Radio interference analysis tool based on GNU Radio

J. A. Yébenes Gálvez$^{(1)}$, E. Martos Naya$^{(1)}$, M. C. Aguayo Torres$^{(1)}$, C. Cárdenas Angelat$^{(2)}$, J. Baños Polglase$^{(2)}$.
jayg@ic.uma.es, eduardo@ic.uma.es, aguayo@ic.uma.es, carlos.cardenas@dekra.com, janie.banos@dekra.com
$^{(1)}$Dept. of Communication Engineering, Universidad de Málaga.
$^{(2)}$DEKRA Testing and Certification, S.A.U. Málaga

***Abstract*—Avoiding interference is one of the main challenges in radio communications. Interference hunting is usually done with costly instruments. In this work, a cost efficient portable tool is described for interference analysis that is based on software defined radio. A generic radio board implements the RF front end, while flexible signal processing is carried out on a personal computer. The proposed tool analyses spectrum characteristics spanning from 70 MHz to 6 GHz band, detects radio interference signals and helps to identify the type of radio technology used by the source transmitter.**

***Index Terms*—GNU radio, spectrum analyser, SDR, energy detector, interference**

I. Introduction

In the last years, radio spectrum is becoming saturated due to the proliferation of devices using radio technologies such as Wi-Fi®, WiMAX™, 3GPP standards (GSM, UMTS, LTE, 5G), Bluetooth®, Zigbee, satellite links, etc. The spectrum of these technologies shows specific characteristics and patterns. For example, Wi-Fi 802.11n [1] transmits periodic beacons with a typical bandwidth of 20MHz, uses Orthogonal Frequency Division Modulation (OFDM) and, on each OFDM symbol, four subcarriers are dedicated to pilot signals. These characteristics give rise to a recognizable spectrum aspect.

A number of technologies simultaneously working in an area produces interference among them that avoid their working properly. During network planning, as well as during optimising and troubleshooting phases, it is useful identifying whether a specific technology is present in a location at a certain moment. Commercially known as "interference hunters", several models are available in the market (e.g., Wi-Spy [2], RF-Catcher [3] or TSMx6 [4]). However, the number of technologies they might capture is limited, as they are constrained to a specific frequency band and bandwidth.

The task of analysing the spectrum to detect active interference sources has been widely investigated in the context of cognitive radios (see, e.g., [5], [6]). Cognitive radios require detecting the presence of a primary transmission but identifying the specific interference source characteristics is not needed. It can be found in literature works that performs blind modulation recognition by classical algorithms [7] or, more recently, helped by machine learning tools [8]. However, interference detection requires identifying the technology (e.g., Bluetooth, Wi-Fi or LTE) and not the specific modulation currently used by that technology.

In this work, we describe an interference hunting tool implemented as a Software Defined Radio (SDR) system which is based on the open source radio software toolkit known as GNU Radio. By measuring the radio spectrum within a certain time interval and bandwidth, the tool analyses the spectrum occupation. By different configuration parameter sets, diverse radio technologies can be detected and identified. Therefore, the proposed tool is highly flexible and suitable for the analysis of any wireless technology.

In the remainder of this paper, we briefly introduce SDR and GNU Radio. Later, we give details about the proposed system architecture and algorithms. Some field test results using the implemented tool are described below. Finally, we conclude the work and give some hints for further work.

II. SDR and GNU Radio

GNU Radio is an open source software toolkit which consists of libraries for signal processing blocks besides the "glue" needed to tie these blocks together [9]. To build and deploy a software radio, a hardware radio module implements the RF front end over a radio board while the radio processing components are implemented as software on a personal computer.

GNU Radio applications are easily built using a graphical interface known as GNU Radio Companion. Also, Python programming language can be used with this purpose. Those blocks which require high effectiveness are implemented in C++ language, as it is the case of libraries. Thanks to the easy reuse of existing GNU Radio blocks, high performance radio systems can be developed in a very effective way.

Reconfigurability is a key feature in software-defined radio systems. A wide set of generic radios with different characteristics (central frequency band, maximum signal bandwidth, etc.) are available in the market [10]. The same processing program can be easily adapted to control different RF front end radio boards by simply changing the "source" block in the GNU Radio application to that provided by the manufacturer.

III. System Architecture

Figure 1 depicts the main modules and interfaces of the interference analysis tool. Three differentiated modules can be identified: hardware (radio board), processing and visualisation. A set of configuration parameters is given as entrance to them.

The hardware module is responsible for receiving the electromagnetic signal and sending it as radio signal records to the processing module. The processing module computes some spectrum metrics described later. Then, it stores the results in system files that the visualization module, developed in Python, transforms into understandable graphs and reports.

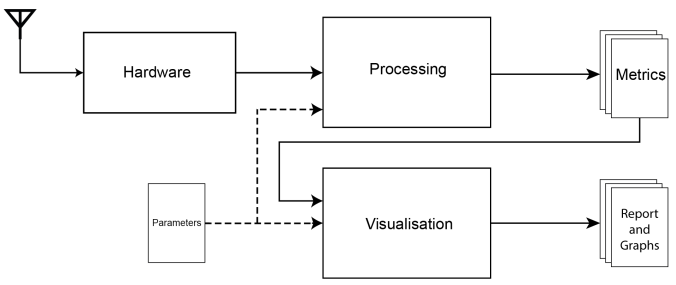


Figure 1. Modules forming the interference analysis tool architecture

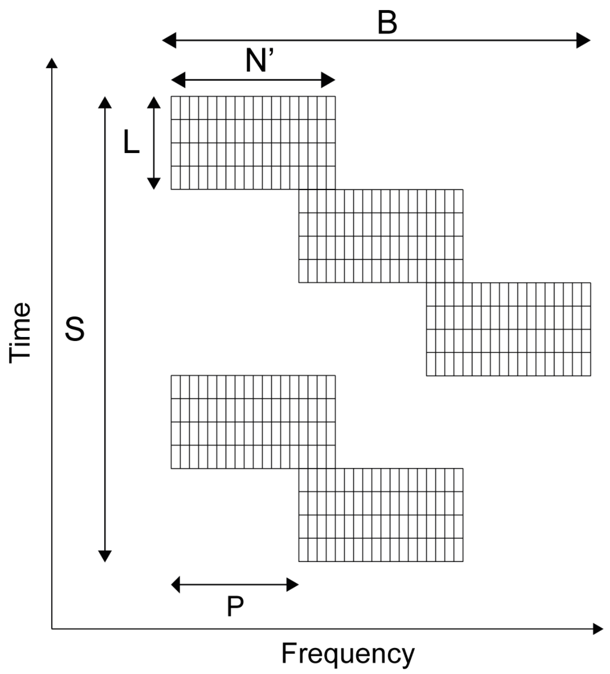


Figure 2. Representation of a capture for metric evaluation

The processing module, implemented on the GNU Radio stack, is the core of the tool. In summary, processing is divided into three parts:

1) The data source gets the input to the system from the hardware board.
2) The signal is transformed by a Fast Fourier Transform (FFT) from the time domain into the frequency domain, where different technologies are more easily recognised.
3) Finally, analysis is carried out over the frequency domain samples and results are stored in files to be processed by the visualisation module.

The analysis of a specific bandwidth is carried out during T seconds divided into S frames of duration T_F each. Each frame corresponds to M consecutive spectrum measurements over a bandwidth W at the center frequency f_o. To perform each spectrum measurement, an N-sized FFT is taken over the signal sampled at $f_s = W$ Hz. Those M measurements are summarised in a single frequency vector by metrics which are described in the next section.

The maximum bandwidth which may be analyzed with a single FFT depends on the radio board available (through its maximum sampling frequency). If a greater bandwidth needs to be analysed, which is often the case, several subbands are sounded by varying the center frequency at the generic radio board during the execution in steps of $f = W \ \frac{P}{N}$ Hz. The same central frequency is kept during L frames and a sweep is later taken on frequency as shown in Figure 2. As there is certain delay between the order to change the center frequency and its application, the radio board should be able to indicate that the change has been executed. Note that the M spectrum measurements in which each frame is further divided are not shown in the figure.

Carried out in this way, the interference analysis is not continuously taken over the whole bandwidth. In order not to miss certain interference patterns (e.g. periodic beacons in WiFi), $t = T_F \ L$ has to be carefully adjusted to capture the expected interference patterns, and several measurements should be taken at each frequency subband. Moreover, the computer available to run the GNU Radio programs might influence the selection for the frame duration (T_F) and the FFT size (N): for high N or low T_F values, processing time increases and it may happen that samples are lost due to computer limitations.

Figure 3 shows the block diagram of the processing module as implemented in the GNU Radio toolkit by the GNU Companion interface. The three component parts are identified by coloured squares surrounding their component blocks. The hardware block (orange square) is configured by external parameters in order to easily adapt the system usage for analysing different radio technologies. This would be the only block to be replaced in order to use a different hardware module. The blue part performing the spectrum analysis is hardware independent. In the next section, the evaluated metrics (green square) are described in detail.

IV. Evaluated metrics

The data stream output from the generic board is formed by samples $x[n]$ of the baseband complex signal measured at certain center frequency f_o with selected bandwidth W and normalized gain G between 0 and 1. An FFT is now applied to N consecutive samples, corresponding to N frequency samples over the bandwidth W Hz. In order to have a better resolution, a Hann window $h[n]$ is first applied to the signal. Thus, the $m \ th$ discrete Fourier transform ($0 \ m \ S \ M$) at the frequency sample k ($0 \ k \ N$), $X[k;m]$, is obtained as:

$$X[k;m] = FFT_N \{x[n + N \ m] \ h[n]; 0 \ n < N\}. \quad (1)$$

As the presence of high signal power in adjacent frequencies might result in aliasing, edge measurements are ignored, reducing the useful frequency samples from N to N'.

Three different metrics are taken over the M spectrum measurements: average, maximum and percentage of use.

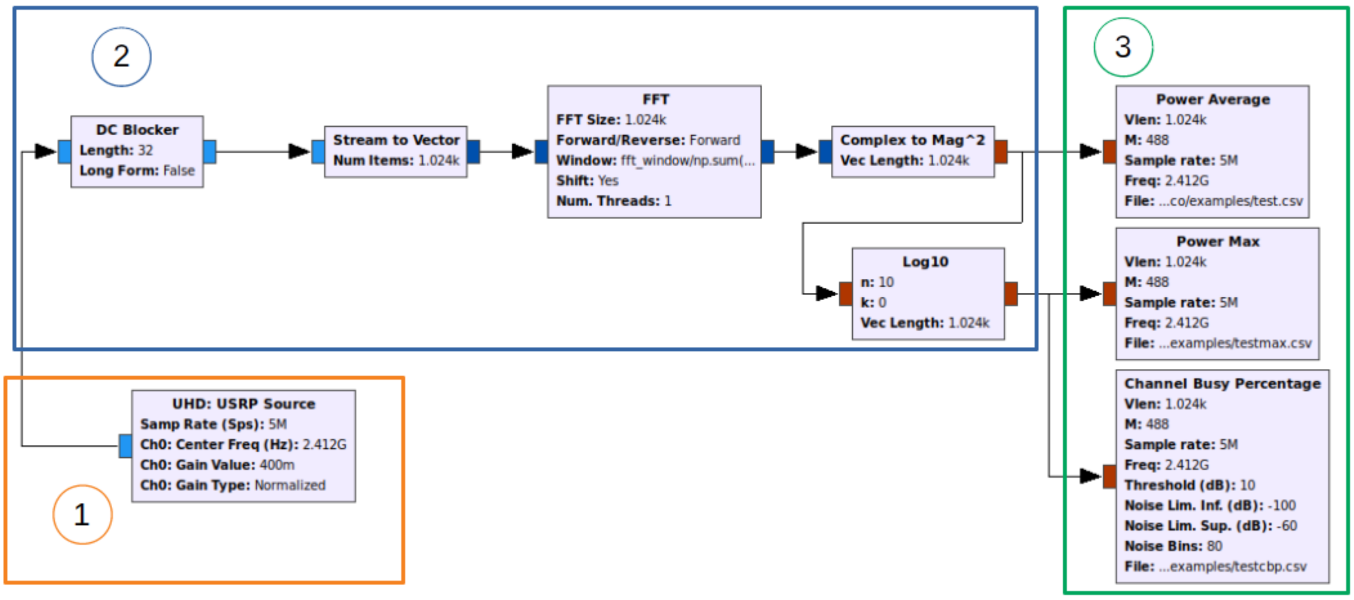
The average metric, M^{Avg}, at frequency k is the logarithmic value of M samples averaged over the l^{th} frame,

$$M^{Avg}[k;l] = 10 \ \log_{10} \frac{\sum_{m=0}^{M} \|X[k;m + M \ l]\|^2}{M} \quad (2)$$

The *maximum* metric, M^{Max}, takes the maximum value instead:

$$M^{Max}[k;l] = \max_{m=0}^{M} \{10 \ \log_{10}(\|X[k;m + M \ l]\|^2)\} \quad (3)$$

We also evaluate the *percentage of use* metric, M^{Per}, i.e., the percentage of time for which received signal power is over a



DC Blocker
Length: 32
Long Form: False

Stream to Vector
Num Items: 1.024k

FFT
FFT Size: 1.024k
Forward/Reverse: Forward
Window: fft_window/np.sum(...
Shift: Yes
Num. Threads: 1

Complex to Mag^2
Vec Length: 1.024k

Log10
n: 10
k: 0
Vec Length: 1.024k

Power Average
Vlen: 1.024k
M: 488
Sample rate: 5M
Freq: 2.412G
File: ...co/examples/test.csv

Power Max
Vlen: 1.024k
M: 488
Sample rate: 5M
Freq: 2.412G
File: ...examples/testmax.csv

Channel Busy Percentage
Vlen: 1.024k
M: 488
Sample rate: 5M
Freq: 2.412G
Threshold (dB): 10
Noise Lim. Inf. (dB): -100
Noise Lim. Sup. (dB): -60
Noise Bins: 80
File: ...examples/testcbp.csv

UHD: USRP Source
Samp Rate (Sps): 5M
Ch0: Center Freq (Hz): 2.412G
Ch0: Gain Value: 400m
Ch0: Gain Type: Normalized

Figure 3. Block diagram in GNURadio

threshold THR. In order to determine that threshold, we first evaluate the noise floor power (NF) and establish a signal to noise threshold (SNT) over it to identify if a signal (and not only noise) is present, that is,

$$M^{Per}[k;l] = \frac{1}{M}\sum_{m=0}^{M} \text{sgn}\left(10\log_{10}\|X[k;m+M\cdot l]\|^2 - THR\right) \quad (4)$$

where $sgn(x)$ is defined as 1 for $x \geq 0$ or 0 in other case.

The three different metrics taken on the received power spectrum within one frame provide complementary information:

- A signal shorter than the frame duration gives a very low mean value in the frame, while the maximum would be easily observed.
- Mean and maximum within the frame duration would be similar for continuous transmissions (e.g., cellular transmission with high load).
- Mean and maximum would be quite different if there is more than one interference source with different powers.
- A high percentage of use but low mean and maximum would be produced by a low power continuous.

V. Field Testing

A set of tests has been carried out using the components listed in Table I. The available board, USRP B210, is able to capture signals up to 56 MHz wide with center frequency from 70 MHz to 6 GHz.

The value of the input parameters is first set in accordance with the target radio technology which is expected to be found: adequate configuration regarding frame duration, bandwidth, etc., is different depending on if it is more likely that, for example, Wi-Fi or Bluetooth signals are present. In order to validate the selected configuration, measurements were taken inside an anechoic chamber, ensuring RF isolation from the environment. A Wi-Fi access point (AP) and the generic radio board were placed inside the chamber.

According to the Wi-Fi standard, access points send beacons periodically, usually every 100 ms. The parameters for this analysis are shown in Table II-Test 1. The average metric measured with the tool when the AP is transmitting beacons and there are no clients active is shown in Figure 4. Beacons are clearly visible as darker lines every 100 ms.

In the experiment shown in Figure 5 (carried out outside the anechoic chamber), there were a set of Wi-Fi access points with connected devices. A new set of parameters was configured for this test, given in Table II-Test 2.

The percentage of spectrum use when, in the previous scenario, a Bluetooth headset was also continuously transmitting

Table I
Overview of components of the test system

Component	Type
CPU	Intel I5 3320M
Operating System	Ubuntu 16.04.2 LTS,64 bit
GNU Radio	v3.7.13.4 Release
SDR	USRP B210 SDR Kit
Antenna	2.4GHz Antenna
Anechoic chamber (when needed)	460 x 600 x 600 mm

Table II
Parameters for captures

Parameter	Test 1	Test 2	Test 3	Units
Measurement duration, T_M	1	1	60	s
Center frequency, f_o	5700	2412	5660	MHz
Gain (normalized), G	0.4	0.4	0.4	
Measurement bandwidth, W	5	10	5	MHz
Frequency resolution, W/N	378.125	39.0625	78.125	kHz
Frame duration, T_F	1	1	100	ms
S/N threshold, SNT	10	10	10	dB
Sweep	Off	Off	On	Bool
Frequency step, Δf	-	-	2.5	MHz
Time step, Δt	-	-	1	s
Frequency end, f_e	-	-	5740	MHz