This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Poveda, Héctor; Navarro, Kiara; Merchan, Fernando; Ramos, Edgar; González González, David

A Software Defined Radio-Based Prototype for Wireless Metrics Studies in IoT Applications

Published in:
Wireless Personal Communications

DOI:
[10.1007/s11277-021-08281-x](https://doi.org/10.1007/s11277-021-08281-x)

Published: 01/10/2021

Document Version
Peer reviewed version

Please cite the original version:
Poveda, H., Navarro, K., Merchan, F., Ramos, E., & González González, D. (2021). A Software Defined Radio-Based Prototype for Wireless Metrics Studies in IoT Applications. *Wireless Personal Communications*, 120(3), 2291-2306. <https://doi.org/10.1007/s11277-021-08281-x>

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

A Software Defined Radio-based Prototype for Wireless Metrics Studies in IoT Applications

Héctor Poveda · Kiara Navarro ·
Fernando Merchan · Edgar Ramos ·
David González

Received: date / Accepted: date

Abstract The spectrum has always been an essential resource of information for wireless communications. With the continued growth of Internet of things (IoT) and 5G, there is a demand to understand how the spectrum is use. One of the challenges of deploying IoT applications is the crowded spectrum in the unlicensed industrial scientific medical bands leading to rising coexistence problems between different wireless protocols. To overcome this congestion, hardware tools supporting spectrum sensing can be used to manage the spectrum more efficiently. In this context, this work presents a prototype that measures a set wireless metrics on raw wireless signals acquired with software defined radio (SDR) technology. This prototype aims to provide mechanisms to sense and monitor spectrum usage that can mitigate one of the issues that IoT faces, the interference being produced by having different technologies using at the same frequency channels. The prototype features configurable radio frequency parameter and programmable periodical tasks execution. It displays wireless metrics such as signal to noise ratio, cumulative density function and power spectral density. This prototype uses web and SDR technologies, highlighting the idea and feasibility of combining these two technologies. In addition, it demonstrates the possibility to obtain wireless metrics with a low-cost

This work is partially supported by the National Bureau of Science, Technology and Innovation of Panama through the project “RAPIDO-5G” (ITE15-021). The National Research System of Panama supports research activities of H. Poveda (SNI 153-2017) and F.Merchan (SNI 17-2020).

H. Poveda, K. Navarro, F. Merchan
Universidad Tecnológica de Panamá, Panama
E-mail: {hector.poveda, kiara.navarro, fernando.merchan}@utp.ac.pa

E. Ramos
Aalto University, Finland
E-mail: edgar.ramos@aalto.fi

D. González
Continental AG, Germany
E-mail: david.gonzalez.gonzalez@continental-corporation.com

hardware based on open source tools in a platform where interaction, debugging and maintaining becomes intuitive and easier. Results of measurements of LoRa protocol signals are presented to demonstrate the capabilities of the prototype.

Keywords IoT · 5G · Interference Characterization · Software Defined Radio · Wireless Communications · Beaglebone Black

1 Introduction

Software defined radio (SDR) has provided a powerful paradigm to explore possibilities with the physical layer that would not been imagined years ago [17]. These devices, being a general hardware without preconceived notion of purpose, allow designers, researches and engineers to rapidly build and test new physical layer protocols. The use cases are vast and bring huge benefits to the academic community and the industry. It is not unknown that SDR encompass several areas ranging from: programming, electronics, communications, hardware, and security. Their versatility make them useful for several applications. However, SDRs devices share drawbacks and challenges to overcome, *e.g.* a learning curve, proprietary software and hardware.

One of the main challenges that face in deploying Internet of things (IoT) [24] applications is the congested spectrum in the unlicensed industrial scientific and medical (ISM) bands [5] leading to rising coexistence issues in assorted wireless protocols. To mitigate this, getting insights rather than information about the spectrum is an approach considered in this work. For this purpose, this work proposes the architecture, design and implementation of a low-cost, small-size, reprogrammable, configurable, and portable prototype composed by a single board computer (SBC) and SDR that senses the spectrum and provides wireless metrics such signal to noise ratio (SNR), cumulative density function (CDF) and power spectral density for frequency bands selected by the user. This information is displayed in a web interface.

To the best knowledge of the authors, there are very few prototypes, not to mention none in the academic literature that presents the implementation a web interface to communicate with a SDR platform to adjust its parameters. Even when SDR it is a well-known topic, most of the research areas are focused in implementation techniques to work with GNU Radio [8], and the digital signal processing framework to process electromagnetic signals. The integration of web technologies with SDR, bring to these systems advantages for online configuration and tasks programming.

There have been an interest in the academic community for the use the SDR in several applications. An overview focused on providing research purposes solutions based on SDR is presented in [26]. The authors use GNUradio to operate over the universal software radio peripheral board and use several implementations based on third-party modules covering widespread wireless protocols such as Bluetooth and IEEE 802.11. The main contribution from this report is that the use of these types of SDR frameworks improve the support

to researchers to study the different protocols at physical and datalink layers without having work with proprietary firmware to achieve the same results. In [25], authors present concepts related to telecommunications and signal processing studies beyond than mathematical equations through the interaction between SDR and *Python*. A low cost experimental kit is presented using *Python* as the programming tool for processing samples and the RTL-SDR dongle as hardware. The results show that it is possible to setup cost efficient experimental testbeds using SDRs with just enough properties to make it feasible in a classroom, encouraging students to be interested in those types of topics.

A design and implementation of a testbed for smart and cognitive radio research was presented in [23]. In this work, the user interface orchestrates the entire platform and PSD sensor component to detect a signal within a pre-defined band. It is demonstrated the possibility to design an user interface that connects with underlying hardware component such as SDR as well as clever algorithms with a high level programming language as *Python* to find solutions that may overcome a widespread of challenges within wireless research.

Usually, fellow researchers working in wireless communications needs to analyze captured data from the spectrum by using some sort of post-processing algorithms on a host machine. The prototype presented in this work offers a solution that keep costs low by using turnkey components that can be connected at a hardware level as a plug and play vision. Instead of creating everything from scratch, the idea is to abstracting the interaction between the SDR platform and the SBC. This will allow researchers, peers and colleagues to easily replicate the same equipment configuration to develop, debug and validate their concepts in their own target scenarios.

The main features of the prototype proposed can be summarized as following:

- **Mobility capability:** It is a compact portable prototype. The results can be stored in the device or accessed online as long as the prototype is connected to the internet (e.g. via WiFi or ethernet).
- **RF sensor:** A bladeRF is used to acquire the raw samples from the spectrum. This is a SDR card that senses the electromagnetic waves within certain span of frequency and pass that data as a 32-bit complex sample format.
- **SBC as the software host:** Is the a single board computer capable of storing momentarily the data taken from the bladeRF and at the same time being house the software that manages the different implementations of the prototype.
- **Periodical execution of measurements:** Raw samples can be recorded automatically following a predefined schedule.
- **Post-processing data:** Data post-processing tasks are executed after the acquisition stage by using sockets between the *Nodejs* [18] web application and the *Python* script manager.

- **Plot of metrics:** Support tools for benchmarking that have been implemented in the prototype to compute and display metrics such CDF, PSD, and SNR.

The novelty of this work is based in the integration of the above features in a low-cost solution for spectrum measurement in an IoT scenario. The exponential growth of IoT devices and applications compel manufacturers to develop low-cost and high scalable test systems that easily meet this increase. This prototype can be used to perform configured remote measurements for different metrics that can be afterwards post-processed within the same interface. This prototype puts in perspective the use of metrics to study the interference signals within the spectrum in IoT ecosystem, to demonstrate as an example, what insights can be obtained from it. This work provides the foundation for implementations required for further research in the area of characterization of spectrum in local area and specific time. For example the results can be used to produce propagation and interference models of bands and technologies that are not well-known or studied yet, which are useful for network planning and new feature development in simulators. The specific capacity of portability allows to understand the situation in particular time and place which allows the operators of the wireless networks to optimize the spectrum usage or to the network implementers to understand the achievable performance and possible requirements of frequency for the equipment to be deployed in such area.

The rest of the work is organized as follows. Section 2 describes the architecture and design of the prototype. The experimental setup is described in section 3. Section 4 presents the measurement results of the prototype deployment. Section 5 encompasses the lessons learned throughout the realization of the prototype. The conclusions and perspectives of this work are presented in section 6.

2 Implementation

Fig. 1 Schematic representation of the designed prototype

The prototype can be described in two main aspects: the components that comprises the hardware structure and the applications implemented on the

software side. The schematic representation of the designed prototype is shown in figure 1 and the actual hardware implementation is shown in figure 2. As it is shown in figure 2, the elements on the hardware stack include: a log periodic antenna, responsible of receiving any electromagnetic signals over the air and converting them to electrical signals; a bladeRF, that captures a flexible and wide bin raw quadrature sample data; and a Beaglebone Black, that provide the bridge between the bladeRF and the web application. In the software side, a graphic user interface (GUI) was built over *Nodejs* as a web application where the user can input data to change RF parameters of the bladeRF, record samples at the moment or in a schedule basis, and visualize plots for benchmarks metrics. For post-processing of raw samples, a *Python* script manager have been programmed. This manager talks over a socket bidirectional channel with the web application to receive the raw data and send results back to the GUI. In the upcoming paragraphs, a detailed description of all these aforementioned will be explained in more detail.

Fig. 2 Hardware components of the prototype

2.1 Hardware

2.1.1 Antenna

One of the advantages that brings SDR technology is to put the software as close to the antenna as possible. The first component used to convert all the spectrum energy into an electrical signal is the antenna, which is part of the analog RF stage in the receiver chain. A log periodic antenna was used. The main reason to select this type of antenna is due their moderate gain over a broad span of frequencies bands ranging from 400 MHz up to 1000 MHz. As the prototype does not target an specific frequency range within the spectrum, this option was a good candidate to start off.

2.1.2 SDR

Most modern SDRs share similar characteristics in their structural design. Nowadays, there are several SDRs that have been presented on the market [1], [4], [10], [13], [21] and that are relative affordable. Choosing an appropriate SDR board may be a not easy task since it depends on many factors and every single one of them must be taken into consideration. By evaluating several possible candidates, the bladeRF x40, was chosen to be used as the SDR. This board presents a high versatility to be employed in a wide variety of IoT and 5G [7] applications. It supports a broad spectrum range from 300 MHz to 3.8 GHz which make a suitable candidate to study cutting-edge wireless protocols that work under unlicensed ISM bands as Bluetooth, LoRaTM, Sig-Fox, Wi-Fi, ZigBee. It also allows full-duplex bidirectional communication, which means that it is possible transmitting and receiving samples at the same time. It reach higher sample rates with a maximum of 40 Msps alongside with a set of digital bandwidth options that goes from 1.5 MHz up to 28 MHz. It provides an on-board RF transceiver [15] capable of set different levels of analog gains for the low-noise amplifier (LNA), and voltage gain amplifier (VGA) control. Other key feature is the use of a non traditional heterodyne approach. Indeed, the RF chip on the bladeRF uses a ZeroIF or direct conversion design which means that compared to their counterparts, does not need an intermediate frequency stage, instead it down converts the signal to baseband immediately. A further look into the trade-offs between using a homodyne and heterodyne architecture is presented in [20]. With all these aforementioned benefits featuring fairly wide sample rates, covering a broad frequency spectrum and being able to work in different bandwidths considered in IoT wireless communications, the authors found out that bladeRF was an excellent choice for this prototype.

2.1.3 SBC

In less than one decade, the microelectronic industry has change the world with novel computer options running very powerful processors. SBCs offer the same technological characteristics than a computer of 10 years ago with a

little more of power. These tiny pieces of hardware are capable to drive high-definition multimedia interface monitors, and perform decent logical tasks. The advantages that SBCs bring to the research cycle are tremendous and for this prototype a Beaglebone Black was used. It contains an universal serial bus port used to connect the bladeRF. With this communication established, the SBC is able to let the USB transmission to occur. The Beaglebone Black is a low-power, hand-held, low-cost, open hardware solution that offers a full customization of the software components prototype that are built on the top of it and simultaneously offers computational capabilities.

2.2 Software

Like depicted in figure 3, the prototype shows a GUI with the main features menu bar located in the top left side. The first menu feature allows to configure: carrier frequency, duration time, LNA, VGAs 1 & 2 gains and to start recording the signals (*i.e.* Record Raw Data button). The first case indicates the name of the file where samples will be stored. The instantaneous bandwidth indicates how much portion within the spectrum the signal of interest will occupy. The values are given in passband and can be set from 1.5 MHz up to 28 MHz. This parameter is relevant regarding the sample rate.

RAPIDO 5G | Hardware Prototype

Record transmission

Sample name:

Sample rate (MHz):

Tunning Frequency (GHz) From 0.3 to 3.8 GHz:

Time to record (s):

Bandwidth:

LNA:

VGA 1 (dB):

VGA 2 (dB):

bladeRF SDR Architecture

```

graph LR
    SDR[SDR] --> EC[Embedded Computer]
    EC --> UI[User Interface]
  
```

bladeRF Information:

- **Serial #:** a3e49bda424efc2893a1a595ab1be1cf
- **VCTCXO DAC calibration:** 0x90a1
- **FPGA size:** 40 KLE
- **FPGA loaded:** no
- **USB bus:** 2
- **USB address:** 3
- **USB speed:** SuperSpeed
- **Backend:** libusb
- **Instance:** 0

Fig. 3 Record sample window

The bandwidth should be less than the current sample rate to avoid aliases in the range of interest. The sample rate can be set up to 40 Msps and must be high enough to avoid aliasing in the signal of interest. The carrier frequency determines the center of the frequency band to be received. The range can be

from 0.3 to 3.8 GHz covering the entire Ultra High Frequency band. The LNA, VGA1 and VGA2 are part of the gain distribution across the receiver chain for optimizing noise figure and linearity. LNAs are used as an alternative to enhance the wireless communication performance increasing the receiver noise figure instead of increasing the transmitted or received power. The value for the LNAs gains goes from 0 to 6 dB, while the VGA1 and VGA2 gains are up to 25 and 30 dB maximum respectively. The duration is an important parameter of the measurement since it will determine the storage size of the sample. This parameter should be set carefully since it can be sizeable due to the fact that each sample is stored in a 32-bit complex number format. When all these parameters are set, the web interface starts to communicate in the background with the bladeRF to initiate taking the measurements. When this process is finished, a message notification will appear in the interface to inform the user that the measurement was taken successfully.

Figure 4 presents another feature in the tab window that allows to program recording samples on a schedule basis. This feature is very desirable since measurements may be taken at a time where the presence of an operator is not possible or the location where the prototype is positioned is difficult to access. Even if at the time being this prototype does not support outdoor environments, this add-on was created thinking as a scalable solution to work in outdoor environments without doing drastic changes on the core. To set a measurement sample on a schedule basis, the prototype needs to generate a task that will contain the already described parameter settings to call the bladeRF everytime the task should be executed. The window contains an "Add task" button that when clicked, it pops-up a window to enter the parameters. Meanwhile, the major region of the window is used to list the previous scheduled tasks through a tabular view. This feature was implemented using already existing Linux command tools as Cron that can be tailored very well for this purpose. Cron is the software in Linux / GNU environment that manages periodic tasks. The prototype takes advantage of this already well designed and tested tool to build the implementation on top of it. The schedule can be set in hourly, daily, weekly or monthly basis.

The last implementation is the post-processing add-on. Once a data measured is taken, this data can be post-processed through the *Python* script manager that will execute algorithms to compute different wireless metrics. To do that, the user must enter the absolute path where the file is stored and select what type of metrics will be applied. See figure 5. Once this information is entered, the GUI will wait for the *Python* manager to finish all the post-processing tasks of the requested data to then display the plots. The returned data is wrapped as a JavaScript Object Notation object over a transmission control protocol socket. The prototype visualizes the results as a form of graphs and tabular overviews of the returned data. An example of a plot and the tabular information will be shown in section 4. The plot is generated with libraries specialized on data science visualization to render high quality plots on web applications to provide features such as zoom, labelling, scale, color and to facilitate code maintainability and reuse. The tabular overviews are extracted

The 'Add task' window is a vertical form with a close button (X) in the top right corner. It contains the following fields and controls:

- Name:** A text input field.
- Sample name:** A text input field with the placeholder text 'your_sample_name'.
- Sample rate (MHz):** A text input field with the value '2'.
- Tuning Frequency (GHz) From 0.3 to 3.8 GHz:** A text input field with the value '2.4'.
- Time to record (s):** A text input field with the value '0.05'.
- Bandwidth:** A dropdown menu showing '1.5 MHz'.
- LNA:** A dropdown menu showing '0 dB'.
- VGA 1 (dB):** A text input field with the placeholder text 'Select First Variable Gain Amplifier'.
- VGA 2 (dB):** A text input field with the placeholder text 'Select Second Variable Gain Amplifier'.
- Schedule:** A group of five buttons: 'Hourly', 'Daily', 'Weekly', 'Monthly', and 'Yearly'. The 'Daily' button is highlighted.
- Task:** A large text input field.
- Buttons:** 'Cancel' and 'Save' buttons at the bottom right.

Fig. 4 Record sample at schedule basis window

through the meta-data encrusted in the measurement file title.

Both *Python* and *Nodejs* programming languages presented interesting features to be selected to interact with the SDR for this implementation. On the one hand, *Python* has become a great alternative to do research calculations [16], while *Nodejs* has been seen as the practical framework to ingrate hardware to the web world thanks to his native asynchronous paradigm [12].

Fig. 5 Selecting research metrics to plot

3 Experimental setup

All the experiments were carried out in an indoor office. The prototype is powered by the Beaglebone Black running Debian 9.3. The graphic interface resides on the Beaglebone Black and it is initiated as a web application in *Nodejs*. The post-processing algorithms are called whenever the web interface starts a request. A detailed information about the components can be seen in table 1.

Table 1 Components used on the prototype testbed

Component	Type
SBC	Beaglebone Black Rev C
Operating System	Debian 9.3
SDR	bladeRF x40
Antenna	Log periodic WA5VJB

To show how the prototype works, two scenarios have been addressed. In the first one, a RF packet is transmitted while in the second one the RF packet is not transmitted. The RF protocol to carry on the experiment is LoRaTM [2], [3], [6], a technology that it is envisioned as one of the promising leading technologies in IoT due its long range, energy efficiency, and resilience to interference compared to other RF technologies. The transmitting commercial device chosen is an end-node RFM95 module [11]. As in IoT wireless protocols there may be a high presence of other devices working at the same frequency range, the prototype research metrics were chosen to characterize the interference that these technologies can experiment. The packet parameters used on the LoRaTM module are shown in the table 2. Details concerning the modulation and coding parameters of this technology can be found in [22]. The measurements of the raw samples will be taken using the parameters

Table 2 LoRaTM packet parameters

Parameter	Value
Bandwith	500 kHz
Spreading Factor	128
Coding Rate	4/6
Tx Power	20 dBm
Tx Frequency	915 MHz

summarized in table 3. A description of the experimental observations will be presented in the following section.

Table 3 Sample configuration

Parameter	Value
Sample rate	2 Msps
Tunning Frequency	915 MHz
Recording time	3s
Bandwidth	1.5 MHz
LNA	6 dB
VGA1	30 dB
VGA2	3 dB

4 Experimental Results

This section describes the results of the empirical analysis carried out with the prototype to highlight how the wireless metrics: CDF [14], PSD and SNR can be used to study the interference characterization of the spectrum within IoT applications. The results are shown through the post-processing tool showing the several plots generated after the post-processing stage. The SNR is defined as follows

$$SNR = 10 \log_{10} \left(\frac{S}{N} \right) \quad (1)$$

where S and N corresponds to the power of the signal and noise, respectively. The estimation of SNR in field recordings is not a trivial task since the exact value of S is unknown. The power of the signal is approximated by the power of the signal during a LoRa packet transmission. The noise power is calculated in the rest of the acquired signal, *i.e.* without LoRa packet.

Following the parameters exposed on section 3, figure 6 shows how the GUI presents a table with the meta-data information saved at the time of taking one sample measurement of three seconds. The aim of the this table was to summarize at a high level along with the plots, the most important configuration parameters of the sample.

One issue was found during the experiments when dealing with the received power on the bladeRF. Most SDRs give the actual received power as relative rather than absolute. Based on this, to ensure the fidelity of the acquired data,

Parameters	Sample
Path file	/home/shared/rx_data_node-2019-06-04-12-34-30-2M-3s-_915G-1_5M.sc16q11
Bandwidth	1.5MHz
Carrier Frequency	.915GHz
Time recorded	3s
Sample rate	2MHz

Fig. 6 Meta-data acquired in one sample

the bladeRF has been calibrated using an already calibrated spectrum analyzer. An offset parameter is available in the GUI to calibrate the bladeRF measurements with the spectrum analyzer. In addition, as the bladeRF uses a direction-conversion architecture causing a large DC offset at the center frequency, an automatic correction [19] utility provided by Nuand was used to mitigate, as much as possible the DC offset.

The plot obtained can be seen in figure 7(a) which shows the PSD of the transmitted LoRaTM packet signal. The bandwidth, sample rate and recording time of the measurement were set to 1.5 MHz, 2 Msps, 3 seconds respectively. As the LoRaTM packet bandwidth was set to 500 kHz [9], a bandwidth selection on the bladeRF of 1.5 MHz was more than enough to cover the entire signal within the spectrum. To select the sample rate of 2 Msps, one must consider the frequency response of the front-end low pass filter used in the bladeRF, which is defined as the bandwidth parameter on the GUI. Sampling slightly above 1.5 MHz, ensured that the filter reached the full rejection before the Nyquist limit, then avoiding aliasing on the received signal. The gain of the LNA was configured to its maximum to receive the best SNR ratio provided by the device, which is 6 dBm. The gains of the first and second VGAs were kept at their minimum since to capture a LoRaTM packet, the dynamic range of the bladeRF did not have to change to much. The PSD in figure 7(a) shows that there is a high overall noise in the frequency span, this result is expected because the LoRaTM transmission was short compared to the recording time. On the other hand, when the LoRaTM packet was not sent, a low power reception sensitivity was captured. In figure 7(b) it can be seen a flat noise floor of -40 dBm indicating the absence of signal within the frequency range. A noteworthy point is the presence of the spike at the center frequency, which is more visible when there is only noise. Even though that the automatic correction utility was used to mitigate this spike, it is still visible. One approach that it can be used for future measurements is moving the tuning frequency to an offset up or down of the signal to minimize even more this shortcoming. It is concluded that plotting the PSD of a RF sample helps to notice the presence

of a signal within the frequency range of interest.

Fig. 7 Power Spectral Density

In addition, to characterize the interference within a frequency band, one way to know when a channel is occupied or free is by using the CDF. The results acquired during the LoRaTM packet transmission at a time of 3 seconds and at their absence are shown in figure 8(a) and 8(b) respectively. It is observed that in figure 8(a) when the LoRaTM packet is transmitted for 3 seconds the slope tends to increase more rapidly with slow changes on the edges. In addition, when the packet is not transmitted, the slope of the curve rapidly increases within the same range and with no slow edges change, pointing out that probably there is no signal of interest present in the frequency range where the measurement was taken. Using a CDF function played a pivotal role in the prototype since it can be used as an analytical model to characterize the interference between signals in the electromagnetic spectrum.

Fig. 8 Cumulative Density Function - Signal Noise to Ratio

Finally, the results obtained for the signal noise ratio is shown in table 4. The findings convey that when a LoRaTM packet is transmitted, the SNR is

increased to 19.82 dB for a time record of three seconds while at absense of the packet, the SNR decrease up to -31.95 dB giving a difference of 51.77 dB. These results are expected since there is a short distance between the prototype and the LoRaTM device, consequently receiving almost all the signal power.

Table 4 SNR Results

Scenario	SNR (dB)
Packet transmitted	19.82 dB
Packet not transmitted	-31.95 dB

5 Conclusions

A prototype to study wireless measurements is presented in this work. The main features of the prototype include: configurable RF parameters, recording samples in a schedule basis and computing and visualization of wireless metrics of acquired raw data. Such prototype addresses challenges impose by the co-existence of different wireless communication protocols in the same frequency bands by proposing a customized interference characterization solution.

The authors demonstrated that it is possible to build a low-cost, small-size, configurable and portable prototype using web and SDR technologies to characterize interference in the context of IoT applications.

The results show that it is possible to apply post-processing algorithms to raw samples taken from a SDR platform and presented on web interface. The lessons learned throughout this experience will help other researches to explore new methodologies on how integrate their algorithms using web technologies with an open source perspective. Furthermore, not only that but leading to a platform where interaction, debugging and maintaining becomes intuitive and therefore, easier.

6 Further Work

In future work, it is expected to add additional wireless measurements such as: spectrogram and phase plot. In addition, as this prototype is focused in applications concerned to IoT and 5G, the post-processing *Python* application manager in conjunction with the web application on *Nodejs* are targeted to be build as a *Docker* image to speed up replication at a software level of several prototypes over a set of various scenarios presented as a out-of-the-box-connectivity solution. Furthermore, it is a plan to introduce an encryption method to improve security. Finally, another notable idea is the development of a custom embedded Linux / GNU image to enhance performance and manage storage efficiently of the Beaglebone Black taking away all extra software that comes with the predefined image and that have been not used in the prototype.

This prototype provides the basis for future systems useful for further studies in the area of characterization of spectrum in local area and specific time.

7 Conflicts of Interest

The authors declare that there is no conflict of interests.

Acknowledgements The authors thank Mr. Oussama Radi, student from the ENSEIRB-MATMECA-Bordeaux INP, France, for his collaboration in some technical aspects of this implementation.

References

1. Adalm-Pluto: <https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html>. [Online; accessed 5-July-2020]
2. Augustin, A., Yi, J., Clausen, T., Townsley, W.: A study of LoRa: Long range & low power networks for the Internet of things. *Sensors* **16**(9), 1466 (2016)
3. Corro, H., Robles, E., Merchan, F., Poveda, H.: Design and implementation of a LoRa-based IoT network for environmental parameters monitoring. In: 7th International Engineering, Sciences and Technology Conference (2019)
4. Ettus Research: USRP. <https://www.ettus.com/>. [Online; accessed 5-July-2020]
5. Federal Communications Commission: Authorization of spread spectrum systems under parts 15 and 90 of the FCC rules and regulations (1985)
6. Ferré, G., Giremus, A.: LoRa physical layer principle and performance analysis. In: 25th IEEE International Conference on Electronics, Circuits and Systems, pp. 65–68. IEEE (2018)
7. Fettweis, G., Alamouti, S.: 5G: Personal mobile Internet beyond what cellular did to telephony. *IEEE Communications Magazine* **52**(2), 140–145 (2014)
8. GNU Radio: <https://www.gnuradio.org/>. [Online; accessed 5-July-2020]
9. Goursaud, C., Gorce, J.M.: Dedicated networks for IoT: PHY/MAC state of the art and challenges. *EAI endorsed transactions on Internet of Things* **1**(1), 1–11 (2015)
10. Great Scott Gadgets: Hackrf. <https://greatscottgadgets.com/hackrf/>. [Online; accessed 5-July-2020]
11. HopeRF: RFM95 Ultra-Long-Range Transceiver Module. <https://www.hoperf.com/modules/lora/RFM95.html>. [Online; accessed 12-July-2020]

12. Johnny-Five: <http://johnny-five.io/>. [Online; accessed 5-July-2020]
13. KiwiSDR: <http://kiwisdr.com>. [Online; accessed 5-July-2020]
14. Kostov, N.: Mobile radio channels modeling in MATLAB. *RadioEngineering-Prague* **12**(4), 12–17 (2003)
15. Lime Microsystems: LMS6002D, Multi-band Multi-standard Transceiver with Integrated Dual DACs and ADCs. <https://limemicro.com/app/uploads/2015/10/LMS6002Dr2-DataSheet-1.2r0.pdf> (2012). [Online; accessed 5-July-2020]
16. Millman, K.J., Aivazis, M.: Python for scientists and engineers. *Computing in Science & Engineering* **13**(2), 9–12 (2011)
17. Navarro, K., Canto, F., Poveda, H.: Software defined radio as an educational learning tool in wireless communications. In: 16th LACCEI International Multi-Conference for Engineering Education Technology, pp. 1–10 (2018)
18. Nodejs Programming language: <https://nodejs.org/en/about/>. [Online; accessed 14-July-2020]
19. Nuand: DC offset and IQ Imbalance Correction. <https://github.com/Nuand/bladeRF/wiki/DC-offset-and-IQ-Imbalance-Correction> (2017). [Online; accessed 14-February-2020]
20. Razavi, B.: Design considerations for direct-conversion receivers. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* **44**(6), 428–435 (1997)
21. RTL-SDR: <https://rtl-sdr.org>. [Online; accessed 5-July-2020]
22. Semtech: LoraTM Modulations Basics. <https://www.semtech.com/uploads/documents/an1200.22.pdf>. [Online; accessed 12-July-2020]
23. Silvius, M.D., Ge, F., Young, A., MacKenzie, A.B., Bostian, C.W.: Smart radio: spectrum access for first responders. In: *Wireless Sensing and Processing III*, vol. 6980. International Society for Optics and Photonics (2008)
24. Stankovic, J.A.: Research directions for the Internet of things. *IEEE Internet of Things Journal* **1**(1), 3–9 (2014)
25. Uengtrakul, B., Bunnjaweht, D.: A cost efficient software defined radio receiver for demonstrating concepts in communication and signal processing using python and rtl-sdr. In: 4th International Conference on Digital Information and Communication Technology and its Applications, pp. 394–399 (2014)
26. Valerio, D.: Open source software-defined radio: A survey on gnuradio and its applications. Forschungszentrum Telekommunikation Wien, Vienna, Technical Report FTW-TR-2008-002 (2008)