ATCZ175 InterOP PROJECT

Interference Measurement Setup

Institute of Electrodynamics, Microwave and Circuit Engineering
Technische Universität Wien

Advisor:
Assoc. Prof. Dipl.-Ing. Dr.techn. Holger Arthaber

by
Proj. Ass. Dipl.-Ing. Christian Spindelberger

October 9, 2019

Contents

Abbreviations

AWGN additive white Gaussian noise

BLE Bluetooth low energy

CP cyclic prefix

DSSS direct spread spectrum sequence

FFT fast Fourier transform

IoT internet of things

ISM industrial, scientific, and medical

IFFT inverse fast Fourier transform

MAC medium access control

MF matched filter

OFDM orthogonal frequency division multiplexing

PAPR peak-to-average power ratio

PER packet error rate

PHY physical

Q-Q quantile-quantile

RF radio frequency

RTS request to send

SINR signal to interference plus noise ratio

TCP transmission control protocol

UDP user datagram protocol

VSG vector signal generator

WLAN wireless local area network

1 Introduction

In this document the realization of a wireless local area network (WLAN) test setup, which is capable of emulating and measuring interference perturbations, is discussed.

At the beginning, the setup and related components are characterized in Section 2. Note that an instruction guide for an appropriate installation and use of the test setup can be found on InterOp homepage[1]. Several alternative interference sources, replacing expensive radio frequency (RF) equipment, are investigated. Section 3 treats the implementation of a Linux PC system injecting interfering signals and emerging issues. Furthermore, another interference source, utilizing modulated noise, is introduced in Section 4. A simple simulation model will compare different interferers, i.e., WLAN DSSS, OFDM, and BLE, to modulated noise. At last, a low-cost implementation of an interference source, modulating noise with the corresponding random variables from the measurement campaign, will be presented.

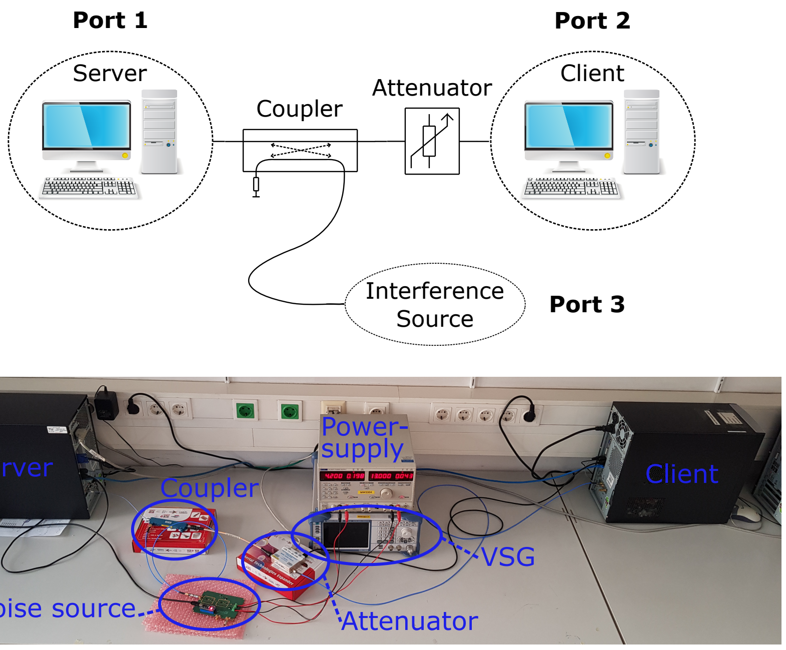


Figure 1: WLAN test setup: block diagram (top), realized arrangement (bottom) with the utilized interference sources (noise source with additional power supply & VSG)

[1]`http://www.interreg-interop.eu/fileadmin/t/InterOp/WLAN_instruction_guide_new.pdf`

2 Test Setup

Regarding WLAN systems, interference causes perturbations by impairing the throughput, packet error rate (PER), retransmissions, and packet delay (jitter). Hence, a test setup (Figure 1) has been created to examine different interference sources by measuring typical performance parameters.

The test arrangement consists of two independent Linux PC systems (server and client) with implemented WLAN modules. They are connected with coaxial cables, combined by a directional coupler and a variable attenuator. Lastly, an interference source, which is connected through the coupler, completes the setup. Details about the utilized components can be found in Table 1.

Component	**Manufacturer**	**Description**
WLAN modules	Atheros	WLE900VX
Coupler	Krytar	MODEL 1850
Attenuator	Mini Circuits	RCDAT-6000-60
VSG	Rohde und Schwarz	SMBV100A

Table 1: WLAN test setup: list of utilized components

The coupler guides the signal from the interference source to the server. This scenario recalls the hidden node problem. The client starts a transmission and the server receives available packets while the data exchange is corrupted by interference. Through the additional insertion loss maintained by the attenuator, the range between client and server can be changed. The adaptation of the range can be interpreted by changing the distance between transmitter (client) and receiver (server). Small ranges relate to close distances and vice versa. Thus, the setup realizes a hidden node scenario with variable ranges between client and server.

Furthermore, the interference source is assumed to be blind to any surrounding traffic established between server and client. It simply injects interference signals without any collision avoidance schemes. internet of things (IoT) systems typically exchange a low amount of data, resulting in a small packet size. The request to send (RTS) threshold is therefore deactivated. The fact that the traffic, caused by the interference source, does not have to respond to RTS-CTS transactions, justifies the hidden node model. It must be mentioned that the isolation realized by the coupler is not infinite. Consequently, the transmitter will receive an amount of interference signals even for a high attenuation.

Figure 2 depicts the relevant S-parameters according to Figure 1 for an attenuation of 0 dB.

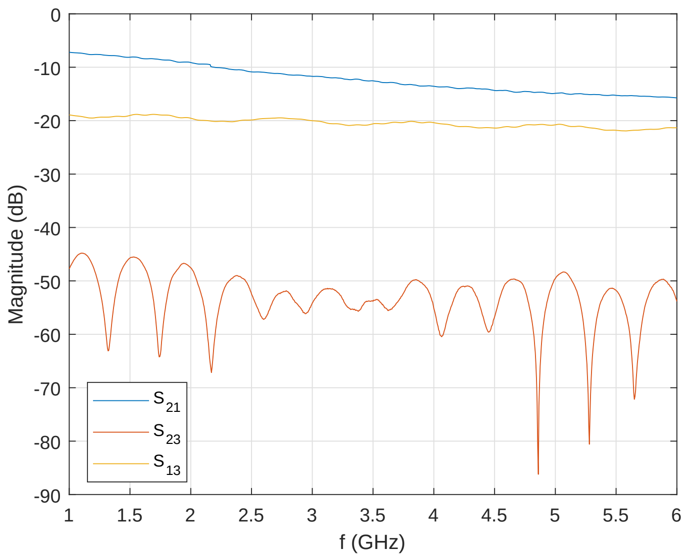


Figure 2: S-parameters of the WLAN test setup: server (port 1), client (port 2), interference source (port 3)

2.1 Measurement Setup Considerations

In the following, occurred issues, leading to the actual test setup (Figure 1), will be discussed. First of all it will be explained, why two separate Linux PCs have been used. Indeed, common PCs offer several PCIe[2] interfaces to connect multiple module extensions. Unfortunately, some issues made it impossible to realize the measurement setup with one PC.

Firmware Issues

To establish a connection between two WLAN modules, an access point (server) and a station (client) have to be configured. In the WLAN configuration guide, it is explained how an association can be established, but utilizing these instructions does not succeed using a single PC, for several reasons.

First of all, it was not possible to mount two identical modules within one PC and configure them individually. The firmware boot loader had problems with loading the desired

[2]Peripheral component interconnect express (PCIe) is a high-speed serial computer expansion bus standard, connecting extensions like WLAN modules.

firmware onto the WLAN modules separately. Therefore, network namespaces were implemented circumventing this problem. Namespaces have the capability to isolate single modules seen by the kernel, making it possible to configure them individually. By implementing this technique, it is possible to set up the two WLAN modules as server and client establishing a valid connection. Unfortunately, the Linux PC system was not able to accomplish performance tests with typical programs. Starting such a test always led to an immediate shut down of the whole system. As the speed of the shut down was very fast, similar to unplugging the power supply, the hardware functionality has been inspected carefully. In addition to this, the IP routing table and the MAC addresses have been verified correctly, but this behavior could not be clarified. Consequently, two separate Linux PC systems were inevitable.

Crosstalk Issues

Using two separate Linux PC systems made it possible to establish a valid connection and to accomplish performance tests, but another problem arose concerning crosstalk. The two systems were capable to associate with each other and exchange data without any cable or antenna connected. The transmit power of WLAN modules in Europe is limited to 20 dBm in the 2.4 GHz industrial, scientific, and medical (ISM) band. Utilizing this high power level makes it possible to overcome large distances of a few hundred meters. Therefore, it is necessary to decrease the output power to a minimum of 0 dBm. At this power level, the two Linux PC systems are not able to maintain an association anymore. Nevertheless, further crosstalk effects due to insufficient decoupling, causing signal leakage over power supplies and wired LAN routers, are presumed.

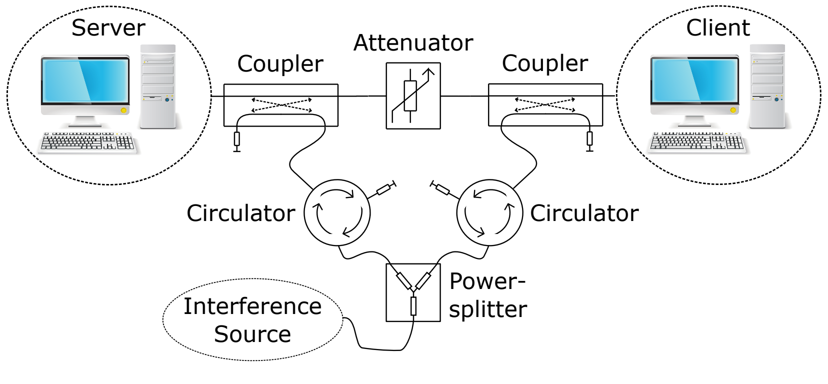


Figure 3: Alternative WLAN test setup

Scenario Issues

The introduced measurement setup refers to the hidden node problem (Figure 1). Another interesting scenario is to simulate WLAN stations which are close to each other, receiving the same interference signals. Therefore, an additional test arrangement was created and analyzed. Figure 3 depicts the setup in detail. It consists of two couplers, two circulators, one attenuator, and a power splitter. The power splitter divides the interference source signal into two paths. Each path is followed by a circulator and a coupler. As before, the coupler provides isolation towards the counterpropagating path to separate the injected interference. Furthermore, the circulators ensure, through their nonreciprocal behavior, that the established traffic between server and client is not passed through the interference source connection network.

It turned out that this realization is impractical. The insertion loss between the interference source and the WLAN stations had an intolerable ripple, which could not be calibrated. The variations within one WLAN channel, at a bandwidth of 20 MHz, led to critical distortions of the interferer power spectral density. Therefore, this approach was discarded and the focus was put on the hidden node scenario.

2.2 Performance Measurements

In order to analyze interference effects, a network performance tool called *iperf* was utilized to establish a data transfer between server and client. *iperf* enables measuring, for instance, the throughput of an IP network using different protocols, such as UDP and TCP. It is possible to modify various parameters in terms of packet size, buffers, delay, protocols, and many more [3]. A performance test works as follows: the *iperf* client is started on both sides, server and client. While the server listens to a specified port for incoming packets, the client addresses the receiver (server) by its IP address. Then, a data stream is sent to the receiver side for one second and values like the throughput and PER are recorded. In order to gain stable performance results, this sequence is repeated ten times within one test cycle. The MTU defines the maximum frame length of an IP network related transmission. Therefore, depending on the packet length, the data to be transmitted is fragmented into frames which fit the MTU best. As the desired packet size is smaller than 1,500 Byte, one transmitted frame consists of several packets. Through resizing data, a block ACK frame is utilized to confirm all valid packets within one reply.

As state-of-the-art modules have plenty of different WLAN standards on board, it is of great interest to have full control over utilized transmission parameters, such as modulation index and transmit power. Indeed, it is possible to control these settings by implementing a specific kernel version from *Candela Technologies* on the respective Linux PC systems [4]. For details about the installation and applications of the whole test setup, refer to the

WLAN setup instruction guide.

Concerning the interference source, the captured baseband data from the measurement campaign is utilized to perturb the communication with a vector signal generator (VSG) during a performance test. Hence, the output of the reference source is a digitally modulated signal, repeating real ISM band traffic. The VSG is started independent of performance measurements and repeats the actual recording endlessly to eliminate timing-related issues. The measurement results of this configuration serve as reference for further verification of different interference sources, discussed in Section 3 and Section 4.

3 Injecting WLAN Frames with Linux

Remembering the results of the measurement campaign, the total amount of classified WLAN frames is more than 90 % for both channels. Consequently, it seems to be natural utilizing another Linux PC system with an implemented WLAN module to emulate ISM band traffic. The main requirements on the desired source are an arbitrary transmit power level and a sufficient timing resolution to fulfill minimum off-times of 1 µs. In the following, two techniques realizing an interference source injecting WLAN frames will be discussed.

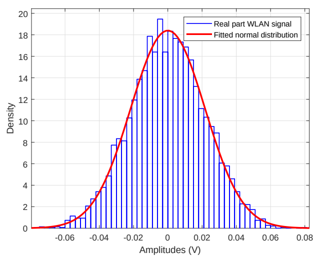
The first technique is based on sockets, enabling IP-based communications by utilizing, for instance, user datagram protocol (UDP) or transmission control protocol (TCP). In order to make use of such higher layer protocols, an association between the two respective nodes, server and interference source, must be established. An association between two WLAN nodes can be understood as connecting a WLAN device to an available access point (server) for entering internet platforms. Examining this working principle of sockets yields a violation of the scenario explained in Section 2. The coupler from Figure 1 provides the same transmission characteristics towards interferer as vice versa. Hence, the interference source will receive some traffic established between server and client during a performance test. Because of collision avoidance schemes, the interferer will stop injecting frames when surrounding traffic is detected. In conclusion, it turned out that the investigated sockets are not sufficient as interference source.

A further approach for interference injection is a Python based program called *Scapy*. It is capable of forming specific WLAN frames and transmitting them independently of any MAC- and PHY-layer constraints. Another important key parameter of *Scapy* is the unrestricted access to MAC-header properties. It is possible to form all kind of sequences, such as management-, control-, and data frames [5]. Since the desired interference source is blind to any surrounding traffic, it is necessary to omit specified transmission rules for medium access. In order to meet these requirements, the operating mode of the respective WLAN module must be changed. The monitor mode offers the opportunity to demodulate all received WLAN packets passively and store their transmission properties. As its name implies, this mode is designed for demodulation only and no rules regarding a valid data exchange have to be fulfilled. Unfortunately, PHY properties cannot be changed. The utilized modulations stick to legacy rates which are 1 Mbit/s (DSSS) regarding the 2.4 GHz band and 6 Mbit/s (OFDM) for 5 GHz-related channels.

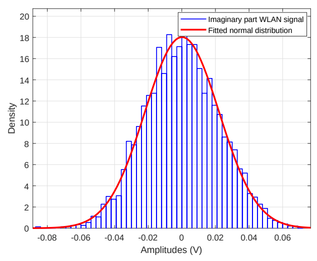
With *Scapy*, a promising interference source has been found, replacing expensive RF equipment, like a VSG. Nonetheless, several issues have been identified. It is not possible to change transmission parameters like modulation and data rate. Although the monitor mode makes use of OFDM in the 5 GHz band, another issue makes this source impractical. As already mentioned, *Scapy* is a Python based program, which is an interpreted, high-level programming language. Unlike procedural languages such as C, Python induces latency effects when it comes to time-critical applications. Linux operating systems offer priority levels for executed scripts, but the occurred timing variations are still too large for the highest priority level. Sockets were also examined regarding timing properties. Since the respective scripts are written in C and the required timing resolution could also not be reached, it is obvious that the problem is caused by the operating system. Furthermore, the actual available firmware and driver of the utilized WLAN module do not support a dynamic regulation of the output power. Thanks to the kernel and adapted firmware from *Candela Technologies*, it is possible to set the power level, but just initially for one single time. However, the dynamic range, which could be achieved (20 dB) is not sufficient to emulate the required power level distribution ~ 60 dB. Because of the mentioned aspects, a different approach for realizing a low-cost interference source must be found.

4 Modulated Noise

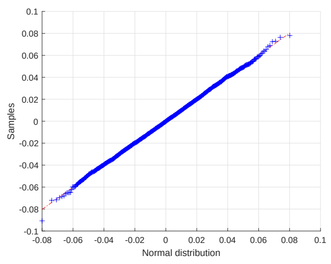
OFDM signals have a similar amplitude distribution compared to white Gaussian noise. In order to prove this statement, an OFDM modulated WLAN frame, captured by the measurement campaign in channel 36, is investigated. Figure 4 depicts the corresponding densities of real- and imaginary parts of the sampled amplitudes. The mean and variance of the observed amplitudes have been calculated to fit a normal distribution to the data. The evaluated distributions are shown in a quantile-quantile (Q-Q) plot to compare the OFDM amplitudes to a Gaussian density function. The Q-Q plots appear to be linear, which means that the observed samples fit a Gaussian distribution function well. In addition, they are uncorrelated and zero-mean. The imaginary- and real parts can be assumed jointly Gaussian and, therefore, independently distributed.



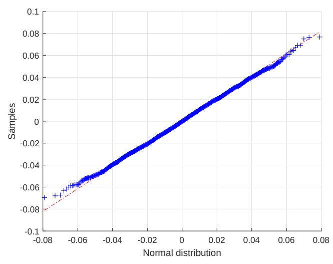
(a) Histogram: real part



(b) Histogram: imaginary part



(c) Q-Q plot: real part



(d) Q-Q plot: imaginary part

Figure 4: Amplitudes of a recorded OFDM WLAN frame: histograms and corresponding Q-Q plots of real and imaginary part

Hence, it is possible to create an interference source by utilizing white Gaussian noise. Instead of injecting WLAN frames with a Linux PC system, noise will be modulated in terms of mean power and burst length, according to the defined random variables from the measurement campaign. As the 2.4 GHz ISM band is also utilized by other communication standards, e.g., IEEE 802.11b (DSSS) and BLE, it will be examined in the following if noise can be used as an interference source, yielding the same performance test results as ISM band signals.

4.1 Simulation Model

In the following, a simulation model based on the IEEE 802.11a standard, in presence of different kind of interference signals, will be investigated. OFDM splits a symbol sequence into $K = 64$ orthogonal parallel data streams and passes them through an inverse fast Fourier transform (IFFT). Figure 5 depicts the subcarrier mapping of the considered communication standard. Binary data is mapped onto a symbol constellation (BPSK, QAM, etc.) and converted into OFDM symbols consisting of 64 subcarriers. The subcarrier mapping shows that an amount of 48 bins is reserved for payload data (deep blue), four are pilots (green), and the remaining 11 subcarriers (cyan) are utilized for oversampling as guard band. Furthermore, the DC carrier in the middle (grey) is not used for data transmission. The OFDM symbols are then passed through the IFFT and at last a cyclic prefix (CP) of 0.8 µs is added.

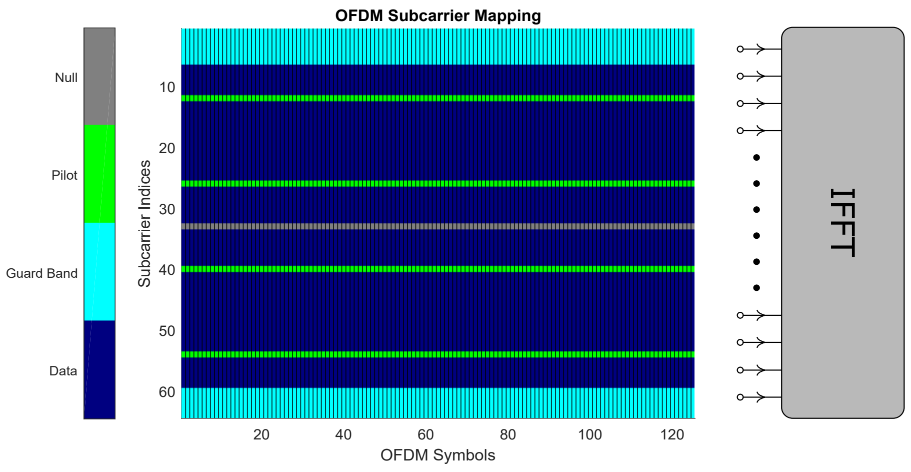


Figure 5: Subcarrier mapping according to IEEE 802.11a

Due to the repeated structure of the preamble, it is possible to detect packets and proceed with further operations, such as channel estimation and frequency offset correction. The short preamble consists of ten repeated sequences with a total length of 8 µs. Utilizing the metric invented by Schmidl and Cox [2], coarse frequency offset estimation and packet detection can be realized. Equation 1 defines the required timing metric $M(d)$ for packet detection, the frequency offset relevant function $P(d)$, and the signal energy $R(d)$. Due to implementing a sliding autocorrelation window with a size of L samples, the described metric is independent of absolute received amplitudes of r. As one short training sequence is 0.8 µs long and assuming a sample rate of $f_s = 20\,\text{MSa/s}$, the correlation window consists of $L = 16$ samples.

$$P(d) = \sum_{m=0}^{L-1} (r^*_{d+m} r_{d+m+L}), \qquad R(d) = \sum_{m=0}^{L-1} |r_{d+m+L}|^2, \qquad M(d) = \frac{|P(d)|^2}{R(d)^2} \tag{1}$$

Through function $P(d)$, it is possible to calculate a coarse frequency offset estimate. According to Equation 2, a shift in frequency domain (δ_k) relates to a multiplication with an exponential function in time domain.

$$e^{j\frac{2\pi\delta_k}{N}n} x_n \quad \circ\!\!-\!\!\bullet \quad X_{k-\delta_k} \tag{2}$$

Furthermore, the autocorrelation function yields an output equal to one within the short training field. Assuming a signal model of $r_i = x_i e^{j\frac{2\pi\delta_k}{N}i}$, the argument of function $P(d)$ can be calculated ($N = N_{\text{fft}} = 64$):

$$\begin{aligned}
P(d) &= \sum_{m=0}^{L-1} x^*_{d+m} e^{-j\frac{2\pi\delta_k}{N}(d+m)} x_{d+m+L} e^{j\frac{2\pi\delta_k}{N}(d+m+L)} \\
&= \sum_{m=0}^{L-1} x^*_{d+m} x_{d+m+L} e^{j\frac{2\pi\delta_k}{N}L} \\
&= \sum_{m=0}^{L-1} e^{j\frac{2\pi\delta_k}{N}L}, \\
\delta_k &= \frac{N}{2\pi L}\arg\left(P(d)\right).
\end{aligned} \tag{3}$$

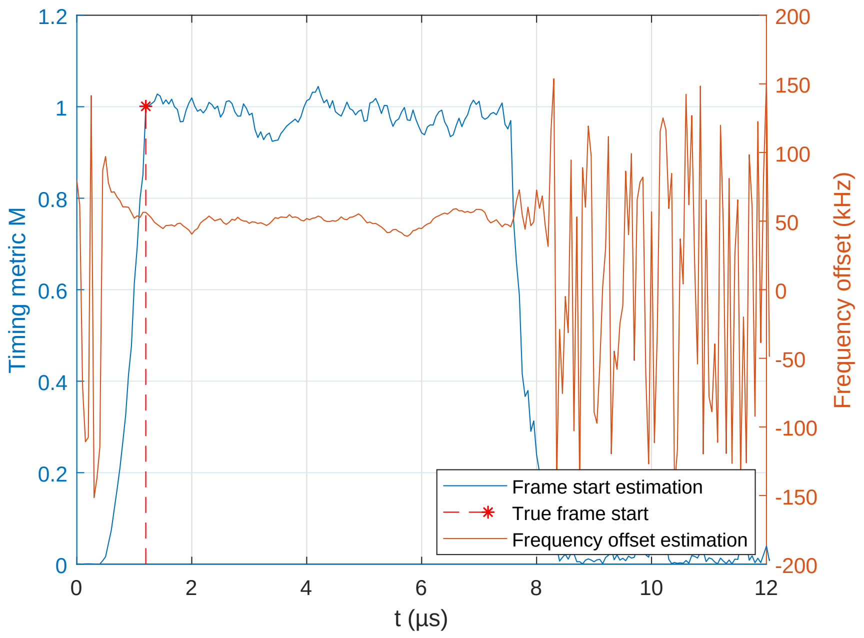


Figure 6: Coarse packet detection and frequency offset estimation: SNR = 20 dB, $f_{\text{offset}} = 50\,\text{kHz}$, $t_{\text{offset}} = 1.2\,\mu\text{s}$

Figure 6 depicts the respective output of timing metric $M(d)$ and frequency offset $f_{\text{offset}} = \delta_k \frac{f_s}{N_{\text{fft}}}$ for the following example: The investigated frame starts after 1.2 µs and has a frequency offset of 50 kHz. Furthermore, AWGN has been added with an SNR of 20 dB.

As the correlation window is just 16 samples long, the frequency offset estimation suffers from inaccuracies. Hence, the mean value is taken into account over a stable time interval. In order to find the beginning of this time interval, the timing metric $M(d)$ is utilized. The coarse detection settings have been set as follows: If more than 16 samples in a row are detected beyond a defined threshold of $M > 0.5$, the beginning of a frame is determined. Since more than two short preamble sequences get lost with this technique (autocorrelation window has a delay of L samples), a stable time of 6.4 µs remains. In order to get useful results also for low-SNR scenarios, a stable time of 2.4 µs is defined. The next step is a fine timing synchronization. Utilizing an matched filter (MF) detection concerning the long training field, correlation peaks appear after the convolution. If two maxima are separated by 3.2 µs, the beginning of the frame is defined precisely. Subsequently, the long training symbols are also used for channel estimation. After synchronizing the received frame in time- and frequency domain, the CP is removed and the fast Fourier transform (FFT) is applied for demodulation. After the FFT, the outcoming OFDM symbols are in frequency domain. Therefore, channel estimation reduces to a simple division of the respective subcarriers (k) with the known preamble. Equation 4 yields a mathematical approach by simply dividing the received long training symbols $r_{k,n}$ by the true reference symbols $x_{k,n}$. As two long training fields exist (n), the mean of two channel estimates is taken into account ($n = 1..2$, $k = 1..48$).

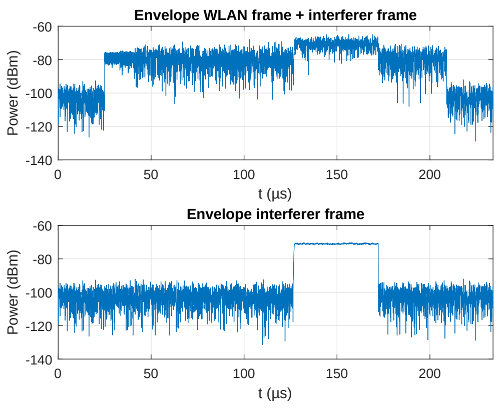
$$\hat{H}_{k,n} = \frac{r_{k,n}}{x_{k,n}} = H_{k,n} + \frac{n_{k,n}}{x_{k,n}} \qquad \hat{H}_k = \frac{1}{2}(\hat{H}_{k,1} + \hat{H}_{k,2}) \tag{4}$$

The frequency offset estimation function of Figure 6 clearly depicts the problem of inaccuracies, caused by the short autocorrelation window. Consequently, the pilots (Figure 5), corrected by initial channel estimation, are investigated for further frequency offset correction. The rotation of the received pilots $\hat{p}_{k,n}$ around the true pilots $p_{k,n}$ is estimated with equation 5. The mean value e_n, is then multiplied with the respective OFDM symbol (n) [1].

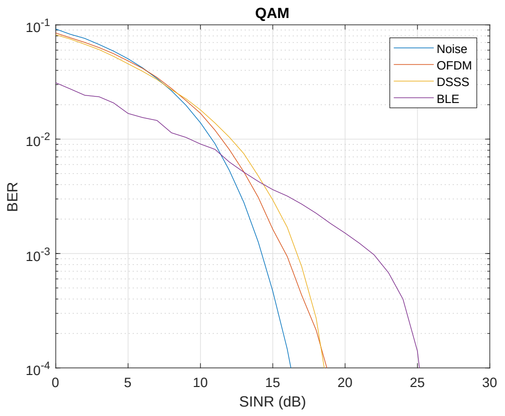
$$e_n = \frac{1}{N_p} \sum_{k=0}^{N_p-1} p_{k,n}\hat{p}^*_{k,n} \tag{5}$$

In the following, the explained techniques will be applied to create a WLAN simulation model. Furthermore, three different communications standards (WLAN DSSS, OFDM, and BLE) will be compared to noise bursts as interferer. Figure 7 depicts the actual interference scenario in subfigure (a) and the respective BER curves (b)–(d) for different constellation mappings. The sent WLAN frame consists of the legacy preamble and payload data according to the MTU size of 1,500 Byte. In order to create realistic conditions, an equally distributed random time delay and frequency offset is applied. Furthermore, an interference signal of additive white Gaussian noise (AWGN) and a certain communication standard are added to the desired WLAN frame. The utilized interfering standard has an SNR of 30 dB, a quarter length of the transmitted WLAN frame, and changes the position anew, for every transmission. Regarding BLE as interferer, an equally distributed frequency hopping scheme, varying the center-frequency for every frame, is implemented. It must be mentioned, that the interfering standards were taken from the captured ISM band data to maintain effects like peak-to-average power ratio (PAPR) reduction.

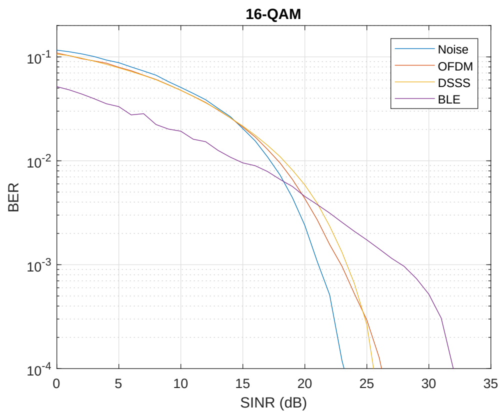
Despite sweeping the SNR for BER curves like in an AWGN scenario, the signal to interference plus noise ratio (SINR) is investigated. Hence, the mean power of the whole interferer- and WLAN frame is divided to gain the respective SINR. Ensuring an appropriate packet detection and initial channel estimation, the utilized interfering standard has an additional time offset to avoid an overlap with the WLAN preamble. Consequently, only the payload data suffers from perturbations caused by additionally induced interfering signals.



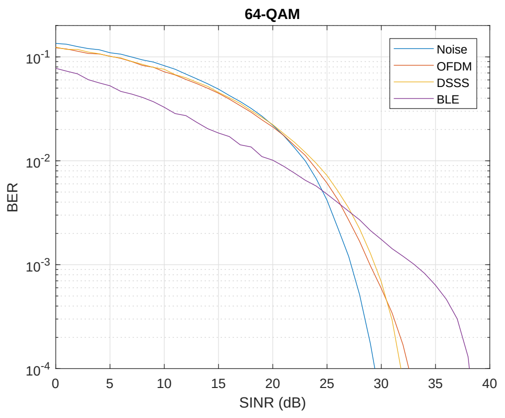
(a) Interference scenario: WLAN frame plus interference (top), AWGN plus interferer (BLE) (bottom)



(b) BER curve: QAM



(c) BER curve: 16-QAM



(d) BER curve: 64-QAM

Figure 7: (a) Current interference scenario: SINR of 0 dB and BLE as interferer, (b)–(d) BER curves over SINR for different modulations and interference sources (Noise, OFDM, DSSS, and BLE)

Obviously, all BER curves (Figure 7) show a similar behavior regarding the interference source. Noise, WLAN DSSS, and OFDM result in BER curves which are close together. For higher QAM orders, the respective curves get even closer. Only distortions caused by BLE differ from noise completely. According to the observed simulation, it is possible to utilize modulated noise instead of digital data, such as WLAN DSSS and OFDM. Unfortunately, BLE cannot be described through this approach. Considering results from measurement campaign, the amount of this interferer is small. Hence, modulated noise is a promising technique for emulating ISM band traffic.

4.2 Implementation of a Low-Cost Interference Source

A low-cost interference source, modeling noise by a digitally-controlled output power level, has been realized [6]. This was done by using off-the-shelf components, such as noise diodes, power amplifiers, and digitally-controlled step attenuators (Figure 8).

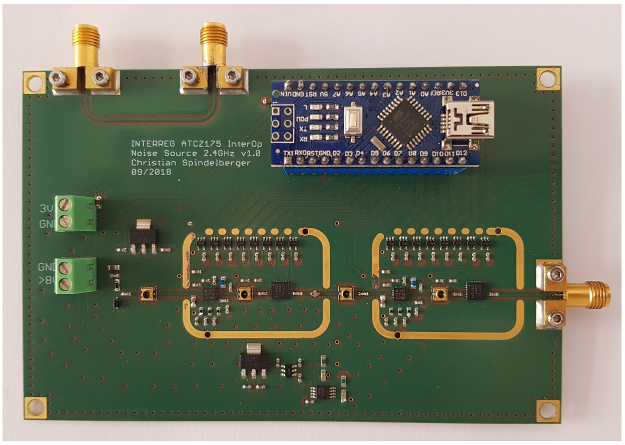


Figure 8: Low-cost interference source

The main system requirements are a high dynamic range DR $> 50\,\text{dB}$ and the opportunity to realize off-times down to $t_{\text{min}} = 1\,\mu\text{s}$. Furthermore, the power level must be large enough to overcome the insertion loss of the coupler, depicted in Figure 1. In order to meet these requirements, an overall gain of $\sim 100\,\text{dB}$, using one noise diode and two power amplifiers, is achieved. Figure 9 depicts the block diagram of the realized source. State-of-the-art noise sources utilize silicon avalanche diodes. Because of the avalanche effect, noise diodes are capable of enhancing the noise floor up to $30\,\text{dB}$ for a wide frequency range ($1\,\text{GHz}...18\,\text{GHz}$). To describe the gain of such diodes, the excess noise ratio (ENR) is investigated. For instance, an ENR of $30\,\text{dB}$ establishes a noise power density of $N_p = -174\,\text{dBm/Hz} + 30\,\text{dB} = -144\,\text{dBm/Hz}$ at room temperature $T_0 = 290\,\text{K}$.

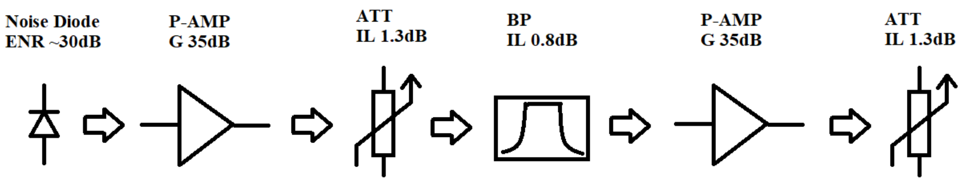


Figure 9: Low-cost interference source: block diagram [6]

In order to control the output power level, two digitally-controlled step attenuators, yielding a dynamic range of 63.5 dB, are implemented. A 14 bit broad parallel interface, controlled by a microcontroller, enables an output power resolution of 0.25 dB. A bandpass filter, optimized for WLAN applications in the 2.4 GHz band, is implemented to ensure a linear operation of the second power amplifier.

Arduino provides a simple structured integrated development environment (IDE) for prototyping applications. The observed noise source uses a serial interface to communicate over USB. Hence, a data sequence, realized by the respective random variables can be streamed to the microcontroller. Unfortunately, the memory depth is not high enough to generate sequences of the same length as a VSG. Therefore, it will be examined if this issue yields different performance results. In addition to this, the PSD of the noise source is optimized for the 2.4 GHz band. For WLAN channel 1, a maximum output power level of −2.67 dBm at a bandwidth of 20 MHz is achieved. Concerning the 5 GHz band, the output power is not high enough to overcome the insertion loss induced by the coupler. Consequently, only channel 1 at 2.412 GHz will be investigated.

References

[1] Travis F. Collins, Robin Getz, Di Pu, Alexander M. Wyglinski. *Software-Defined Radio for Engineers.* Artech House, 2018. 11

[2] Timothy M. Schmidl, Donald C. Cox. *Robust Frequency and Timing Synchronization for OFDM.* IEEE Trans. Commun., Vol 45, No. 12, December 1997. 9

[3] Jon Dugan, Seth Elliott, Bruce A. Mah, Jeff Poskanzer, Kaustubh Prabhu. *iPerf - The ultimate speed test tool for TCP, UDP and SCTP.* `https://iperf.fr/` 5

[4] Candela Technologies. `https://www.candelatech.com/` 5

[5] Philippe Biondi. *Scapy documentation.* `https://scapy.readthedocs.io/en/latest/` 6

[6] Christian Spindelberger BSc. *Noise source with a digitally-stepped output power level*, 2018. `http://www.interreg-interop.eu/results/wlan_interference_analysis/low_cost_interference_emulator/` 13, 14