana, Antonio Valdovinos, Joan Olmos, Jose Luis Valenzuela

PII: S0263-2241(18)31138-2
DOI: https://doi.org/10.1016/j.measurement.2018.11.082
Reference: MEASUR 6125

To appear in: *Measurement*

Received Date: 10 July 2018
Revised Date: 23 November 2018
Accepted Date: 26 November 2018

Please cite this article as: D. Perez-Diaz-de-Cerio, A. Hernández-Solana, A. Valdovinos, J. Olmos, J.L. Valenzuela, Low-cost test measurement setup for real IoT BLE sensor device characterization, *Measurement* (2018), doi: https://doi.org/10.1016/j.measurement.2018.11.082

Low-cost test measurement setup for real IoT BLE sensor device characterization

David Perez-Diaz-de-Cerio [1,*], Ángela Hernández-Solana [2], Antonio Valdovinos [2], Joan Olmos[1] and Jose Luis Valenzuela [1]

[1] Signal Theory and Communications Department, Universitat Politècnica de Catalunya, Esteve Terradas 7, 08860 Castelldefels, Spain; olmos@tsc.upc.edu (J.O.) valens@tsc.upc.edu (J.L.V.)

[2] Aragon Institute for Engineering Research (I3A), University of Zaragoza, 50018 Zaragoza, Spain; anhersol@unizar.es (A.H.) toni@unizar.es (A.V.)

[*] Correspondence: dperez@tsc.upc.edu; Tel.: +34-93-413-7209

Abstract: The methodology presented in this paper aims to characterize impairments shown by real devices which are usually neglected on standardized tests but that become very important in massive IoT scenarios. For instance, we have measured that real BLE scanners are not able to scan continuously even though they are configured to do so. Besides, we have also found and demonstrated that some manufacturers seem not to apply any backoff mechanism although it is mandatory. These two unexpected behaviors have a significant impact on the performance of massive wireless sensor networks based on BLE. So, it becomes necessary to characterize these and other impairments.
The proposed tests are based on device current consumption measurements and their association with the information obtained from upper layers.
We describe a new low-cost generic measurement setup and provide all the necessary data (configuration parameters, scripts, etc.) for applying the proposed methodology. As an example, we use it to profile the behavior of Bluetooth Low Energy devices. Furthermore, the proposed setup can also inspire researchers to characterize other wireless technology devices, like Wi-Fi, Zigbee, LoRa, etc.

Keywords: IoT; sensor networks; current sensor; BLE; low-cost

I. Introduction

Internet of Things (IoT) has enabled the possibility of implantation of wireless industrial systems for sensors, data and control applications. The Industrial Internet of Things (IIoT) will be part of the next revolution of industry, which is usually referred to as Industry 4.0, and will rely on the capacity of the manufacturing processes to implement real-time functionalities based on the management and transmission of all relevant information in real-time. Bluetooth, LoRa, Zigbee, IEEE 802.15.4, NB-IoT or IEEE 802.11ah have been proposed as a viable connectivity solution for the IoT.

The device design and implementation for high-density wireless networks is a challenge that should take into account several factors such as power consumption, latency, throughput, interference, reliability, etc. It is important to accurately analyze and characterize the device real-time behavior in a stressed and dense environment and apply the results to obtain precise mathematical and simulation models. Methodologies usually include analytical modelling and simulations [1] considering ideal device behavior based on the specification descriptions [2]. However, device hardware impairments have a huge impact on the system performance when data is generated by hundreds or even thousands of devices. These real-world technological limitations or practical implementations in the physical, MAC (Medium Access Control) or upper control layers should be methodically measured with different test setups.

This paper has been motivated by the discovery of significant differences between the results of ideal models and real measurements when analyzing the performance of massive IoT deployments. These non-idealities are caused by limitations of real devices and depend on the firmware and hardware implementation of the chipset manufacturers. We have demonstrated [3] that these non-idealities can be neglected when just a few devices are present, but become very noticeable in stressed scenarios.

The Bluetooth Test Specification includes the Test Suite Structure and Test Purpose to test the Bluetooth RF layer, including Basic Rate, Enhanced Data Rate and Low Energy. The objective of this Test Specification is to provide a basis for conformance tests for Bluetooth devices giving a high probability of air interface inter-operability between Bluetooth devices of different manufacturers [4]. These tests have been designed to comply with radio emission standards, verify the system reliability and ensure interoperability between devices from different manufacturers. For example:

- RF-PHY/TRM-LE/CA/BV-01-C: Output power
- RF-PHY/RCV-LE/CA/BV-01-C: Receiver sensitivity, uncoded data at 1 Mbit/s
- RF-PHY/RCV-LE/CA/BV-03-C: C/I and Receiver Selectivity Performance, uncoded data at 1 Mbit/s
- RF-PHY/RCV-LE/CA/BV-07-C: PER Report Integrity, uncoded data at 1 Mbit/s

However, these tests defined in the standard do not examine all the Bluetooth functionalities and do not analyze the real behavior of Bluetooth devices when there are multiple transmitters and receivers, like the delay required to switch between scanning frequencies and many other non-idealities. In addition to the reception gaps related to the change of frequency, real operation is usually affected by other periodic pauses scattered along the scanning intervals, blind gaps after packet processing, etc. Also, the standard tests do not analyze whether the devices meet specifications at MAC level or higher layer protocols such as HCI (Host Controller Interface).

On the other hand, several device manufacturers such as Texas Instruments [5,6], NXP Semiconductors [7] or Silicon Labs [8] provide application notes describing the setup and procedures to measure power consumption on some of their devices. In a similar way, instrumentation device manufacturers, such as Keysight or Rohde & Schwarz, offer several equipment [9] and applications [10] to test and measure power consumption of Bluetooth devices.

The research community shows also the interest on this subject by using different measuring methods. For example, in [11] the authors compare the datasheet specifications of real devices with measurements for the most important phases of the communications (start-up, advertising, connection) using a the N6705B Agilent power analyzer. In [12], the sleep current consumption of BLE (Bluetooth Low Energy), ANT and Zigbee devices is measured using a multimeter (Fluke 287) and the active state current using the Texas Instruments INA226 current measurement chip. Authors in [13] use a power monitor to derive models of the basic energy consumption behavior of BLE. Other studies [14–17] employ sniffers and network protocol analyzers like Wireshark [18] to analyze the behavior and detect intrusions or other security issues.

The aim and novelty of this paper is to propose a new series of functional and parametric test setups addressed to enhance and extend the current tests. The main goal is to measure with accuracy the real behavior of IoT devices, whether they transmit, receive or remain idle. The Bluetooth standard defines the state machine diagram of BLE devices when they are in scanning mode and receiving advertisement broadcast frames. The ideal state diagrams for the passive and active scanning modes we will analyze are represented in Figure 1 and Figure 2, respectively. However, after analyzing a wide number of BLE chipset manufacturers, we have verified that the implemented state diagram significantly differs from the ideal operation. As an example, the real behavior of the most common of those chipsets during the scanning state is depicted in Figure 3 and Figure 4. It is important to notice that the measured state diagram reveals additional unideal states that modify the behavior of the devices.

The measured state diagrams reveal idle gaps associated with processes such as frequency change, erroneous demodulation, frame processing, etc. The tests defined in this paper, combined with upper layer interaction, allow to determine these unideal reception states. For example, when there are multiple devices transmitting simultaneously and there are frame collisions, errors, capture effect, etc. The proposed design is composed by a synchronized measurement system of current consumption, a HCI command controller and a protocol analyzer.

In previous works, see [3,19], we have also followed these procedures to characterize and analyze BLE devices.

Although we use BLE as an example for the target of the measurements, the methodology employed can be used with other technologies in a similar way as we did for analyzing Wi-Fi devices in [20].

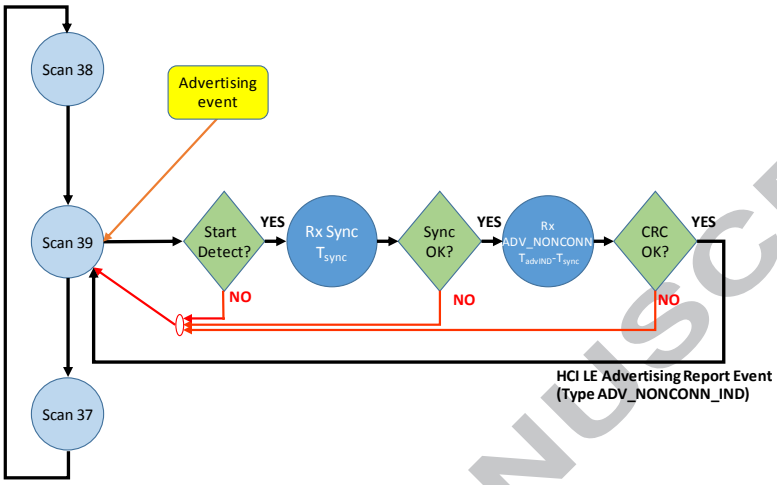

Figure 1. Ideal non-connectable advertisement reception state machine diagram

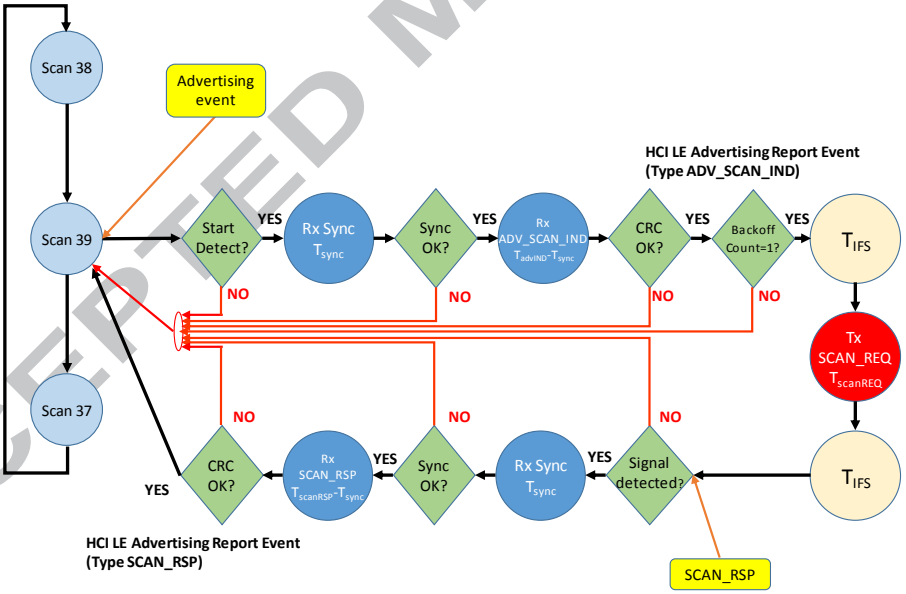

Figure 2. Ideal scannable advertisement reception state machine diagram

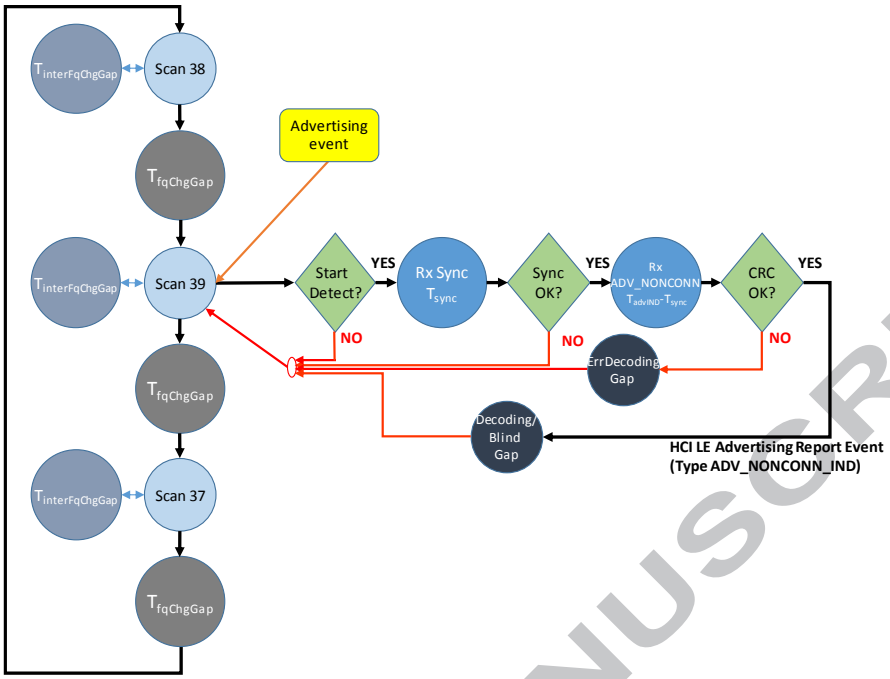

Figure 3. Real non-connectable advertisement reception state machine diagram

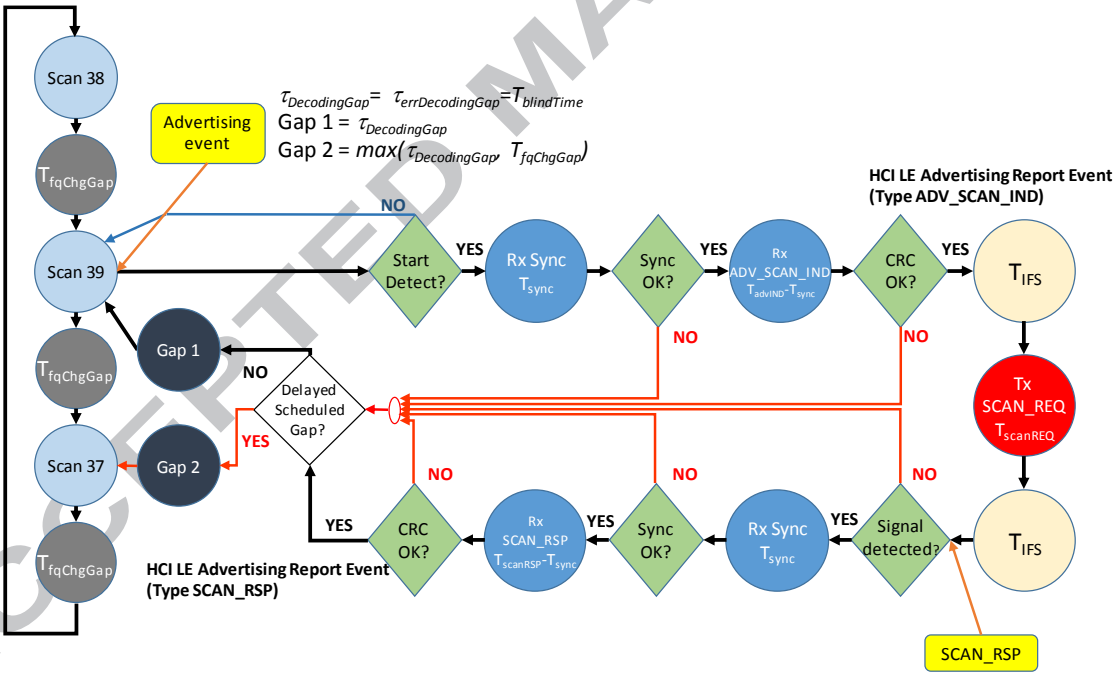

Figure 4. Real scannable advertisement reception state machine diagram

The paper is structured as follows. In first place, a generic measurement setup with the description of the needed components for all the tests is presented. Then, for each specific test, we indicate which sections (subsets) of the generic setup are necessary. As we have stated before, we focus our measurements on a BLE sensor network. So, we include a brief summary of the basic BLE concepts needed for a better understanding of the devices under test. The specific equipment variations employed for the characterization of BLE devices are also introduced in this section. Following, we describe in detail each one of the tests: objective, setup, configuration scripts, etc. and we present the obtained results. Finally, we include a conclusion section to summarize the most important topics discussed.

II. Measurement setup

The aim of the proposed measurement setup is to determine the behavior of wireless devices by observing, in a combined way, the actual state at RF level along with the behavior of upper layers (e.g., considering MAC retransmissions, or whether a frame has been correctly received by upper protocol layers). This is achieved by combining measurements of instantaneous power consumption of the transmitting and receiving wireless devices under test (DUTs) with network packet analysis obtained at the receiver side. It should be stressed that the proposed setup and methodology is not limited to a specific type of devices, but can be applied to a wide range of wireless standards, such as 802.11, Bluetooth, BLE, 802.15.4, etc.

A. GENERIC MEASUREMENT SETUP

A schematic representation of the setup topology is shown in Figure 5. The key components of this setup are:

i) The wireless device (DUT) configured for the intended test.
ii) A signal generator or a set of wireless devices configured as signal generators.
iii) An RF shield box for the DUT placement, aiming to isolate it from external sources of interference.
iv) RF connectors, antennae for the DUT, a circulator and an attenuator to model the different test cases.
v) A set of current sensors, providing a voltage output that is proportional to the current consumption of the transmitting and receiving DUT.
vi) A digital oscilloscope, to measure the voltage signal and register the obtained samples.
vii) A power supply, to power the DUT and the current sensors.
viii) A laptop with a packet network analyzer tool, to capture and analyze the packets of the receiving DUT.

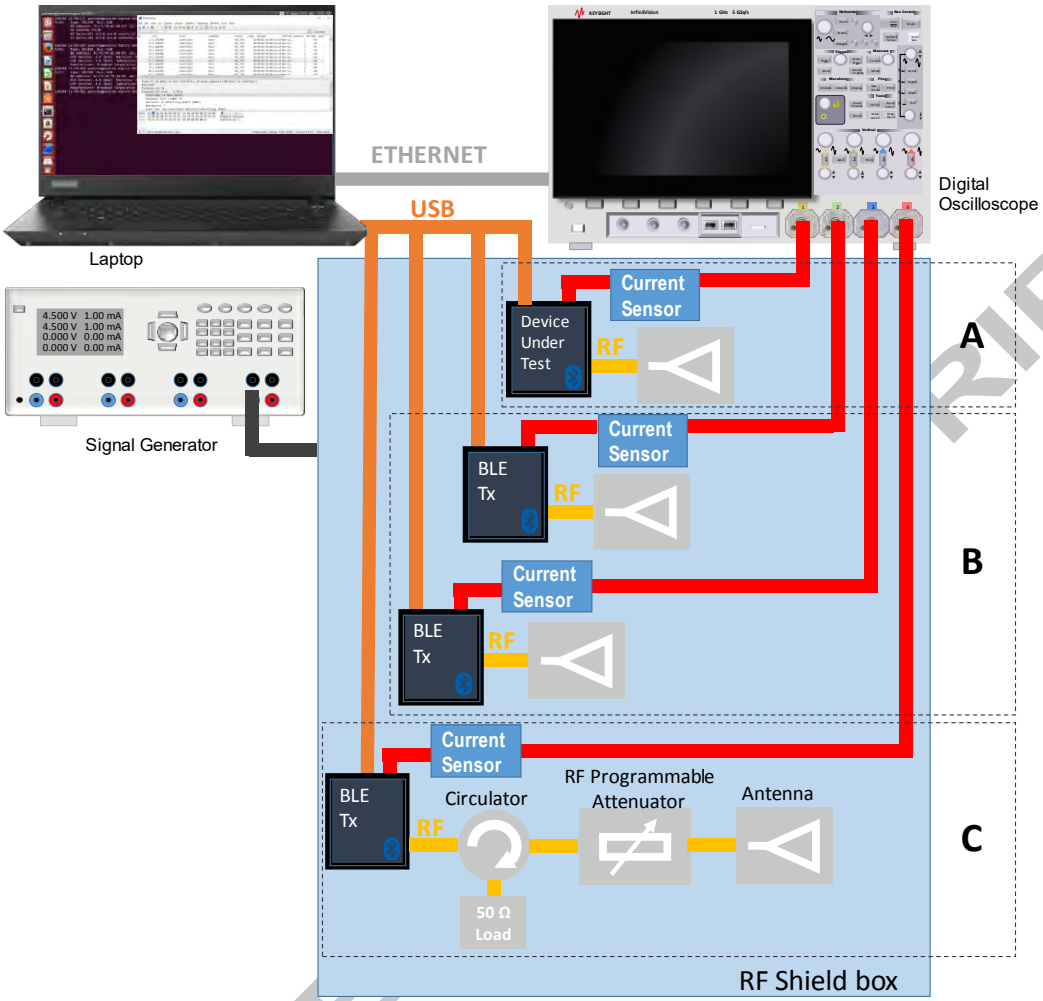

Figure 5. Generic setup topology scheme

B. BLUETOOTH LOW ENERGY FUNDAMENTALS

Since version 4.0 the Bluetooth standard includes a low energy mode. This mode is ideal for IoT applications. One of the main differences between this mode and the classic Bluetooth is the use of advertising events previous to the establishment of a connection or just to transmit short amounts of data.

Within an advertising event the transmitter, known as advertiser, is able to transmit a frame over up to three different frequencies (channels 37, 38 and 39). The amount of data of these frames is limited, reaching a maximum of 31 bytes. This limit has been extended to 256 bytes in version 5 of the standard. After a fixed period (advertisement interval) and a random delay ranging from 0 ms to 10 ms (to avoid collisions) a new advertising event is generated.

The receiver is known as scanner. The scanner listens for a period of time named *Scan_Window* on one of the three advertising channels waiting for an advertisement. After a *Scan_Interval* it switches to the next advertising channel and listens again during a *Scan_Window* and so on.

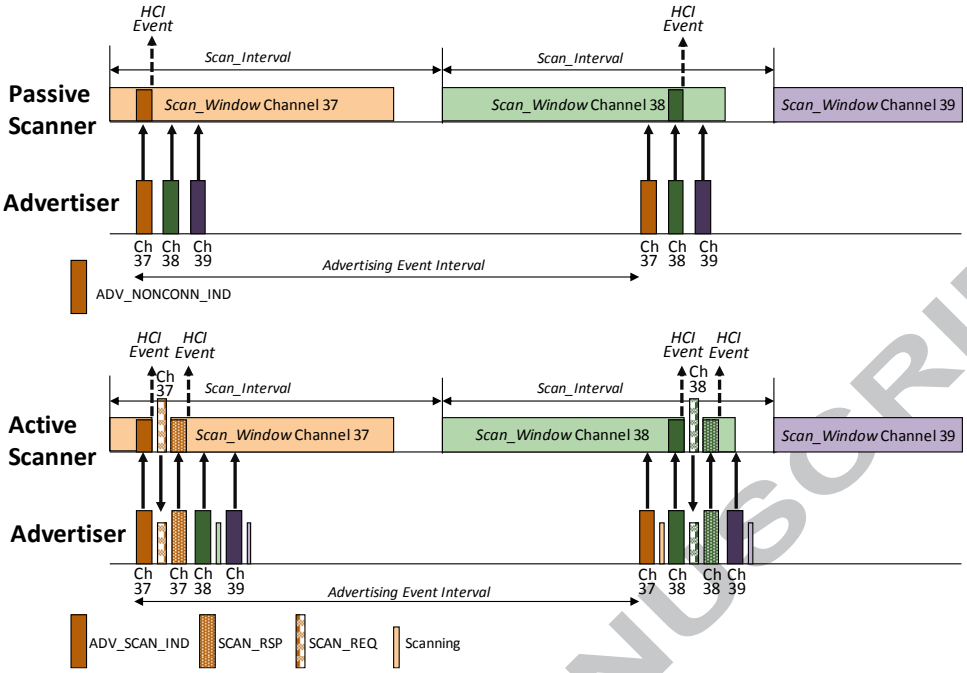

Figure 6. Passive and active scanning modes

Additionally, the scanner can perform an active or passive scanning (see Figure 6). When using the active scanning the scanner sends automatically a new frame (*scan request*) after 150 μs of receiving a valid advertisement frame. Once this request is received by the advertiser it answers with a *scan response* frame to provide additional data. In passive scanning the scanner never requests additional data. However, to be able to send a *scan request,* in addition to the scanner being in active mode the advertisement received should belong to a specific type. The standard specifies different types of advertising events. In this paper we will focus on just two of them: non-connectable advertising events (ADV_NONCONN_IND) and scannable advertising events (ADV_SCAN_IND). The second one allows the exchange of *scan request and scan response* frames (named SCAN_RSP and SCAN_REQ) while the first one does not.

Another aspect to take into account when using active scanning is that two scanners may receive at the same time an advertisement and so both of them will transmit simultaneously a scan request, thus producing a collision. The standard requires that a backoff algorithm must be implemented, but the actual implementation depends on the manufacturers because the standard just proposes an example algorithm and leaves it open.

The algorithm proposed in the standard defines two parameters: *backoffCount* and *upperLimit.* Initially, these two values are set to one. Then, on every received advertisement which is liable to be followed by a SCAN_REQ, the *backoffCount* is decremented by one until it reaches zero. Only when the backoffCount is zero the scanner transmits the SCAN_REQ. If this request is not answered, or the SCAN_RSP is not received from the advertiser, it is considered a failure. On every two consecutive failures, the *upperLimit* is doubled (until it reaches a maximum of 256) and on every two consecutive successes the *upperLimit* is halved (until it reaches one). After any success or failure, the scanner sets the *backoffCount* to a new pseudo-random integer between one and *upperLimit* (inclusive).

C. MEASUREMENT SETUP FOR BLE DEVICE CHARACTERIZATION

Next, we will describe the specific setup implemented in our laboratory for Bluetooth devices, serving as a proof of concept of the proposed methodology and used to acquire the results presented in this paper.

In this particular setup we have considered BLE USB dongle devices, from different manufacturers, controlled by a laptop through the BlueZ protocol stack for Linux.

The current consumption of BLE devices is completely different depending on the state of the device: idle, scanning, or transmitting. So we use these differences to infer the behavior of the devices by analyzing their consumption.

All devices have been placed inside a Rohde and Schwarz RF shield-box to limit external interference. A set of Texas Instruments current sensing boards [21] have been employed to measure the current consumption of the transmitting and receiving BLE devices. The sensing boards require a 5 V supply and can accurately detect a load current between 0 and 1 A. Their design is based on an OPA320 amplifier with a rail-to-rail input/output and a relative low offset voltage. The voltage output of this design is proportional to the measured current with a sensitivity of 5V/1A. The OPA320 model has a unity-gain bandwidth of 20 MHz. A power supply, operating in the range of 0-30V with a maximum current of 2.5 A, has been employed to power the wireless devices and the current sensors.

The measured voltage signal from the current sensors has been fed to a Keysight Mixed Signal Oscilloscope model MSOX4104A4. The oscilloscope has a 1GHz bandwidth and a resolution of 5 GSample/s, enabling the sampling of the measured signal with high accuracy. Furthermore, the oscilloscope offers several triggering options (by pulse width, voltage level, signal form, etc.), thus enabling the synchronization between the voltage sampling and the frame capturing process.

Finally, a laptop operating under Linux and equipped with the Wireshark network protocol analyzer and the BlueZ kernel modules, libraries and utilities, has been employed. The laptop is connected to the DUT for the purposes of: 1) Controlling the BLE USB dongle configuration and 2) Monitoring the received frames with Wireshark. For this connection, a modified USB cable has been prepared to enable the transmission and reception of data between the DUT and the laptop without interfering the measurements of the current sensors (see Figure 7).

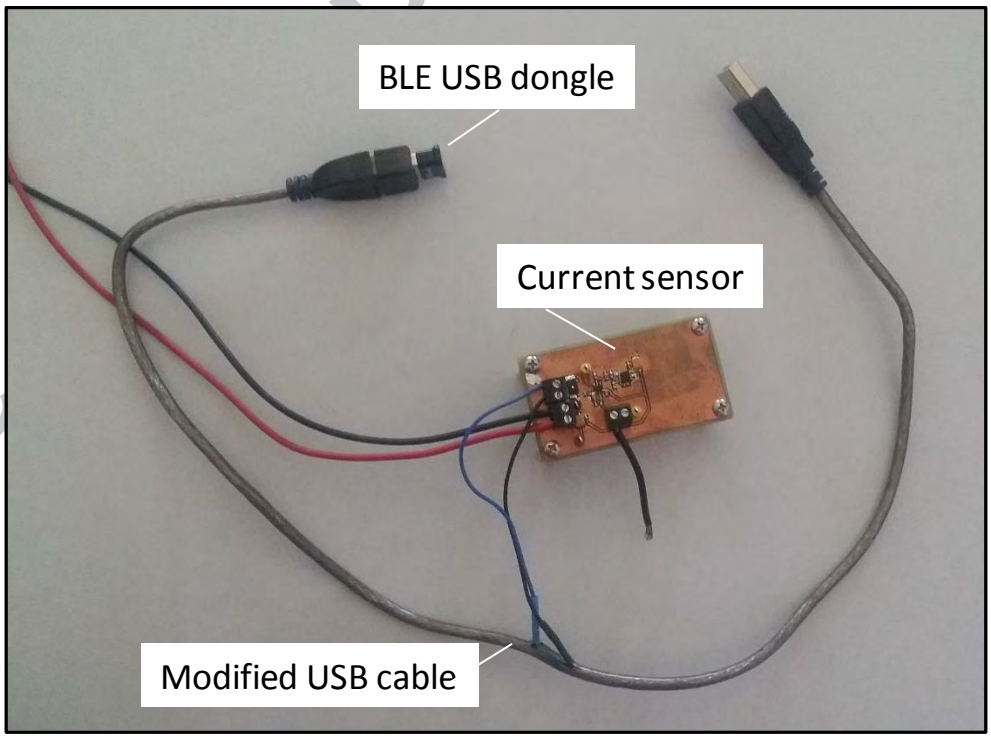

Figure 7. Detail of the modified USB cable

III. Tests and results

By adequately adjusting the generic setup we are able to perform the following battery of tests:

A. Characterization of the receiver in scanning mode
B. Characterization of the receiver during frame reception
C. Characterization of the behavior in presence of frame collisions
D. Backoff implementation test

These tests allow us to derive the real behavior diagrams included in Figure 3 and Figure 4 for scanning devices of any manufacturer. For each test we provide a summary with the test description, settings, list of necessary equipment, the setup scheme, and the configuration scripts. Finally, we analyze the obtained results.

A. Characterization of the receiver in scanning mode

The aim of this test is to determine the real scanning timing performance of the DUT when specific *Scan_Window* and *Scan_Interval* values are configured. In this mode the scanner device listens for advertising messages during an interval called *Scan_Window* and every *Scan_Interval* the device changes its scanning frequency to the next advertising channel. If both values are equal, the device is supposed to be scanning continuously. In this case, the ideal behavior is as depicted in Figure 1 and Figure 2. However, in practice, the device performance does not correspond to this assumption.

To characterize this behavior, just the section A of the generic setup scheme (Figure 5) is used. The computer and the DUT are connected via a USB cable. Although the DUT is powered through the USB cable, the power supply unit is still necessary for powering the current sensor.

The Bluetooth specifications define a HCI command that allows to configure the *Scan_Window* and *Scan_Interval* parameters. This command is accessible through the “hcitool” executable, which is part of the BlueZ Linux package. We use it, in combination with the “hciconfig” command, to configure the different devices present in the scenario by means of bash configuration scripts. To fully understand these scripts it is necessary to know that these commands implement part of the standard host controller interface (HCI) layer defined and detailed in the Bluetooth specifications [22]. The HCI layer carries commands and events, between the host and the Bluetooth hardware, which are composed of two main parts: the operation code and the parameters. The operation code is composed of two fields: OpCode Group Field (OGF) and OpCode Command Field (OCF).

For this scenario we only require one scanning device. We achieve the required configuration by running the bash script of Figure 8. The script first disables the Inquire and Page Scan. Next, the LE (Low Energy) Scan is first disabled and then configured and enabled with the following parameters: type (active/passive), scan interval, scan window, own address type and filtering policy. Notice that multiple byte parameters must be passed to the “hcitool” command starting by the less significant byte. For example, the LE_Scan_Interval (duration of the scan interval in slots) value is 0x0320 (800d), which corresponds to 800*625μs=500ms, but the bytes must be given as 20 03.

```
#!/bin/bash
# Disable classic Inquiry Scan and Page Scan
    sudo hciconfig -a hci0 noscan

# Disable BLE scan with HCI_LE_Set_Scan_Enable command
# OGF=0x08 OCF=0x000C
# LE_Scan_Enable=0x00, Filter_Duplicates=0x00
    sudo hcitool -i hci0 cmd 0x08 0x000C 00 00

# LE Scan parameters configuration with HCI_LE_Set_Scan_Parameters
# Command: OGF=0x08 OCF=0x000B
# LE_Scan_Type=0x00, LE_Scan_Interval= 20 03
# LE_Scan_Window=20 03, Own_Address_Type=0x00
# Scanning_Filter_Policy=0x00
    sudo hcitool -i hci0 cmd 0x08 0x000B 00 20 03 20 03 00 00

#Enable BLE scanning with HCI_LE_Set_Scan_Enable command
# OGF=0x08 OCF=0x000C
# LE_Scan_Enable=0x01, Filter_Duplicates=0x00
    sudo hcitool -i hci0 cmd 0x08 0x000C 01 00
```

Figure 8. Passive mode and continuous scanning configuration script

Figure 9 shows the measured device current after the execution of the script. The marker on the left side is the ground reference. From these measurements we can determine the real behavior of the BLE device. In this case, when the current consumption is high the device is in the scanning process, waiting for the reception of BLE frames. When the consumption is under this level the receiver is disabled. This fact can be confirmed by setting a *Scan_Window* different to the *Scan_Interval* parameter and comparing the results.

Additionally, we can observe in Figure 9 that every 500ms (the preconfigured *Scan_Interval*) there is a noticeable drop in the current consumption. We have empirically tested that if a frame is received in this precise moment it is not detected. Therefore, it can be inferred that these drops correspond to the deactivation of the RF stage. It is important to remark that even when the device is configured to be in continuous scan (*Scan_Window=Scan_Interval*), in practice the device is not scanning 100% of the time. We presume that this behavior is due to hardware limitations, which require a readjustment period when changing the scanning frequency and thus blocking the detection of new frames. Figure 10 shows a detailed zoom where the duration of this “blind time” can be measured. On the other hand, we have also checked that the duration of this gap depends on the specific values of the *Scan_Window* and *Scan_Interval* parameters.

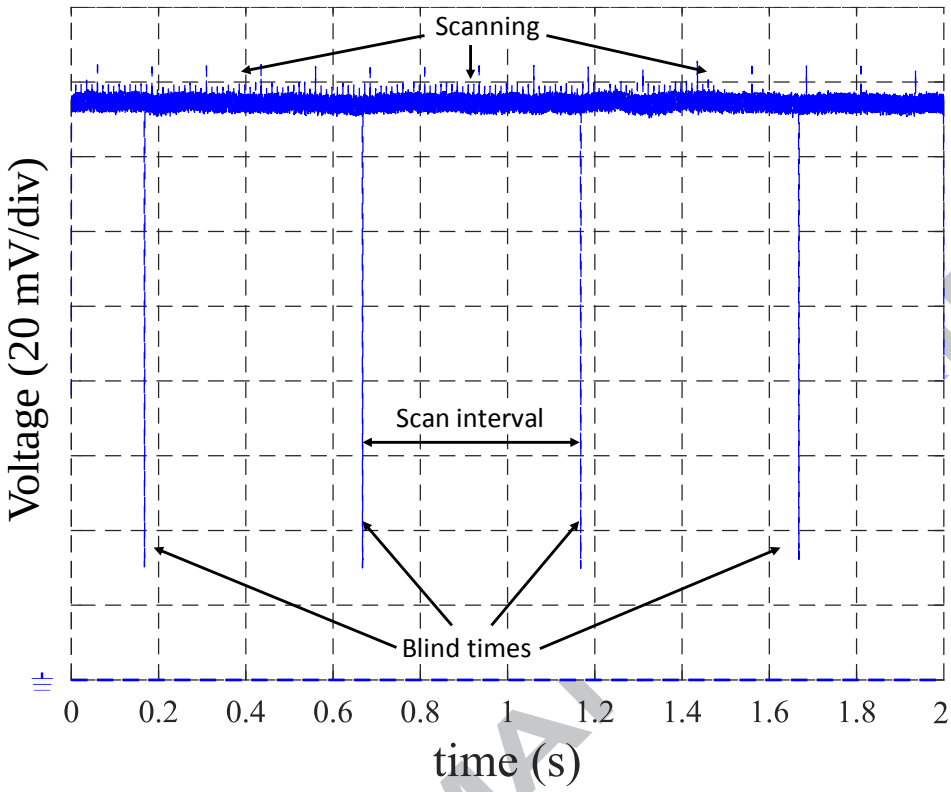

Figure 9. First device: current consumption pattern of one scanner in a scenario without advertisers

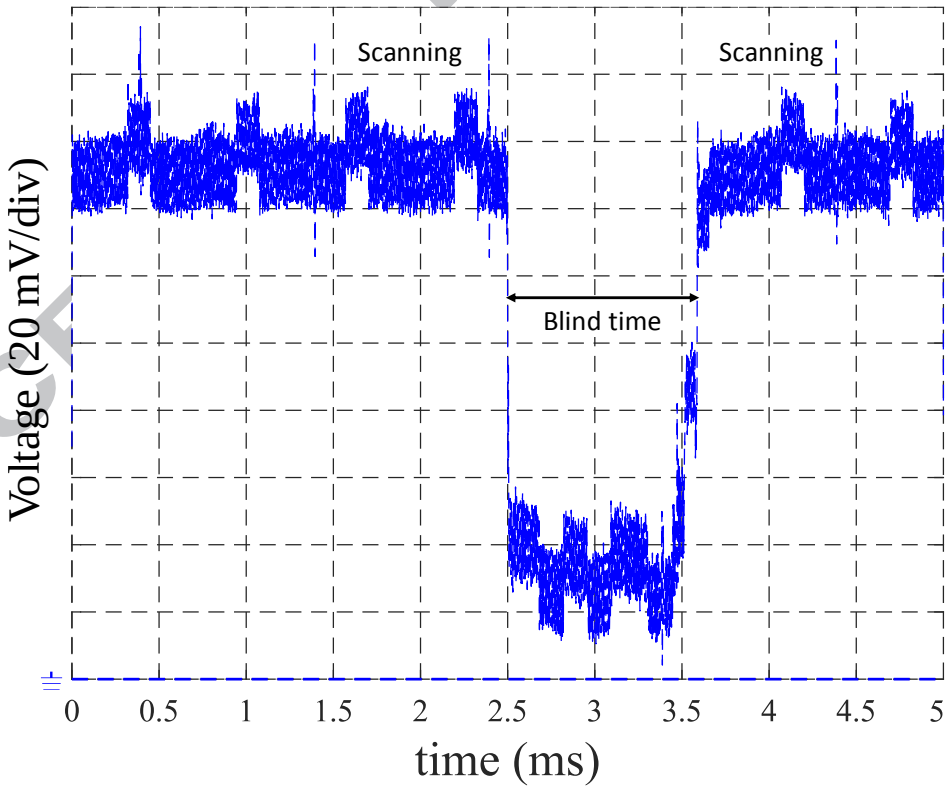

Figure 10. First device: Detailed zoom of a scanning blind time

In Figure 11 and Figure 12 the behavior of a different, but identically configured, chipset is depicted. It can be observed that there is also a drop in the consumption every 500 ms. However, this device presents many other additional “blind times” that we associate to unknown processes

caused by the manufacturer's implementation and programming of the firmware. The duration of the first gap can be measured in the zoomed version of the figure depicted in Figure 12. Nevertheless, the duration of all these "blind times" in this case does not depend on the configured *Scan_Window* and *Scan_Interval* parameters.

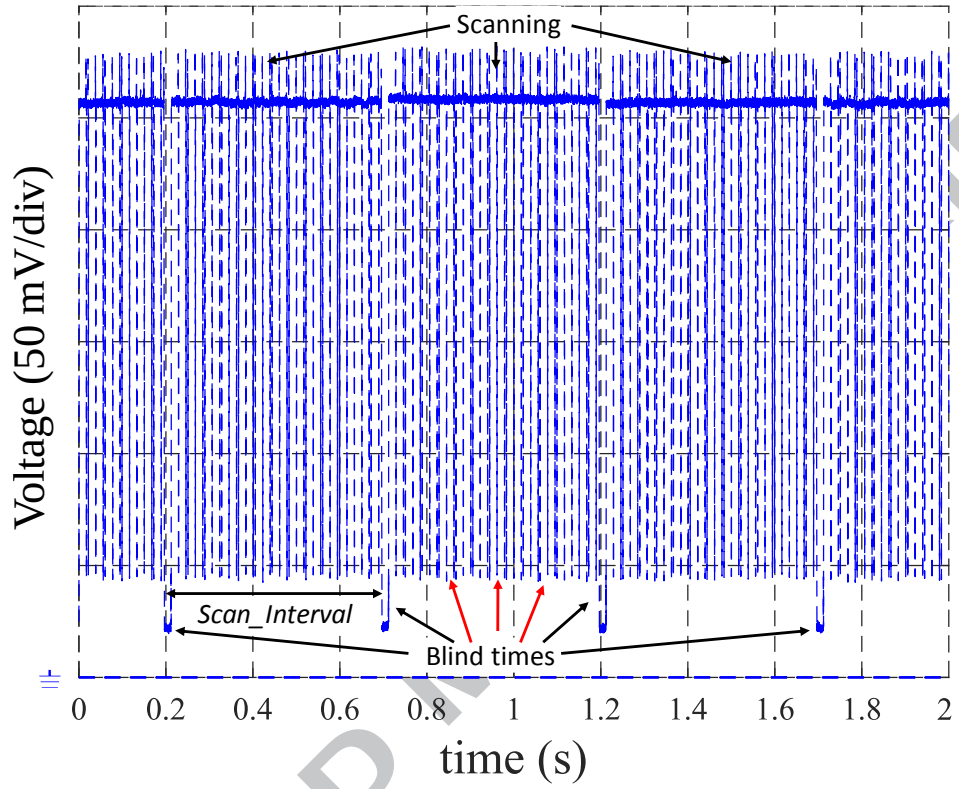

Figure 11. Second device: current consumption pattern of one scanner in a scenario without advertisers

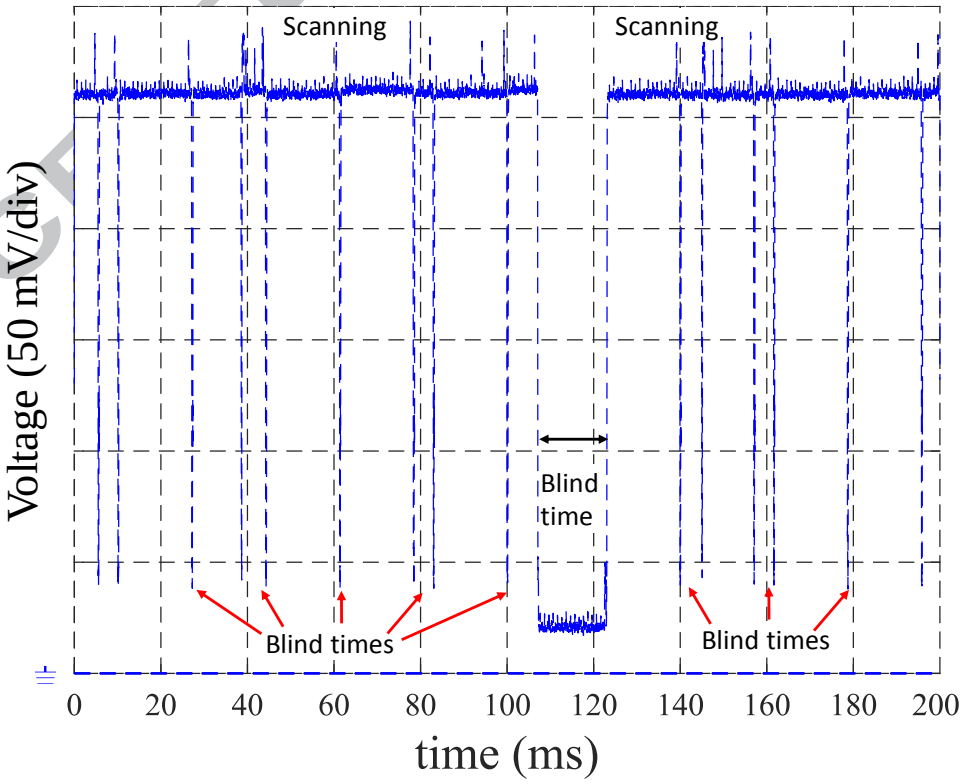

Figure 12. Second device: Detailed zoom of a scanning blind time

Summarizing, real BLE scanners cannot scan continuously and present blind times. Some devices present only these scanning gaps when they change their scan frequency. The duration of these gaps is variable and depends on the selected values of *Scan_Window* and *Scan_Interval.* Other devices also present these frequency change gaps (depicted in black) but they also introduce other periodic blind times (depicted in red) associated with unknown processes of the manufacturer firmware implementation [3,19].

B. Characterization of the receiver during frame reception

The next test determines the receiver timing performance of the DUT when it receives a BLE frame. The results are obtained by jointly analyzing the transmitter (signal generator) and receiver (DUT) current measurements and the time synchronized information obtained from Wireshark. The scanner is configured in passive mode and continuous scanning, i.e., *Scan_Window* equal to *Scan_Interval*. A couple of BLE devices or a signal generator are configured to transmit non-connectable advertising frames. If the device transmits at high power level, an attenuator could be inserted between the antenna and the DUT to protect it.

From the general setup depicted in Figure 5, we use the sections marked as A and B. In this case the DUT is the scanner and we include two transmitters. All three devices are controlled and powered via USB and monitored using current sensors. Again, we use a power supply unit for powering the current sensors.

In this scenario the DUT is configured using the script of Figure 8 while the other two devices are configured using the script of Figure 13, which configures one BLE device as an advertiser.

```
#!/bin/bash
# Disable classic Inquiry Scan and Page Scan
    sudo hciconfig -a hci1 noscan

#Transmitter device(ADV_NONCONN_IND)
#Disable Advertising transmission
# OGF=0x08 OCF=0x000A
# Advertising_Enable: 0x00
    sudo hcitool -i hci1 cmd 0x08 0x000A 0x00

# LE Advertising parameters configuration with HCI_LE_Set_
# Advertising_Parameters command
# OGF=0x08 OCF=0x0006
# Adv_Interval_Min= Adv_Interval_Max 0x00A0 Adv_Type=0x03
# Own_Address_Type= 0x00 Peer_Address_Type= 0x00,
# Peer_Address= 00:00:00:00:00:00, Advertising_Channel_Map=0x07,
# Advertising_Filter_Policy= 00
    sudo hcitool -i hci1 cmd 0x08 0x0006 A0 00 A0 00 03 00 00 00 00
00 00 00 00 07 00

# Set Advertising Data with HCI_LE_Set_Advertising_Data Command.
# OGF=0x08 OCF=0x0008
# Advertising_Data_Length=1F (376 microseconds frame)
# Data (length + Data AD type + 26 bytes). 1B 09 73 74 6F 70 73 73
# 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 74 6F 70
# Flags field 3 bytes (length + flag AD type + 1 byte) 02 01 08
    sudo hcitool -i hci1 cmd 0x08 0x0008 1F 1B 09 73 74 6F 70 73 73
73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 74 6F 70 02 01 08

# Enable Advertising transmission
# OGF=0x08 OCF=0x000A
# Advertising_Enable: 0x01
sudo hcitool -i hci1 cmd 0x08 0x000A 0x01
```

Figure 13. BLE advertiser configuration script

In this script, after disabling the Inquire and Page scan, the advertising transmission is also deactivated. This allows to configure the advertising parameters such as (more information in [22] Vol 2. Part E- 7.8.5):

- Minimum advertising interval and maximum advertising interval (0x00A0): both equal in order to fix the advertising interval to a unique value and, at the same time, to the minimum advertising interval allowed by the standard for this advertising type (100 ms).
- Advertising type (0x03): Non-connectable undirected advertisement.
- Peer address type (0x00) and peer address (00:00:00:00:00:00): to select a public device address type and its value.
- Advertising channel map (0x07): to use the three available advertising channels.
- Advertising filtering policy (0x00): no filters applied.

Once the advertising parameters are configured, the advertisement data to be transmitted are defined. In this case we decided to fill the frame to the maximum allowed capacity by transmitting a 26-bytes device name and a “flags” field to enable the reading by a smartphone application (nRF-connect by Nordic semiconductors). Finally, the transmission is enabled again.

Figure 14 and Figure 15 show the results after the execution of the configuration scripts. In this case the scenario consisted on one scanner (pink line) and two advertisers (blue and red lines). The “ground” markers on the left of the figure refer to the ground reference level for each oscilloscope channel. Figure 14 represents the transmission of a full advertisement event composed of three frames, transmitted in channels 37, 38 and 39, for each of the advertisers. As can be observed, when the transmission of the first frame of the blue transmitter is finished, the scanner decrements its consumption. At this precise moment, the protocol analyzer captures a message with the contents of the transmitted frame. Then the scanner returns to the original consumption level after which there is a new transmission event by the red advertiser. Again, after the end of the frame, a reduction of the consumption shows up in the scanner and the corresponding frame is received at the protocol analyzer. So, we can conclude that the scanner is tuned to channel 37. The figure also shows that, for this manufacturer, the duration of the gap after the reception of a frame is variable.

Figure 15 confirms that this type of gaps are in fact “blind times”, i.e. time periods where, if a frame is transmitted, it is not actually received by the scanner. In this example the scanner is tuned to channel 38 (the drop of consumption comes after the second frame of the blue advertiser). It can be seen that, during this gap, there is a new frame transmission on channel 38 made by the red advertiser. However this frame does not show up in the protocol analyzer. So we can conclude that it has not be detected and this gap is, in fact, another “blind time”.

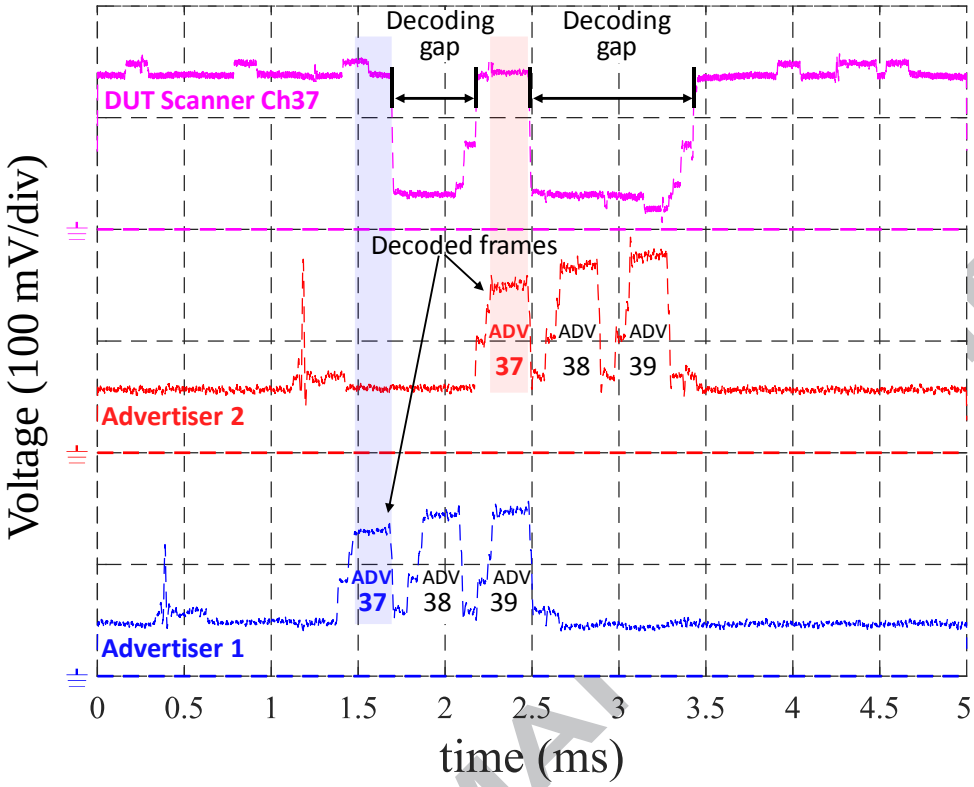

Figure 14. First device: decoding gaps

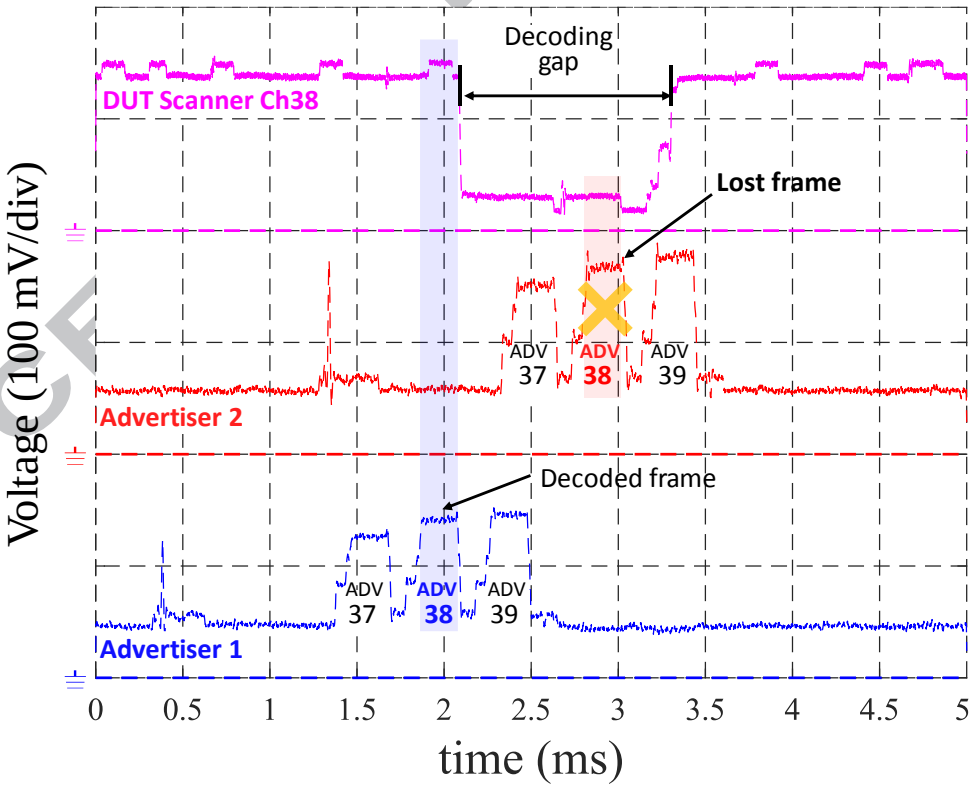

Figure 15. First device: example of frame losses caused by the decoding gaps

The following pictures depict the same scenario but with a scanner of a different manufacturer. It can be observed (Figure 16) that the gaps after the reception of a frame are also present. However, in this case their duration is constant and considerably shorter (194 μs). Figure 17 also confirms that the decoding gaps produce a blind time during which any received frame is not processed.

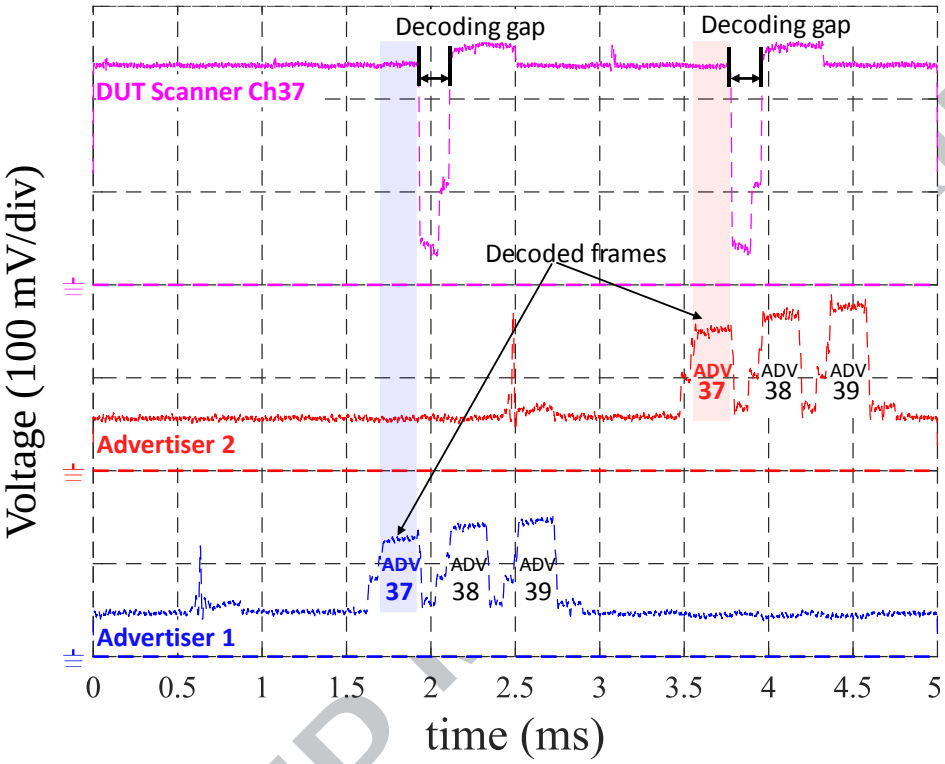

Figure 16. Second device: shorter and constant decoding gaps

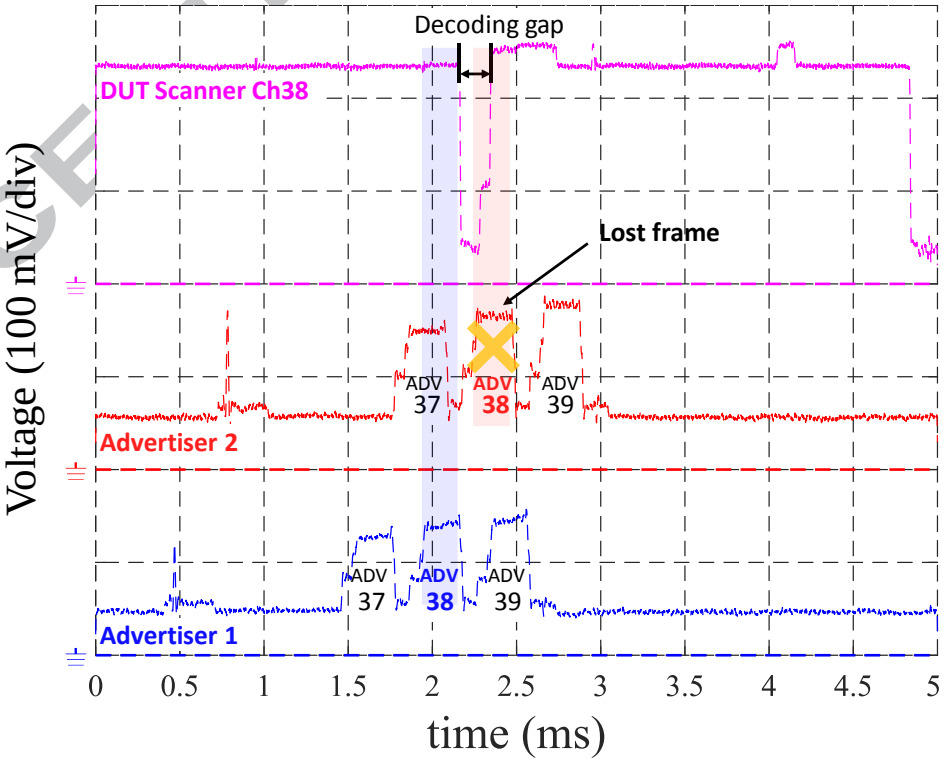

Figure 17. Second device: Blind time produced by the decoding gap

In this case we have found that the scanners present additional blind times whenever they receive a frame. Depending on the manufacturer implementation these gaps are variable or fixed. They may also be different when the received frame is demodulated correctly or not. More details can be found in [3,19].

C. Characterization of the behavior in presence of frame collisions

The next two tests evaluate the receiver timing performance of the DUT when it receives two overlapped Bluetooth frames. We analyze together the data obtained from Wireshark and the measurements from the current sensors. We will consider two cases: when the scanner is in passive mode (non-connectable mode) and also when it requests additional data (scannable mode).

C.1.- Non-connectable advertising frames collisions

In this case the scanner device just listens for advertising messages continuously (*Scan_Window* = *Scan_Interval*). Two Bluetooth devices or signal generators are configured to transmit non-connectable advertising frames. Thus, only the A and B sections of the generic measurement setup (Figure 5) are needed. Again, if the advertisers transmit at high power levels, an attenuator may be inserted to avoid the saturation of the DUT. The configuration scripts are the same of the previous section. The scanner follows the script of Figure 8 and the advertisers the script of Figure 13. However, in this case the oscilloscope trigger is specifically configured to detect overlapped frames. Figure 18 and Figure 19 depict the obtained results.

First of all, in some cases a collision between two frames could imply that the two frames are unsuccessfully received, but it is also possible that one of them could be demodulated (capture effect). In this case, and due to the sequential nature of the receiver, only the first detected frame would be received successfully if the signal-to-interference-plus-noise ratio (SINR) is good enough to avoid reception errors.

Figure 18 and Figure 19 show the cases where the frame cannot be correctly demodulated. In both cases the DUT starts the demodulation of the first incoming frame and, during the process, another frame on the same frequency arrives. As the receiver is working on a previous demodulation process, the second frame is lost. Additionally, the part of the first frame that collides with the second presents errors, so the first frame is not reported either.

In Figure 18, it can be observed that the device introduces a “blind time” similar to the one introduced for a successful reception. Nevertheless, for the other DUT (Figure 19), the blind time is shorter (144 μs) than the one corresponding to a successful reception (194 μs).

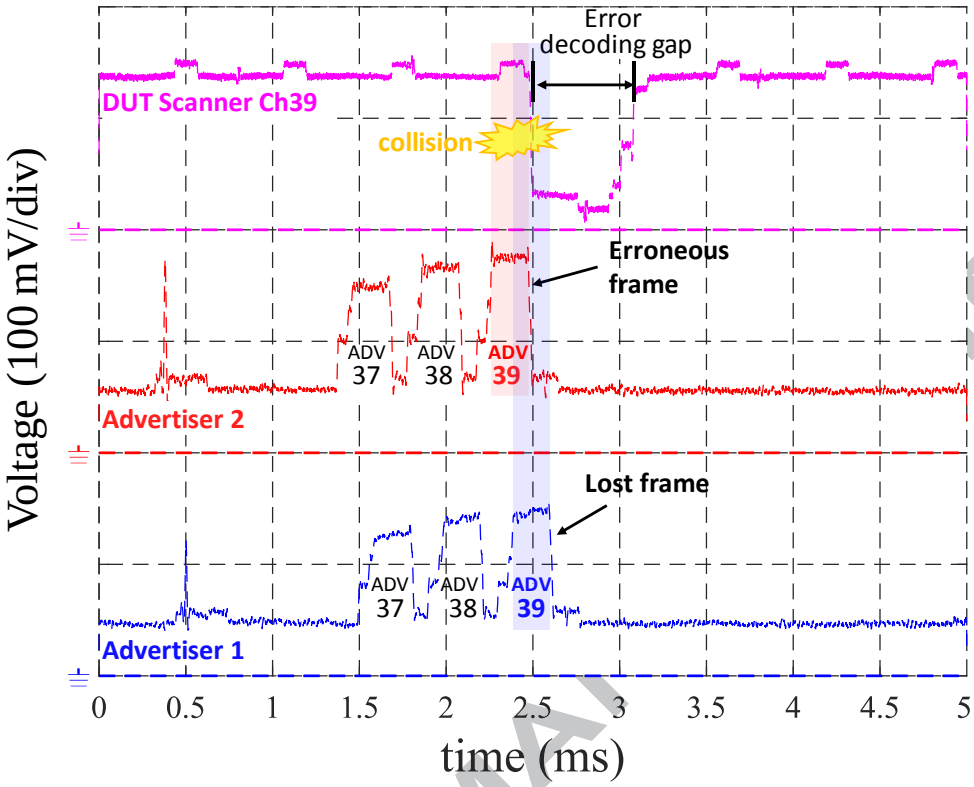

Figure 18. First device: two frames collide, the first is demodulated erroneously the second is lost. The scanner shows a new gap.

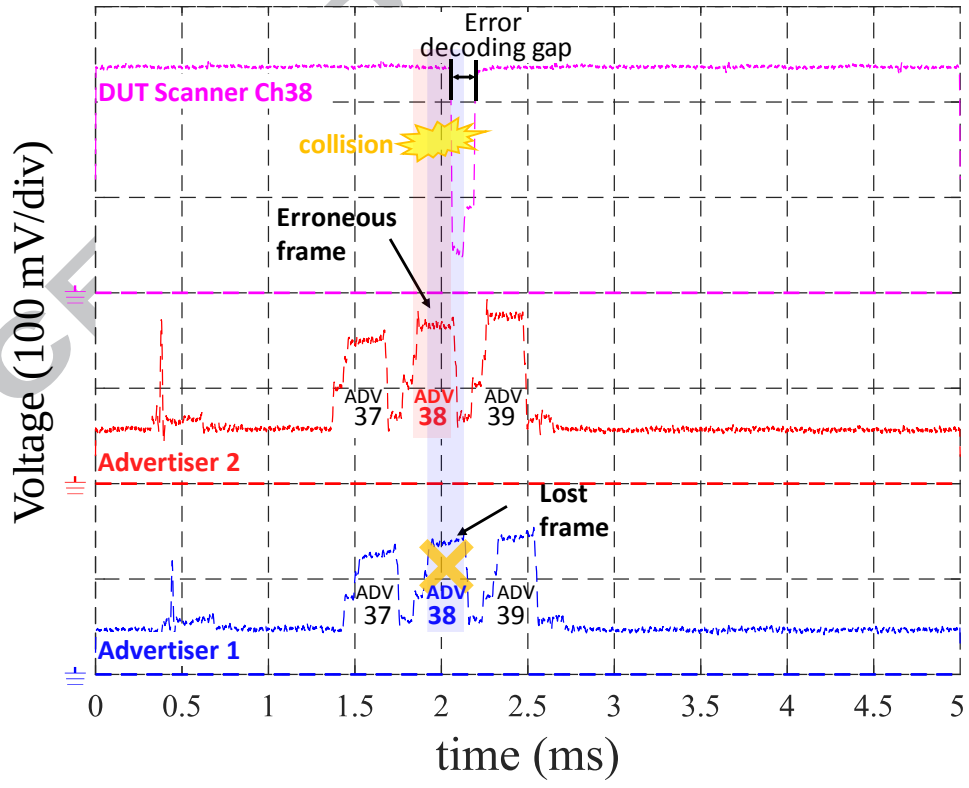

Figure 19. Second device: erroneous and lost frame due to a collision and its corresponding blind time.

C.2.- Scannable advertising frames collision

In this second test the scenario is practically the same. However, the scanner is configured in active mode and the advertisers use scannable advertisement events. In order to enable the active scanning, the LE_Scan_Type parameter of the scanner configuration script (Figure 8) should be set to 0x01. The rest is kept unmodified. However, for the advertisers it is necessary to configure the contents of the SCAN_RSP frame and modify the advertisement type. The new script is shown in Figure 20.

```
#!/bin/bash
# Disable classic Inquiry Scan and Page Scan
    sudo hciconfig -a hci1 noscan
    sudo hciconfig -a hci2 noscan

#Transmitter device (ADV_SCAN_IND)
#Disable Advertising transmission
# OGF=0x08 OCF=0x000A
# Advertising_Enable: 0x00
    sudo hcitool -i hci1 cmd 0x08 0x000A 0x00

# LE Advertising parameters configuration with HCI_LE_Set_
# Advertising_Parameters command
# OGF=0x08 OCF=0x0006
# Adv_Interval_Min=A0 00 Adv_Interval_Max A0 00 Adv_Type=02
# Own_Address_Type= 00 Peer_Address_Type= 00,
# Peer_Address 00 00 00 00 00 00, Advertising_Channel_Map= 07,
# Advertising_Filter_Policy= 00
    sudo hcitool -i hci1 cmd 0x08 0x0006 A0 00 A0 00 02 00 00 00 00
00 00 00 00 07 00

# Set Advertising Data with HCI_LE_Set_Advertising_Data Command.
# OGF=0x08 OCF=0x0008
# Advertising_Data_Length=1F (376 microseconds frame)
# Data (length + Data AD type + 26 bytes). 1B 09 73 74 6F 70 73 73
# 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 74 6F 70
# Flags field 3 bytes (length + flag AD type + 1 byte) 02 01 08
    sudo hcitool -i hci1 cmd 0x08 0x0008 1F 1B 09 73 74 6F 70 73 73
73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 74 6F 70 02 01 08

# Set Advertising scan response Data with
# HCI_LE_Set_Scan_Response_Data Command.
# OGF=0x08 OCF=0x0009
# Advertising_Data_Length=1F (376 microseconds frame)
# Data (length + Data AD type + 26 bytes). 1B 09 73 74 6F 70 73 73
# 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 74 6F 70
# Flags field 3 bytes (length + flag AD type + 1 byte) 02 01 08
    sudo hcitool -i hci1 cmd 0x08 0x0008 1F 1B 09 73 74 6F 70 73 73
73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 74 6F 70 02 01 08

# Enable Advertising transmission
# OGF=0x08 OCF=0x000A
# Advertising_Enable: 01
    sudo hcitool -i hci1 cmd 0x08 0x000A 0x01
```

Figure 20. Scannable advertisement configuration script

Figure 21 exemplifies the case where the advertisement and response of both transmitters are successfully received. In this instance the scanner is tuned to channel 38. This figure demonstrates that the receiver detects the advertisement which arrives in the first place (blue). After the TIFS=150 μs required by the Bluetooth specifications, it transmits a request (pink). Then, the advertiser answers with a response which is successfully decoded by the scanner. The protocol analyzer records both frames: the advertisement and the response. Later, the procedure is repeated with the red advertiser.

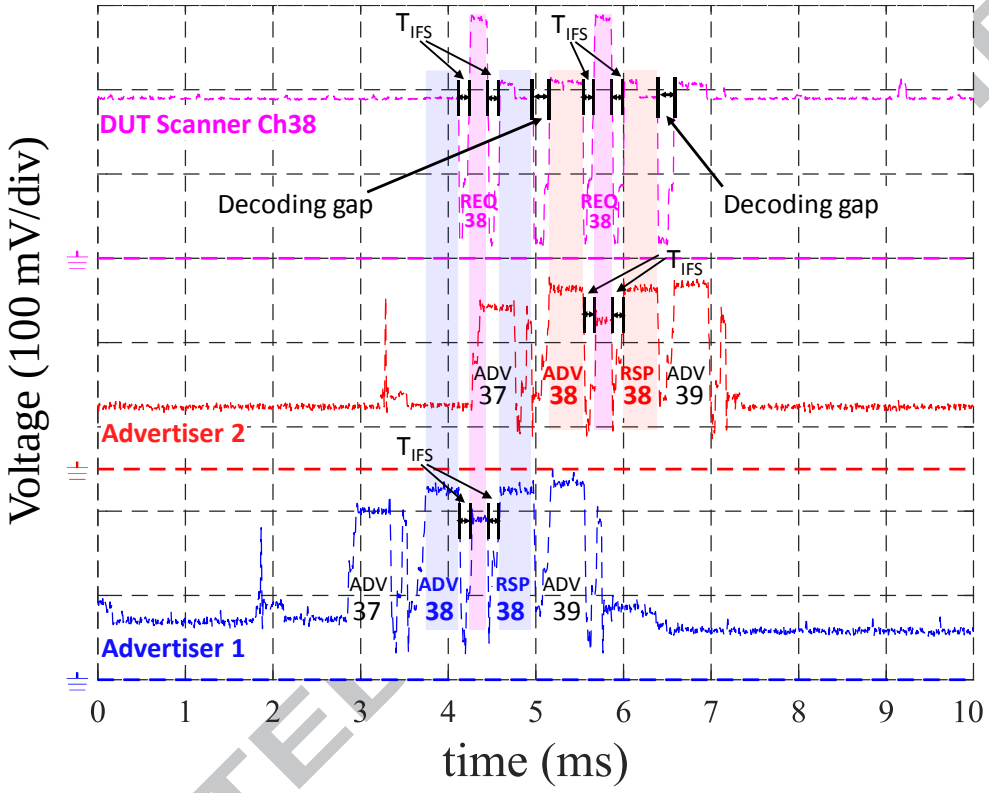

Figure 21. Successful exchange of advertisement-request-response frames

In Figure 22, however, the advertisement of the first transmitter (blue) is not detected, as the DUT is in a blind time when the frame starts. Although there is no collision in this case, strictly speaking, Wireshark only reports the first advertisement and response frames.

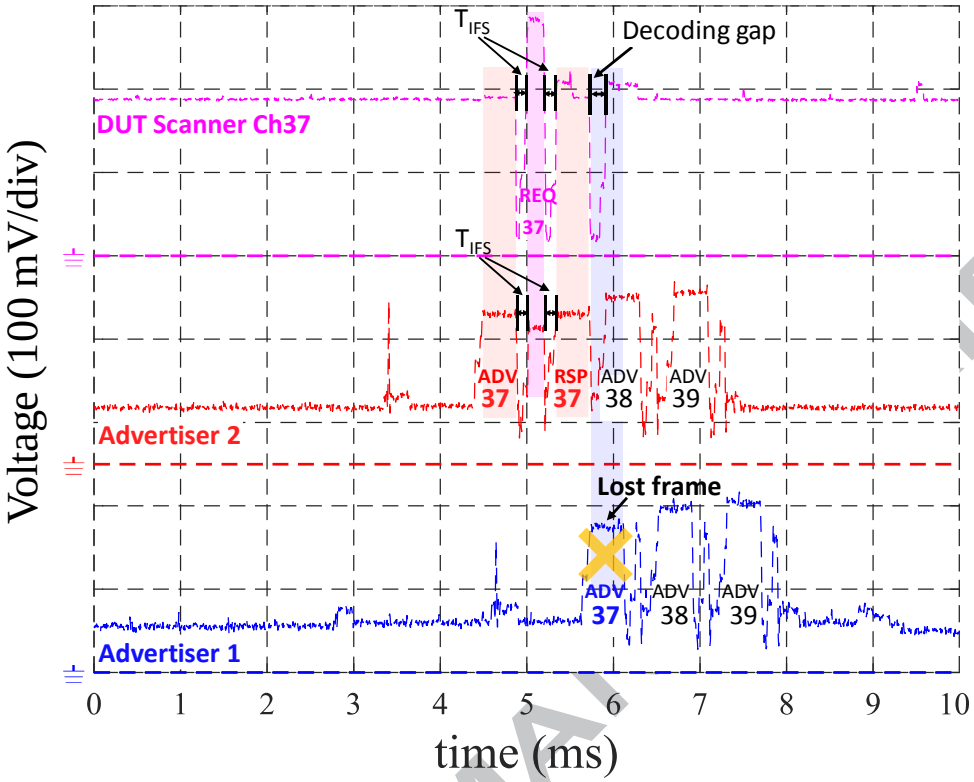

Figure 22. Advertiser 1 (channel 37) arrives during a blind time and it is lost

The next case (Figure 23) is also similar, the difference is that the response is not detected because it is overlapped with another advertisement and the SINR is not enough to avoid reception errors. In this case the protocol analyzer only shows up the content of the advertisement, but not the content of the response.

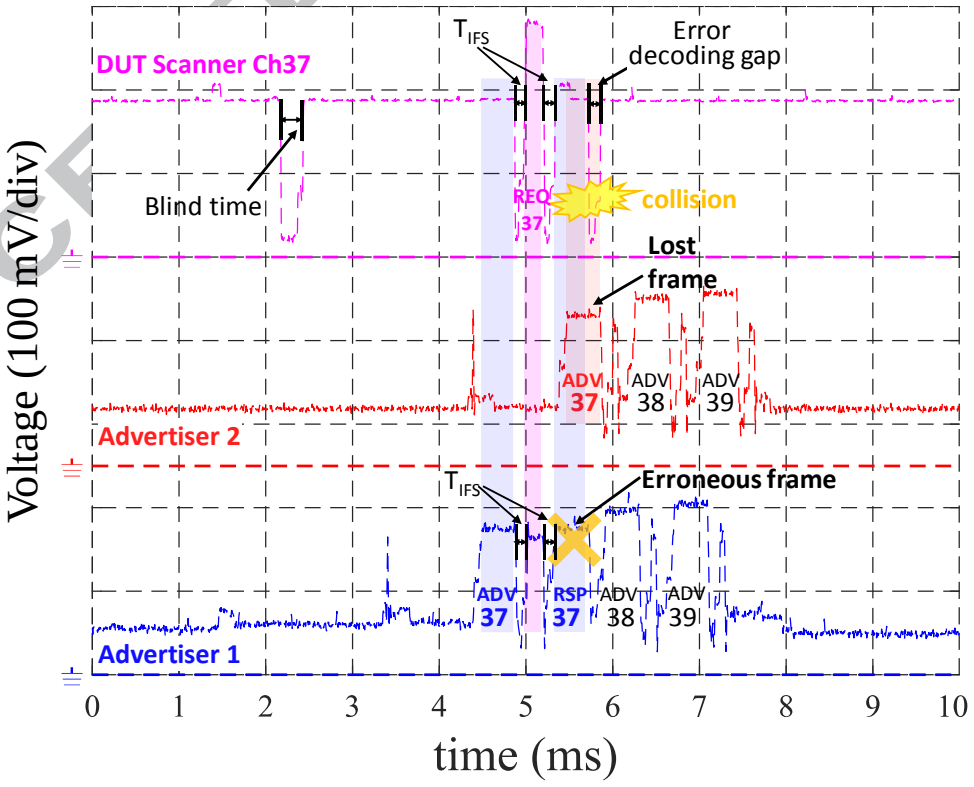

Figure 23. Collision of an incoming advertisement with a previous response frame

The following case (Figure 24) reproduces the simultaneous reception of two advertisements. The scanner demodulates successfully one of them (red) with the corresponding Wireshark report and then the DUT transmits a request that is detected by both advertisers. Advertiser 2 answers with a response, while Advertiser 1 ends its operation because the request frame does not contain its address.

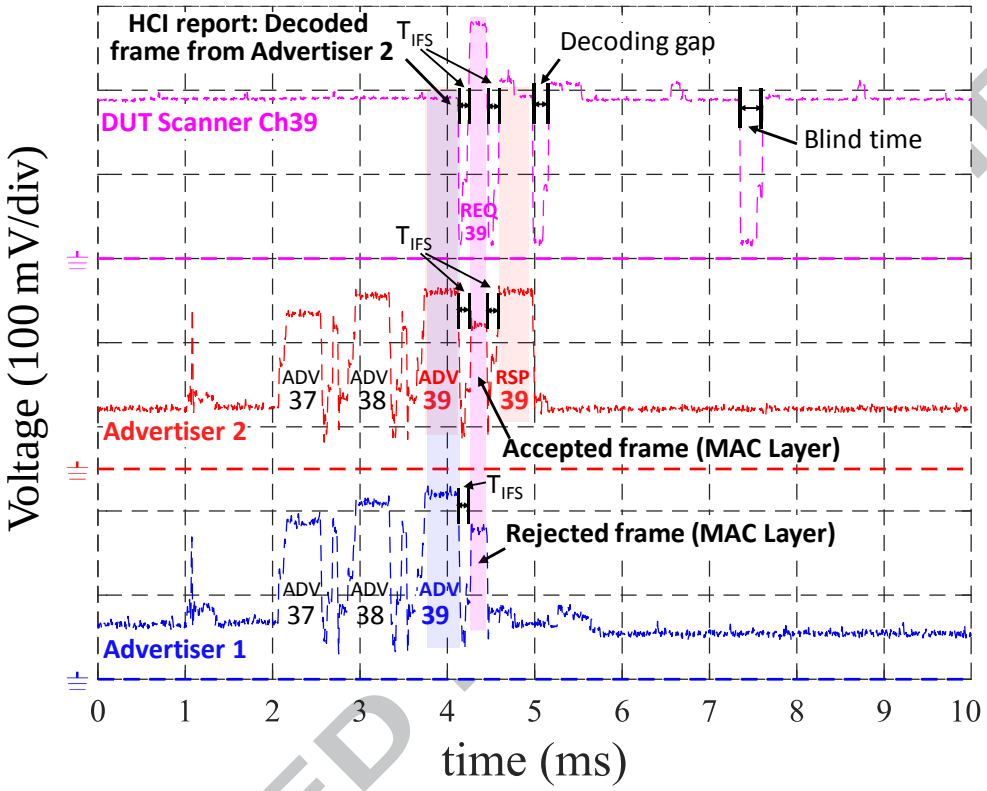

Figure 24. Collision between two advertisements

In Figure 25 there is a simultaneous transmission of an advertisement and a response frame. The DUT demodulates the first incoming advertisement and makes a request. Next, the blue transmitter answers with a response, but it is overlapped with an advertisement with better channel conditions. The DUT demodulates the advertisement and transmits another request whose response, in this case, is successfully received. In this scenario, the protocol analyzer reports only an advertisement from the blue transmitter and both an advertisement and a response from the red transmitter.

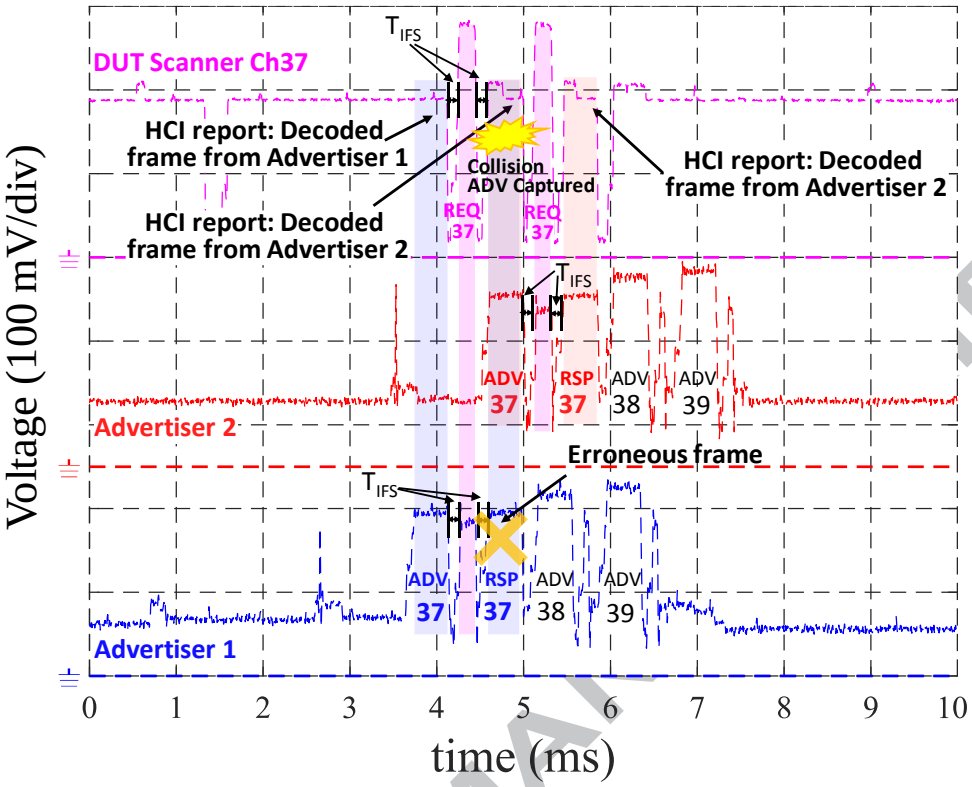

Figure 25. Collision between an advertisement and a response frame

Figure 26 shows a collision between an advertisement and a request frame. In this case, the advertiser does not generate the response because it is not aware of the request. Wireshark reports only an advertisement in this occasion.

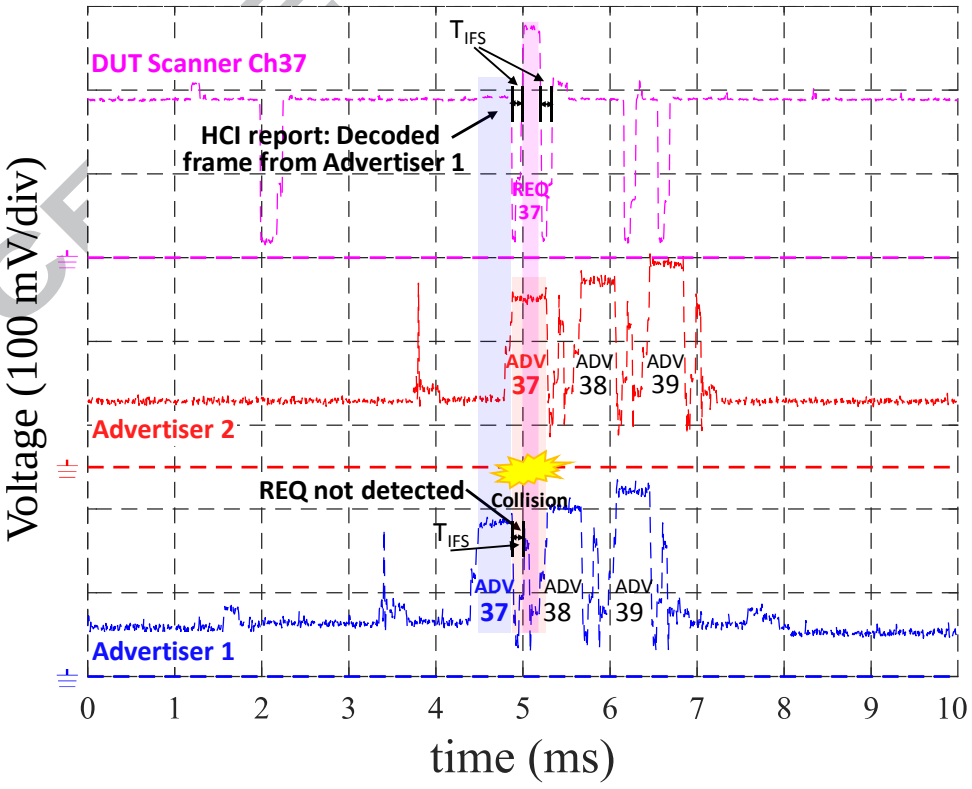

Figure 26. Collision between an advertisement and a request frame

In summary, these tests allow us to determine the precise duration of the processing gaps considering different frame overlapping cases. The specific values are manufacturer dependent. With the information provided by the protocol analyzer we can also deduce which frames are going to be discarded under certain conditions.

D. Backoff implementation test

This last test determines the receiver behavior of the DUT when the backoff mechanism should be activated. To evaluate this scenario only one advertiser and a scanner are necessary, but it is necessary to control the reception of request frames. Thus, from the general setup of Figure 5, we use sections A and C. The RF output of the transmitter is connected to the first port of the circulator. The second port is connected to a variable attenuator and its output goes to an antenna. A 50 Ω matching load is placed on the third port of the circulator. The idea beyond this is that the transmitted frames (advertisements) reach the antenna and are successfully received by the scanner, but the requests generated by the DUT never arrive to the advertiser. In this way the advertiser does not generate the response. The scanner interprets that a collision happened and initiates its backoff mechanism.

The configuration scripts of the advertiser and the scanner follow the configuration explained on the previous section (C.2.Scannable advertising frames collision). That is, continuous active scanning and scannable advertisements.

Figure 27 shows how we force the activation of the backoff mechanism. It can be seen that after the reception of the advertisement the DUT sends a request, but the advertiser never detects it because it has been redirected to the 50 Ω load by the circulator.

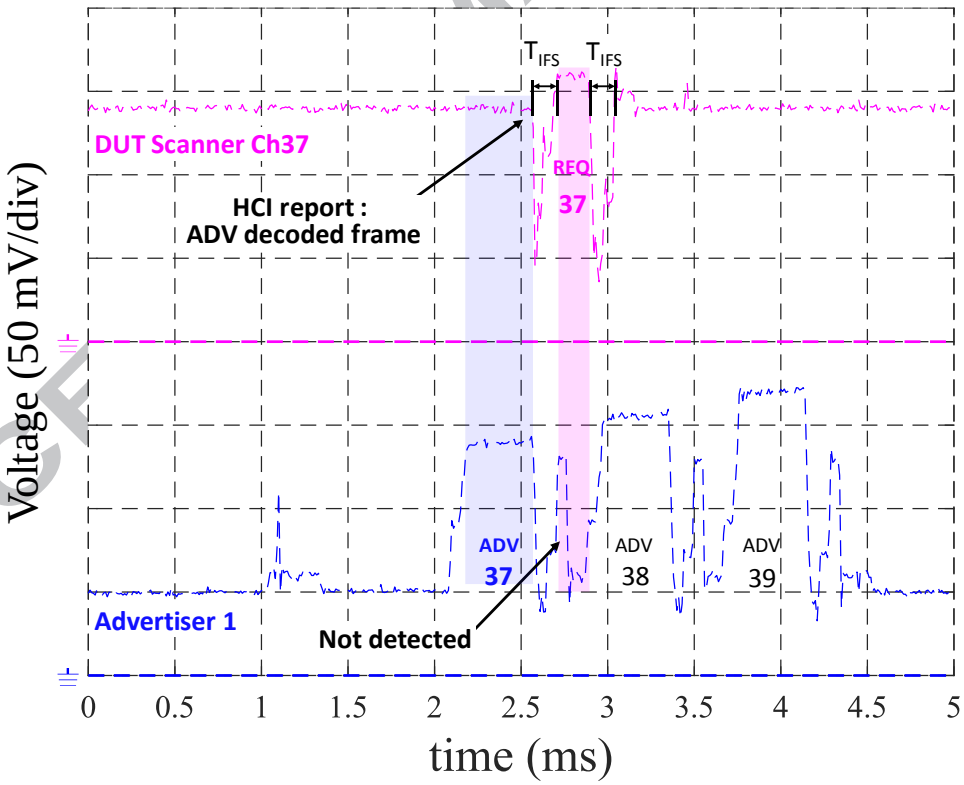

Figure 27. Forced suppression of the request frame

After having activated the backoff procedure of the DUT during a fixed period of time we remove the circulator and allow the normal transmission/reception of requests. Thus, if the device implements the backoff algorithm proposed by the standard (see section II.B) the *upperLimit* would be high enough to apply the backoff to the next successfully received advertisement frames. In Figure 28 we can observe how the scanner applies the backoff procedure. After the correct demodulation of an advertisement with its corresponding Wireshark report, the DUT does not generate the expected request.

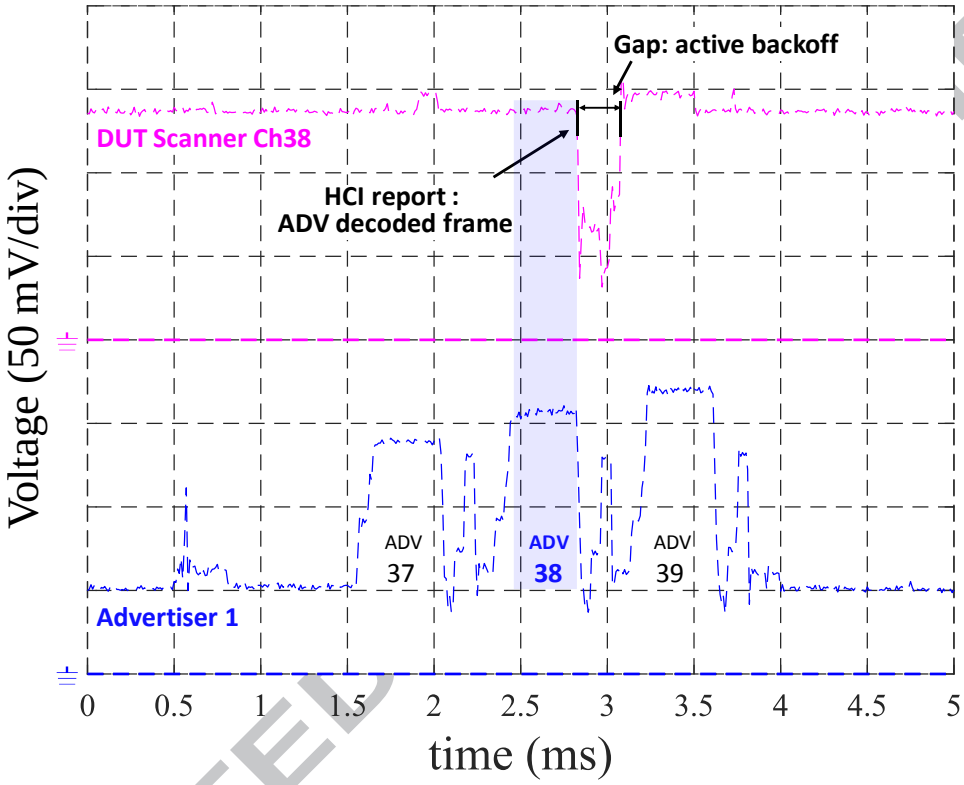

Figure 28. Backoff activated: successfully reported advertisement and request frame not sent

However, we have found that some manufacturers do not implement neither this nor any other kind of backoff procedure, so that after restoring the normal transmission/reception of request frames the DUT always transmits a request frame, as can be seen in Figure 29. This behavior does not fulfill the specification requirements, which stablish that the scanner should run a backoff procedure that allows to share the medium responsibly.

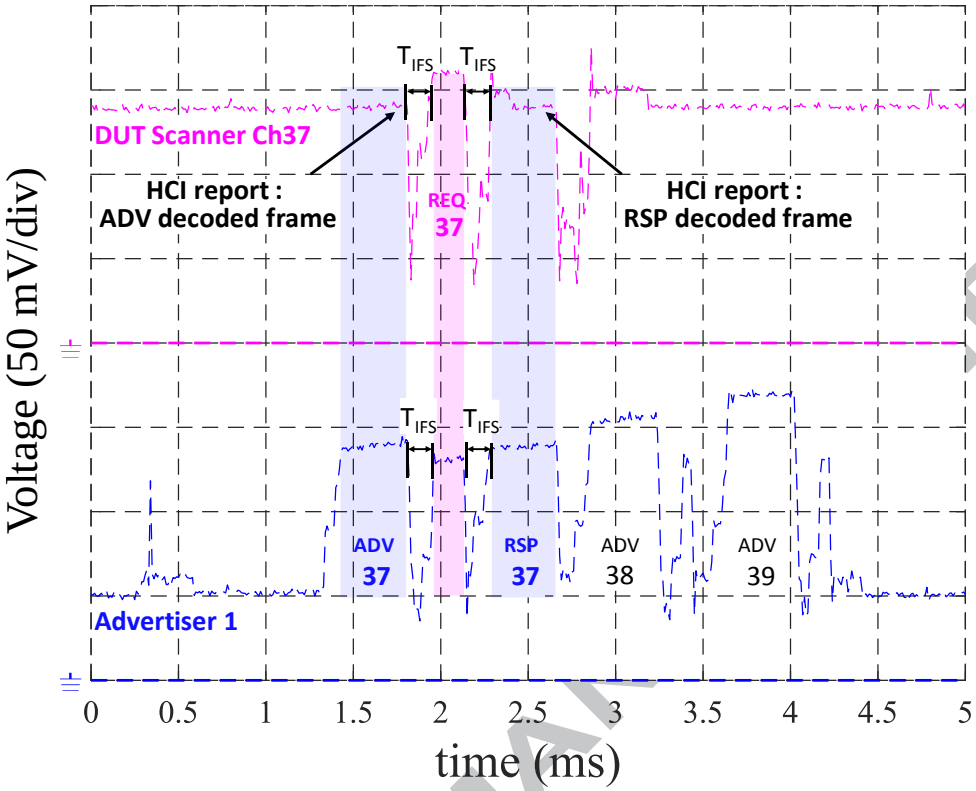

Figure 29. Anomaly: Backoff procedure not implemented

IV. Conclusions

In this paper we have defined several tests that allow us to characterize the real time behavior of BLE devices. These tests ease measuring impairments that are usually not considered but become very important in dense IoT wireless scenarios. By associating the current consumption variations and the upper layer information provided by a protocol analyzer we can infer and quantify with precision the unexpected performance shown by real devices.

To reflect the use of the methods described in the paper we have analyzed, just as an example, some of the impairments present in real Bluetooth Low Energy devices. That is, we have applied this methodology to profile the behavior of BLE scanners. We have measured that real BLE scanners cannot scan continuously and present blind times. These blind times appear under different circumstances associated with different processes of the scanner: frequency changes, frame decoding and other internal processes. Their duration is variable. It depends on the configuration parameters, if the frame decoding is successful or not and, of course, the device firmware. So by means of the proposed tests, we have detailed with accuracy the duration and appearance patterns of these gaps for several devices from different chipset manufacturers and situations.

Lastly, we have also checked the receiver behavior of the devices under test when the backoff mechanism should be activated. Even though it is mandatory, we have found out that some chipsets do not implement any backoff mechanism.

A main advantage of the proposed method is that it is low-cost. Besides the usual laboratory equipment (digital oscilloscopes and power sources) it only requires simple current sensors, and small modifications of the powering cables. An RF shield box would also be desirable if the measuring environment is not enough isolated. On the other hand, although along the paper we have used these tests to measure typical BLE impairments, the methodology could be easily applicable to other technologies like Wi-Fi, Zigbee, LoRa, etc.

V. Bibliography

[1] S. Shabdanov, P. Mitran, C. Rosenberg, Cross-layer optimization using advanced physical layer techniques in wireless mesh networks, IEEE Trans. Wirel. Commun. (2012). doi:10.1109/TWC.2012.070212.111859.

[2] C. Jung, K. Kim, J. Seo, B.N. Silva, K. Han, Topology Configuration and Multihop Routing Protocol for Bluetooth Low Energy Networks, IEEE Access. (2017). doi:10.1109/ACCESS.2017.2707556.

[3] D. Perez Diaz de Cerio, A. Hernandez, J.L. Valenzuela, A. Valdovinos, Analytical and experimental performance evaluation of BLE neighbor discovery process including non-idealities of real chipsets, Sensors (Switzerland). (2017). doi:10.3390/s17030499.

[4] BTI, RF PHY - Bluetooth® Test Specification RF-PHY.TS.5.0.1, (2017) 95.

[5] S. Kamath, J.L. Keywords, Measuring Bluetooth® Low Energy Power Consumption, (n.d.). http://www.ti.com/lit/an/swra347a/swra347a.pdf (accessed December 11, 2017).

[6] J. Lindh, C. Lee, M. Hernes, Measuring Bluetooth Low Energy Power Consumption, (2015). www.ti.com/ble-power-calculator. (accessed December 11, 2017).

[7] NXP Semiconductors, MKW40Z Power Consumption Analysis, (n.d.). https://www.nxp.com/docs/en/application-note/AN5272.pdf (accessed December 11, 2017).

[8] Silicon Labs, AN969: Measuring Power Consumption of Blue Gecko Bluetooth ® Smart Devices, (n.d.). https://www.silabs.com/documents/public/application-notes/an969-measuring-power-consumption.pdf (accessed December 11, 2017).

[9] Rohde & Schwarz, Wireless Device Testers by Rohde & Schwarz | Rohde & Schwarz, (n.d.). https://www.rohde-schwarz.com/us/products/test-measurement/wireless-communications-testers-systems/wireless-communication-testers-systems/wireless-device-testers_86475.html (accessed December 11, 2017).

[10] Keysight Technologies, N9081A Bluetooth® Measurement Application | Keysight (formerly Agilent's Electronic Measurement), (n.d.). https://www.keysight.com/en/pd-1867595-pn-N9081A/bluetooth-measurement-application?nid=-32129.955967&cc=ES&lc=eng (accessed December 11, 2017).

[11] J. Bernegger, M. Meli, Comparing the energy requirements of current bluetooth smart solutions, in: InES Inst. Embed. Syst. Nuremb. Embed. World Conf., Nuremberg, 2014. https://pd.zhaw.ch/publikation/upload/207967.pdf (accessed December 11, 2017).

[12] A. Dementyev, S. Hodges, S. Taylor, J. Smith, Power consumption analysis of Bluetooth Low Energy, ZigBee and ANT sensor nodes in a cyclic sleep scenario, in: 2013 IEEE Int. Wirel. Symp., IEEE, 2013: pp. 1–4. doi:10.1109/IEEE-IWS.2013.6616827.

[13] M. Siekkinen, M. Hiienkari, J.K. Nurminen, J. Nieminen, How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4, in: 2012 IEEE Wirel. Commun. Netw. Conf. Work., IEEE, 2012: pp. 232–237. doi:10.1109/WCNCW.2012.6215496.

[14] G.M. Al-Saadoon, B. Manama, Applying Packet Analysis as New Approach for Discovering Bluetooth Intrusion, in: Proc. 2011 ICICS Conf., 2011.

[15] W.K. Zegeye, Exploiting Bluetooth Low Energy Pairing Vulnerability in Telemedicine, in: Int. Telemetering Conf. Proc., 2015: p. Volume 51.

[16] S. Siby, R.R. Maiti, N.O. Tippenhauer, IoTScanner: Detecting and Classifying Privacy Threats in IoT Neighborhoods, (n.d.). https://arxiv.org/pdf/1701.05007.pdf (accessed December 12, 2017).

[17] W.B. Pöttner, L. Wolf, IEEE 802.15.4 packet analysis with Wireshark and off-the-shelf hardware, in: Proc. Seventh Int. Conf. Networked Sens. Syst., Kassel, Germany, 2010.

[18] G. Combs, Wireshark, (n.d.). www.wireshark.org.

[19] Á. Hernández-Solana, D. Perez-Diaz-de-Cerio, A. Valdovinos, J.L. Valenzuela, Proposal and Evaluation of BLE Discovery Process Based on New Features of Bluetooth 5.0, Sensors. 17 (2017) 1988. doi:10.3390/s17091988.

[20] D. Pérez Díaz de Cerio, M. González Rodríguez, J.L. Valenzuela González, J.M. González Arbesú, Trimming the power consumption of domestic Wi-Fi networks: how much power does your router need?, Proc. Jt. Newcom. Work. Wirel. Commun. (JNCW 2015), Barcelona, Spain, Oct. 14-15, 2015. (1000) 1–5. https://upcommons.upc.edu/handle/2117/83076 (accessed June 14, 2018).

[21] M. Mock, 0-1A, Single-Supply, Low-Side Current Sensing Solution. Texas Instruments., (2013). http://www.ti.com/lit/ug/tidu040b/tidu040b.pdf (accessed August 22, 2017).

[22] Bluetooth SIG, Bluetooth Core Specification 5.0, (n.d.). https://www.bluetooth.com/specifications/adopted-specifications (accessed August 15, 2017).

Acknowledgments: This work has been supported in part by the MINECO/ERDF under the projects TEC2014-58341-C4-2-R, TEC2014-60258-C2-2-R and RTI2018-099880-B-C32 and Gobierno de Aragón/FEDER (Research Group T31_17R)

HIGHLIGHTS:

- A low-cost generic measurement setup applied to test BLE procedures and devices.
- New test setups to enhance and measure the real behavior of IoT devices.
- Experimental characterization combining current measurement and upper layer response.
- Hardware design and software configuration of the proposed measurement system.