Radio logicielle pour des réseaux de capteurs sans fil cognitifs : un standard IEEE 802.15.4 reconfigurable

Rafik Zitouni

UNIVERSITÉ PARIS-EST

THÈSE

en vue de l'obtention du titre de

DOCTEUR DE L'UNIVERSITÉ DE PARIS EST

Spécialité : INFORMATIQUE

Présentée Par: Rafik ZITOUNI

SOFTWARE DEFINED RADIO FOR COGNITIVE WIRELESS SENSOR NETWORKS: A RECONFIGURABLE IEEE 802.15.4 STANDARD

Soutenue le 14 Octobre, 2015

Jury

Directeur :	**Laurent GEORGE**	-	CNRS/UPEMLV/ESIEE/ENPC (LIGM), France
Rapporteurs :	**Michel MISSON**	-	Université de Blaise Pascal/LIMOS, France
	Pascale MINET	-	INRIA, France
Examinateurs :	**Didier LE RUYET**	-	CNAM, France
	Fabien LIGREAU	-	Safran Engineering Services, France
	Stefan ATAMAN	-	ECE Paris, France
Invitée :	**Marie MATHIAN**	-	ECE Paris, France

PhD prepared at
ECE Paris – LACSC
Laboratoire d'Analyse et Contrôle des Systèmes Complexe
37, Quai de Grenelle
CS 71520
75725 PARIS CEDEX 15

PhD in collaboration with
UPEC – LiSSi (EA 3956)
Laboratoire Images, Signaux et Systèmes Intelligents
Domaine Chérioux
122 rue Paul Armangot
94400 Vitry sur Seine

Software Defined Radio for Cognitive Wireless Sensor Networks: A reconfigurable Ieee 802.15.4 Standard

Abstract

The Increasing number of Wireless Sensor Networks (WSNs) applications has led industries to design the physical layer (PHY) of these networks following the IEEE 802.15.4 standard. The traditional design of that layer is on hardware suffering from a lack of flexibility of radio parameters, such as changing both frequency bands and modulations. This problem is emphasized by the scarcity of the radio-frequency spectrum. Software Defined Radio (SDR) is an attracting solution to easily reconfigure radio parameters. In addition to SDR, a cognitive radio concept can be proposed by spectrum sensing and Dynamic Spectrum Access (DSA) both to overcome the spectrum scarcity problem. This thesis proposes a new SDR solution for WSNs based on the IEEE 802.15.4 standard. Our aim is to characterize an SDR platform that implements two standardized PHY layers and cognitive radio for WSNs.

In this thesis, we carried out SDR implementations using a GNU Radio and Universal Software Peripheral Radio (USRP) platform. We chose this particular platform because it is one of the most practical and well-performed ones. A thorough study was performed to analyze GNU Radio software architecture before its usage. USRPs and their daughter boards were also analyzed through experimental radio-frequency measurements. The analysis of the GNU Radio USRP platform brought a detailed description of its architecture and performances as well as the way to implement an SDR. This description particularly assists researchers to quickly develop efficient SDR receivers and transmitters. We show through our experiments that the measured performances of daughter boards mounted on a USRP are lower than expected ones. Despite these results, some daughter boards have many interesting features such as large covered frequency bands and with a linear output power. An empirical model was introduced to accurately characterize the average output power of a particular daughter board.

Then, we implemented a new possible standardized PHY layer for the 868/915 MHz frequency band. A reverse engineering process of another implementation was performed for the 2450 MHz frequency band. These two PHY layers were described by communication chains or flow graphs. We suggested a new Cognitive Radio by a reconfiguration of these flow graphs within the corresponding frequency bands. The particularity of our cognitive radio is to reconfigure flow graphs in function to the selected frequency. This selection is performed by both DSA and spectrum sensing based on energy detection through real wireless communications. We introduced a message based algorithm in order to reconfigure the flow graphs and to synchronize the selection of a carrier frequency.

Our two implemented PHY layers for the 2450 MHz and the 868/915 frequency bands were found functional. The first one was tested by exchanging data packets with real sensor nodes. The second was also experienced by a packet exchange, but via GNURadio/USRP communications. Both tests were carried out through real communications. We were also able to measure two wireless communication parameters: Bit Error Rate (BER) and the Packet Success Rate (PSR). The result of functional PHY layers was beneficial for realization and experiments of our cognitive radio. We found that our DSA significantly improves the packet success rate compared to that obtained with static spectrum access in an indoor environment. The results of this thesis lead to experiment a cognitive radio with an SDR not only for a WSN, but also for other wireless networks and radio standards.

Keywords: Software Defined Radio (SDR), IEEE 802.15.4, Wireless Sensor Networks (WSN), Physical Layer, Cognitive Radio, Dynamic Spectrum Access (DSA), Universal Software Radio Peripheral (USRP), GNU Radio

Radio Logicielle pour des Réseaux de Capteurs Sans Fil Cognitifs: Un Standard IEEE 802.15.4 Reconfigurable

Résumé

Le nombre croissant d'applications des Réseaux de Capteurs Sans Fils (RCSFs) a conduit les industriels à concevoir ces réseaux avec une couche Physique (PHY) suivant le standard IEEE 802.15.4. Actuellement, cette couche est implémentée en matériel souffrant d'un manque de flexibilité du changement des paramètres radio, telles que bandes de fréquences et modulations. Ce problème est accentué par la rareté du spectre radio fréquences. La Radio Logicielle (RL) est une nouvelle solution pour reconfigurer plus facilement des paramètres radio. A partir d'une RL, il est possible de développer une radio cognitive permettant une écoute de spectre et un Accès Dynamic au Spectre (ADS). Ces deux possibilités sont utiles pour surmonter le problème de la rareté du spectre. Cette thèse propose une nouvelle solution Radio logicielle pour un RCSF basé sur le standard IEEE 802.15.4. Notre objectif est de caractériser une plate-forme RL qui implémente à la fois deux couches PHY standardisées et une radio cognitive pour des RCSFs.

Dans cette thèse, nous avons réalisé des implémentations RL en utilisant une plateforme composée de la solution Universal Software Peripheral Radio (USRP) d'Ettus Research et de GNU Radio. Nous avons choisi cette plateforme particulière puisqu'elle est parmi les outils les plus performants et les plus pratiques d'après notre état de l'art. Une étude minutieuse a été effectuée pour analyser l'architecture logicielle de la GNU Radio avant son utilisation. Des USRPs et leurs cartes filles ont été aussi analysés à travers des mesures expérimentales radio fréquences. L'analyse de la plate-forme GNU Radio USRP a apporté une description détaillée de son architecture et de ses performances pour la réalisation d'une RL. Cette description doit permettre à la communauté de chercheurs de développer rapidement des récepteurs et émetteurs radio logiciels. Nous avons prouvé à travers nos expériences que les performances mesurées sont plus faibles que celles attendues pour certaines cartes filles d'USRP. Malgré ces résultats, certaines cartes ont de nombreuses caractéristiques intéressantes, comme de grandes bandes de fréquences couvertes et avec une puissance de sortie linéaire. Un modèle empirique a été introduit pour caractériser avec précision la puissance de sortie moyenne d'une carte fille particulière.

Nous avons ensuite implémenté une nouvelle couche PHY standardisée pour la bande de fréquence 868/915 MHz basée sur le standard 802.15.4. Un processus de rétro-ingénierie d'une autre implémentation développée pour la bande 2.4GHz a été effectué. Ces deux couches ont été décrites par des chaines de communications ou des graphes de flux. Nous avons finalement proposé une nouvelle radio cognitive par une reconfiguration de ces graphes de flux dans les deux bandes de fréquences correspondantes. La particularité de notre radio cognitive est de reconfigurer les graphes de flux en fonction de la fréquence sélectionnée. Cette sélection est effectuée par un ADS et une écoute de spectre basé sur une détection d'énergie, validés tous les deux au travers des réelles communications sans fil. Nous avons introduit un algorithme à base de messages afin de reconfigurer les graphes de flux et de synchroniser la sélection sur une fréquence porteuse.

Les deux couches physiques en radio logicielle pour les bandes 2.4 GHz et 868/915 MHz ont été testées et sont fonctionnelles. La première a été testée en échangeant des paquets de données avec des nœuds capteurs réels. La deuxième a été expérimentée par l'échange de paquets, à travers une communication entre deux radios logicielles USRP/GNU Radio. Les deux tests ont été réalisés à travers des communications réelles. Nous avons réussi à mesurer deux paramètres réels d'une communication sans fil : le taux d'erreur binaire et le taux de succès des paquets. Les couches PHY résultantes ont servi à la réalisation et à l'expérimentation d'un ADS de notre radio cognitive. Un ADS a amélioré significativement le taux de succès de paquets par rapport à celui obtenu avec un accès statique dans un environnement indoor. Les résultats de cette thèse conduisent à expérimenter une radio cognitive avec une radio logicielle non seulement pour un RCSF, mais pour d'autres réseaux sans fil et standards radio.

Mots-clés : Radio Logicielle, IEEE 802.15.4, Réseaux de Capteurs Sans fils (RCSFs), Couche physique, Radio Cognitive, Accès Dynamique au Spectre (ADS), Universal Software Radio Peripheral (USRP), GNU Radio

Remerciement

Je tiens tout d'abord à remercier Laurent George pour avoir dirigé cette thèse. Laurent m'a fait confiance dès le début en me proposant un sujet de recherche d'actualité. Il m'a guidé, encouragé, conseillé tout en me laissant une grande liberté de travail et en me déléguant plusieurs responsabilités ; ce fut un honneur et j'espère avoir été à la hauteur.

Cette thèse a été traitée en marge d'un projet de recherche auquel l'ECE Paris participait. Le consortium du projet a financé une partie de la thèse, en facilitant l'acquisition de nouveaux matériels de test et d'expérimentation. Je remercie les financeurs de ce projet qui se reconnaîtront en lisant ces lignes.

Je remercie également Stefan Ataman qui était mon encadrant de thèse. Stefan m'a permis avec sa rigueur d'appréhender certains concepts nécessaires à l'avancement de cette thèse. Il m'a facilité l'acquisition des connaissances en traitement de signal et en communication numériques. Mes remerciements vont aussi à Marie Mathian qui était mon encadrante au début de cette thèse et qui m'a épaulé dans le projet de recherche.

Je remercie aussi Michel Misson et Pascale Minet d'avoir bien voulu être mes rapporteurs. Je remercie vivement Didier Le Ruyet et Fabien Ligreau pour leur participation au jury.

J'adresse un remerciement particulier à Yacine Amirat directeur du laboratoire LISSI de l'Université de Paris Est Créteil. Yacine m'a initié à la recherche et ses conseils étaient utiles pour la concrétisation de ce doctorat.

Mon arrivée à Paris peut être considérée comme une "nouvelle" date de naissance. Paris m'a permis d'accéder facilement à de multiples sources de savoir : conférences en philosophie, en culture d'orient et en psychologie, etc. Ces activités m'ont enrichi, m'ont marqué et ont nourri ma soif de savoir. Cela m'a permis également de prendre du recul par rapport à mon domaine de recherche. Bien que ces savoirs soient accessibles virtuellement via le web, elles restent beaucoup plus stimulantes à travers un contact réel. Comme le dit si bien la sagesse arabe "Il est préférable d'acquérir le savoir en étant proche des Hommes de la science". Je remercie donc le destin qui m'a donné la chance de m'enrichir à travers cette ville.

Mon travail à l'ECE Paris était particulièrement riche d'expériences. Je reconnais l'apport de l'environnement stimulant de l'ECE et je dis merci à ceux qui y ont participé de près ou de loin. Je commence par les collègues doctorants, ceux qui ont fini leurs thèses : Pierre et Vincent, c'étaient des collègues sympathiques et modestes. Pierre était proche, bienveillant et m'a initié à la culture française. La rencontre des autres doctorants comme Clément, Thomas, Olivier, Ermis ou Nora était aussi enrichissante humainement. Les enseignants chercheurs de l'ECE ont été une source de courage, en l'occurence ceux qui partagent mon bureau : Fréderic Fauberteau, Xiaoting Li, Benaoumeur Senouci, Jean François

Hermant et Yves Rakotondratsimba. Je n'oublie pas ceux avec qui j'ai partagé des discussions intéressantes : Max Agueh et Houari Mechkour. Un merci aux collègues généreux et généreuses, qui ont relu mes résumés en anglais et en français, en particulier : Waleed Mouhali et Kristen Bini. Je remercie aussi Assia Soukane et Valéria Nuzzo qui m'ont permis de rédiger ma thèse dans les meilleures conditions possibles.

J'adresse ensuite une dédicace aux personnes que j'ai eu le plaisir de côtoyer durant ces dernières années. Hafid, Ihssen, Mouhamed et Mehdi m'ont soutenu depuis le début de ma thèse. C'étaient pour moi des compagnons présents et proches durant les bons et les mauvais moments.

Je remercie Claire Bozec, son mari Jean-Jacques et leur fille Sylvie pour leur générosité. En effet, ils m'ont beaucoup aidé en acceptant de relire une partie de mon manuscrit... Les premières corrections de mon manuscrit m'ont permis d'améliorer mon anglais.

Ma gratitude et mes remerciements vont aussi à toute ma famille pour leur soutien et leur amour. Un grand Merci à ma mère et mon père qui m'ont transmis les valeurs nécessaires pour avancer et surmonter les difficultés de cette thèse. Merci à mes grand-mères et ma tante qui ont été une source d'affection et d'énergie mentale, et à mes frères et sœurs, Imad et Mourad, Nassima, Nadia, Sihem, Habiba et Randa, eux aussi une source de courage et d'inspiration. Encore une fois : merci à tous !

Je finis mes remerciements et mes dédicaces par ma femme. Merci à Yasmine qui m'a accompagné durant les mois de rédaction de ce manuscrit. Sa patience et sa confiance en moi m'ont permis d'avancer avec assurance jusqu'au bout de cette thèse. Elle était la lumière me montrant le bout du chemin de cette aventure : la thèse.

Ceux qui par la science vont au plus haut du monde
Qui par l'intelligence, scrutent le fond des cieux
Ceux-là, pareils aussi à la coupe du ciel
La tête renversée, vivent dans leur vertige
[Omar Khayyam]

Author's publication list

International Conference Papers

SDR'2012 **Rafik Zitouni**, Stefan Ataman, Marie Mathian, and Laurent George. "IEEE 802.15.4 transceiver for the 868/915 MHz band using Software Defined Radio". In: *Proceedings of Wireless Innovation Forum SDR'12* abs/1304.8028 (2012)

CyberC'2013 **Rafik Zitouni**, Stefan Ataman, and Laurent George. "RF Measurements of the RFX 900 and RFX 2400 Daughter Boards with the USRP N210 Driven by the GNU Radio Software". In: *Proceedings of 5ths International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. IEEE, Oct. 2013, pages 490–494. ISBN: 978-0-7695-5106-7

SSD'2015 **Rafik Zitouni** and Stefan Ataman. "An empirical model of the SBX daughter board output power driven by USRP N210 and GNU Radio based Software Defined Radio". In: *Proceedings of 12th Multi-Conference on Systems, Signals and Devices SSD'15*. IEEE, Mar. 2015

SDR'2015 **Rafik Zitouni**, Laurent George, and Yacine Abouda. "A Dynamic Spectrum Access on SDR for IEEE 802.15.4e networks". In: *Proceedings of Wireless Innovation Forum SDR'15* (Mar. 2015)

M&N'2015 **Rafik Zitouni**, Stefan Ataman, Mathian Marie, and Laurent George. "Radio Frequency Measurements on a SBX Daughter Board using GNU Radio and USRP N-210". In: *Proceedings of 3rd IEEE International Workshop on Measurements and Networking 2015*. IEEE, Oct. 2015

Posters in International Conferences

GR'2015 **Rafik Zitouni**, Stefan Ataman, and Laurent George. "Radio Frequency Measurements on SBX Daughter Boards using GNU Radio/USRP N-210". In: *GNU Radio 2015 conference* (Aug. 2015)

Résumé étendu de la Thèse

Résumé

Ce résumé étendu récapitule l'essentiel des travaux réalisés dans le cadre de cette thèse intitulée "*Radio Logicielle pour des Réseaux de Capteurs Sans fils Cognitifs : Un standard IEEE 802.15.4 reconfigurable*". Cette thèse s'intéresse au problème de la reconfiguration des paramètres radios de la couche **PHY**sique (PHY) des **R**éseaux de **C**apteurs **S**ans **F**ils (RCSFs) et en particulier à la problématique de la rareté du spectre radiofréquence. Notre solution est basée sur une **R**adio **L**ogicielle (RL). Nous avons analysé et évalué une plate-forme RL afin de réaliser cette solution. Nous avons implémenté sur cette plate-forme des spécifications IEEE 802.15.4 pour la bande de fréquences 868/915 MHz. Cette nouvelle implémentation a été détaillée conjointement avec celle pour la bande de fréquence 2.4 GHz. En outre, nous avons introduit une radio cognitive définie par un **A**ccès **D**ynamique au **S**pectre (ADS) à deux bandes de fréquences 868/915 MHz et 2.4 GHz. Un ADS a amélioré significativement la robustesse de communication des paquets de données. Ce résultat ainsi que la caractérisation de la plate-forme contribuent à la réalisation de nouvelles solutions basées sur une RL.

1 Introduction et objectifs

Le nombre croissant d'applications basées sur les RCSFs a conduit les industriels à concevoir ces réseaux avec une couche PHY suivant le standard IEEE 802.15.4 [16] [17]. Un des problèmes de cette couche est la reconfiguration de ses paramètres radios tels que la bande de fréquence ou la technique de modulation. La source de ce problème est liée à la nature physique de cette couche et à la nécessité de conformité au standard. De plus, le nombre croissant de réseaux sans fils et de technologies radios conduit à une rareté du spectre radiofréquence. En outre, la communauté de recherche traite ces problèmes ainsi que d'autres en rapport avec la couche PHY en se basant généralement sur des outils de simulation.

Une RL est une solution émergente d'expérimentation réelle de transmissions sans fils. Elle intéresse à la fois les industriels et la communauté de recherche, puisqu'elle permet de définir les paramètres et les fonctions d'une radio en logiciel à la place du matériel (c.-à-d. puce électronique). Ces paramètres peuvent être la bande de fréquence, le type de modulation et la puissance radio d'émission [10]. La reconfigurabilité est parmi les principaux avantages d'une RL. Étant donné cet avantage, un nombre croissant de plates-formes, d'architectures et de standards ont été proposés dans la littérature. Dans cette thèse, nous présentons un état de l'art sur

la radio logicielle. Nous nous intéressons principalement à des solutions performantes et simple à mettre en oeuvre.

L'**U**niversal **S**oftware **R**adio **P**eripheral (USRP) de *Ettus Research* [34] et l'outil logiciel GNU Radio [72] constituent tous les deux une plate-forme RL basée sur une architecture avec un processeur générique (ou **G**eneral **P**urpose **P**rocessor (GPP)). L'USRP/GNU Radio est facilement accessible et avec des performances radiofréquences élevées (disposant d'une large bande de fréquence et d'un taux d'échantillonnage élevé). La GNU Radio est un logiciel libre permettant l'implémentation (ou le prototypage) d'une radio à l'aide d'un ordinateur hôte disposant d'un GPP. Une implémentation se présente sous la forme de chaînes de blocs de communication (ou graphes de flux) pour un émetteur et un récepteur radio. Un USRP est un périphérique qui porte des cartes filles permettant à un graphe de flux GNU Radio de recevoir un signal en bande de base et de transmettre un signal modulé. Dans la littérature [35] [36] [37] [82] [83], la plate-forme USRP/GNU Radio a été utilisée pour expérimenter des solutions dans la recherche sans une connaissance à priori des performances radios. Notre thèse propose une analyse minutieuse de la plate-forme. En effet, nous proposons une évaluation des performances des cartes filles des USRPs [2] [3].

Le standard IEEE 802.15.4 [16][17] spécifie plusieurs couches PHYs en fonction de la bande de fréquence et de la modulation. Les spécifications pour la bande de fréquences **I**ndustrial **S**cientific **M**edical (ISM) à 2.4 GHz avec la modulation **O**ffset-**Q**uadtrature **P**hase **S**hift **K**eying (O-QPSK) ont été implémentées sous la GNU Radio/USRP [35]. Cette implémentation a été testée et trouvée fonctionnelle dans la littérature. Par contre, nos premiers tests ont montré certaines difficultés à recevoir et à transmettre des paquets de données. Notre thèse présente une rétro-ingénierie de cette implémentation. Notre objectif est de détailler les chaînes de l'émetteur et du récepteur radio. L'émission doit être décrite de la génération des paquets de données jusqu'à leurs transmissions sous forme d'un signal modulé. De même, la réception doit être détaillée de la réception d'un signal en bande de base, en passant par une démodulation et en finissant par une construction des paquets.

Plusieurs couches PHY spécifiées dans le standard 802.15.4 ne sont explorées ni par l'industrie ni par la communauté de recherche. La couche PHY pour la bande de fréquence 868/915 MHz permet une portée de transmission plus grande que celle pour la bande 2.4 GHz [1]. Cette thèse présente une nouvelle implémentation en radio logicielle pour la bande 868/915 MHz.

Une radio cognitive est un concept proposé comme une solution au problème de la rareté du spectre radiofréquence. Ce concept est réalisé concrètement par une écoute de spectre et un ADS. Dans la bande de fréquence ISM 2.4 GHz, un RCSF basé sur le standard 802.15.4 partage cette bande avec d'autres réseaux, par exemple IEEE 802.11 b/g (WiFi). Le problème d'interférence des canaux de ces réseaux conduit à un manque de robustesse des communications radios. Bien que la nouvelle version 802.15.4e [17] propose un nouveau mécanisme de saut de fréquence (ou canaux),

ce mécanisme définit en avance des valeurs statiques (ou pseudo aléatoires) à ces fréquences. Dans cette thèse, nous introduisons une nouvelle solution radio cognitive réalisée avec un ADS et une écoute de spectre basée sur une détection d'énergie. Notre ADS est effectuée à travers les deux bandes de fréquences 2.4 GHz et 868/915 MHz avec une reconfiguration des chaînes d'émission et de réception.

Ce résumé étendu est organisé en quatre principales parties. La Section 2 présente une brève présentation de notre état de l'art sur la radio logicielle. L'analyse de la GNU Radio et les mesures expérimentales sur des cartes filles des USRPs sont présentées, respectivement dans Section 3 et Section 4. Les implémentations radio logicielles, en suivant le standard IEEE 802.15.4, sont décrites dans la Section 5. Un réseau de capteurs sans fil cognitif basé sur le 802.15.4 est détaillé dans la Section 6. Nous finissons notre résumé par une conclusion et des perspectives.

2 État de l'art sur la Radio Logicielle

Notre objectif est de découvrir les solutions radios logicielles qui permettent une réalisation en logiciel des nœuds d'un RCSF basé sur le standard IEEE 802.15.4. Nous avons dressé un état de l'art sur les solutions existantes dans la littérature.

Une architecture typique d'une RL est composée de trois parties : Une partie RadioFréquence (RF) de front-end, une partie de **F**réquence **I**ntermédiaire (FI) et une dernière pour un traitement en bande de base (voir Figure 2.2) [39]. La partie RF permet la réception et l'émission d'un signal en bande de base via des antennes. La partie FI est introduite dans cette architecture générale, puisque le Convertisseur Analogique Numérique (A/N) nécessite une puissance de calcul et une fréquence d'échantillonnage élevées. En général, cette conversion peut être accomplie par un **F**ield **P**rogrammable **G**ate **A**rray (FPGA), par exemple dans un USRP. La dernière partie de traitement en bande de base remplace des fonctionnalités analogiques par d'autres numériques. Des technologies programmables permettent ce remplacement par exemple en utilisant un FPGA, un Processeur de traitement de signal (ou **D**igital **S**ignal **P**rocessor (DSP)) ou un GPP.

La reconfigurabilité est la principale propriété de cette architecture. Elle permet à un émetteur ou à un récepteur radio de changer ces paramètres (fréquence) et fonctions (modulations) en logiciel. Ce changement est effectué par un concepteur radio ou par la radio elle-même (c.-à-d. en-ligne ou durant l'exécution). La portabilité et l'interopérabilité sont aussi parmi les avantages d'une telle architecture. La portabilité est définie par la mobilité de traitement des formes d'ondes d'une plate-forme matérielle à une autre. L'interopérabilité est la possibilité d'interconnecter des technologies et des réseaux sans fils via la couche PHY. Par ailleurs, l'optimisation des performances des dispositifs matériels comportant les trois parties d'une RL représente un enjeu majeur. Le traitement en bande de base peut être accompli via un **A**pplication **S**pecific **I**ntegrated **C**ircuit (ASIC). Ce dernier est préféré pour un

calcul efficace sur une puce miniaturisée, mais au détriment de la flexibilité [42]. Un FPGA, un DSP ou un GPP offrent plus de flexibilité. Le choix de ces dispositifs matériels et leur disposition (centralisée et distribuée) peut impacter la facilité de la prise en main d'une RL. La programmation peut se faire avec des langages de haut niveau dans un GPP et un DSP alors qu'elle est en **V**ery high speed integrated circuit **H**ardware **D**escription **L**anguage (VHDL) avec un FPGA. Par conséquent, un nombre important d'architectures et de plates-formes sont proposées dans la littérature.

L'architecture de communication logicielle **S**oftware **C**ommunication **A**rchitecture (SCA) [44] et le système radio reconfigurable **R**econfigurable **R**adio **S**ystem (RRS) [49] sont les principales architectures qui tentent de standardiser l'usage d'une RL. La première se base sur une architecture distribuée de type **C**ommon **O**bject **R**equest **B**roker **A**rchitecture (CORBA) [44]. La deuxième réutilise une plate-forme GNU Radio/USRP modifiée [49].

Nous avons suggéré une classification des radios logicielles adaptées aux périphériques embarqués. Deux classes ont été définies: l'une basée sur un GPP et une deuxième basée sur un matériel reconfigurable [52][53]. Le dernier désigne l'utilisation d'un FPGA ou une architecture granulaire (en anglais *coarse-grained architecture*). Le développement logiciel en utilisant un matériel reconfigurable peut être lent avec des langages de programmation comme le VHDL. Par contre, un GPP permet un portage facile d'applications radios programmées en langage de haut niveau, par exemple en C++, Python, C, etc. Un ordinateur hôte est facilement accessible par une grande communauté d'usagers. Par conséquent, contrairement à l'utilisation d'un FPGA, un ordinateur hôte permet un prototypage rapide d'une RL. Il est à noter qu'une architecture peut se baser sur la combinaison d'un FPGA et d'un GPP. Un FPGA peut être dédié aux fonctions lourdes en puissance de calcul de la partie FI, et un GPP pour un traitement en bande de base. Un exemple d'une telle combinaison est la GNU Radio/USRP (voir la prochaine Section).

Dans notre état de l'art, nous avons synthétisé deux classifications de radios logicielles [54] [55] : une classification qui prend en compte le modèle de programmation et une deuxième qui considère le matériel utilisé pour le traitement en bande de base. Les tableaux 2.1 et 2.2 récapitulent les classes avec un nombre de solutions radio logicielles. De plus, nous avons compté plus de soixante solutions et plates-formes radio logicielles (une sélection est illustrée dans le tableau 2.3). Nous avons déterminé deux principaux paramètres de performances pour le choix d'une plate-forme. Il s'agit du taux d'échantillonnage et de la bande passante de radiofréquence d'un matériel front-end d'une plate-forme. Dans notre état de l'art, les valeurs de ces paramètres sont parmi les plus élevées pour un périphérique USRP [34] quand nous les comparons avec les autres solutions matérielles. Ce périphérique peut couvrir une bande de fréquence allant de 4 MHz à 6 GHz avec un taux d'échantillonnage allant jusqu'à 400 MS/s (Mega Samples per second). En outre, un USRP peut être connecté avec un GPP. Ce dernier, en exécutant des programmes GNU Radio, réalise une RL. La

GNU Radio est un logiciel libre facilement accessible sur le réseau Internet. À travers notre étude théorique, nous avons caractérisés les limites du matériel pour l'obtention d'une RL pure. La partie de FI de l'architecture générale est nécessaire pour la conversion numérique analogique (N/A) et surtout pour la conversion analogique numérique (A/N). La reconfiguration d'une radio (ou la flexibilité) a été identifiée comme l'avantage principal de cette architecture.

Cette partie de la thèse nous a permis de comparer les performances des plates-formes permettant le prototypage/implémentation d'une RL. Une plate-forme basée sur un GPP offre un accès facile à ses ressources logicielles et matérielles. La GNU Radio/USRP a montré, en plus de son architecture basée sur un GPP, des performances radios élevées par rapport aux autres plates-formes existantes. Néanmoins, ces performances n'ont pas été explorées par la communauté de recherche. De plus, cette plate-forme n'a pas été suffisamment documentée pour faciliter sa prise en main.

3 Analyse de la plate-forme Radio Logicielle USRP/GNU Radio

Le but de cette section est de résumer le Chapitre 3 consacré à l'architecture logicielle et matérielle de l'USRP/GNU Radio. Nous avons établi notre analyse en nous basant sur notre recherche bibliographique et notre rétro-ingénierie de la plate-forme. Par analogie avec l'architecture générale, le GPP en exécutant un graphe de flux GNU Radio représente la partie bande de base. Les fonctions des deux parties de FI et de front-end radiofréquence sont assurées par un USRP. Ce dernier est disponible en plusieurs versions avec différents performances. De même, différentes liaisons physiques sont possibles avec un ordinateur hôte (ou un GPP). Dans nos travaux d'expérimentations et d'implémentations, nous avons utilisé les deux versions USRP 1 et USRP N210. Le Tableau 3.1 récapitule les performances de ces deux USRPs.

La GNU Radio est exécutée par un ordinateur hôte. Une implémentation sous la GNU Radio reprend la conception de Mitola [39] qui consiste à définir une RL sous forme d'une chaîne de blocs. Chaque bloc permet une fonction de traitement de signal ou de données, de modulation et de démodulation numérique, etc. Ces fonctions sont disponibles dans une boîte à outils avec une possibilité d'intégrer de nouveaux blocs. Une chaîne de réception ou d'émission radio est décrite par un graphe de flux. Un graphe de flux est exécuté par un GPP. Il traite en entrée un signal en bande de base et génère en sortie un signal modulé. Il s'agit de l'interconnexion d'un ensemble de blocs logiciels. Ces derniers sont programmés, généralement en C++ et connectés via un script Python. La pile des langages de programmation a été détaillée avec sept couches dans la Figure 3.1. L'implémentation d'une RL se base principalement sur la maîtrise des deux langages Python et C++. Les blocs possèdent des ports en entrée, en sortie ou en entrée et en sortie. Les entrées et les sorties des blocs

sont des échantillons (ou *Samples* en anglais) ayant différents types de données, par exemple *Char*, *Short*, *Complex*, etc. Ces types définissent les tailles des échantillons et indiquent les types de ports d'entrée et de sortie d'un bloc. Le design d'un bloc se fait par une instanciation d'objet en C++ de quatre classes : `gr_sync_block`, `interpolator`, `gr_sync_decimator` et `gr_block`. Un bloc peut aussi être créé en instanciant un objet bloc de la classe `gr.hier_block2`. La connexion des blocs peut se faire facilement avec l'interface graphique de la GNU Radio. Des arcs connectent les blocs à condition que les types des ports des blocs soient compatibles. Cette condition permet à un flux numérique (une succession d'échantillons) de traverser ces blocs. En programmation, un graphe de flux peut être dérivé de deux classes : `top_block` du module `gr` ou `std_top_block` du module `stdgui2` (voir Figure 3.4). Un graphe de flux est exécuté soit via une commande sur un terminal ou via l'interface graphique. Pour éviter un temps de calcul supplémentaire nécessaire à l'affichage, il est recommandé d'utiliser un terminal de commande tout en évitant aussi les commandes d'affichage.

La GNU Radio est dotée d'un ordonnanceur qui ordonnance l'exécution des blocs en permettant une gestion de l'écoulement des échantillons du flux à travers ses blocs. Notre débogage de graphes de flux nous a montré l'existence de deux types d'ordonnanceurs : Single Threaded Scheduler (STS) et Thread-Per-Block Scheduler (TPS). Ce résultat est confirmé dans la littérature [40]. Le premier ordonnanceur alloue un seul thread à un graphe de flux. Par contre, le deuxième, comme son nom l'indique, alloue pour chaque bloc un thread d'exécution.

Afin de réduire et d'uniformiser le temps d'exécution des blocs à travers les différents types de processeur, une solution **S**ingle **I**nstruction **M**ultiple **D**ata (SIMD) a été proposée dans [77]. Cette solution est appelée **V**ector **O**ptimazed **L**ibraray of **K**ernels (VOLK). Elle permet à un bloc d'exécuter une seule instruction avec des vecteurs d'échantillons en entrée. La difficulté du VOLK réside dans la gestion des tailles des vecteurs avec la variation des temps d'inter-arrivée des échantillons.

Nous avons analysé les performances d'une implémentation radio logicielle sous GNU Radio/USRP. Ces performances dépendent du délai (ou *latency*) et du débit d'écoulement du flux (ou *throughput*). Le délai est engendré par la séparation via un lien physique entre l'ordinateur hôte qui exécute un graphe de flux et l'USRP. Ce délai est dû à la nature des graphes (une succession de blocs séparés). Pour réduire ce délai, des buffers sont intégrés entre les blocs logiciels et les composants matériels (entre les deux convertisseurs A/N, N/A et le FPGA, entre un USRP et un GPP). La Figure 3.10 schématise l'existence de buffers entre ces différents composants.

Le recours à des buffers introduit la notion de débit d'écoulement qui est défini par le nombre d'échantillons passant par un buffer durant une unité de temps. L'effet du délai et du débit sont pris en compte en premier lieu par l'ordonnanceur. Cet effet peut aussi être traité en imposant un paramètre de délai pour chaque bloc ou pour tout le graphe de flux [80]. De plus, des outils d'estimation du temps d'exécution des blocs sont proposés par la communauté [41]. Il s'agit d'un compteur de performances

(Performance counters et ControlPort). Ces deux outils sont utiles pour le débogage d'un graphe de flux et la détection des blocs provoquant le plus de temps de calcul. Dans le chapitre 3 l'architecture générale d'un USRP a été proposée en se basant sur la littérature [34] (voir Figure 3.8). La carte mère d'un USRP porte un FPGA, des cartes filles et convertisseurs A/N et N/A. Le rôle du FPGA est principalement de convertir la fréquence du signal en entrée en une fréquence adaptée au contrôleur de la liaison physique avec le GPP. Les cartes filles (en anglais *daughter boards*) sont les front-ends radio-fréquence de la radio logicielle. La bande de fréquence couverte est parmi les principaux paramètres d'une carte fille (voir Table 3.2). Un USRP est reconnu par un pilote logiciel appelé l'**USRP H**ardware **D**river (UHD). Ce dernier permet l'initialisation des paramètres radio par exemple : gain, fréquence centrale, taux d'échantillonnage, etc.

Notre analyse nous a apporté une compréhension de l'architecture logicielle de la GNU Radio. Les langages de programmation ont été organisés en couches pour plus de clarté. Nous avons mis l'accent sur la méthode à suivre pour implémenter une chaine de communication radio. Une implémentation radio logicielle est une interconnexion de blocs logiciels ou un graphe de flux. Nous avons aussi identifié les techniques d'ordonnancement d'exécution de ces blocs. Cet ordonnanceur peut profiter des outils comme le VOLK et de la présence de buffers entre les blocs logiciels. Les sources de génération de délai au niveau logiciel et matériel ont été identifiées. Par contre, les performances radiofréquences de l'architecture USRP n'ont pas été mesurées d'une façon précise. En particulier, la partie front-end de radiofréquence ou les cartes filles d'un USRP.

4 Mesures de performances radios sur des cartes filles des USRPs

Les travaux du Chapitre 4 sont motivés par l'obtention de résultats inattendus lors de tests effectués en utilisant des cartes filles des USRPs. L'interprétation de certains résultats a été difficile en raison du manque d'informations précises sur ces cartes. L'objectif de nos tests a été d'estimer le rapport signal sur bruit (ou **S**ignal-to-**N**oise **R**atio (SNR)) via un bloc de la GNU Radio. Cette dernière comporte déjà des estimateurs proposés dans la littérature [84] [85]. Nous avons simulé en boucle fermée une transmission via un canal radio. Ensuite, nous avons calculé des valeurs estimées du SNR (voir Figure 4.3). En se basant sur les résultats obtenus, un estimateur a été choisi pour calculer le **B**it **E**rror **R**ate (BER) en fonction du SNR (BER est l'erreur en bit) pour une modulation/démodulation **B**inary **P**hase-**S**hift **K**eying (BPSK) simulée et expérimentée en réel. Les résultats obtenu par simulation ont été en accord avec la théorie (voir Figure 4.5). Par contre, l'expérimentation réelle avec une carte fille RFX 2400 a donné des résultats non cohérents. La Figure 4.7 montre un nuage de points au-dessus de la limite théorique. Ce nuage de points a été obtenu avec

des valeurs de BER élevées pour des puissances radios de sortie faibles et vice-versa. Ce résultat a été interprété par l'inégalité entre la valeur de la puissance fixée dans un graphe de flux et la vraie puissance radio de sortie. Par ailleurs, Ettus Research [34] annonce des informations non détaillées de la puissance de sortie des cartes. La Figure 4.8 affiche les bandes passantes radiofréquence annoncées dans [34]. En outre, les travaux ayant traité les performances des USRPs dans [87] [88] ne se sont pas intéressés aux paramètres : bandes de fréquences et puissance de sortie des cartes filles. Cependant, nos travaux publiés dans [2] et [3] sont les premiers à avoir reporté des mesures sur ces deux paramètres. Notre caractérisation des performances des cartes filles est basée sur une approche expérimentale. L'objectif a été de calculer précisément pour des cartes filles les bandes de fréquences couvertes et la linéarité de puissance de sortie. Par conséquent, une puissance en sortie d'une carte a été mesurée pour des valeurs de fréquence centrale et de puissance fixées dans un graphe de flux. Les mesures ont été accomplies via deux outils : avec un analyseur de spectre et en utilisant la GNU Radio elle-même. Ces outils interceptent le signal en bande de base généré par un graphe de flux source. Nous nous sommes intéressés aux cartes couvrant des bandes ISM et des bandes spécifiées dans le standard IEEE 802.15.4. Les cartes filles explorées par nos mesures sont la RFX 2450, la RFX 900, la SBX et l'USRP B210 qui couvrent respectivement les bandes de fréquences [2.3 GHz, 2.9 MHz], [750 MHz, 1.050 GHz], [400 MHz, 4.4 GHz] et [70 MHz, 6 Ghz]. Ces intervalles sont annoncés par Ettus Research dans [34] (voir aussi Figure 4.8).
Dans les mesures accomplies via un analyseur de spectre, nous avons utilisé un oscilloscope LeCroy 640 Zi [89] (voir Figure 4.9). Ce dernier a été connecté via un câble coaxial [90] à un USRP N210 qui a porté les cartes RFX 2450, RFX 900 et SBX. La carte USRP B210 a été connectée directement à l'oscilloscope, puisque cette carte porte à la fois une carte mère et une carte fille. Par ailleurs, un graphe de flux source a été implémenté avec la possibilité de générer deux types de signaux : un signal sinusoïdal à bande étroite (ou porteuse) et un signal modulé BPSK (voir Figure 4.10). Dans ce graphe de flux, la puissance de ces signaux peut être ajustée via deux paramètres : DAC et UHD_G. Le DAC est défini dans un bloc séparé du graphe de flux. Le deuxième paramètre UHD_G est un gain radio défini dans le dernier bloc d'un graphe appelé USRP puits. Ces deux paramètres peuvent être modifiés par l'utilisateur. Nous avons établi la relation quadratique espérée entre la puissance mesurée réelle et la valeur du DAC introduite via le graphe de flux (voir équation (4.6)).
Les résultats de la linéarité de puissance de sortie en fonction des valeurs de DAC ont été obtenus pour les deux signaux : BPSK modulé et sinusoïdal. Cette linéarité est meilleure avec une carte RFX 2400 qu'une carte RFX 900 (voir Figures 4.12, 4.13 et 4.15). La relation quadratique espérée (équation (4.6)) est vérifiée pour la carte RFX 2400 et elle dévie légèrement avec une carte RFX 900. Les deux cartes RFX 2400 et RFX 900 couvrent respectivement seulement 24 % et 18 % des bandes de fréquences annoncées par Ettus Research. Précisément, une carte RFX 900 couvrant une bande

est de 72 MHz dans une fenêtre de puissance entre 0 et -30 dB contrairement aux 300 MHz annoncés. De plus, pour une carte RFX 2400 et dans la même fenêtre de puissance cette bande est de 168 MHz au lieu de 600 MHz. Ces résultats sont reportés dans les courbes des Figures 4.11 et 4.14. Pour les deux cartes SBX et l'USRP B210, les bandes de fréquences sont bien égales à celles annoncées. Néanmoins, la puissance de sortie diminue avec des valeurs de fréquences centrales élevées (voir les Figures 4.16 et Figures 4.22). La courbe obtenue dans la Figure 4.16 pour la carte SBX suit une forme modélisable par un modèle analytique. Par ailleurs, l'UHD$_G$ a été identifié comme un amplificateur supplémentaire pour les deux cartes SBX et USRP B210, son augmentation devrait se faire avec précaution. En effet, nous avons observé une distorsion du signal et une émission de puissance sur des fréquences harmoniques au-delà d'une valeur seuil de l'UHD$_G$ (voir Figures 4.18 et 4.19). Ces deux phénomènes ont été caractérisés par le calcul d'un **T**otal **H**armonic **D**istortion (THD). La Figure 4.17 montre des courbes des valeurs mesurées de THD en fonction des puissances de sortie sur des harmoniques avec l'augmentation de l'UHD$_G$.
Étant donné la forme modélisable des courbes obtenues pour des cartes SBX, nous avons effectué des mesures complémentaires sur d'autres cartes (au total quatre). Les résultats obtenus nous ont permis de proposer un modèle empirique par interpolation (voir Figures 4.24, 4.25, 4.26 et 4.27). Il s'agit de la définition d'une fonction qui prédit la puissance de sortie en fonction des variables du DAC, de l'UHD$_G$ et de la fréquence centrale. Ce modèle a été simplifié pour permettre sa réutilisation facile par la communauté ou pour son implémentation sous la GNU Radio (voir formule 4.11).
La deuxième partie des mesures a été accomplie uniquement avec des graphes de flux. L'objectif est la caractérisation de la puissance d'une carte par un graphe de flux en recevant une porteuse générée également par un graphe de flux. Pour cela, nous avons réutilisé le même graphe de flux source en émission. En réception, nous avons implémenté un graphe de flux capable de calculer l'énergie reçue sur une fréquence en utilisant une densité spectrale de puissance. Nous nous sommes limités dans ces mesures à l'usage des cartes RFX 2400 et SBX. En effet, une carte SBX a été utilisée en sortie et des cartes SBX et RFX 2400 en entrée. Les résultats obtenus, et ce pour les deux cartes, montrent l'effet du paramètre UHD$_G$ du récepteur (voir Figure 4.23). L'amplificateur sature avec un effet inverse de perte de puissance sur des fréquences harmoniques, si la valeur de l'UHD$_G$ est supérieure à 30 dB.
Nous avons apporté de nouvelles informations sur les performances d'un USRP. Ces informations ont contribué à la maîtrise de la réaction des cartes filles. Particulièrement, pour des implémentations radio logicielles sur la plate-forme USRP/GNU Radio.

5 Implémentations Radio Logicielles pour les RCSFs basés IEEE 802.15.4

Notre objectif est de présenter le standard IEEE 802.15.4 en première étape et après de le traduire sous forme d'implémentations radio logicielle. Nous avons présenté deux implémentations de deux couches PHYs pour les deux bandes de fréquences 2.4 GHz et 868/915 MHz. Nous avons détaillé les spécifications ou les attributs de ces deux couches en deux parties : des spécifications communes et d'autres propres à chaque bande. Le format d'un paquet et le type d'étalement de spectre **D**irect **S**equence **S**pread **S**pectrum (DSSS) sont les deux spécifications communes. L'occupation du spectre, les séquences **P**seudo **A**léatoires (PAs) du DSSS ainsi que la modulation sont des attributs propres à chaque couche. La modulation pour la couche de la bande 2.4 GHz est l'O-QPSK, alors que c'est la **D**ifferential-**P**hase **S**hift **K**eying (D-BPSK) dans la bande 868/915 MHz. Les plans d'allocation des canaux (ou fréquences centrales), les séquences PAs sont récapitulés dans Figures 5.1, Figure 5.2 et Tableaux 5.2, 5.3.

La couche de la bande 2.4 GHz a été implémentée sous la GNU Radio/USRP et présentée dans [35] [36]. Dans la littérature, nous n'avons pas trouvé de détails sur les graphes de flux représentant l'émetteur et le récepteur GNU Radio. De plus, au début de notre exploration, nous avons observé certaines difficultés en testant ces graphes de flux. Cependant, notre travail a apporté le maximum de détails sur les paramètres de configuration des graphes de flux. Ces détails ont servi, en partie, à proposer une nouvelle implémentation pour la couche PHY de la bande 868/915 MHz.

Les Figures 5.3 et 5.4 reportent respectivement les graphes de flux de l'émetteur et du récepteur de la couche PHY 2.4 GHz. L'émetteur est décrit allant du bloc de génération de paquets jusqu'au bloc puits d'envoi d'un signal modulé en O-QPSK. Le récepteur représente l'opération inverse commençant par une démodulation O-QPSK et finissant par une construction de paquets. Cette dernière étape est détaillée sous forme d'un algorithme divisé en trois sous-parties. La recherche du début ou du préambule de paquet est la première partie (voir l'Algorithme 1). La deuxième partie est la recherche et la construction de l'entête de paquet, elle est présentée dans l'Algorithme 2. La dernière étape compacte les fragments d'octets en donnant des paquets insérés dans une file d'attente. Les instructions de condition des trois algorithmes contrôlent la compatibilité du flux d'octets décodé avec la séquence PA correspondante (voir Tableau 5.2). Notre nouvelle implémentation est basée, principalement, sur la rétro-ingénierie de celle de la bande 2.4 GHz. Les graphes de flux de l'émetteur et du récepteur sont présentés respectivement dans la Figure 5.5 et la Figure 5.6 [1]. L'émetteur génère des paquets et construit une constellation des symboles en D-BPSK tout en suivant un étalement de spectre DSSS. Ce dernier reprend la séquence PA spécifiée par le standard pour la bande 868/915 MHz (voir Tableau 5.3). Inversement, le récepteur démodule le signal en première étape avec

un démodulateur BPSK. Le décodage différentiel est retardé pour la dernière étape de décodage des paquets pour construire à la fin un démodulateur D-BPSK. Le décodeur de paquets reprend le même principe de celui de la bande 2.4 GHz à quelques différences. La première partie de recherche du préambule construit des bits différemment en fonction de la séquence PA correspondante tout en appliquant un décodage différentiel. La construction d'entête et le compactage d'octets sont aussi repris avec l'usage de la séquence PA correspondante. Le fonctionnement des graphes des flux dans la bande 2.4 GHz a été testé via deux communications réelles de paquets de données. La première communication expérimente une émission et une réception entre les graphes de flux (ou USRP/GNU Radio configurations). Nous avons mesuré le taux de réussite d'envoi de paquet (**P**acket **S**uccess **R**ate (PSR)) en fixant l'ensemble des paramètres logiciels et matériels nécessaires. Ce taux varie en fonction de la taille des paquets (voir Figure 5.7). Nous avons trouvé qu'une augmentation de ce taux est liée aux taux d'échantillonnage à la réception/émission. La deuxième communication utilise des nœuds capteurs (ou motes Telos B et Raven stiks) et les graphes de flux de l'émetteur et du récepteur. Nous avons réussi à échanger des paquets en réception et en émission avec les nœuds capteurs. Les implémentations pour la bande 868/915 MHz ont été testées avec deux USRP 1 pilotées par les graphes de flux. Nous avons mesuré le taux d'erreur en bit et en paquet d'une communication (voir Figures 5.9 et 5.10). Une constellation des symboles a été obtenue dans la Figure 5.9.2. La perte de paquet augmente dans le cas d'un déphasage entre l'émetteur et le récepteur. En effet, cette désynchronisation peut être provoquée par deux sources : le déséquilibre entre les temps de calcul des graphes de flux et la gestion des buffers entre GNU Radio et USRP. Malgré ces difficultés et l'environnent indoor bruité des expérimentations, nos implémentations sont fonctionnelles.

6 RCSF cognitif basé sur le standard IEEE 802.15.4

Le problème de la rareté du spectre radiofréquence est une conséquence principale de la régulation et la rigidité des standards de télécommunication. La bande ISM est définie libre pour de nombreuses applications. Un réseau de capteurs sans fil basé sur le standard IEEE 802.15.4 partage la bande 2.4 GHz avec d'autres réseaux IEEE 802.11 b/g et IEEE 802.15.1. Les canaux de ces réseaux se chevauchent comme présentés dans la Figure 6.1. De plus, même si la nouvelle version du standard IEEE 802.15.4e [17] définie un mécanisme de saut de fréquences (ou saut de canal), les valeurs de ces fréquences sont statiques une fois attribuées. Cependant, un réseau de capteurs sans fil cognitif apporte une nouvelle solution au problème de la rareté du spectre.

Une radio cognitive existe concrètement si la radio peut écouter le spectre radio

et peut aussi accéder dynamiquement et en ligne à d'autres bandes de fréquences. Notre ADS suit un modèle ouvert de partage (ou en anglais *spectrum common*). Ce modèle est celui où tous les réseaux ont les mêmes droits en accédant à une bande de fréquences sans licence. Nous avons considéré un RCSF comme un **U**tilisateur **S**econdaire (US) et les autres réseaux comme des **U**tilisateurs **P**rimaires (UPs). Les deux implémentations pour les 2.4 GHz et 868/915 MHz bandes de fréquences ont été réutilisées.

Notre RCSF cognitif est composé d'un émetteur et d'un récepteur, où chacun est composé d'un nombre de graphes de flux. Le spectre radiofréquence comprend les deux bandes 2.4 GHz et 868/915 MHz. Par conséquent, le ADS peut conduire à un changement de graphe de flux de réception et d'émission. Cependant, le récepteur US est obtenu avec cinq graphes de flux (voir Figure 6.3). Il comporte un détecteur d'énergie, un récepteur/émetteur de messages avec un démodulateur/modulateur **G**aussian **M**inimum **S**hift **K**eying (GMSK) et les deux récepteurs pour les deux bandes de fréquences 2.4 GHz et 868/915 MHz. L'émetteur quant à lui comporte les émetteurs correspondants aux bandes de fréquences pour les mêmes graphes de flux de modulateur/démodulateur GMSK (voir Figure 6.4). L'écoute du spectre est basée sur une détection d'énergie effectuée uniquement par le récepteur. Dans notre configuration de radio logicielle, l'UP est un modulateur **O**rthogonal **F**requency-**D**ivision **M**ultiplexing (OFDM) de signal généré aléatoirement (voir Figure 6.5). Cet UP simule la modulation OFDM ou le perturbateur de notre réseau US (ou IEEE 802.15.4).

Notre détecteur d'énergie réutilise le graphe de flux proposé dans [113]. Ce dernier a été adapté et intégré au récepteur de l'US (ou IEEE 802.15.4). Le principe de la détection est basé sur l'estimation de la moyenne d'une densité spectrale de puissance de sortie (voir Figure 6.6). Il s'agit de recevoir un flux de signal et de le transformer en vecteurs de 512 échantillons complexes pour une fréquence donnée. Ensuite, chaque vecteur est traité dans une fenêtre Balkman-Harris avec une Transformée de Fourrier Rapide (ou **F**ast **F**ourier **T**ransform (FFT)). Cette fenêtre se déplace afin de balayer une bande de fréquence. L'énergie détectée sur une fréquence centrale est donnée par l'équation (6.1).

La sélection dynamique de fréquence porteuse est accomplie après l'exécution d'un algorithme de coordination par échange de messages. Cet algorithme est inspiré des requêtes automatiques de répétition (ou **A**utomatique **R**epeat re**Q**uest (ARQ)). Les Algorithmes 5 et 6 sont exécutés respectivement par le récepteur et l'émetteur USs. La sélection est effectuée par le récepteur après la détection de l'énergie minimale sur une bande de fréquence. Un seuil minimal d'énergie est fixé afin d'arrêter la détection dans un temps limité. La valeur initiale de ce seuil est définie à partir des expériences de détection précédentes. L'échange de message d'acquittement de changement de fréquence est assuré via les modulateurs/démodulateurs GMSK. Cette modulation est choisie, puisque sous la GNU Radio/USRP nous avons observé un taux de PSR élevé avec une modulation GMSK. Le récepteur envoie continuellement une requête

avec la valeur de la fréquence sélectionnée. Entre-temps, il attend un acquittement de l'émetteur avant d'envoyer de nouveau un message de confirmation indiquant qu'il est prêt à recevoir des données. Du côté de l'émetteur, l'attente est continuelle jusqu'à ce qu'il reçoive une requête. Cette réception déclenche l'envoi répétitif du message d'acquittement jusqu'à la réception d'une confirmation d'aptitude à recevoir des données. Ensuite, l'émetteur déclenche l'envoi des paquets de données IEEE 802.15.4.

Nous avons réalisé une expérimentation avec trois USRP 1 dans un environnement indoor. Deux USRP 1 sont dédiés à l'émetteur et au récepteur de notre RCSF cognitif (ou US). Un USRP 1 est avantageux, puisqu'il permet le montage de deux cartes filles avec pour chacune deux sorties antenne, une pour la réception (Rx) et une deuxième pour réception/émission (Rx/Tx). Nous avons utilisé quatre cartes filles SBX. Le détecteur d'énergie et le modulateur/démodulateur GMSK occupent une carte fille avec pour chacun, respectivement, les antennes Rx et RX/Tx. Par ailleurs, l'UP (modulateur OFDM) est joué par un troisième USRP 1.

Une fréquence de communication centrale est initialisée pour commencer l'expérimentation. Ensuite, le détecteur d'énergie balaye par division de 8 MHz dans la bande de fréquence 2.4 GHz. Cette division est due à la limitation en 8 MHz de la liaison USB 2 de l'USRP 1. D'autres paramètres hors-ligne et en-ligne sont définis, respectivement, avant l'exécution et au moment de l'exécution des graphes de flux. La bande passante d'un canal et le taux d'échantillonnage sont définis hors-ligne. Par contre, les paramètres en-ligne sont : la fréquence centrale, la tranche de balayage et le nombre de pistes (ou *bins* en anglais) d'une fenêtre FFT (voir Tableau 6.1). Une fenêtre FFT est définie par son début, sa taille et sa fin calculés respectivement, par les équations (6.2), (6.3) et (6.4).

Avant d'évaluer les performances de notre ADS, une caractérisation préalable a été effectuée sur ce spectre en détectant l'énergie reçue pour chaque fréquence (voir Figure 6.7 et 6.8). En effet, nous avons détecté plus de pics de puissance dans la bande 2.4 GHz par rapport à la bande 868/915 MHz. Après la caractérisation, nous avons considéré deux scénarios : avec un ADS et avec une définition statique d'un canal du standard, mais les deux ont été perturbés par un PU. La robustesse de l'ADS est mesurée par le taux PSR et le taux des paquets reçus (ou **P**acket **R**eceived **R**ate (PRR)). Ce dernier mesure le nombre de paquets reçus avec erreur par rapport à la somme totale des paquets envoyés. La Figure 6.9 et la Figure 6.10 montrent, respectivement, les résultats d'une communication sans l'ADS et avec l'ADS. Une amélioration de 80 % du PSR est obtenue avec notre ADS par rapport à un accès statique.

7 Conclusion et perspectives

Après la description résumée des différentes parties de la thèse, nous rappelons les contributions de notre travail comme suit :

- Notre état de l'art a suggéré deux classes de radio logicielles pour des périphériques embarqués. Il s'agit d'une classe d'architecture avec un GPP et une autre avec un matériel reconfigurable. De plus, nous avons synthétisé d'autres classifications avec une étude comparative des performances des plates-formes.

- Nous avons apporté une caractérisation de la plate-forme GNU Radio/USRP. La GNU Radio a été analysée avec une description de l'architecture logicielle et du modèle en graphe de flux d'une implémentation radio logicielle. Nous avons identifié les techniques d'ordonnancement et les sources de délais d'exécution de ces graphes. En outre, l'architecture SIMD (ou VOLK), les outils ControlPort et compteur de performances ont été étudiés et présentés.

- Nos mesures de performance sont les premiers travaux de recherche en rapport avec les cartes filles RFX 2400, RFX 900, SBX et USRP B210 des USRPs. Les bandes de fréquences et les puissances de sorties de ces cartes ont été mesurées via une méthode expérimentale. Des cartes SBX et USRP B210 ont été trouvées avec des bandes de fréquences larges compatibles avec les intervalles annoncés par Ettus Research [34]. Néanmoins, la puissance de sortie de ces deux cartes diminue en augmentant la fréquence. Un modèle empirique reprend cette relation afin de prédire la puissance de sortie d'une SBX en fonction des amplificateurs logiciels. Par ailleurs, les cartes RFX 2400 et RFX 900 couvrent, respectivement, seulement 24 % et 18 % des bandes annoncées. Ces résultats sont utiles pour l'implémentation de radios logicielles en utilisant la GNU Radio/USRP.

- Nous avons implémenté une nouvelle couche PHY possible pour un RCSF basé sur le standard IEEE 802.15.4. Des graphes de flux ont été proposés pour la bande de fréquences 868/915 MHz. De plus, la couche PHY pour la bande 2.4 GHz a été détaillée via une rétro-ingénierie de son implémentation sous forme de graphes de flux. Ces graphes sont fonctionnels pour ces deux bandes, puisqu'ils ont été testés via une communication réelle de paquets de données avec des nœuds capteurs.

- Nous avons proposé un ADS en exploitant les deux couches PHYs IEEE 802.15.4 implémentées. Cet ADS est l'étape la plus importante vers le développement d'un réseau de capteur sans fil cognitif. Ce réseau a été considéré comme un US des deux bandes 2.4 GHz et 868/915 MHz. Nous avons réussi à assembler dans un nœud du réseau un ensemble de graphes de flux : modulateurs et démodulateurs GMSK de messages, les deux implémentations pour le standard

IEEE 802.15.4 et un détecteur d'énergie. La synchronisation et la coordination de changement de fréquences et de graphes de flux ont été assurées via un algorithme avec échange de messages. Notre ADS a été expérimentée à travers une communication réelle de paquets dans un environnement indoor. Le PSR mesuré de cette communication est amélioré de 80 % avec notre ADS par rapport à un accès statique.

Les principales perspectives de notre thèse sont au nombre de trois :

- Nous pourrions améliorer la synchronisation des graphes de flux émetteur et récepteur. Une estimation précise du temps d'exécution des blocs de graphe est possible à travers les outils : compteur de performances et ControlPort.
- Pour la couche PHY proposée, les tests peuvent être étendus à un échange de paquets avec un émetteur/récepteur matériel.
- D'autres couches PHYs du standard IEEE 802.15.4e peuvent être explorées. Nous pouvons considérer celle avec une modulation **D**ifferential-**Q**uadtrature **P**hase **S**hift **K**eying (D-QPSK) et un étalement de spectre **C**hirp **S**pread **S**pectrum (CSS) pour la bande 2.4 GHz.

Contents

1 Introduction et objectifs 5
2 État de l'art sur la Radio Logicielle 7
3 Analyse de la plate-forme Radio Logicielle USRP/GNU Radio 9
4 Mesures de performances radios sur des cartes filles des USRPs 11
5 Implémentations Radio Logicielles pour les RCSFs basés IEEE 802.15.4 14
6 RCSF cognitif basé sur le standard IEEE 802.15.4 15
7 Conclusion et perspectives 18

1 Introduction **31**
1 Historical Elements and Context 32
2 Definitions 35
2.1 Software Radio and Software Defined Radio 35
2.2 Cognitive Radio 35
2.2.1 Spectrum Sensing 36
2.2.2 Dynamic Spectrum Access 36
2.3 Wireless Sensor Networks 37
2.3.1 IEEE 802.15.4 standard 37
3 Thesis motivations 38
3.1 Context of WSNs 38
3.2 Applications of WSNs and SDRs 40
4 Thesis objectives 41
4.1 SDR platform to implement standardized PHY layer of WSNs 41
4.2 Cognitive Radio for Spectrum Scarcity 42
5 Thesis Organization and Contributions 43
5.1 State of the art on Software Defined Radio 43
5.2 Analysis of GNU Radio and experimental measurements on USRP's Daughter boards 44
5.3 SDR implementations of IEEE 802.15.4 standard 45
5.4 Cognitive Wireless Sensor Network based on IEEE 802.15.4 . 45

2 State of the art on Software Defined Radio-SDR **47**
1 Introduction 48
2 Typical architecture of an SDR 48
2.1 SDR Receiver (Receiver (Rx)) 49
2.2 SDR Transmitter (Tx) 50
3 Features and Challenges of SDR 50
3.1 Features 50
3.1.1 Reconfigurability 50
3.1.2 Portability 51

3.1.3 Interoperability . . . 51
3.2 Challenges . . . 51
3.2.1 Handling of an SDR platform . . . 51
3.2.2 Hardware physical dimensions . . . 52
3.2.3 Radio frequency performances . . . 52
3.2.4 Baseband processing hardware . . . 53
4 SDR standards and architectures . . . 53
4.1 Software Communication Architecture SCA . . . 53
4.1.1 **O**pen **S**ource **S**CA **I**mplementation::**E**mbedded (OSSIE) . . . 54
4.2 Reconfigurable Radio System RRS . . . 54
5 SDR for Embedded Devices . . . 56
5.1 GPP based architecture . . . 56
5.2 Reconfigurable hardware based architecture . . . 56
6 SDR classifications . . . 57
6.1 Programming model . . . 57
6.2 Used hardware . . . 59
6.3 SDR platforms . . . 61
7 Summary . . . 64

3 Analysis of GNU Radio and USRP SDR 65
1 Introduction . . . 66
2 GNU Radio . . . 66
2.1 Programming language layers . . . 67
2.2 Software blocks . . . 67
2.3 Flow graphs . . . 69
2.4 Software scheduler . . . 70
2.5 SIMD programming (Volk) . . . 71
3 Universal Software Radio Peripheral . . . 72
3.1 USRP Architecture . . . 74
3.2 Transmit and Receive Paths . . . 74
3.3 RF daughter boards . . . 75
3.4 Firmware and FPGA images . . . 75
3.5 Universal Hardware Driver (UHD) . . . 76
4 GNU Radio and USRP properties . . . 76
4.1 Latency and throughput . . . 77
4.2 Buffers organization . . . 78
4.3 Performance counters and ControlPort . . . 78
5 Advantages of GNU Radio and USRP . . . 79
6 Summary . . . 80

4 Radio Frequency Measurements on USRP Daughter boards 83
1 Introduction . . . 84
2 Problem statement . . . 84
2.1 An overview of BPSK modulation . . . 85
2.1.1 The BER and SNR parameters . . . 85
2.1.2 BER/SNR estimators on GNU Radio simulation . . . 86
2.1.3 BER/SNR estimators in real experiment . . . 88
2.2 Related Works . . . 90
3 Experimental approach . . . 91
3.1 Hardware Setup . . . 91
3.2 Software Setup . . . 92
3.2.1 The expected DAC vs ouput power relationship . . . 93
4 Spectrum Analyzer measurements . . . 93
4.1 RFX2400 Daughter board . . . 93
4.1.1 Frequency bandwidth . . . 94
4.1.2 Output power versus DAC value . . . 95
4.2 RFX900 Daughter board . . . 95
4.2.1 Frequency bandwidth . . . 97
4.2.2 Output power versus DAC value . . . 97
4.3 SBX Daughter board . . . 98
4.3.1 Frequency bandwidth . . . 99
4.3.2 Total Harmonic Distortion (THD) . . . 99
4.3.3 Output power versus DAC value . . . 101
4.4 MIMO USRP B210 . . . 103
4.4.1 Frequency bandwidth . . . 104
5 Measurements through flow graphs . . . 104
5.1 RFX and SBX Daughter boards . . . 105
6 Empirical model for SBX daughter boards . . . 106
7 Summary . . . 109

5 SDR implementations for IEEE 802.15.4-based WSN 113
1 Introduction . . . 114
2 Problem statement . . . 114
3 Related works . . . 116
4 IEEE 802.15.4 PHY layers . . . 118
4.1 Common specifications for 868/915 MHz and 2450 MHz PHY layers . . . 118
4.2 2450 MHz specifications . . . 119
4.3 868/915 MHz specifications . . . 120
5 Software Implementations . . . 121
5.1 Software transmitter/receiver for 2450 MHz PHY . . . 121
5.1.1 Tx flow graph . . . 121

5.1.2 Rx flow graph . . . 122
5.1.3 Packet decoder . . . 123
5.2 Software transmitter/receiver for 868/915 MHz PHY . . . 124
5.2.1 Tx flow graph . . . 127
5.2.2 Rx flow graph . . . 128
5.2.3 Packet decoder . . . 129
6 SDR communications for 2450 MHz . . . 131
6.1 Communications between two SDRs . . . 131
6.2 Communications between sensor motes and SDRs . . . 132
7 SDR communications for 868/915 MHz . . . 133
8 Summary . . . 136

6 Cognitive Wireless Sensor Network based on IEEE 802.15.4 139
1 Introduction . . . 139
2 Problem statement . . . 140
3 Related works . . . 142
3.0.1 Related specifications . . . 142
3.0.2 Related implementations . . . 143
4 Dynamic spectrum access on GNU Radio USRP SDR . . . 143
4.1 Reconfigurable SDR settings . . . 144
4.2 Energy Detector . . . 146
4.3 Dynamic frequency selection . . . 146
5 Experiments and results . . . 148
6 Summary . . . 152

7 Conclusions and Future Work 153
1 Conclusions . . . 153
2 Future Work . . . 156

Glossaries 159
Acronyms . . . 159
Glossary . . . 166

List of Figures

1.1 The opportunity to replace Wireless communications devices by Software Defined Radio . . . 32
1.2 The crowded frequency spectrum from 300 MHz to 3 GHz in USA [9] 34
1.3 Interaction stack of CR, SDR, DSA and SS . . . 36

2.1 Ideal architecture of a Software Radio . . . 49
2.2 General architecture of SDR . . . 49
2.3 SDR receiver block diagram . . . 49
2.4 SDR transmitter block diagram . . . 50
2.5 Version 4 of the SCA architecture [49] . . . 54
2.6 ETSI architecture [51] . . . 55
2.7 Hardware tasks associated with process intensity and flexibility . . . 61

3.1 Software layers of the GNU Radio . . . 67
3.2 Source, Sink and Intermediate blocks . . . 68
3.3 C++ signal processing modules . . . 68
3.4 Programming layers of GNU Radio implementation . . . 69
3.5 An example of a flow graph . . . 70
3.6 GNU Radio Software layers [78] . . . 70
3.7 VOLK programming model . . . 72
3.8 General USRP Architecture with a daughter board (USRP N210 with WBX) [37] . . . 75
3.9 UHD in GNU Radio/USRP . . . 76
3.10 Latencies between GPP, FPGA and DAC/ADC . . . 77
3.11 Latencies between Software blocks . . . 78
3.12 Buffers in GNU Radio USRP SDR . . . 78
3.13 Controlport clients with GNU Radio applications over a TCP connection 79
3.14 An example of the performance's counters graph of a given flow graph 79

4.1 Constellation diagram for BPSK modulation . . . 85
4.2 Loop back flow graph for SNR estimation . . . 87
4.3 Known theoretical SNR compared to the estimated ones through GNU Radio and python . . . 87
4.3.1 Simple estimator of the SNR. . . . 87
4.3.2 Skew estimator of the SNR. . . . 87
4.3.3 M2M4 estimator of the SNR. . . . 87
4.3.4 SVR estimator of the SNR. . . . 87
4.4 Loop back flow graph for BER estimation . . . 88

4.5 Compared BER versus Eb/N_0 obtained under simulation and theoretical processing . . . 88
4.6 Two USRP 1 connected to a host computer which run a BPSK modulator/demodulator flow graph . . . 89
4.7 BER versus SNR obtained with a BPSK modulation on RFX2400 daughter board . . . 89
4.8 Daughter boards frequency band coverage . . . 90
4.9 Experimental setup for the measurements . . . 92
4.10 A simplified sinusoidal and BPSK transmitter flow graph . . . 93
4.11 Measured bandwidth of the RFX2400 daughter board. The reference level of 0 dB was taken at 2434 MHz. Units are logarithmic (dB). . . . 94
4.12 The average output power of the RFX2400 versus the DAC value for 6 frequencies (unmodulated carrier). The $\langle P_{out} \rangle \sim \text{DAC}^2$ law is closely followed. . . . 96
4.13 The average output power of the RFX2400 versus the DAC value for 3 frequencies in the cases: unmodulated carrier and BPSK transmission. The $\langle P_{out} \rangle \sim \text{DAC}^2$ law breaks down for BPSK at DAC = 0.7. . . . 96
4.14 Measured bandwidth of the RFX900 daughter board. The reference level of 0 dB was taken at 910 MHz. Units are logarithmic (dB). . . . 97
4.15 The average output power of RFX900 versus the DAC value at 900 MHz (unmodulated carrier). . . . 98
4.16 Measured bandwidth of the SBX daughter boards for UHD_G of 0, 10, 20 dB. The small oscillation between 2.2 and 3.4 GHz is probably due to a slight mis-adaptation in the SBX board. . . . 99
4.17 THD measured at carrier frequencies $f_1 = 600$ MHz, 900 MHz and 2400 MHz. . . . 100
4.18 Time domain waveform at 600 MHz for UHD_G from 10 to 40 dB. Beyond the UHD gain value of 20 dB, the waveform becomes heavily distorted. . . . 101
4.19 The measured output power for a carrier frequency $f_1 = 600$ MHz and $f_1 = 900$ MHz versus the UHD_G. . . . 102
4.19.1 $f_1 = 600$ MHz. The second harmonic at $f_2 = 1800$ shows a sharp increase between 20 and 30 dB, partially explaining the increase in the THD. . . . 102
4.19.2 $f_1 = 900$ MHz. . . . 102
4.20 The output power on the carrier frequency and the total output power versus the DAC value for $\text{UHD}_G = 15$ and $\text{UHD}_G = 20$ dB at $f_1 = 900$ MHz. The square law from Equation 4.6 is closely followed for a UHD_G of 15 dB, however it breaks down for a gain of 20 dB and for DAC ≥ 0.7. . . . 103
4.21 A simplified sinusoidal MIMO flow graph . . . 103

4.22 The measured frequency bandwidth of the MIMO B210 USRP board for different UHD_G values . . . 104
4.23 Output power results measured over SBX and RFX 2400 daughter boards within the GNU Radio USRP SDR . . . 106
4.23.1 Relative average of output power measured with RFX daughter board. . . . 106
4.23.2 Relative average of output power obtained with SBX daughter boards. . . . 106
4.24 Average output power versus carrier frequency for UHD_G gains of 0, 10, 20 dB for the four measured SBX boards. . . . 107
4.25 The average output (in mW) for the four measured SBX boards and the average output power (thick line) versus the empirical model given by Equation (4.10). . . . 108
4.26 The average output (in mW) for the four measured SBX boards versus the empirical model Equation (4.11) (converted to mW). . . . 109
4.27 The average output power (in dB) versus the empirical model from Equation (4.11) for $UHD_G = 0$ (lower curve), $UHD_G = 10$ (moddle curve) and $UHD_G = 20$ (upper curve). . . . 110

5.1 IEEE 802.15.4 packet structure and size . . . 119
5.2 Channel allocation in 868/915 MHz and 2450 MHz . . . 119
5.3 Transmitter (Tx) flow graph for 2450 PHY layer . . . 122
5.4 Receiver (Rx) flow graph for 2450 PHY layer . . . 123
5.5 Transmitter (Tx) flow graph for 868/915 PHY layer . . . 128
5.6 Receiver (Rx) flow graph for 868/915 PHY layer . . . 129
5.7 The PSR and received packet rates versus pocket size using SDR implementation for 2450 MHz . . . 132
5.8 TelosB and Raven sticks . . . 133
5.9.1 Power spectrum of our software transceiver recorded with the USRP and drawn by FFT gnuradio plot. . . . 134
5.9.2 Receiver symbol constellations. . . . 134
5.9 The BER versus received SNR for central frequency 868.3 MHz and for the MFB . . . 135
5.10 The PER versus SNR using two central frequencies 916 MHz and 868 MHz . . . 135

6.1 Overlapping of IEEE 802.15.4 channels with that of IEEE 802.11b/g in 2450 MHz band . . . 141
6.2 Basic cognitive cycle [15] . . . 141
6.3 Software chain of SU receiver (Rx) . . . 145
6.4 Software chain of SU transmitter (Tx) . . . 145
6.5 Flow graph of PU transmitter (Tx) . . . 145
6.6 Flow graph of our energy detector based spectrum sensing . . . 146

6.7 Spectrum Sensing of frequency band 2.4 GHz to 2.5 GHz 149
6.8 Spectrum Sensing of frequency band 850 MHz to 950 MHz 150
6.9 Packet Success Rate (PSR) and Packet Received Rate (PRR) function of spectrum distance between PU and SU without **D**ynamic **S**pectrum **A**ccess (DSA). 151
6.10 Packet Success Rate (PSR) and Packet Received Rate (PRR) function of spectrum distance between PU and SU with DSA. 152

List of Tables

2.1 SDR classification refereed to a programming model 59
2.2 Synthesised classification of SDR hardware given in [56] based on the DSP architecture 60
2.3 Non-exhaustive list of **S**oftware **D**efined **R**adio (SDR) platforms . . . 63

3.1 USRPs and their performances [37] 74
3.2 Some daughter boards and their performances [37] 76

4.1 Output Power at high frequencies and high UHD_G values 101

5.1 Synthesized specifications of IEEE 802.15.4 [19], [20] 115
5.2 Symbol-to-chip mapping for the 2.4 GHz band 120
5.3 Symbol-to-chip mapping for the 868/915 MHz band 121
5.4 Parameters of packet transmissions between two SDRs 131

6.1 Parameters of energy detector 149

Chapter 1

Introduction

The fundamental problem of communication is that of reproducing at one point, either exactly or approximately, a message selected at another point.

Claude Shannon

Contents

1 Historical Elements and Context 32

2 Definitions 35

2.1 Software Radio and Software Defined Radio 35

2.2 Cognitive Radio 35

2.3 Wireless Sensor Networks 37

3 Thesis motivations 38

3.1 Context of WSNs 38

3.2 Applications of WSNs and SDRs 40

4 Thesis objectives 41

4.1 SDR platform to implement standardized PHY layer of WSNs 41

4.2 Cognitive Radio for Spectrum Scarcity 42

5 Thesis Organization and Contributions 43

5.1 State of the art on Software Defined Radio 43

5.2 Analysis of GNU Radio and experimental measurements on USRP's Daughter boards 44

5.3 SDR implementations of IEEE 802.15.4 standard 45

5.4 Cognitive Wireless Sensor Network based on IEEE 802.15.4 . 45

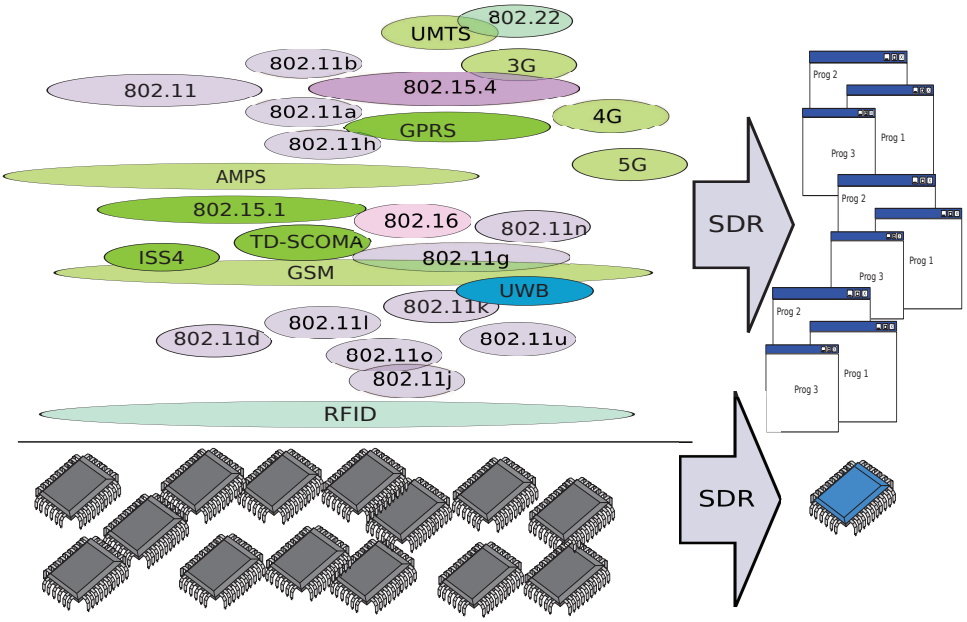


Figure 1.1: The opportunity to replace Wireless communications devices by Software Defined Radio

1 Historical Elements and Context

The birth of wireless technology is traced back to an equation published by James Maxwell around 1865 [7], which caused Heinrich Hertz to prove the existence of electromagnetic waves. Around the beginning of the 20th century, Marconi established the first long-distance communication via a radio telegraph. During the following five decades, analog radio communication was brought to perfection with different analog modulations. Shannon and Nyquist launched a big change in radio development since they publish the Nyquist-Shannon sampling theorem [8]. This theorem states that a perfect signal reconstruction is possible when the sampling rate (or sampling frequency) is at least twice the signal bandwidth being sampled. The digital signal processing used in data transmission is a result of transistors and integrated circuits on chip computing. It ensures a radio link or wireless connections between electronic devices. Commonly, the radio interconnection of these devices (or nodes) are the basis of wireless networks.

The wireless networks support several applications in our modern life, for example at home, office and car. Each network answers to particular end-user needs, *e.g.* internet connection, mobile phone communications and home automation, etc. In order to answer to some questions related to networking; standard organizations and groups have been formed such as the **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers (IEEE), the **I**nternet **E**ngineering **T**ask **F**orce (IETF) and the **I**nternational **T**elecommunication **U**nion (ITU). They impose specifications in order to ensure the interoperability between developed wireless devices.

Figure 1.1 shows the growing number of wireless networks and technologies over the last twenty years. The end-user expectations, industrial constraints and market opportunities are the main driving force of this significant progress. Diverse applications can be supported, such as voice, video, and data communications. Each one needs an appropriate hardware air-interface supporting specific radio techniques and protocols. In fact, each application has its packet structures, data types, and signal processing techniques, and each radio has to communicate and decode signals using a dedicated circuitry, *e.g.* one Smartphone contains 3G, 4G, GSM, Bluetooth and Wifi modules. In addition, hardware manufacturers wish to develop quickly and cheaper new wireless technologies. Moreover, radio modulation techniques are static, since they are implemented in hardware; neither radio designer nor the radio itself can change these techniques without replacing the hardware. The physical layer (PHY layer) of wireless networks uses these hardware radios, and cope with their limitations. Furthermore, with the high number of networks and technologies, the radio-frequency spectrum is scarce and radio communications can interfere and leading to a radio performances degradation. As shown in Figure 1.2, the radio-frequency spectrum in USA is crowded by the large number of wireless technologies and networks.

Wireless Sensor Networks (WSN) is a particular case with high number of applications. It also requires self-awareness of its environment and self-adapting of its radio parameters to improve its performances.

Form the above limitations, a virtual definition of radio transceivers (transmitter and receiver) could be an suitable solution. The objective is to substitute the signal processing objects and operation with software running on computer machine. However, writing a set of programs for reproduce signal processing is more efficient, since software can be customized more easily than hardware. Thus, the ideal solution for an industrial or a standard organization is to have existing wireless technologies defined in software running a top a common hardware. As shown in Figure 1.1, **S**oftware **D**efined **R**adio (SDR) can substitute a traditional hardware radio by a software one.

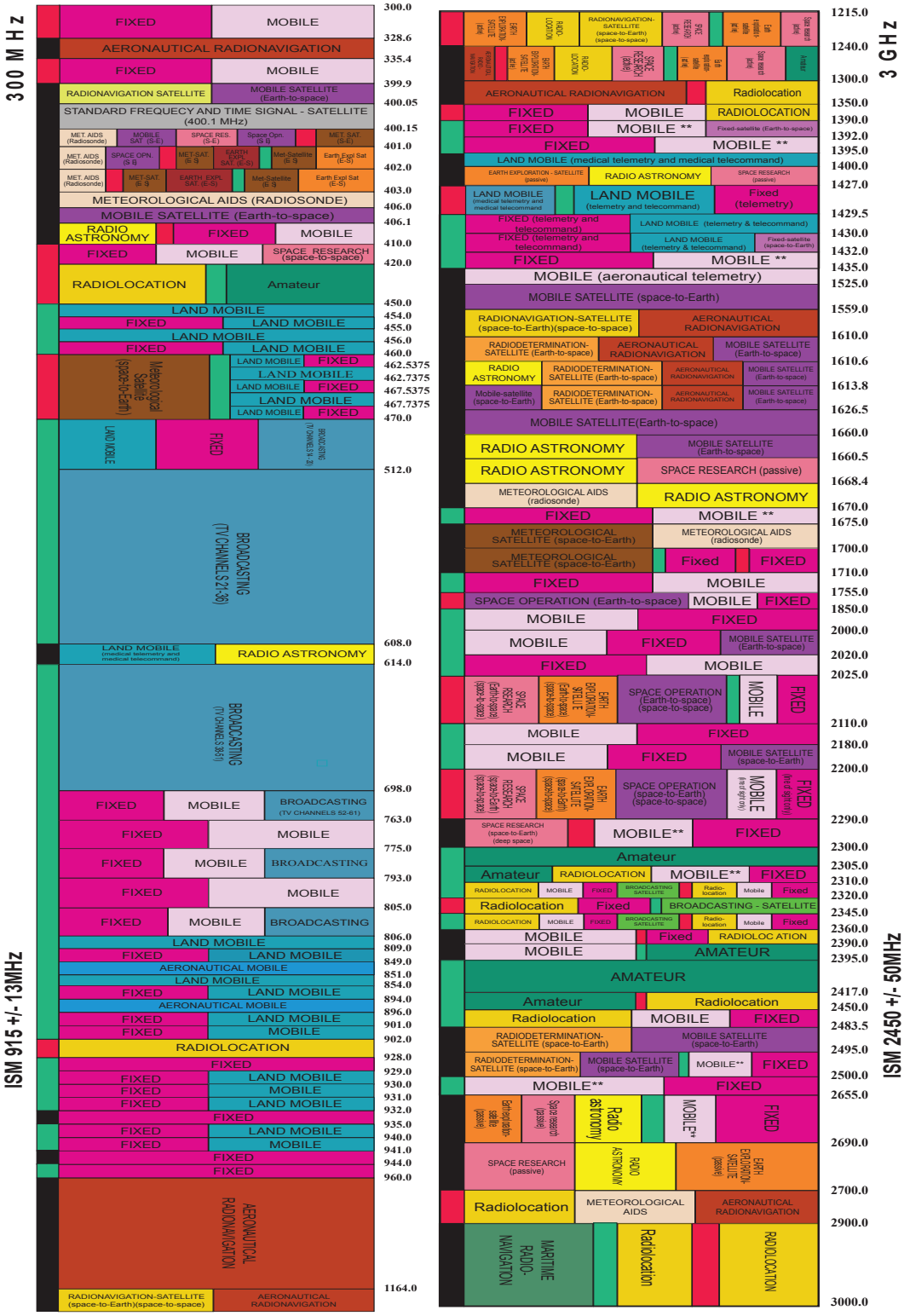


Figure 1.2: The crowded frequency spectrum from 300 MHz to 3 GHz in USA [9]

2 Definitions

Before giving the motivations behind this thesis and its contributions, we introduce the primary involved concepts.

2.1 Software Radio and Software Defined Radio

Software Radio (SR) is a set of technologies for defining radio transceiver parameters and functions in software, including carrier frequency, modulation bandwidth and frequency/space/time/code agility [10] [11] [12]. An ideal SR technology refers to the complete software control of the entire system. The aim of SR is to have an analog conversion only at antennas, ensuring the support for a wide frequency band.

SDR is a realizable implementation of Software Radio, as defined above; it is a reconfigurable radio, in which the radio functionalities are defined as much as possible in software. The SDR should provide a radio architecture which allows us changing these functionalities in real-time. The formal definition of SDR is given by ITU [13] as:

*"A radio transmitter and/or receiver employing a technology that allows the RF **R**adio **F**requency (RF) operating parameters including, but not limited to, frequency range, modulation type, or output power to be set or altered by software."*

Mitola [12] argues for the substitution of chains of circuitry signal processing systems by software processing blocks. We give more details about that objective in the next chapter.

2.2 Cognitive Radio

The SDR capabilities provide an opportunity for a system designer to build a new intelligent radio system with different functions. The **C**ognitive **R**adio (CR) is a conceptual layer over SDR. It is an abstraction layer to program the SDR satisfying application and user requirements. CR intends to describe an intelligent radio that can autonomously make decisions using Radio Frequency environment information. The intelligence of the CR is defined by the level of self-awareness to the environment and user requirements. The **F**ederal **C**ommunication **C**ommission (FCC) introduces it as:

"a radio that can change its transmitter parameters based on the environment in which it operates." [14]

In the research community, Haykin [15] defines CR as a radio capable of being aware of its surroundings, learning, and adaptively change its operating parameters in real-time. According to this definition, the objective is to provide reliable wireless communications, most of the time, anywhere, and spectrally efficient. The cognitive

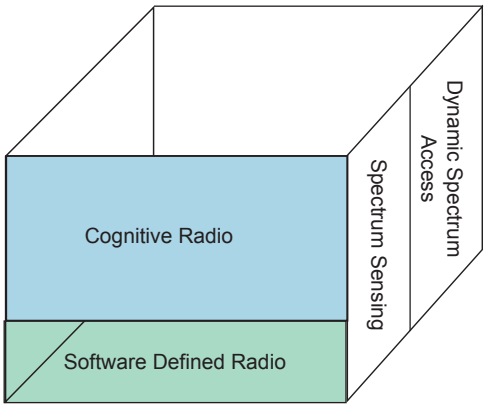


Figure 1.3: Interaction stack of CR, SDR, DSA and SS

entity is a node in a network managed locally or globally depending on the problem definition. In this thesis, we consider the network as a unique entity.

2.2.1 Spectrum Sensing

Spectrum Sensing (SS) is a very important component in order to establish a cognitive radio. It is a mechanism to get awareness about the spectrum usage in a geographical area. The mechanism consists in detecting radio frequency activities in a given spectrum. By default, only licensed users (or Primary Users (PUs)) are specified to occupy a spectrum. Opportunistic users (or Secondary Users (SUs)) accomplish a spectrum sensing for cognitive radio operations. The conventional spectrum sensing exploits only three dimensions of the spectrum space: frequency, time, and space. Commonly, it is treated as a detection and an estimation of the signal problem.

2.2.2 Dynamic Spectrum Access

Dynamic Spectrum Access (DSA) is also an important operation in a cognitive radio. It is the spectrum sharing paradigm that allows SUs to solve the inefficiency of spectrum usage. The Dynamic Spectrum Access Networks (DySPAN) [16] standards committee claims that:

"DSA is real time adjustment of spectrum utilization in response to changing circumstance and objectives".

In fact, SUs use the unused frequency spectrum based on returned information of spectrum sensing.

We summarize the interaction of CR with SDR, DSA and SS by a conceptual stack shown in Figure 1.3. The base of Cognitive Radio is Software Defined Radio allowing Dynamic Spectrum Access and Spectrum Sensing operations.

2.3 Wireless Sensor Networks

A **W**ireless **S**ensor **N**etwork (WSN) consists of distributed wireless nodes used to measure and communicate physical information [17]. The nodes can measure, process and communicate sensed data to a sink node or a base station through a wireless ad-hoc network[1]. The sensor nodes communicate their data via radio in a single or multiple hops manner. Generally, they are deployed in a hostile environment and typically for event-driven applications. When an event occurs, the source nodes sense, generate and communicate data packets. WSNs applications are constrained by the characteristics of sensor nodes, which are limited in power processing, memory resources and in their residual energy.

Commonly, every network is defined by its conceptual stack of layers or **O**pen **S**ystems **I**nterconnection (OSI) model. Seven layers features an OSI model: Physical (PHY), Link (with **M**edium **A**ccess **C**ontrol (MAC) sub-layer), Network, Transport, Session, Presentation and Application [18]. Their main objectives are to ensure the interoperability of diverse networks and the modularity of development. WSN can be featured by mainly four layers, PHY, MAC, Network and Application. Of course, the application layer covers the upper ones of the OSI model. Each layer serves the upper one, by performing techniques and/or protocols, which are commonly standardized by competent organizations. IEEE, ITU, IETF are well-known standard organizations.

2.3.1 IEEE 802.15.4 standard

The suffix number of each IEEE standard defines a working group (or committee) on a particular network, *e.g.* 802.11 for Wifi and 802.15.1 for Bluetooth networks and 802.15.4 for WSN. IEEE 802.15.4 is a standard specified for PHY and MAC layers of **L**ow-rate **W**ireless **P**ersonal **A**rea **N**etwork (LR-WPAN). The IEEE is an association which approves specifications of several communication protocols.

The IEEE 802.15.4 standard is designed for data communication devices using low data-rate, low-power, and low-complexity short-range radio frequency (RF) transmissions in a wireless personal area network (WPAN). However, many WSN implement standard specifications, regarding node's capabilities. The IEEE 802.15.4 has been enhanced from the first version of 2003 [19] to that of 2012 IEEE 802.15.4e [20]. The main contribution of this last version is introducing a channel-hopping mechanism in addition to old specifications of the standard.

Zigbee is only an example of a suitable product available commercially. Zigbee Alliance organization proposes the two network and application layers, above the IEEE 802.15.4 standard. Obviously, other protocols can be defined for specific applications, such as IEEE 802.15.5 standard [21], WirelessHART [22], ISA100 [23] and 6LoWPAN [24].

[1]The nodes of the network forward data dynamically based on the network connectivity

3 Thesis motivations

The motivations are primarily related to the WSN and SDR issues. We regroup both issues and motivations from the context of WSNs to the interesting applications of WSNs and SDRs.

3.1 Context of WSNs

Several research works deal with network layer issues for Wireless Sensor Networks. Data routing under energy constraints and node positioning in an indoor environment are two active research problems [25] [26]. Some solutions have been proposed based on PHY layer parameters. For the data routing problem, the objective is to define for each node an efficient metric. The latter can be based on nodes residual-energy and energy consumed for each communication mode, *i.e.* transmission, reception and idle mode. In general, the efficiency of the data routing is measured through simulators, *e.g.* NS-2 simulator, OPNET, OMNET, etc. However, the obtained results depend on the ability to manipulate available PHY parameters, in particular the energy model. Numerous energy consumption models have been defined, each one, for a specific simulator [27]. The problem of node localization in an indoor environment has been addressed in some works using **R**eceived **S**ignal **S**trength **I**ndication (RSSI) parameter [26]. The latter can be calculated by each sensor node in a real world network. Each node estimates the distance between its position and those of other neighboring nodes. The nodes calculate this distance when they receive a signal, and after they estimate its output power. Commonly, the obtained positions are with notable variations due to the interference, fading and multi-path in an indoor environment [28]. The effectiveness of the calculated positions is related to the measured output power, which is defined by default in the hardware transceiver of nodes.

These two research problems, *i.e.* energy-aware data routing and localization in WSN, showed that the PHY layer's parameters are, at the same time, essential and useful for OSI model's upper layers. The output power of a transmitted signal is managed by the PHY layer. Since it features the node's energy consumption, its capture by the network layer helps defining efficient routing metrics. The PHY layer can also exploit information passed by the network layer. If the requested data rate is low, then the PHY layer can choose an adapted digital modulation. To realize this inter layer communication Cross layer concept has been proposed [29]. The idea behind the Cross Layer design is to break down the rigid separation between the OSI's layers. The drawback of this design is the technology dependency which means that the designed architecture is not portable to multiple technologies. Furthermore, software platforms are needed to test and to implement this architecture.

Diverse software tools have been proposed to establish a model for PHY layer parameters of WSNs. A Simulator is a traditional and simple possibility to design

and to emulate wireless hardware and transmission environment *e.g.* MATLAB with Simulink, NS-2, OPNET, and OMNET , etc. Commonly, they are the base of tests and validation of proposed solutions for networking problems. These solutions tested on a simulator, cannot be neither reproduced nor verified without real-world examples. In addition, several simulators are optimistic when they model network environment, such as the channel model between network's nodes. In simulation, we assume that the inter-node communications are ensured without possible changes of digital signal processing techniques. From the point of view of problem complexity and interoperability, it is more efficient to isolate problems by layers. But if the solution needs only a little modification at another layer, it would be more efficient to facilitate layer interactions. For example, if we deal with the data rate improvement at the network layer, nodes queue management and routing path optimization could be possible solutions. Whereas this problem can be solved only by changing the digital modulation, *e.g.* data rate can be improved at least 6 times with 64-**Q**uadrature **A**mplitude **M**odulation (QAM) instead of **B**inary **P**hase-**S**hift **K**eying (BPSK) modulation. Furthermore, the sponsors of research and development projects often criticize fundamental research, mostly if its proposal is not realistic and without real-world tests [30].

Standards help ensuring product functionality, compatibility and facilitate interoperability. A good example of the power of standardization is the GSM mobile. This technology has been deployed world-wide [31]. As seen in Section 2.3.1, IEEE 802.15.4 is the defacto standard for WSNs. It addresses general requirements of WSNs, such as energy and processing limitation of sensor nodes. However, all node's manufacturers must follow one standard instead of developing new proprietary techniques. But the applications of WSNs are diverse with specific requirements. To test a standard adaptation or exploration, the manufacturer requires passing through a manufacturing process. In addition, standards evolve gradually from a first version to an improved one. For example, 802.15.4e is a new version of the IEEE 802.15.4 including additional specifications. Thus, the challenge is to find a tool able to explore these specifications faster and cheaper.

The frequency scarcity mentioned in Section 1 is also a problem for WSNs. Currently, **I**ndustrial **S**cientific **M**edical (ISM) available frequency bands for WSN are shared with many other wireless communication standards and technologies, such as IEEE 802.15.1, IEEE 802.11b/g/n and Microwave oven. As we have explained above, standard organizations specify a carrier frequency in those frequency bands. A carrier frequency (or central frequency)defines a channel of communication between network nodes. The value in Hertz of that frequency is specified statically even if its selection can be done dynamically. However, these values should be chosen also dynamically to guarantee more robustness to spectrum perturbations. Of course, central frequencies are fixed and integrated in the PHY layer of WSN.

The research community of wireless networks is interested in the SDR opportunity to explore PHY layer parameters in real time. These parameters are implemented

in software to offer more flexibility for possible functions tests and experiments. The researcher can adjust, quite easily, some parameters. Thus, these opportunities could be explored throughout new proposals solving some research issues, and complementary knowledge could be acquired in addition to existing experiences. In addition, I think that we have more chances to convince industrial companies to sponsor research projects, when we offer real-world tests rather than simulations. Hence, we can lead our research and prototyping works on well performed and convincing SDR platform for real profs of concepts.

3.2 Applications of WSNs and SDRs

The SDR emergent technology allows researchers and manufacturers to build real time experiments and realistic characterizations. It also tries to meet to requirements raised by researchers, hobbyists, developers and manufacturers. These requirements can be summarized by fast and less costly access to PHY layer techniques and parameters, such as radio modulation and central frequency. In addition, the SDR usage is adapted to a rapid evolution of user applications. Some wireless communication standards are not able to follow permanently commercial needs. For example, users of 3G smart-phones are not able to benefit from a 4G even if mobile operators offer a 4G service. Hence, a standard defined in software is more attractive to upgrade its specifications than a costly hardware replacement.

Several applications can benefit from SDR. In military, the **J**oint **T**actical **R**adio **S**ystems (JTRS) project is an SDR based solution for soldier to soldier communications under radio systems heterogeneity. Other confidential projects are based on GNU Radio [32] [32] software despite its open-source aspect. They are funded by Department of Defence (DoD) in USA and Thales in France. However, the SDR's business market evolves gradually, it will reach 27.3 $ billion by 2020, according to ASDReports [33]. The major actors are the companies of aerospace, defense and transportation manufacturers.

The open-source propriety of some SDR platforms allows the hobbyists and scientists to develop applications under **G**eneral **P**ublic **L**icense (GPL). It results in the appearance of diverse projects from simple radio broadcast transceivers to aerospace receivers. For example, we can obtain easily a source code of Radio Data System (RDS) [34] receiver on GNU Radio. Similarly, satellite communications can be performed using International Sun/Earth Explorer 3 (ISEE-3) [35] software. The latter allows the user to control an old NASA (National Aeronautics and Space Administration) satellite. We can also receive the Automatic Dependent Surveillance ADS-B signal for airplanes localization [36].

SDR addresses multi-standard challenges of mobile phone applications. An increasing number of cellular, broadcast and multimedia standards push manufacturers, such as Intel, to propose new mobile SDR-based devices. The objective is to quickly adapt mobile devices to the rapid growth of mobile system technologies, *e.g.* 2G,

3G, 4G and 5G.

WSNs are playing a key role in several scenarios such as healthcare, agriculture, environment monitoring, and smart metering. Some manufacturer companies try to replace wired sensors by wireless ones to reduce the weight of machines, *e.g.* planes, trains. Thus, the energy consumption and the cost of such machines are reduced. We can also quote smart buildings, cities, transports, etc. where each application needs a WSN. A new paradigm called "Internet of Things" (IoT) [24] integrates WSNs as a key element where nodes join the Internet network dynamically, and use it to collaborate and accomplish specific tasks. For example, 6LoWPAN/IPv6 technology [24] has been developed to integrate IP-based sensor networks.

4 Thesis objectives

We can summarize the objectives of the thesis as follows: We specify an SDR platform adapted to WSNs and we address the problem of spectrum scarcity via a cognitive radio.

4.1 SDR platform to implement standardized PHY layer of WSNs

Some PHY layer's parameters of wireless networks are easily accessible through SDR, such as output power, modulation, frequency spectrum. To realize software transmitter/receiver chains for a given wireless technology, we need a software and hardware SDR platform. Numerous SDR platforms have been proposed for general and particular applications. It is useful to classify these platforms by analyzing their properties and performances. The goal of study existing architectures is to find one, adapted to implement the PHY layer of IEEE 802.15.4. A comparative study throughout numerous platforms and architectures is necessarily to select this adapted SDR platform.

Existing SDR platforms based on **G**eneral **P**urpose **P**rocessor (GPP) or a combination of GPP and **D**igital **S**ignal **P**rocessor (DSP), are relatively inexpensive. Since the programming can be written in high level languages (C, C++, python), several open source projects have been developed. GNU Radio software with **U**niversal **S**oftware **R**adio **P**eripheral (USRP) [37] SDR platform has been considered by the researcher community as a suitable solution for rapid development and prototyping. Its main advantage is that it is open-source. An open-source software often suffers from a lack of rigor in source program writing and organization. Thus, the analysis of software and hardware architecture is needed to provide more information about the manner to prototype an SDR transmitter/receiver.

USRP is also an open-source hardware proposed by Ettus Research [37]. It is the front end of the GNU Radio USRP based SDR. Its main parameters interacting with

GNU Radio software are frequency and output power. The unknown performances of this hardware, requires a deep analysis which can be done through an experimental approach. The objective is to see if the announced performances of some parameters, *i.e.* frequency bandwidth and output power conform to specifications [37]. The characterization of the hardware can be obtained with an experimental plan. The research can be carefully carried, and the measurements can be obtained under controlled conditions. In general, the measured parameters can be controlled independently and an empirical model, can be also formulated to predict parameters' behaviors. Such characterization of the USRP is essential to realize a powerful prototype.

The possibility to customize (with scalability issues) the IEEE 802.15.4 standard motivates its prototyping via an SDR implementation. Furthermore, not all the standard specifications have been implemented by manufacturers of sensor nodes. However, it is interesting to have these specification in software to facilitate their upgrading. In the literature we can find some SDR realizations for IEEE 802.15.4 standard on GNU Radio USRP SDR [38] [39] [40] and others on different SDR platforms [41]. That of Thomas Schmid [38] has been proposed for 2450 MHz frequency band.

Reverse engineering process or back engineering allows an engineer and a researcher to extract a knowledge or design information from existing prototypes. In our case, the objective is to disassemble an already proposed SDR prototype for 2450 MHz frequency [38]. SDR transmitter and receiver steps should be explained from data generation to baseband signal transmissions and vice versa. Furthermore, our aim is to reuse some source codes and implement a new prototype for 868/915 MHz frequency band of the IEEE 802.15.4 standard. Note that this band offers a wider coverage for network devices compared to the frequency band of 2450 MHz. These two SDR prototypes for two particular bands lead us to think about a Cognitive Radio.

4.2 Cognitive Radio for Spectrum Scarcity

ISM frequency band for WSN based on IEEE 802.15.4 is shared with many other communication standards and technologies, such as IEEE 802.15.1, IEEE 802.11b/g/n and Microwave ovens. The coexistence of these technologies in the same frequency band degrades the transmission performances. The packet collisions at MAC layer and signal interference at physical layer are the main sources of performance degradation. Frequency band overlapping can be avoided by an intelligent use of three types of diversity, namely frequency, space and time. The agile use of frequency spectrum and transmission parameters of sensor nodes can be insured with their software transceivers.

Cognitive Wireless Sensor Network is a new solution. Primarily, it has been proposed as an answer to frequency scarcity issue. The growth of the number of

applications and the standard rigidity is a fertile ground for this solution. In some geographical areas over the world, for example, in America and Europe, the spectrum frequency is crowded and wireless technologies can share the same frequency band. Traditional radio defines static carrier frequencies (or channels) for each network or technology. For example, only 16 and 13 channels are respectively available for IEEE 802.15.4 and IEEE 802.11 networks in 2450 MHz frequency band. If these channels interfere, only a static definition of these channels is possible. By contrast, over a software radio, more flexibility can be ensured, especially by a dynamic change of channels definition, *i.e.* a channel can take any central frequency in a frequency band. Such changes can improve at the same time interference avoidance and packet success rate. The communication context needs to be analyzed before a dynamic spectrum access. Spectrum sensing and dynamic spectrum access are possibles operations which can be implemented on SDR platform.

We have seen, in Section 2.2, that the cognitive radio is a high level of abstraction of software radio. It involves two operations: Spectrum Sensing (SS) and Dynamic Spectrum Access (DSA). We have also mentioned that IEEE 802.15.4 defines numerous specifications, each one for a particular frequency band. Based on these two operations DSA and SS, we would define Cognitive Radio for WSNs. We can also deal with that using real world communications, since the cognitive radio has been largely addressed but with theoretical works. Analyzing all the available resources within GNU Radio USRP SDR, we can realize proofs of concept of cognitive WSN on this SDR.

5 Thesis Organization and Contributions

This dissertation is organized in four main parts. Chapter 2 presents the state of the art on Software Defined Radio. Analysis of GNU Radio and experimental measurements on USRP hardware are presented in Chapter 3 and 4. SDR implementations of IEEE 802.15.4 standard are described in Chapter 5. A Cognitive Wireless Sensor Network based IEEE 802.15.4 is detailed in Chapter 6. We conclude the thesis with final remarks and suggestions for further works in Chapter 7.

5.1 State of the art on Software Defined Radio

We start by defining, featuring and classifying existing Software Defined Radios. We put the light on the general architecture of an SDR transmitter and receiver as well as its advantages. We can summarize this architecture in three parts: the radio frequency front-end part, the intermediate frequency and the baseband processing part. The two first parts cannot be obtained in software regarding some processing constraints, which are explained in Chapter 2. The second part can be featured by a chain of blocks programmed and executed on specific architecture. After that, we discuss the advantages of an SDR. The reconfigurability is its main feature. The

software part of an SDR can be easily reconfigured autonomously or by a designer. We show also the limits of the SDR performances as well as its constraints. We also notice the large panel of SDR technology platforms. Since we are interested in implementing WSN physical layer on an SDR, we propose to classify the possible SDR platforms into two classes: The GPP based architecture and the reconfigurable hardware-based architecture. This classification allowed us to present a general one which is based on hardware and programming model of SDRs. We end Chapter 2 by comparing the announced performances for most used SDR platforms. The GNU Radio and USRP SDR promote more flexibility and high level programming languages, as well as a high covered frequency band.A difference between their announced performances and the observed ones led us to analyze in details the GNU Radio and USRP SDR in Chapter 3 and Chapter 4.

5.2 Analysis of GNU Radio and experimental measurements on USRP's Daughter boards

We propose to analyze the architecture of the GNU Radio USRP SDR since it is an open-source platform. One drawback of an open-source platform's is the lack of an accurate and detailed documentation. However, we describe this SDR from bottom to top starting from USRP to GNU Radio. The USRP's architecture handles two parts: the radio frequency front end and the intermediate frequency. One USRP version can carry up to four daughter boards and several antennas. The daughter boards are the radio frequency interface between antennas and the USRP motherboard. In Chapter 3, we summarize the different versions of the USRP and daughter boards as well as their properties. We concentrate our study on those which are used in our thesis. We show that the radio frequency bandwidth, and the output power of the SDR can be affected by the daughter boards behavior. Furthermore, we detail how the USRP's motherboard interacts with GNU Radio throughout a particular Universal Hardware Driver (UHD).

GNU Radio software provides signal and data processing blocks, which construct the transmitter and receiver chains or flow graphs. It is based on the Mitola's [10] thought as briefly defined in Section 2.1. Its usage in our research works is also motivated in the previous sections. We characterize the software architecture describing the stack of programming languages. Python simplicity helps the radio designer to construct communication chains, which are called flow graphs. The C++ language supported by GPP architecture permits to develop fast processing blocks. We summarize the C++ source code object oriented architecture by presenting its main classes to generate flow graphs. In addition, we approve the result obtained by Chiang *et .al* [42] towards the two types of schedulers: single thread and Thread Per block Scheduler (TPS). The debugging of the flow graphs' programs can be performed easily knowing the scheduler approach. We also report an abstract of the last modifications to improve the processing time of blocks using SIMD architecture

(or VOLK). Before summarizing the advantages of GNU Radio and USRP SDR, we discuss the trade-offs between its performances and its features. In fact, we precise the sources of processing latency between hardware and software. The processing time within each block of a given flow graph can be estimated using Performance counters and ControlPort tools, which are proposed in [43]. We explain their operating principle regarding their importance for program debugging, processing synchronization and processing optimization, *.e.g* we need to know which block spends more time than others.

In Chapter 4, we measure, throughout experiments, two main parameters: the frequency bandwidth and the output power of some USRP's daughter boards. We select the RFX2400, RFX900, SBX and the MIMO B210 board. We motivate the experimental approach by performing loop-back GNU Radio simulations. We discuss the estimated quality of BPSK communication using a traditional BER/SNR parameter. Although the communication is in the GNU Radio loop-back, we show that the BER/SNR deviates slightly from expected theoretical results. We also show a similar behavior using a USRP and an RFX 2400 daughter board. We prove that this behavior is partly due to daughter boards' output power and frequency bandwidth. We feature this insufficiency throughout curves for each selected daughter board. We choose SBX daughter board's measurements for a deep analysis regarding its large frequency bandwidth. We construct an empirical model drown upon the obtained results,. This model is simple and can be implemented to calibrate and predict the output power at each central frequency under the covered bandwidth.

5.3 SDR implementations of IEEE 802.15.4 standard

This part contains a documented reverse engineering process of existing SDR transmitter/receiver, and it reports a new one. Both are based on IEEE 802.15.4 standard specifications. Existing one reproduces the specifications of the worldwide ISM frequency band of 2450 MHz. We bring more details about transmitter and receiver flow graphs compared to that available in the literature [38]. Furthermore, we formulate the packet decoding operation in pseudo code algorithms. Then we propose a new implementation of the IEEE 802.15.4 for a frequency band of 868/915 MHz. In addition to the flow graphs, we bring more details about the setup of USRP parameters. We achieve real-world communications between true hardware transceivers of WSNs and SDR implementations. The particularity of our work is the ability to measure two different parameters in PHY and network layers.

5.4 Cognitive Wireless Sensor Network based on IEEE 802.15.4

In chapter 6, we realize a first step toward a cognitive wireless sensor network. We propose a Dynamic Spectrum Access drawn upon the two implemented SDR for the

frequency bands of 868/915 MHz and 2450 MHz of IEEE 802.15.4. For that, we use a Spectrum Sensing technique to quantify the quality of a central frequency under disturbed transmissions. We propose a message exchange algorithm to synchronize transmitter and receiver in the same central frequency. Then, we detail the SDR settings for each node. We show the interest of using DSA through experiments in an indoor environment. Under a list of setup parameters, the DSA improves by 80 % the packet success rate in a one to one communication.

Chapter 2

State of the art on Software Defined Radio-SDR

Is it possible to replace all processing devices by software?

Me

Contents

1 Introduction . . . 48

2 Typical architecture of an SDR . . . 48

2.1 SDR Receiver (Receiver (Rx)) . . . 49

2.2 SDR Transmitter (Tx) . . . 50

3 Features and Challenges of SDR . . . 50

3.1 Features . . . 50

3.2 Challenges . . . 51

4 SDR standards and architectures . . . 53

4.1 Software Communication Architecture SCA . . . 53

4.2 Reconfigurable Radio System RRS . . . 54

5 SDR for Embedded Devices . . . 56

5.1 GPP based architecture . . . 56

5.2 Reconfigurable hardware based architecture . . . 56

6 SDR classifications . . . 57

6.1 Programming model . . . 57

6.2 Used hardware . . . 59

6.3 SDR platforms . . . 61

7 Summary . . . 64

1 Introduction

This chapter introduces a feasible design of SDR based on Transmitter (Tx) and Rx chains. It shows detailed architectural diagrams for both radio transmission and reception. It highlights the limits of software definition of hardware radio. However, the SDRs present several advantages for industrial users and researchers in wireless communications. Indeed, a lot of hardware/software platforms and architecture have been developed. And some standards have also been proposed to harmonize existing SDRs. These SDRs must be compared and classified considering their architectures, properties and performances. This comparative study facilitates the selection of a suitable SDR platform. The time and cost to build software transmitter and receiver are the main parameters to select an SDR platform.

2 Typical architecture of an SDR

Mitola describes an ideal Software Radio in his paper [10]. He recommends that, a **D**igital to **A**nalog **C**onverter (DAC) and an **A**nalog to **D**igital **C**onverter (ADC) are respectively at the closest proximity to antennas of transmitter and receiver chains. Mitola assumes that the communication chain is achieved by a set of software blocks which perform signal or data processing functions, such as signal generation and modulation/demodulation functions (see Figure 2.1).

Figure 2.2 highlights a classical SDR architecture [10]. The first part is Radio Frequency (RF) Front End (FE). It receives or transmits a baseband signal through an input and output antenna. An **I**ntermediate **F**requency (IF) part is introduced since the processing capabilities of an ADC are slower than the **S**oftware **D**efined **R**adio (SDR) function expectations. This leads to the impossibility for ADCs and DACs to cope up with high-frequency signals. The ADC requires a high-sampling rate, high-power consumption, and a reduced frequency bandwidth [44]. Thus, an intermediate Frequency (IF) part is used to ensure a pragmatic Software Radio or Software Defined Radio. The IF section selects a bandwidth and shifts it from RF to an Intermediate section. It is considered as a brake of the SDR domain, as it is limited in sampling rate and a frequency bandwidth. In addition, it accomplishes a channelization, *i.e.* choose a specific central frequency, which cannot be performed at a Radio Frequency (RF) Front End (FE). In the last section, the baseband processing replaces analog functionalities by digital ones. Programmable-processing technologies perform this function using, for example, **F**ield **P**rogrammable **G**ate **A**rray (FPGA), **D**igital **S**ignal **P**rocessor (DSP), **G**eneral **P**urpose **P**rocessor (GPP), Programmable **S**ystem-**o**n-**C**hip (SoC) and other application specific-processing entities *e.g.* **A**pplication **S**pecific **I**ntegrated **C**ircuit (ASIC). The following section allows us to understand the several steps of Tx/Rx chains.

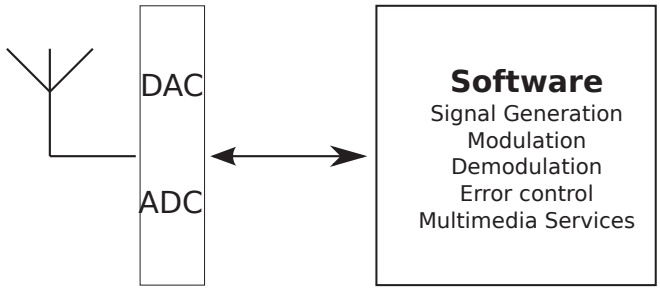


Figure 2.1: Ideal architecture of a Software Radio

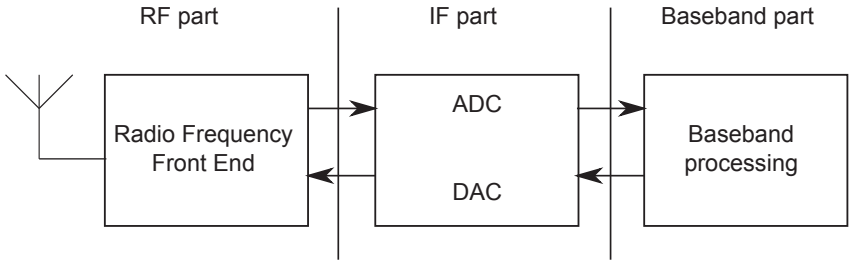


Figure 2.2: General architecture of SDR

2.1 SDR Receiver (Rx)

Figure 2.3 shows a general SDR Rx, starting with a RF tuner which converts analog RF signal to analog IF frequency. Then, the ADC digitizes the IF thereby converting it into digital samples. These samples enter to a **D**igital **D**own **C**onversion (DDC) which could be a monolithic single chip or an FPGA. A DDC decimates a data stream to a lower sampling rate and adapts a stream to a lower speed processor. It contains a digital mixer, a digital local oscillator, and a **F**inite **I**nput **R**esponse (FIR) lowpass filter. The digital mixer and the local oscillator translate IF samples down to a baseband signal. The primary function of FIR low-pass filter is to limit the signal bandwidth. The DSP performs a demodulation, decoding, and other processing tasks via software radio programs [45].

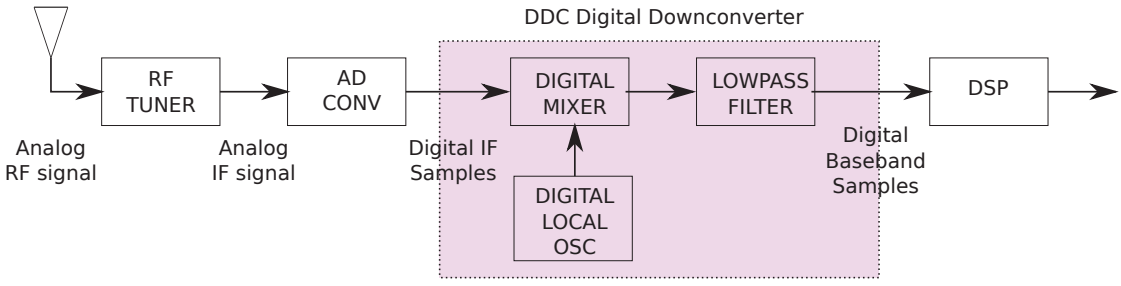


Figure 2.3: SDR receiver block diagram

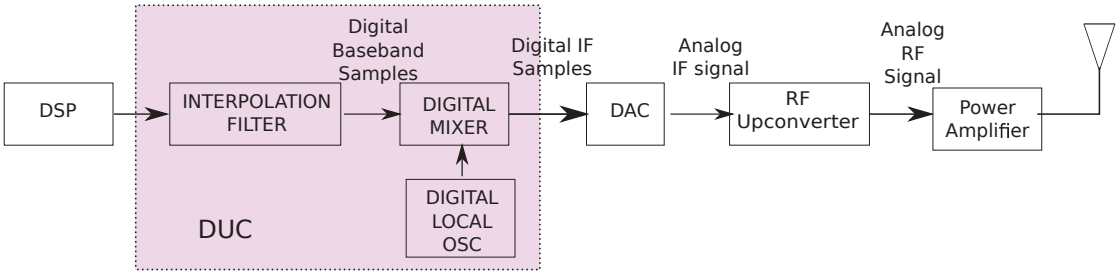


Figure 2.4: SDR transmitter block diagram

2.2 SDR Transmitter (Tx)

Similarly to Rx, Figure 2.4 shows SDR Tx chain. Tx firstly generates a digital baseband signal of a transmission chain. **D**igital **U**p **C**onverter (DUC) interpolates a baseband signal and up-converts it to an intermediate frequency band. Then DAC converts a digital IF samples into an IF analog signal. The RF Upconverter that follows, converts an IF analog signal to RF frequencies. Finally, a power amplifier boosts the signal energy into the antenna [45].

3 Features and Challenges of SDR

The wireless transmission has multiple requirements and properties depending on user expectations and hardware performances. Some advantages are reported in the first chapter and in Section 2. In this section, we report features and challenges of SDR based transmissions.

3.1 Features

The features of a software defined radio are for different types of application as follows:

- Reconfigurability,
- Portability,
- Interoperability,

3.1.1 Reconfigurability

The behavior of traditional radio transceivers is static and defined by a manufacturer. It cannot be changed by the transceivers' designer or by itself. Thus, devices (or nodes) of wireless networks handle only specific applications. A reconfigurable system has the ability to dynamically change the transceiver behavior. Since all blocks of communication chains are in software, a simple modification is possible during, before

or after a transmission. For an SDR designer, modifying these chains should be performed transparently. However, wireless communication standards implemented in SDR can be maintained only by changing software. For an SDR itself, a dynamic auto configuration is interesting, since it brings cognitive radio capabilities [44].

3.1.2 Portability

The portability refers to the waveform mobility. From end user point of view, it is the ability to implement and migrate waveform processing from one SDR platform to another. From end-user view, migration should be carried out with a maximum of transparency. This means that waveform processing should be possible on another platform without rewriting the whole application. However, some specific architecture of SDR defines its components as software programming objects. In this case, a waveform can be handled by one of these objects. For example, **S**oftware **C**ommunication **A**rchitecture (SCA) details an object oriented architecture which is based on a transport mechanism of waveform components mapped on DSPs and FPGAs [46] (see Section 4). Nevertheless, portability requires a correct migration from and to a new platform without information losses. In addition, hardware performances should be the same when changing the hardware platform or the programming language.

3.1.3 Interoperability

The interoperability or radio bridging enables heterogeneous radio networks to inter operate by dissimulating their differences. The challenge is to have a multi-standard, multiband and an open radio system. Traditionally, a radio transceiver is limited to communications only with nodes that share the same radio properties. The latter could be waveform, frequency band, modulation type and data communication protocol. However, a radio bridge acts as a translator between heterogeneous transceivers. It receives a signal encoded in one format and retransmits it following another specification. Primarily, the interoperability is addressed in military applications [47].

3.2 Challenges

In section 2, we saw that the main SDR challenge is to overcome the limited processing power. Consequently, Intermediate Frequency part (see Section 2) is unavoidable to get an SDR processing look like Software Radio. In this section, other challenges of SDR solution are addressed.

3.2.1 Handling of an SDR platform

Numerous SDR platforms have been developed in the last decade with different integration levels of hardware and software architecture. For software-designer,

high-level programming could be an easy way to handle software, but at the expense of a high hardware physical dimensions. Furthermore, the type of the architecture, *i.e.* distributed or centralized, is an important element to understand how to use different components. A centralized architecture could be easier to manage than a distributed one. For example, the SCA [46] is distributed and interoperability is guaranteed via **C**ommon **O**bject **R**equest **B**roker **A**rchitecture (CORBA) middleware. Understanding software architecture and programming with CORBA require more efforts than programming on a host computer. In CORBA, the software/hardware object locations of architecture are hidden to designers. Whereas, programming on host computer is more simple, since all objects are present in the same physical place. GNU Radio and **U**niversal **S**oftware **R**adio **P**eripheral (USRP) [37] follow a centralized architecture with a host computer programming. On GNU Radio software, mastering high-level programming languages is required to construct software communication chains. In fact, source codes are written in C++ and Python and most of the time not commented (lack of comments) making them hard to understand. In addition, these codes are commonly generated by a machine. RF performances of USRP hardware, which is connected to GNU Radio are not documented. Thus, implementations on USRP and GNU Radio can be time consuming. In spite of these drawbacks, this platform presents several advantages which will be discussed in the next Chapter.

3.2.2 Hardware physical dimensions

Physical dimensions issue comes from the need to integrate SDRs in constrained applications, such as WSNs applications. It affects SDR capabilities, especially, power processing and energy autonomy. Commonly, it depends on the hardware front-end size and on the baseband processing software. Signal processing is a greedy operation, and it is performed at various stages of the transmitting and receiving chains. Thus, the capabilities of baseband processing part can be considered as a reference to define hardware dimensions and programming languages of an SDR platform. In the case of a GPP configuration, a high-level programming language on a host computer is preferred by designers. Thus, the platform is cumbersome and less usable for **W**ireless **S**ensor **N**etworks (WSNs). In the case of an interoperable architecture like SCA [46], a middleware manages several technologies, *e.g.* FPGA, DSP, and GPP, etc. Actually, SDR nodes for WSNs is a futuristic technology, regarding constraints of one SDR.

3.2.3 Radio frequency performances

We have shown in Section 2 the general architecture of an SDR, IF is the bottleneck of an ideal SDR. In fact, SDR performances depend on hardware's computational resources and front end design. Dedicated purpose and non programmable chips offer high-performances, but they are avoided, since they are limited in terms of

flexibility. However, an IF or a baseband filter sets the analog bandwidth from a minimum to a maximum frequency. Furthermore, ADCs and DACs define SDR's sampling rate or a processing bandwidth.

3.2.4 Baseband processing hardware

Several hardware components of baseband processing could match with SDR requirements. ASIC is not a reconfigurable processing component, but it is useful when efficient processing is preferred [44]. However, an FPGA is an alternative to an ASIC. It rapidly reconfigures any waveform component through a specific program. Hence, it is a preferred choice for an environment with real time constraints, particularly when an application needs some permanent connectivity. Furthermore, an FPGA requires significant energy and produces unpleasant heat for a handheld SDR.

A host computer simplifies the reconfiguration of baseband signal processing. It runs SDR's source codes over GPPs. The Operating System (OS) of host computers creates an abstraction layer over a GPP by running different applications. In fact, the OS handles software programs with little or no knowledge of underlying hardware management [48]. The memory manager of such architecture also combines numerous programs without a special care on **R**andom-**A**ccess **M**emory (RAM) of software development.

DSPs are similar to GPPs, since they can be programmed with a high-level language such as C or C++ for processing and running under an **O**perating **S**ystem (OS). However, the difference comes from instructions set and on memory management. The instructions of a DSP are dedicated to a particular application.

4 SDR standards and architectures

In this chapter, we quote a list of existing Software Defined Radios. This list is organized in standards and architectures. Furthermore, they are classified, regarding main parameters, *i.e.* baseband programming languages and front-end hardware.

4.1 Software Communication Architecture SCA

Software Communication Architecture (SCA) is an open and common architecture allowing a designer to define hardware and software elements running on **J**oint **T**actical **R**adio **S**ystems (JTRS) SDR [46]. Software radio programs are defined to process waveforms in different layers while CORBA ensures interactions among them [48]. For an embedded application, CORBA is greedy in processing power and in memory resources. Hence, the focus of embedded SDR community shifted away from CORBA. Architecture improvements have been made by SDR forums and by the **O**bject **M**anagement **G**roup (OMG). The objective was to simplify the architecture

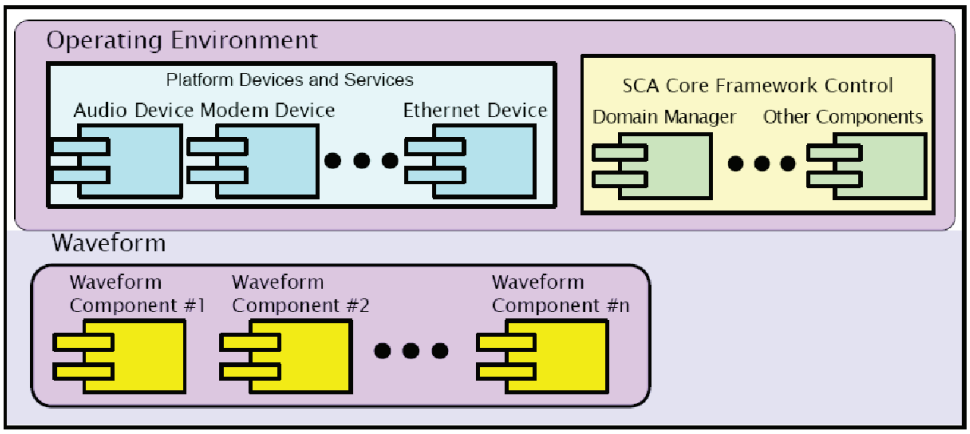


Figure 2.5: Version 4 of the SCA architecture [49]

and to separate the SCA and waveform components from the operating environment (see Figure 2.5).

The latest specifications of the SCA in versions 2, 3 and 4 discuss the use of an FPGA [49]. As explained in Section 3.2.4, an FPGA is more reconfigurable and has a higher processing power than a DSP or an ASIC. The FPGA can accomplish digital processing of the IF and that of the baseband processing part of an SDR. It can also performs specific processing such as cryptography algorithms. Figure 2.5 shows the main parts of the SCA architecture: Operating environment, SCA Core Framework and Waveform. The SDR's core framework handles all underling software responsible for running waveform processing.

4.1.1 Open Source SCA Implementation::Embedded (OSSIE)

OSSIE is an open-source SDR and Object Oriented SCA architecture [50]. It is implemented over a CORBA middleware working on a Linux operating system and carried out by Intel or **A**dvanced **M**icro **D**evices (AMD) based computer. The core framework of OSSIE contains signal-processing components and software interfaces based on CORBA. The objective behind OSSIE is to reduce the designer's learning curve of the SCA and the cognitive radio. Furthermore, for a proof of concepts, SDR prototyping can be faster than other software architecture. However, CORBA requires a lot of memory resources, and the objects communicate with fewer throughput. In fact, SCA performances depend on the object size, since several objects have to be exchanged within the SCA [44].

4.2 Reconfigurable Radio System RRS

European **T**elecommunication **S**tandards **I**nstitute (ETSI) introduces a **R**econfigurable **R**adio **S**ystem (RRS) for mobile phone networks [51]. This architecture is more specialized than the SCA architecture. It specifies the required

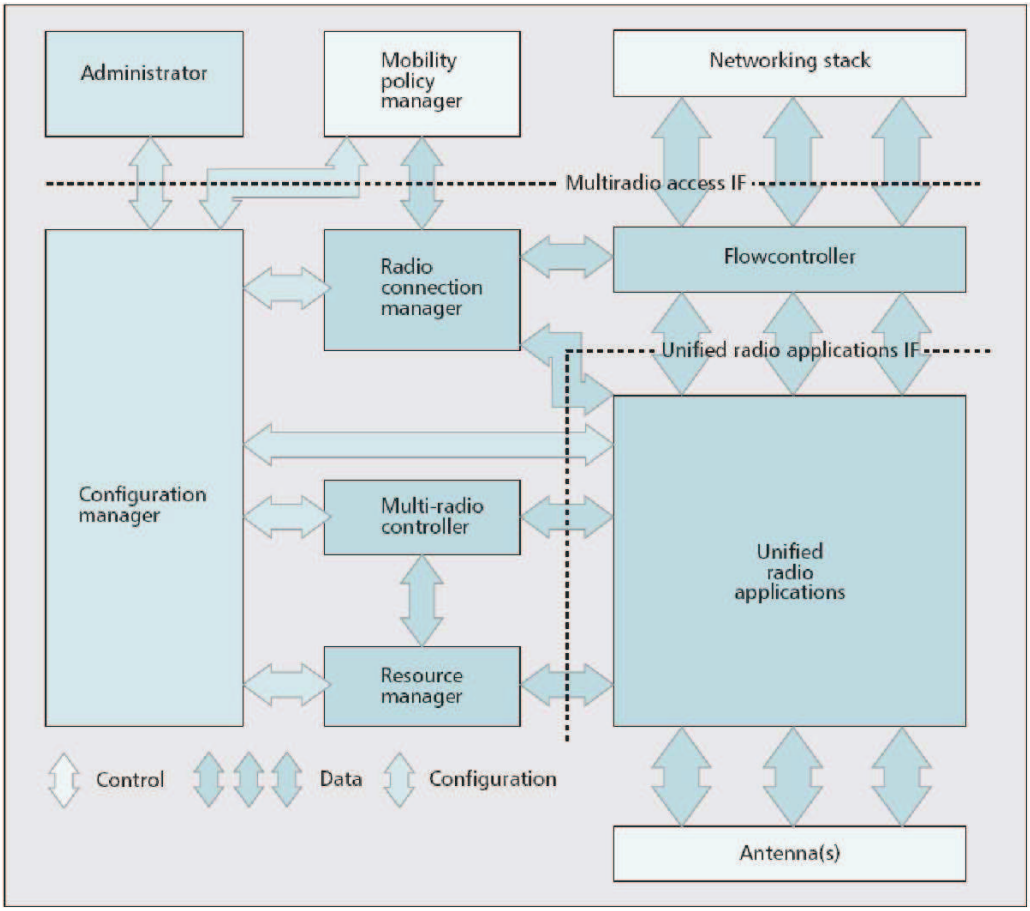


Figure 2.6: ETSI architecture [51]

radio resources and interfaces. Figure 2.6 shows interactions between different abstract managers of the architecture. The configuration manager installs radio implementation in radio computers and updates radio parameters. The Radio Connection Manager interacts with designer or inter-implementation requests to activate radio implementations or to switch among them. The multi-radio controller anticipates interoperability problems by scheduling simultaneous spectrum requests. Finally, the resources manager controls the allocation of radio hardware resources, such as spectrum or hardware devices, according to the application importance.

RRS test bed is the only SDR platform found in the literature following the ETSI specifications. It is a modified USRP and GNU Radio based SDR, dedicated to security research for military applications. Next chapter gives more details about the USRP and GNU radio platform.

5 SDR for Embedded Devices

In the literature, there is neither a general architecture nor a standard proposed for embedded devices or wireless sensor nodes. The design of a such SDR is constrained by the application type and hardware capabilities. From our state of the art we can notice two main architectures:

- GPP based architecture
- Reconfigurable hardware based architecture

5.1 GPP based architecture

The GPP base architecture is the case of platforms such as GNU Radio/USRP [37] [37] and Sora platforms [52]. The GPP receives/generates baseband signal replacing a DSP in Tx and Rx SDR architectures shown in Figures 2.3 and Figure 2.4. The SDR implementations, prototypes or realization are designed at a GPP using high-level programming languages. Commonly, the research community simply considers these prototypes as SDRs, software transmitters/receivers or software transceivers.

The SDRs based on host computers are more accessible than other possible SDR architectures. The developing and debugging become easier, especially with the collaboration of a high number of users. Furthermore, the GPP power processing can be significantly increased by integrating a multicore in a monolithic processor. These cores can coherently handle a distribution of waveform processing blocks [52]. Thus, prototyping of OSI's physical layer is simplified and efficient using GPPs.

Multi core GPP based SDR architecture is less deterministic than those combined with an FPGA or a DSP. Moreover, without a specific processing device, reliability decreases. The FPGA offers a good ratio between real time processing and reconfigurability. It can thus be used to accomplish high-power processing. What's more, the GPP handles software frameworks for waveform processing. An example of such architecture is that of USRP and GNU Radio SDR.

A DSP can replace an FPGA if the GPP coordinates the assigned signal processing operations among programmable DSPs. However, for an SDR designer, the advantage is to have a possibility to build separately DSP programs and execute them on **M**ulti **P**rocessors **S**ystem **o**n **C**hips (MPSoC). Table 2.1 presents some DSP based SDRs [53]

5.2 Reconfigurable hardware based architecture

In order to benefit from the flexibility and the high performances of dedicated hardware, the entire radio communication stacks can be implemented on an FPGA. Hardware synthesizer tools should be used, for example Register Transfer Level (RTL) [54]. These tools allow a designer to implement a hardware conception, but

with an extra time needed to handle the associated programming languages, *e.g.* **V**ery high speed integrated circuit **H**ardware **D**escription **L**anguage (VHDL).

An SDR can be implemented via an **I**ntellectual **P**ropriety (IP) core, which is data or logic blocks. Ideally, an IP core should be entirely portable and easily inserted into marketed technologies. However, IPs are expensive and consume a lot of logic resources as well as require powerful hardware [45]. Furthermore, coarse-grained reconfigurable architecture consists of a large number of function units interconnected through an embedded network. Comparing this architecture to an FPGA, the advantage is low in terms of power consumption and time needed to set up the platform. Therefore, the gate-level reconfigurability is limited, but with a large increase in hardware efficiency [55].

6 SDR classifications

In the literature, several classifications can be found in [56] and [57]. SDRs have been classified by considering hardware architecture and programming model. Our objective is to synthetize these classifications throughout comparative tables. Table 2.1 synthesizes Dardaillon *et .al* work [57]. It contains a classified SDRs function of a programming model.

6.1 Programming model

From the programmer's point of view, the architecture has a crucial impact on programming models and tools. Six classes are proposed in [57]:

- GPP General Purpose Processor approach
- Coprocessor
- Processor centric approach
- Configurable units approach
- Programmable blocks approach
- Distributed approach

The SDR based GPPclass uses a computer processor as a computing platform, with programming at a high-level for more flexibility. However, high-energy consumption is proportional to a high demand of hardware resources. Thus, the GPP approach is improved by integrating a coprocessor to perform heavy processing and to reduce energy consumption. In this case, a GPP is associated to an FPGA (or DSP), chip rate accelerators or coarse-grained cores. The processor's centric approach increases the SDR efficiency. For example, a dedicated processor like an **A**cron **R**ISC

Machine (ARM) can be used for the signal-processing part. This approach guarantees high programmability but reduces flexibility from its specific architecture. The configurable units are proposed to offer lower energy consumption by substituting DSPs with reconfigurable processors. Programmable blocks refer to an FPGA based architecture. They provide programmability with a great flexibility to create tailored architecture. Finally, a distributed approach distinguishes signal-processing cores or a distributed asynchronous array from simple processors.

Classes	Name	Programming	Hardware
GPP approach	USRP [37]	C++, Python (GNU Radio)	GPP
	QuickSilver (QS1R) [58]	SDRMAX pre-build	GPP
	Microsoft SORA [59]	SORA SDK pre-build	GPP
	RTL-SDR [60]	SDR# Pre-built and limited use of GNU Radio	GPP
	SDR4All [61]	C++, Matlab	GPP
Coprocessor approach	KUAR Kansas University Agile Radio	VHDL[1] implementation or GNU radio flow	FPGA
	Texas Instrument SDR	CHDL or MATLAB Simulink	Programmable DSP
	Imec ADRES [2]	C on DRESC compiler	GPP, ADRES accelerator
	Hiveflex	HiveCC SDK	HiveFlex accelerator
Processor-centric approach	Infeneon MuSic	C and ASM	ARM processor, DSP
	Sansblaster	ANSI C on didecated compiler	ARM processor, Sandblaster cores
	SODA University of Michigan ARDBEG	C programming language	ARM processor, DSP
	Tomahawk (University of Dresden)	C programming language	Tensilica RISC[3] processor, DSP
Configurable units approach	Imec BEAR: The evolution of Imec ADRES	C programming language, MATLAB	ARM processor, ASIP[4], veterbi accelerator
	CEA Magali chip	C programming language, ASM	ARM processor, coarse grain reconfigurable cores Merphisto

[1] VHDL
[2] **A**rchitecture for **D**ynamically **R**econfigurable **E**mbedded **S**ystems (ADRES)
[3] **R**educed **I**nstruction **S**et **C**omputing (RISC)
[4] **A**pplication-**S**pecific **I**nstruction-set **P**rocessor (ASIP)

	EURECOM ExpressMIMO (reconfigurable FPGA)	C programming language	FPGA
Programmable blocks approach	XiSystem	C programming language	PiCoGA FPGA
	WARP (Rice University)	VHDL	Xilinx Virtex FPGA
	WINC2R (Rutgers University)	C++, python (GNU Radio)	FPGA, Soft Core Processors and accelerators
	Lytech	Simulink, MATLAB	FPGA
Distributed approach	Picochip	C programming language	Matrix of small cores
	CEA Genepy	C programming language, ASM	Coarse grained based on Magali with ARM processor

Table 2.1: SDR classification refereed to a programming model

6.2 Used hardware

As shown in Table 2.2 SDRs can be categorized in three classes [56]:

- Coarse-grained reconfigurable
- Processor centred architecture
- Multi core and multi thread architecture

The first class is based on reconfigurable hardware, whereas the second one consists of DSP-centred and accelerator-assisted architecture. Although an SDR baseband processing can be done by a GPP or an FPGA, the general-purpose DSP appears to be the most used solution for embedded systems. The processor centred architecture is based on an ASIP, a DSP and many-cores SDRs. The DSPs exploit a native date and a parallelism instruction level of radio kernels. In some cases, they are assisted by accelerators, *e.g.* ASIC in LeoCore SDR [62]. Some SDR platforms are based on splitting a bigger task into smaller ones and distribute them among the cores. Behind the task's distribution, power consumption is reduced to an acceptable level, *e.g.* SODA [63]. In Section 5.2, we explained that coarse-grained reconfigurable architecture offers more flexibility than processor centric architectures. For example, we can cite ADRES, HERS and CREMA platforms (see table 2.2).

Architecture	**Name**	**Programming**	**Hardware**

Coarse-grained reconfigurable	Montium	Montium Sensation Suite Simulator and Editor. Language used is Montium Configuration Design Language (CDL)	Chameleon SoC
	BUTTER and CREMA	VHDL	SoftCore, CREMA is synthesized on FPGA
	HERS	FireTool for C language extensions	SoftCore
	EURECOM ExpressMIMO	C programming language	FPGA
	Imec ADRES	ADRESC compiler with ANSI C	ADRES processor, ASIP, FlexFec processor
Processor centred architecture	Leocore	Coresoninc developer studio	Network on Chip Noc
	Sandblaster	C language	Multi-Core and Multi-Thread processor
	Connx BBE	TIE language, C++ language	Tensilica Xtensa processor
	EVP Embedded Vector Processor	EVP-C compler	ASIC
Multi core and multi thread architecture	SODA Signal processing On demand architecture	C compiler generated by OptimoDE's Framework, Matlab C model supported	Imagine Processors and IBM Cell Processor
	Tomahawk MP-Soc	C compiler	Two Tensilica processors, ASIP
	MuSIC	MuSIC specific C compiler to support SIMD [5] C extensions	Two Tensilica processors, ASIP
	Imec BEAR	Matlab and C programming language	ARM processor for control and three ASIPs

Table 2.2: Synthesised classification of SDR hardware given in [56] based on the DSP architecture

[5] **S**ingle **I**nstruction **M**ultiple **D**ata (SIMD)

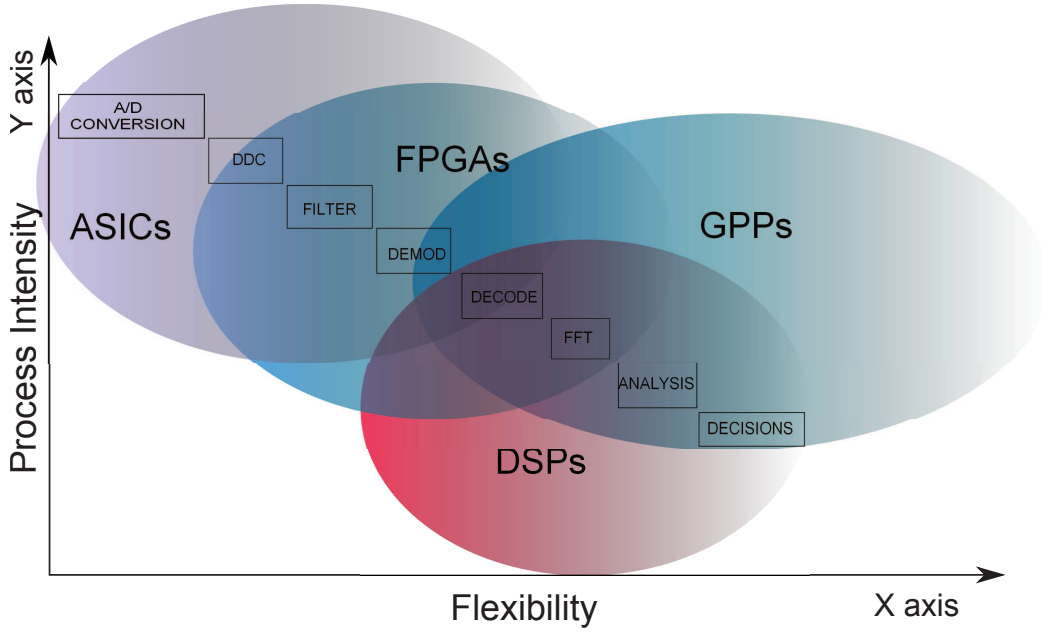


Figure 2.7: Hardware tasks associated with process intensity and flexibility

6.3 SDR platforms

SDR platforms are software and hardware toolkits that allow the designer to construct an SDR prototype or implementation. Of course, an implementation can consist of separated transmitter or receiver. It is also possible to gather transmitter and receiver giving an SDR transceiver.

Figure 2.7 shows the main signal processing tasks associated to an SDR implementation. They are on two vertical and horizontal axis. Intensity of processing is associated to the vertical axis and flexibility to the horizontal one. The degree of highly repetitive and rather primitive operations denotes the process intensity. Flexibility refers to the uniqueness/variability of the processing and how likely the function may have to be customized for any specific application. Figure 2.7 illustrates the degree of flexibility vs the processing intensity. In the upper left of this figure we find hardware structures for real time operations, such as ADC, DAC and DDCs that can be done by ASICs. In the lower right of this Figure, we find other tasks, such as analysis and decision tasks that need more flexibility, and can be accomplished by DSPs and GPPs.

We notice the existence of a large panel of SDR technology platforms. More than sixty platforms could be counted over the world in 2014. Table 2.3 summarizes a non exhaustive list of research commercial and open source projects SDR technologies. Numerous SDR platforms are proposed with different RF performances, programming languages and hardware architecture. The sampling rate and the covered frequency range are two primary parameters which define the capabilities of an SDR implementation. Compared to other SDRs, the USRP brings high performances. It

is proposed by Ettus Rsearch [37] in different versions . Its frequency range grows up to 6 GHz with a DAC and ADC sampling rate up to 400 **M**ega **S**amples **P**er **S**econd (MSPS) and 100 MSPS, respectively. Furthermore, baseband processing is handled via a GPP which executes a software toolkit.

The announced performances of the USRP led to the development of those software toolkit. Particularly, GNU Radio [32] has been developed to drive the USRP despite existing general software, such as Simulink MATLAB and LabView. The GNU Radio is an open-source software used to drive not only the USRP, but also some other SDR hardware (*e.g.* HackRF [64] and Nutaq ZeptoSDR [65]). However, several communication standards and prototypes have been implemented on GNU Radio and USRP (*i.e.* IEEE 802.15.4, IEEE 802.11a, IEEE 802.11p, **A**utomatic **D**ependent **S**urveillance-**B**roadcast (ADS-B) and **H**ight **D**efinition **T**elevision (HDTV) etc). Of course, the SDR based USRP and GNU Radio will be detailed deeply in the next chapter.

Name	Architecture	Software and Programming	Frequency range	Sampling rate	connecting mechanism	OS	Applications
USRP family [37]	GPP + FPGA	GNU Radio	Up to 6 GHz	Up to 100 MS/s (Rx) and Up to 400 MS/s (Tx) according to their version	Gigabit Ethernet, USB2, USB3 and PCIe depending on their version	Windows, Linux and Mac	General Applications
Microsoft SORA [59]	GPP	SORA SDK	2.4 GHz-5 GHz	40 MS/s-44MS/s	PCI	Windows	General applications
SDR4all [66]	GPP + FPGA	MATLAB	400 MHz - 4 GHz	20 MS/s, 200 MS/s	USB	WINDOWS, LINUX	General applications
UDPSDR-HF1 [67]	FPGA	BeMicro SDR	100 KHz - 30 MHz	80 MS/s	USB	Windows	Military and Commercial applications
Realtek RTL2832U DVB-T tuner	CMOS[6] + GPP and Audio card	Pre-build, Limited GNU Radio	52 MHz-2200 MHz	2.8 MS/s	USB	Windows, Linux and Mac	Amateur radio
RDP-100 [68]	FPGA + PowerPc	Pre-build	RX 0-125 MHz; Tx 0-200MHz	Rx-250 MS/s; TX-800 MS/s	PCI	Embedded System	General applications
BladeRF [69]	FPGA + ARM processor	Pre-build	300 MHz-3.8 GHz	40 MS/s	USB 3.0 SS	Windows, Linux, Mac	Amateur Radio, Research
Bitshark Express RX [70]	FPGA + GPP	Kit	300 MHz - 4 GHz	105 MS/s (Rx only)	PCIe	Windows and Linux	GSM, iDEN, CDMA2K, UMTS, TD-SCDMA
FlexRadio SDR-1000 [71]	DSP	Pre-build (PowerSDR)	12 KHz - 60 MHz	Not indicated	Parallel port	Windows	Amateur radio
Matchstiq [72]	GPP and FPGA	Prebuild SDK	300 MHz - 3.8 GHz	40 MS/s	USB	Windows, Linux and Mac	General applications
NI Flex RIO [73]	FPGA	NI LabView Matlab	200 MHz-4.4 GHz	1.6 GS/s	PXI Express x4	Windows	General applications
HackRF [64]	GPGA and GPP	GNU Radio	30 MHz - 6 GHz	20 MS/s	USB	Windows, Linux and Mac	General applications

Table 2.3: Non-exhaustive list of SDR platforms

63

7 Summary

Through a theoretical study, differences between a pure SR and Software Defined Radio have been detailed. A limitation of a pure Software Radio is the intermediate frequency part, since DAC and ADC sample at a low-rate with high-power consumption. The hardware dimensions and radio-frequency performances are also other constraints for SDR implementations. Flexibility is the main opportunity of the SDR. The radio parameters can be changed easily by the radio designer or by the radio itself.

We established a survey of SDR standards, architecture and platforms. We focused on existing SDR architectures proposed for embedded devices. We showed two approaches: reconfigurable hardware architecture and GPP based architecture. The latter brings more facility with high level programming languages. Then, we expanded our survey to the classes of SDRs considering their programming model and hardware. We analyzed most important SDR platforms. From the performed study, we have chosen an open source GNU Radio/USRP platform as it provides, open-source high level languages with a well established community.

Chapter 3

Analysis of GNU Radio and USRP SDR

Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousands of slaves.

Alan Kay

Contents

1 Introduction 66

2 GNU Radio 66

2.1 Programming language layers 67

2.2 Software blocks 67

2.3 Flow graphs 69

2.4 Software scheduler 70

2.5 SIMD programming (Volk) 71

3 Universal Software Radio Peripheral 72

3.1 USRP Architecture 74

3.2 Transmit and Receive Paths 74

3.3 RF daughter boards 75

3.4 Firmware and FPGA images 75

3.5 Universal Hardware Driver (UHD) 76

4 GNU Radio and USRP properties 76

4.1 Latency and throughput 77

4.2 Buffers organization 78

4.3 Performance counters and ControlPort 78

5 Advantages of GNU Radio and USRP 79

6 Summary 80

1 Introduction

An SDR platform architecture based on General-Purpose Processor (GPP) or a combination of GPP and DSP is relatively inexpensive. It is advantageous in terms of programming environment and tools. Thus, several open source projects have been developed by the community of researchers, hobbyists and industries. As shown in chapter 2, the GNU Radio and USRP are suitable for rapid implementation of wireless networks' specifications. In this chapter, this platform is described from top to down, starting from GNU Radio to **U**niversal **S**oftware **R**adio **P**eripheral (USRP).

The GNU Radio toolkit can be downloaded via the Internet, and a USRP can be purchased around 800 euros from *Ettus Research*. However, the biggest difficulty in an open-source software is the rigor in program writing throughout upgrading GNU Radio versions, *i.e.* change name of objects and methods. In most cases, these changes are not commented, and programmers need more time to adjust radio programs from old versions to new ones. In addition, the performances of SDRs might depend on both: Software/Hardware processing units and links, *e.g.* PCI, USB, Ethernet, etc, between these units. Hence, the analysis is needed to provide programmers with more information about software/hardware organization. Especially, we need to estimate the delay between two processing units. The analysis comes from the state of the art, and from reverse engineering performed through several experiments on the platform.

2 GNU Radio

GNU Radio is an open-source project toolkit for building software radios that run on (**G**eneral **P**urpose **P**rocessor (GPP)) [74]. The project was founded by Eric Blossom in early 2000, and revived by Thomas Rondeau in 2010. It can be used with readily-available low-cost external RF hardware to create software-defined radios, *i.e.* USRP, or without hardware in a simulated environment. It is widely used in hobbyist, academic and commercial environments to support both wireless communications research and real-world radio systems. Furthermore, the GNU Radio can be supported by Windows, Linux and Mac OS.

Based on the philosophy of Mitola [10], the toolkit provides signal processing blocks for modulation, demodulation, filtering and various data processing operations. In addition, new blocks can be easily added to the toolkit. Furthermore, a software radio program can be created by connecting these blocks to form flow graphs. Each

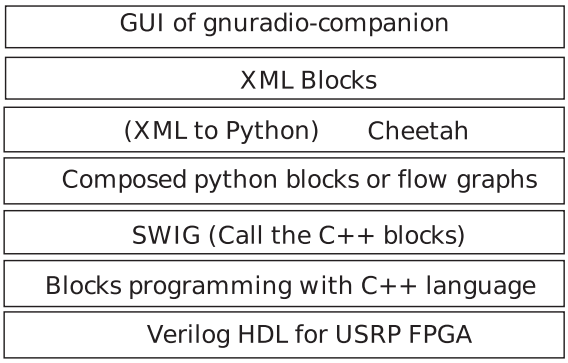


Figure 3.1: Software layers of the GNU Radio

block can be developed in Python or C++ programming languages. In section 2.3, we explain how to use these blocks to form flow graphs (or software radio programs).

2.1 Programming language layers

Figure 3.1 depicts the programming language layers of the GNU Radio. Processing blocks are written in C++ and then connected through a Python script. These C++ blocks are integrated in python script via an interface compiler called a **S**implified **W**rapper and **I**nterface **G**enerator (SWIG). To create a flow graph, we can use a graphical user interface called gnuradio-companion or directly via a python code. In this case, an E**X**tensible **M**arkup **L**anguage (XML) code describes C++ blocks to facilitate the visibility of the graph's blocks. The XML script is interpreted to a python code via the cheetah[1] tool. Finally, a Verilog HDL layer can be used to configure the **F**ield **P**rogrammable **G**ate **A**rray (FPGA) of a USRP board.

Recently, a new layer of QT programming language has been added to these layers. A graphical user interface of some blocks, such as the **F**ast **F**ourier **T**ransform (FFT) sink or the IQ (In and Quadrature) constellation, can be imported from QT package [75].

We note that several GNU Radio versions have been published in Git (git) repository of the GNU radio project. In each version, users can add new blocks or update old ones. Usually, the Python and the C++ codes are the most updated source codes, since they are the kernel of the toolkit.

2.2 Software blocks

From the previous sections, we can see that the blocks are the basic data structure of a GNU Radio system. They are connected to construct directional flow graphs. A stream of samples flows through the blocks from a source block to a sink block and vice versa. Generally, C++ is suitable for signal processing functions that need

[1] http://www.cheetahtemplate.org/

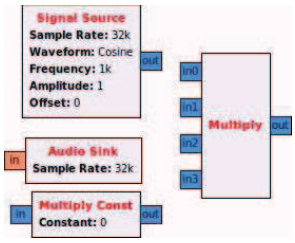


Figure 3.2: Source, Sink and Intermediate blocks

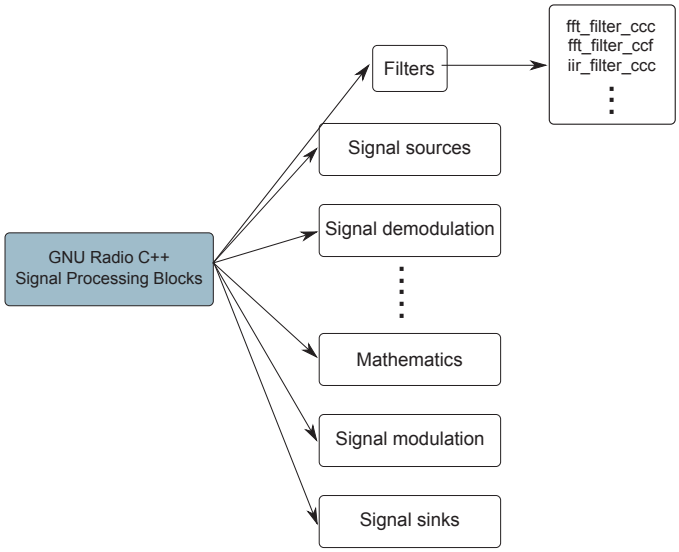


Figure 3.3: C++ signal processing modules

a short time response. Hence, Input/output streams are processed at high level without loss of stream's samples. For example, signal modulation and demodulation, equalization and Fourier's analysis have been implemented in C++. Data processing blocks, such as vector source or message generator, can be written in Python.

The type of data samples is commonly complex floats or interleaved floats. A first part is treated as real, and a second as imaginary part. In this case, the size of each sample is up to 8 bytes. Other sizes exist in order to represent Float (4 bytes), Short (2 bytes integer values) and Char (1 byte integer values). At a definition of a given block, a new type can be created from the predefined blocks. For example, vector types are defined to deal with a fixed number of samples. These types are useful, since they define one or several input and output ports of a block. As shown in Figure 3.2, source and sink blocks have only one side with ports, the output for sources and the input for sinks.

The design is object oriented based on a hierarchical organization with inheritance propriety [76]. It allows a community of GNU Radio to contribute easily and to change class diagrams. Figure 3.3 shows some modules of C++ signal processing classes. It was summarized from the Doxygen documentation of the project.

Signal processing blocks can be either synchronous or asynchronous. Synchronous

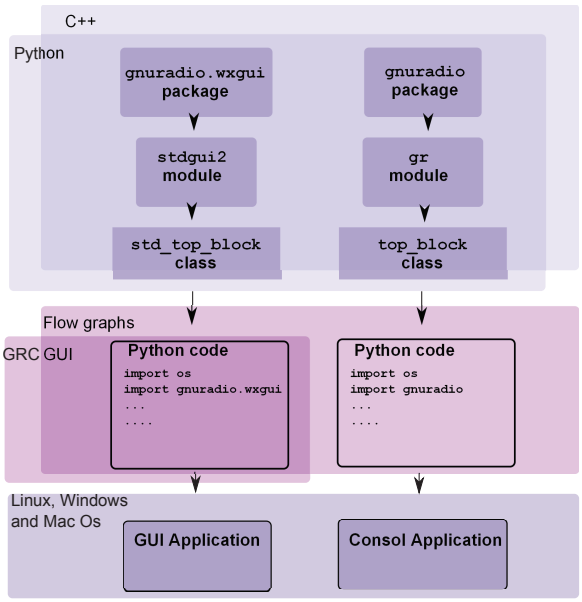


Figure 3.4: Programming layers of GNU Radio implementation

blocks need an integer relationship between the sample rate at input and output ports. Usually, they are derived from classes `gr_sync_block`, `gr_sync_ interpolator` and `gr_sync_decimator`. Asynchronous blocks are directly derived from `gr_block`. Blocks are usually written in C++ while a `work` function is the main function which processes an input flow. Furthermore, it is possible to wrap up several blocks into a higher level block. This can be done in Python by deriving a class from `gr.hier_block2`.

2.3 Flow graphs

The designer can create a flow graph in two ways, through a GUI interface called Gnu Radio Companion (GRC) or python code. In GRC panel list, blocks are organized in groups, regarding their functions. For example, modulators are grouped in one drop-down list which contains AM, FM, Phase modulators. Furthermore, blocks drag-and-drop is possible for fast and easy block's handling. Through Python, designers of flow graphs need a deep knowledge about class diagram and how to instantiate new blocks. Figure 3.5 shows an example of GUI flow graph's representation. The three blocks are connected by edges from an audio source to a Wav File sink. Two blocks can be connected if the type of the source block's output, and the destination block's input is the same. For example, float is the type of both the output of the audio source and the input of the low-pass filter (see Figure 3.5).

Flow graphs are derived from two possible classes: the class `top_block` of the module `gr`, or `std_top_block` of the module `stdgui2`. The difference between two possibilities depends on application requirements. If flow graphs are executed on console, `gr` module is imported from `gnuradio` package. Otherwise, when designers

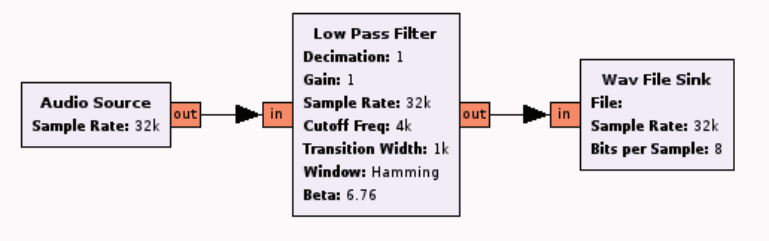


Figure 3.5: An example of a flow graph

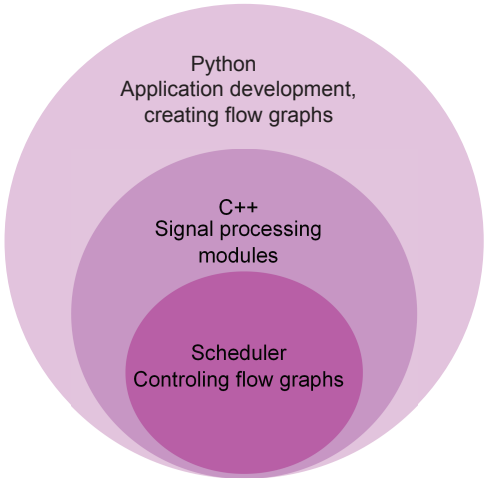


Figure 3.6: GNU Radio Software layers [78]

need an FFT **G**raphical **U**ser **I**nterface (GUI) in order to analyze a spectrum, the `stdgui2` is imported from `gnuradio.wxgui` (see Figure 3.4).

Actually, flow graphs are only built in Python, whereas to get them in a pure C++ is more efficient for embedded devices. However, some works are carried on handling GNU Radio for embedded architecture. In [77], **F**inite **I**nput **R**esponse (FIR) filter block has been mapped on Zynq based FPGA architecture.

2.4 Software scheduler

Figure 3.6 shows that a scheduler is at the heart of GNU Radio functioning. It interacts with C++ methods by managing blocks executions, and samples flow between them. Precisely, it is assisted by customized `forecast()` method, which defines the number of input samples required to produce a given output number of samples. In addition, the scheduler triggers block processing through `general_work` or `work` functions.

From flow graph debugging, we found two types of scheduler: single *threaded scheduler* and *Thread-Per-block Scheduler* (TPS). This result is confirmed in the literature by Chiang *et al.* [42].

By calling each block's executor, the Single Threaded Scheduler (STS) allocates

only one thread for each flow graph. It scans through the flow graph from source to sink looking for a block with sufficient input data and available output buffer (see section 3.10). When it finds an eligible block, it schedules the block for processing. Then, it continues with the next block. When it reaches the sink, it resumes at the source block. The scheduler is essentially a cyclic poller, calling each block in turn to perform its processing function, always cycling in the same order.

GNU Radio new versions (from 3.3.0) use TPS scheduler, which allocates a distinct thread for each block's execution. There is no global scheduling among blocks at runtime. A block's associated thread loops until GNU Radio code is terminated. In each loop, the thread first calls a block executor, which checks whether this block has sufficient input data and available output buffer. In this case, the executor calls the `work` function for current data processing. After that, it notifies neighbor block(s) on its status changes. Otherwise, this block waits for status change of its neighbor(s), and then checks its status again. Following the above mechanism, all the blocks in a flow graph together process incoming data stream.

We discovered the two types of schedulers by debugging C++ source codes of flow graphs. The open source Data Display Debugger (DDD [2]) has been used. It is a graphical front end of the command-line debugger Gnu Project DeBugger (GDB [3]).

2.5 SIMD programming (Volk)

Not far from a scheduler, the parallel programming architecture like Single Instruction Multiple-Data (SIMD) allows a programmer to perform one operation on multiple data points simultaneously. Commonly, signal processing blocks need more efficiency and processing time. SIMD is a solution to run faster a signal processing application.

Vector Optimization Library of Kernels (VOLK) has recently been proposed to support SIMD on GNU Radio [79]. It deals with code optimization within different SIMD architectures. For example, a SIMD architecture on x86, ARM and AMD are different. With Streaming SIMD Extensions (SSE), NEON and 3DNow! are dedicated to x86 processor, ARM and AMD, respectively. The VOLK's objective is to ensure the same processing time even if the programmer changes processor. Furthermore, VOLK provides a significant speed-up to signal processing blocks. In fact, the VOLK is a platform-agnostic interface (or an API) called kernel for each conceptual execution unit subject to SIMD vectorization. It has a set of proto-kernels designed for particular platforms, SIMD architecture versions, or run-time conditions.

When we analyzed the programming model of VOLK, we found three conceptual objects: kernels, archs and machines. The kernel is a C++ header file (extension `.h`) containing definitions of its proto-kernels which are static C++ inline functions. The arch is an abstraction for any hardware specific property. For a compiler, an arch corresponds to one or several flags, such as `-msse3` flag in a command line of GCC

[2] http://www.gnu.org/software/ddd/
[3] http://www.gnu.org/software/gdb/gdb.html

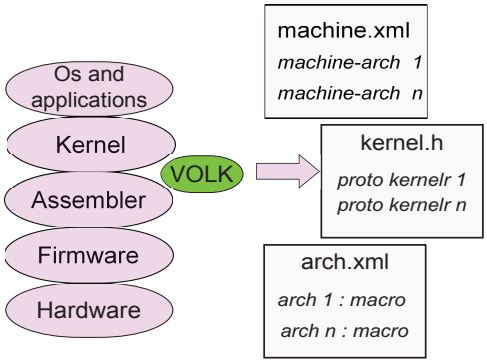


Figure 3.7: VOLK programming model

compilation [79]. For processor, an arch describes firmware attributes enabling the processor to execute a machine code of particular assembly instruction. Each arch corresponds to a VOLK macro and is an entry in an XML file. Finally, a VOLK machine is an abstraction of SIMD architecture for a processor. It describes the various architectural and software attributes required for a processor to run binaries within a shared object. The machine set is defined in an XML file, and it can be considered as a list of archs. The VOLK's position is between assembler and kernel of a host-computer's OS (see Figure 3.7).

Two primary conditions, buffer alignment and correctness, must be verified by a VOLK based block. The alignment is defined by buffer requirements for vector loads and vector stores. For example, in an SSE architecture, processing block needs 4 float per SSE register. However, forcing an output multiple can keep input and output buffers aligned to SIMD architecture requirements. As we have seen in Section 2.4, scheduler is involved in buffer management, since it passes sufficient data to produce output-multiple items. The VOLK ensures that all proto-kernels have the same behavior on a given machine. In addition, the Quality Assurance (QA) is a programming code used by designers to measure a behavior variance.

Dynamic arrival times of samples from source blocks are difficult to manage for the VOLK mechanism. When a number of input samples is not proportional to the output requirement, a sample processing is delayed. In fact, GNU Radio flow graphs keep wait remaining samples until enough samples are received ensuring alignment requirements. In the case of a significant inter-arrival time of samples, we then experience significant latency.

3 Universal Software Radio Peripheral

Universal Software Radio Peripheral (USRP) is the most common hardware platform to receive/transmit analog signals for SDR. It has been developed by Ettus Research [37] under GPL license to serve as a digital baseband and IF section of a radio communication system. For the time being, there are four USRPs versions available

on the market, categorized according to their connecting mechanism to a host computer and to their hardware performances.

Table 3.1 shows the versions (series) those we have used further in our work, *i.e.* Serial (USRP 1 and USRP B210), Ethernet (USRP N210) and Embedded (E110). The X series remain unused, since it is the newest version. All these versions can be connected to a host computer via multiple high speed interfaces, *e.g.* **P**eripheral **C**omponent **I**nterconnect **e**xpress (PCIe), dual 10 GigE, dual 1 GigE.

From the previous classifications, USRP hardware belongs to the GPP based architecture. The waveform-specific processing, such as modulation and demodulation can be executed on a host computer. A USRP consists of a mother board which can carry up to four daughter boards, depending on their versions. For example, USRP 1 can contain four Basic Rx/Tx daughter boards. In fact, the mother board holds analog interfaces connected to DACs and ADCs. In addition, it holds an FPGA which accomplishes high-speed general-purpose operations, *i.e.* digital up and down conversion, decimation and interpolation.

The performances of a USRP depend on frequency coverage and on analog bandwidth of daughter boards. Analog bandwidth is a useful bandwidth between an RF port and an IF/baseband interface of an RF channel. Typically, the analog bandwidth is set by IF or baseband filters on the daughter board. These filters are designed to avoid aliasing when paired with a USRP motherboard with ADC/DAC sample rates. Table 3.2 shows the announced analog RF coverage of some daughter boards.

	Name	**Host sample rate**	**ADC rate**	**DAC rate**	**Host Connection**	**RF bandwidth**
Bus series	USRP 1	8 MS/s	64 MS/s	128 MS/s	USB	Defined by a daughter boards placed in 2 slot
	USRP B200 and B210	61.44 MS/s	61.44 MS/s	61.44 MS/s	USB 3.0	MIMO card, 70 MHz-6 GHz
Networked series	USRP N200 and N210	50 MS/s	100 MS/s	400 MS/s	GigaBit Ethernet	Defined by a daughter boards placed in 1 slot
	Quad Receiver QR 210	50 MS/s	120 MS/s	120 MS/s	1 and 10 Gigabit Ethernet	700 MHz to 4 GHz
Embedded series	USRP E100 and E110	4 MS/s	64 MS/s	128 MS/s	OMAP [4] GPMC [5]	Defined by a daughter boards placed in 1 space

[4] **O**pen **M**ultimedia **A**pplications **P**latform (OMAP)
[5] **G**roup **P**olicy **M**anagment **C**onsole (GPMC)

X series	USRP E100 and E110	50 MS/s	200 MS/s	200 MS/s	PCIe, dual 10 GigE, dual 1 GigE	Defined by a daughter boards placed in 2 slots

Table 3.1: USRPs and their performances [37]

3.1 USRP Architecture

There is no general architecture proposed by Ettus Research for USRPs [37]. We can consider the architecture of an USRP N210 mounted with WBX daughter boards as a general one. Due to the numerous versions of USRPs, there is no general architecture. A USRP requires an RF front end, mixers, filters, oscillators and amplifiers, to translate the signal from the RF domain to the complex baseband or IF signals (see figure 3.8). As explained in Section 2.1, the baseband of IF signals are sampled by ADCs, and the obtained digital samples are clocked into the FPGA. The latter provides down-conversion, which includes fine-frequency tuning and several filters for decimation [37]. After decimating and through the host interface, raw samples (or data) are streamed to a host computer. The reverse process is applied to the transmission chain. Figure 3.8 presents a schematic organization of the USRP N210 from and toward a host computer. This organization can be considered as the general USRP architecture.

The bandwidth of a USRP device is a function of the analog and the FPGA processing bandwidth and the GPP bandwidth. The minimum of these three bandwidths is the system bandwidth. Thus, care should also be taken to prevent the analog bandwidth to be wider than the ADC/DAC sample rate of any device. For example, the USRP 1 has an Altera Cyclone FPGA with 64 MS/s dual ADC and 128 MS/s dual DAC. This USRP allows a host computer to receive/transmit data through a USB 2.0 connection. Since the sample rate of a USB 2.0 is limited to 8 MS/s then the bandwidth of the USRP 1 is only 8 MS/s.

3.2 Transmit and Receive Paths

Over GPP, radio application generates complex baseband, *i.e.* In and Quadrature signal components. This signal is sent to the USRP through USB, Ethernet or PCI Express. It is interpolated and digitally up-converted to IF by Digital Up Converters (DUCs) on an analog device. Then, DAC converts a digital signal into analog one and the RF daughter board takes over. Note that DUC is not performed by an FPGA (see Figure 3.8). However, the unique signal processing blocks of the transmitter in the FPGA are the Cascaded Integrator-Comb (CIC) interpolators [37]. The performances of the USRP DACs are recalled in Table 3.1.

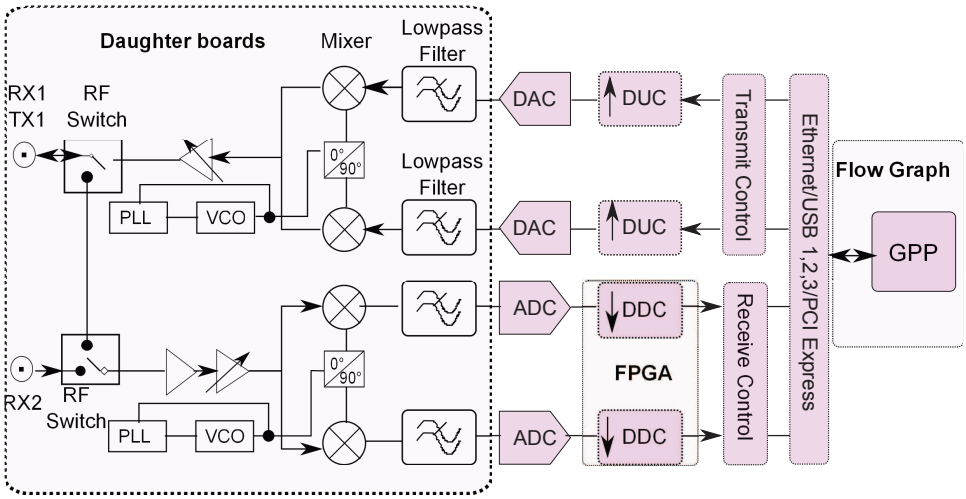


Figure 3.8: General USRP Architecture with a daughter board (USRP N210 with WBX) [37]

In the receiver path, the daughter boards down convert an RF signal to a baseband. The signal is sampled and converted to a digital stream through the ADC (see Figure 3.8). This stream is transferred to the FPGA which down converts (DDC) the received signal from IF to the baseband. Furthermore, the FPGA decimates the signal samples to adapt the sample rate to the data rate of the communication interface, *i.e.* Gigabit Ethernet or USB bus. Thus, the resulting signal is a complex baseband signal (waveform) at a given frequency, and the data rates are low enough to be transferred to the GPP.

3.3 RF daughter boards

RF daughter boards are the RF front end in a transmission chain of the USRP GNU Radio SDR. The USRP mother board can hold up to four slots, depending on USRP version, *e.g.* USRP N210 up to two, USRP 1 up to four. These slots allow us to plug in up to 2 basic receiver/transmiter daughter boards [80]. Primarily, the frequency tuning is accomplished in two steps. In the daughter board through Phase-Locked Loop (PLL) and via DDC. In an FPGA, the phase generator in Numerically Controlled Oscillator (NCO) is clocked at a frequency of the FPGA, *e.g.* 64 MHz in a USRP 1. Table 3.2 lists some daughter boards and their performances.

3.4 Firmware and FPGA images

Each USRP device must be loaded with special firmware and FPGA images. The methods of loading images into the device vary among device version. In a USRP 1, USRP-B and USRP-X, the host will automatically load the firmware and FPGA

Daughter board	Frequency coverage	Analog bandwidth
RFX 2400	2300 MHz-2900 MHz	Not specified
RFX 900	750 MHz- 1050 MHz	Not specified
SBX	400 MHz - 4.4 GHz	40 MHz
CBX	1.2 GHz - 6 GHz	40 MHz
WBX	50 MHz - 2.2 GHz	40 MHz

Table 3.2: Some daughter boards and their performances [37]

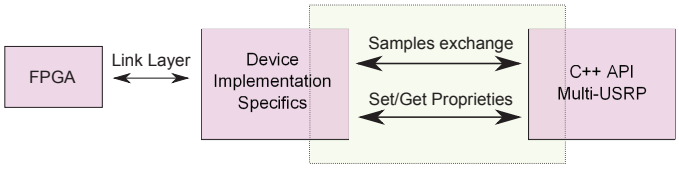


Figure 3.9: UHD in GNU Radio/USRP

image at run time. By contrast, the user must manually write firmware images onto the USRP 2 Secure Digital (SD) card. In USRP N series, designer programs an image into on-board storage, which then is automatically loaded at runtime. FPGA image is written in the VHDL language (see section 2.1).

3.5 Universal Hardware Driver (UHD)

UHD is a hardware driver library for all USRP versions and daughter boards. It provides a consistent Application Program Interface (API). It can be used as UHD driver standalone with other applications, such as Labview and Simulink. UHD finds devices on a USRP system and instantiates a device object in GNU Radio toolkit. It also allows the designer to update the desired parameters. The UHD sets/gets radio properties (*e.g.* gain, amplitude, center frequency, sample rate, and time) and transmits samples by using Operating System (OS) read() and write() operations. The UHD creates a sending or receiving stream between the host computer and the FPGA in the USRP to send and receive samples from/to the USRP. UHD also supports control and management messages such as Overflow, Stream command error (Rx path), Underflow and Sequence error (Tx path) [81]. UHD functionalities are wrapped into radio application, *e.g.* GNU Radio, by using USRP source and sink blocks (see Figure 3.9).

4 GNU Radio and USRP properties

Different operating systems and software applications can be executed on a GPP architecture. Hence, software radio applications might behave differently, and their performances can become difficult to predict by a designer. In particular, the problem

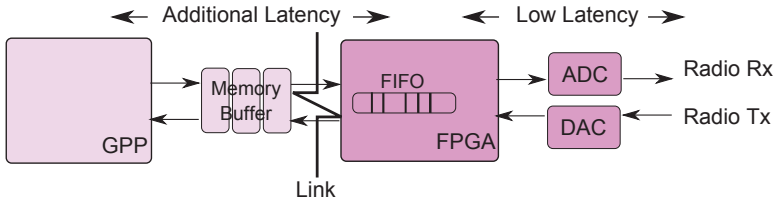


Figure 3.10: Latencies between GPP, FPGA and DAC/ADC

of identifying sources of a high latency and low throughput in communication chains is difficult. In this section, we discuss GNU Radio and USRP features and performances trade-offs.

4.1 Latency and throughput

The benefits of a GPP based architecture is associated to trade-off in performances. Typically, latency is the most undesirable drawback if processing chains with separated components. This latency comes from two sources: hardware and software setups. At hardware level, a latency is produced by the hardware link between the GPP and DAC/ADC. The delay between an FPGA and DAC/ADC is small compared to that between GPP and DAC/ADC. In fact, the latency depends on the link technology, *i.e.* USB, Gigabit Ethernet, PCI Express (see Figure 3.10). Thus, to deal with a hardware latency and data is buffered in the both directions from and to GPP.

At software level, additional latency is introduced by the scheduling and buffering of individual processing blocks (see Figure 3.11). Typically, block processing function is a function that reads a data stream from memory, processes it, and stores the output back to memory. Latency of processing blocks is not negligible, considering that it sums-up throughout a whole flow graph. Usually, buffer sizes are increased to limit the frequency of memory read/write operations. If the buffer size is small, then the data are sliced to chunks and the number of read /write operations increase. Thus, latency is increased across the flow graph since the relationship between buffer sizes and latency is typically proportional.

The throughput of a given buffer is the number of sample passing through the buffer time unit. The throughput of a block characterizes how fast an input signal can be processed. If the block's throughput is smaller than the latency generated in acquiring the samples to process, then the thread handling processing can either handle some other block's processing or sleep waiting.

To overcome the drawback of data buffering at GPP level, an FPGA can be proposed to interconnect processing blocks. However, this solution needs a soft-core processor to implement or to synthesize flow graphs in VHDL. GNU Radio addresses this problem by providing a possibility to impose a latency parameter individually, for each processing block or for all flow graphs [82].

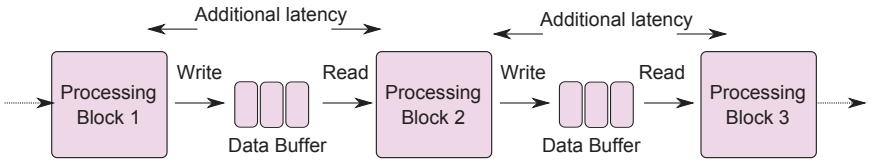


Figure 3.11: Latencies between Software blocks

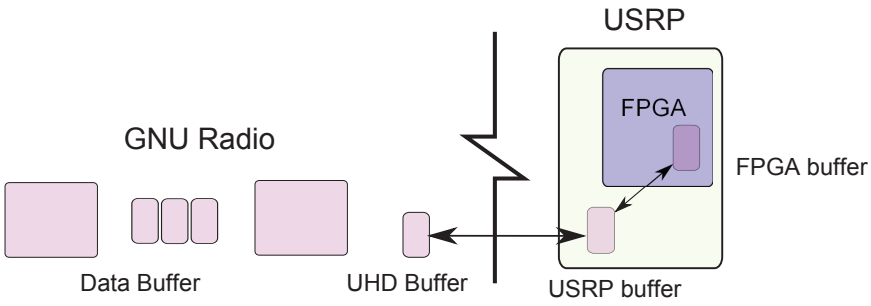


Figure 3.12: Buffers in GNU Radio USRP SDR

4.2 Buffers organization

The previous section motivates us to identify all types of buffers in the GNU Radio USRP SDR. We start from GNU Radio down to USRP hardware. The first buffer is that between signal processing blocks. The second is before Kernel bus driver. It allows GNU Radio to wait for sufficient data to generate a packet (USB, Ethernet or GPMC (General-Purpose Memory Controller) packets). The third buffer is the internal bus controller buffer on the USRP mother board. Finally, the FPGA buffer gets samples before processing them. Figure 3.12 summarizes all the buffers involved in a GNU Radio USRP SDR.

4.3 Performance counters and ControlPort

In order to estimate the processing time within each block, Thomas Rondeau *et .al* [43] introduce two inspection tools: *ControlPort* and *Performance counters*. The objective of these tools is to measure the work time of each block running in a GNU Radio flow graph. These tools allow a designer to find which block causes the maximum delay and try to optimize it at run time.

ControlPort is a GNU Radio tool that creates an integrated *remote* procedure or call interface to GNU Radio flow graphs [43]. It enables us to debug without requiring extra debug streams. Any GNU Radio block can register interfaces with ControlPort. Through these interfaces, ControlPort client interacts with the GNU Radio application, *i.e.* query and update properties of blocks. The general form of these interfaces is a "set" and/or "get" functions to adjust or query the state of a GNU Radio block's parameter. Figure 3.13 shows two ControlPort blocks with GNU Radio applications.

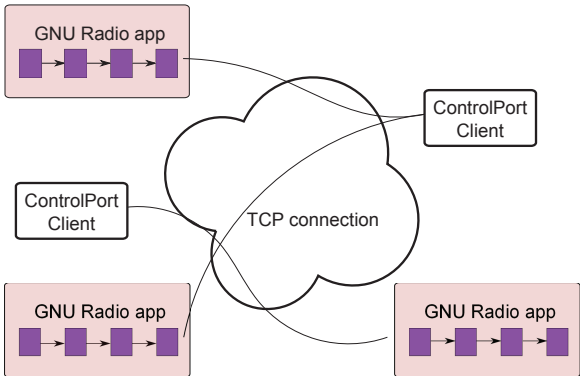


Figure 3.13: Controlport clients with GNU Radio applications over a TCP connection

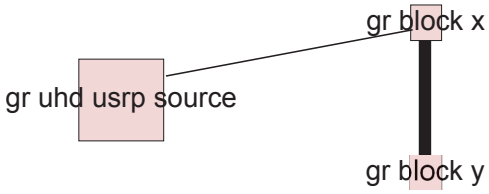


Figure 3.14: An example of the performance's counters graph of a given flow graph

Performance Counters are a way to generating performance measurements for each block running a GNU Radio flow graph. These counters keep track of various states of each block that can then be used for analyzing the performance and behavior of a running flowgraph. A specific package is launched by a designer to represent the GNU Radio flow graph over ControlPort. It constructs a graph of nodes (see Figure 3.14), which represents blocks and where edges are the buffers between blocks. The size of each node is proportional to the work time while the darkness of an edge is a function of buffers filling [43].

5 Advantages of GNU Radio and USRP

The GNU Radio has several advantages that we can summarize as follows:

- GNU Radio allows a designer to process in real time a signal stream (or waveform). Furthermore, the radio program can be simulated in a loop back.
- Over 100 blocks are available to easily develop new standards and applications as well as wireless networks.
- Python language is relatively easy to master. It describes processing steps throughout successive and connected blocks.
- Thanks to a Python script, a data flow can reach a maximum rate under processing blocks.

- The community of GNU Radio's designers is large. The toolkit can be carried out by a simple host computer on Linux, Windows and Mac.
- A flow graph (or communication chain) can be reconfigured even at run time. Several parameters of signal processing can be changed, such as frequency, power and sampling rates.

We can summarize some advantages of USRPs as follows :

- USRP is an open source hardware.
- It offers more flexibility to develop several applications with high level programming languages.
- It is compatible with Linux (2.6 kernel, any distribution), Mac OSX (PPC and Intel), Windows XP/2000, NetBSD and FreeBSD (Berkeley Software Distribution).
- An embedded SDR can be implemented with embedded version of USRP such as USRP E100 and E110.
- With Rx/Tx daughter boards, USRP can cover a large frequency band from 4 MHz to 6 GHz.

6 Summary

The USRP and GNU Radio GPP based SDR have been detailed. It is a well established open-source SDR platform. It allows a radio designer to prototype/implement a SDR transmitter/receiver. A prototype or an implementation requires a USRP to receive/transmit a radio signal. The latter is transformed to a digital stream and processed by flow graphs executed on host computer.

The GNU Radio's architecture has been analyzed by describing the programming language layers. It supports interesting and numerous signal/data processing functions, materialized by software blocks. These blocks are primarily developed in C++ and they are connected to define flow graphs or radio applications. The design of the GNU Radio is an object oriented with a vast class hierarchy. The SDR designer can use and instantiate the blocks as modules via a graphical user interface. However, for advanced use and evolutive modification, the designer needs more time to understand the class dependency and to master the C++ programming language.

The scheduler is a kernel of the executed flow graph. It can be of two types: single threaded scheduler and Thread-Per-Block Scheduler (TPS). The objective of the latest scheduler is to reduce the latency and improve the throughput of a flow graph at GPP level. Furthermore, blocks can be accelerated by programming them using Volk's library, which is based on SIMD architecture. Outside a block, data

stream is buffered in several levels of the GPP based SDR architecture. The buffers are not only between software blocks; a buffer is needed between OS kernel and GNU Radio. Hence, performance counters and ControlPort are proposed to estimate the processing delay within blocks and among them.

A general architecture of a USRP hardware composed of two parts: the mother board and a number of daughter boards. FPGA is irreplaceable across different USRP versions, since it accomplishes high-speed general-purpose operations. In the other side, the RF Front End or daughter boards define the frequency coverage for software applications at GPP level. However, technical information remains unexplored, especially, output power and the RF bandwidth of daughter boards. Thus, this unknown behavior leads us to carry a deep analysis of some daughter boards in the next chapter.

Chapter 4

Radio Frequency Measurements on USRP Daughter boards

No man's knowledge can go beyond his experience.

John Locke

Contents

1 Introduction 84

2 Problem statement 84

2.1 An overview of BPSK modulation 85

2.2 Related Works 90

3 Experimental approach 91

3.1 Hardware Setup 91

3.2 Software Setup 92

4 Spectrum Analyzer measurements 93

4.1 RFX2400 Daughter board 93

4.2 RFX900 Daughter board 95

4.3 SBX Daughter board 98

4.4 MIMO USRP B210 103

5 Measurements through flow graphs 104

5.1 RFX and SBX Daughter boards 105

6 Empirical model for SBX daughter boards 106

7 Summary 109

1 Introduction

Before performing a real implementation of wireless communications on a specific SDR platform, we would like to analyze performances of USRP daughter boards. More precisely, we would like to evaluate how these boards answer to GNU Radio commands? This question came from our first experiment when we were trying to get **B**it **E**rror **R**ate (BER)/**S**ignal-to-**N**oise **R**atio (SNR) parameters from a simple **B**inary **P**hase-**S**hift **K**eying (BPSK) modulation/demodulation. The RFX2400, SBX, RFX900 and B210 daughter boards are widely used to test and to validate research works [83] [84] while their RF performances have not been reported through the literature in conjunction with GNU Radio.

In this chapter, we adopt an experimental approach in order to test and to evaluate these performances. Thus, measurements are carried out via two tools: spectrum analyzer and GNU Radio itself. The obtained results are highlighted and discussed thoroughly. Furthermore, a new accurate empirical model is proposed, which is based on these results. Its usefulness comes from the possibility to predict the daughter board's output-power versus flow graph parameters.

2 Problem statement

The first stage of our exploration of the GNU Radio and USRP SDR, we have done simple digital and analog communications, such as an FM receiver/transmitter, an **A**utomatic **D**ependent **S**urveillance-**B**roadcast (ADS-B) receiver, and a simple **F**ast **F**ourier **T**ransform (FFT) or a spectrum analyzer. In addition, some digital modulations have been performed. For example, digital transmission of bits' frame through BPSK modulation has been accomplished. Over GNU Radio, two flow graphs can be constructed by using all in one modulator/demodulator block or connected elementary blocks. With both techniques, we observed that transmission was functional but very sensitive to the adjustment of flow graphs and USRP parameters.

The usability of the chosen SDR can be shown via these tests, but without a precise information about RF front end behavior. Commonly, daughter boards' output power and covered frequency bandwidth are useful information to feature. Furthermore, the behavior of flow graphs has not been featured when a flow graph controls these parameters. Particularly, UHD parameters of a USRP source and sink block are not documented. Thus, UHD Gain and Frequency parameters can be another aspect to clarify their impact via our experiments.

This section shows some results obtained when we implemented a BPSK modulator, motivating advanced tests and measurements.

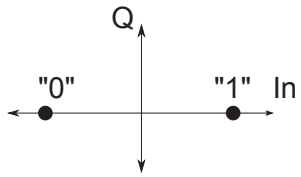


Figure 4.1: Constellation diagram for BPSK modulation

2.1 An overview of BPSK modulation

BPSK is a simple and robust digital modulation compared to other Phase Shift modulations. The robustness of BPSK comes from the difficulty to alter a demodulator decision. In the constellation diagram depicted in Figure 4.1, we distinguish without difficulty the symbols (points) in the complex plane. The real and imaginary axes are termed the In-phase and Quadrature axis. The two possible signals correspond to two bits, 0 and 1 , respectively, separated by a phase shift of π radians on the In-phase axis (see Figure 4.1). This yields to two phases, 0 and π, in the specific form with the following signals over a time t:

$$s_0 = \sqrt{\frac{2E_b}{T_b}}\mathbf{cos}(2\pi f_c t + \pi) = -\sqrt{\frac{2E_b}{T_b}}\mathbf{cos}(2\pi f_c t) \tag{4.1}$$

$$s_1 = \sqrt{\frac{2E_b}{T_b}}\mathbf{cos}(2\pi f_c t) \tag{4.2}$$

where :

- f_c is the central frequency of the carrier wave $\mathbf{cos}(2\pi f_c t)$
- $\frac{2E_b}{T_b}$ is the energy ratio per bit divided by time period T_b

The previous formulas simply highlight the simplicity of the modulation. A more detailed study of BPSK modulation can be found in [85] and [86].

2.1.1 The BER and SNR parameters

Bit Error Rate (BER) is a common parameter used to evaluate a digital modulation. It is a non linear function of the Signal to Noise Ratio (SNR). A bit error occurs whenever the transmitted bit and the corresponding received bit do not match; this is a random event depending on the noise level. Let n denote the number of bit errors observed in a sequence of bits of length N [86]. The BER is defined as:

$$BER = \lim_{N\to\infty} \frac{n}{N} \tag{4.3}$$

The SNR is a dimensionless parameter. It is measured at the receiver and represents the ratio of the average power of the received signal (*i.e.*, channel output) over the average power of noise measured at the receiver input. It is a common

practice to express the SNR in decibels (dBs), which defined as 10 times the logarithm (in base 10) of the power ratio. For example, signal-to noise ratios of 10, 100, and 1000 are 10, 20, and 30 dBs, respectively. The SNR can be used to measure the quality of analog systems [86].

In communication systems, a received signal $x(t)$ can be modeled as the sum of the desired signal, $s(t)$, and a narrowband noise signal, $n(t)$:

$$x(t) = s(t) + n(t) \tag{4.4}$$

The signal quality is the ratio of the variances of the desired and undesired signals [87]. On this basis, the signal-to-noise ratio is formally defined by:

$$SNR = 10.\log_{10}\left[\frac{\sigma^2_{s(t)}}{\sigma^2_{n(t)}}\right] \tag{4.5}$$

2.1.2 BER/SNR estimators on GNU Radio simulation

It is difficult to implement a real time SNR estimator. The estimation depends on the modulation scheme, channel model and synchronization. It requires more time of design than other signal processing blocks. A receiver and a transmitter must be synchronized. We primarily need three synchronization: frequency, timing, and phase. For each one, the issue is to have a reactive change of SNR's values. In the literature [88], we can find a number of estimators but without real computational possibilities.

Tom Rondeau [89] has implemented four estimators on GNU Radio, based on the works published in [88]. Primarily, they are dedicated to Multiple Phase Shift Keying signals and an **A**dditive **W**hite **G**aussian **N**oise (AWGN) channel. They inherit from one GNU radio block, and they are named the "simple," "skew," "M2M4," and "SVR" estimators. The M2M4 and SVR estimators are inspired from [88].

To highlight the difference between the four estimators, we performed loop-back GNU Radio simulations. The simulation's purpose is to evaluate the SNR estimators. The SNR is calculated via both techniques: a direct processing and python code. Apparently, the flow graph needs high processing, since it involves several programming layers. The two techniques are compared to a reference: the expected theoretical SNR.

The SNR estimator flow graph is depicted in Figure 4.2. It contains an SNR block which handles four estimator types adapted to an MPSK signal. A first block is a vector source, which generates a data bit stream. Then, White Gaussian Noise (AWGN) channel which carries a stream to the SNR estimator. Before flow graph execution, one SNR type can be selected, *i.e.* "simple," "skew," "M2M4," and "SVR". Finally, null sink drops the processed stream.

Figure 4.3 presents four curves of an estimated SNR from -5 dB to 30 dB. For each estimator "Simple", "Skew", "M2M4" and "SVR", the obtained values are depicted in

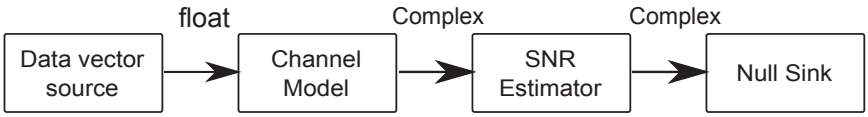
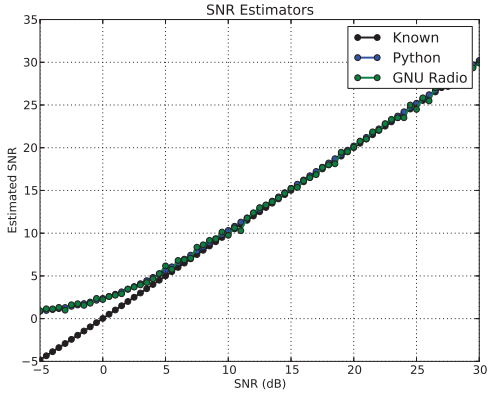
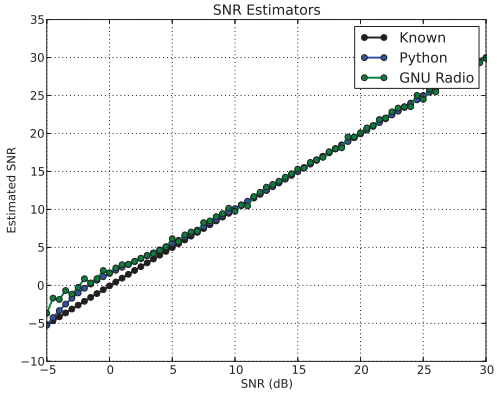


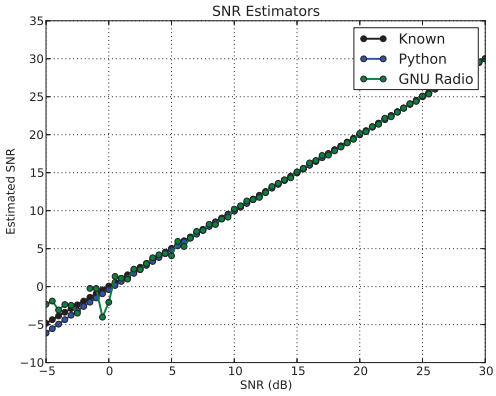
Figure 4.2: Loop back flow graph for SNR estimation



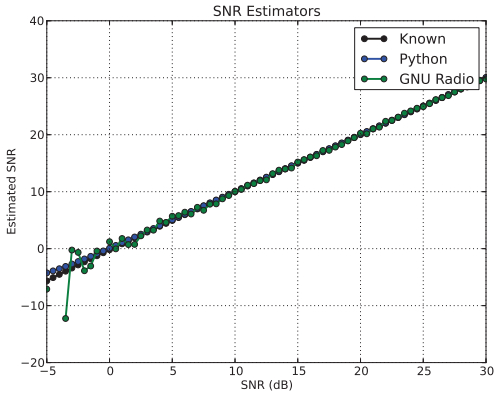
4.3.1: Simple estimator of the SNR.



4.3.2: Skew estimator of the SNR.



4.3.3: M2M4 estimator of the SNR.



4.3.4: SVR estimator of the SNR.

Figure 4.3: Known theoretical SNR compared to the estimated ones through GNU Radio and python

Figure 4.3.1 4.3.2 4.3.3 4.3.4, respectively. The SNR is calculated by each estimator using a GNU Radio flow graph and using a Python program. The obtained results are compared to a linear function of expected SNR values. Higher than 5 dB, the four estimators behave the same near the theoretical results. For SNR values less than 5 dB, the estimator's results are different from each other. The values of "Simple" estimator can be calibrated above 5 dB. By contrast, the three remaining estimators present a harsh variation. Hence, the "Simple" estimator can be selected to perform a real time transmission.

Similarly to the SNR estimator, BER block can calculate the BER by comparing

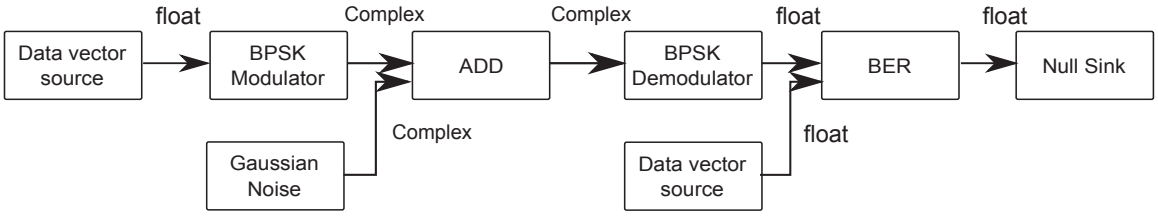


Figure 4.4: Loop back flow graph for BER estimation

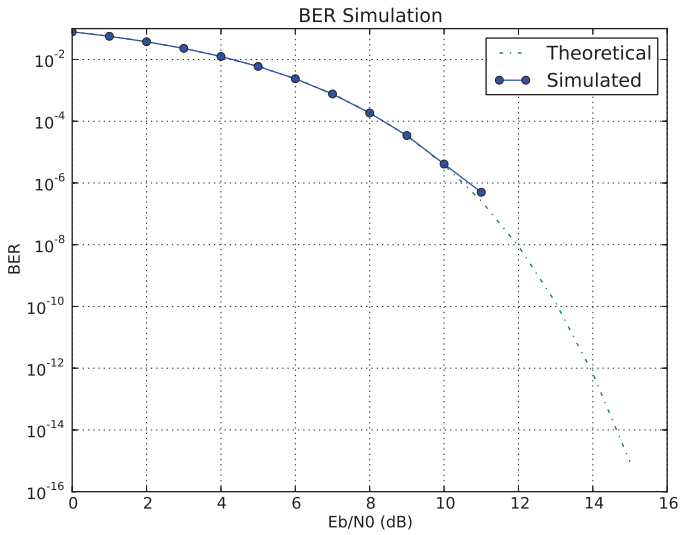


Figure 4.5: Compared BER versus Eb/N_0 obtained under simulation and theoretical processing

the demodulated vector stream with a vector source. Figure 4.4 shows a corresponding flow graph with a BPSK modulator/demodulator and a Gaussian Noise channel. The results of the simulation are compared to the theoretical ones. They are depicted in Figure 4.5. The BER is calculated versus Eb/N_0 (the energy per bit to noise power spectral density ratio). The curve of BER versus Eb/N_0 obtained by simulation matches that which is theoretically calculated as shown in Figure 4.5. The simulation results deviate slightly from the theoretical ones when output power is high. This result can be explained by an extra processing time needed by the flow graph.

2.1.3 BER/SNR estimators in real experiment

We focus in our experiments on a Hardware (Hw) setup with two USRP 1 connected to a host computer. Each one carries an RFX 2400 daughter board (see Figure 4.6). The two USRPs are within an indoor environment, and they are separated by a distance of 1m. The SNR as function of the BER are obtained using BPSK modulator/demodulator flow graph. The software implementation of the BPSK modulators is detailed in the next chapter when we report our implementation of

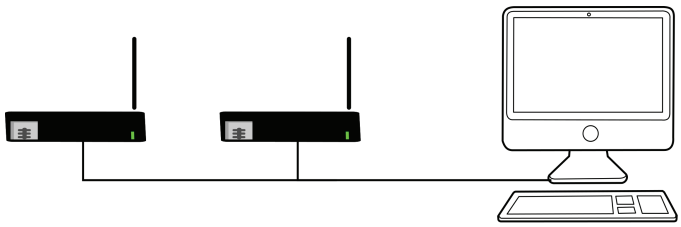


Figure 4.6: Two USRP 1 connected to a host computer which run a BPSK modulator/demodulator flow graph

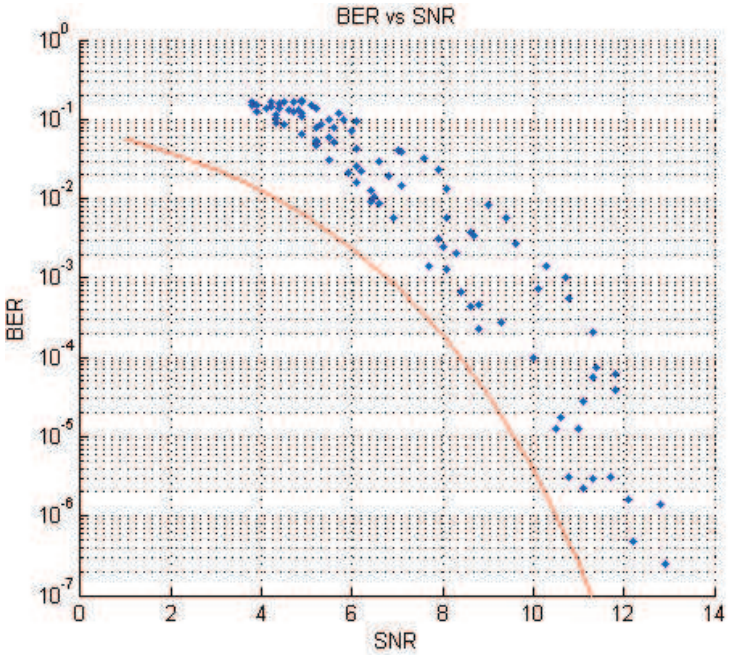


Figure 4.7: BER versus SNR obtained with a BPSK modulation on RFX2400 daughter board

the IEEE 802.15.4 specifications. The USRP 1 contains two RFX2400 daughter boards for a receiver and transmitter. Through a transmitter flow graph, the signal is amplified by sweeping an interval of possible output powers.

Figure 4.7 shows the estimated SNR/BER and a **M**atched **F**ilter **B**ound (MFB) curve in red. The SNR is estimated using Simple SNR estimator in black [89] [88]. In another hand, BER is calculated by comparing the demodulated bits sequence with the known expected one. MFB curve is the theoretical threshold of the SNR/BER. It can be calculated via Matlab tool [90]. The transmitter amplifies the signal strength through the GNU Radio flow graph. The amplifier is handled by a software parameter called DAC. It is implemented in GNU Radio block and it will be detailed in Section 3.2. Consequently, the receiver's flow graph measures the SNR and BER. However, the measurements reported for each output power must be accomplished in the same conditions *i.e.* same distance between the two USRP 1 and fixed carrier frequency.

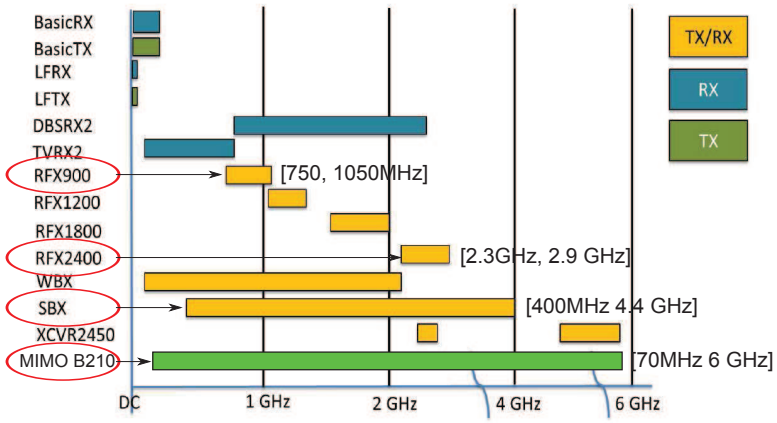


Figure 4.8: Daughter boards frequency band coverage

Results of BER depending on SNR give a scatter plot above the MFB. The obtained points do not exceed the MFB limit, so the SNR estimator is useful even though the shape of results is not linear. This non-linearity can be explained by the non-equalized power amplifier. Since the analog amplifier is handled by Front End Radio frequency, so we suspect its miss-behavior. Hence, the daughter board's amplifier should be analyzed via real time experiments and through several radio-frequency measurements.

2.2 Related Works

Before presenting our experimental method, we give a brief state of the art about works dealing with daughter boards' performances.

Ettus research [37] claims that the selection of an RF daughter board is made solely on the application requirements for frequency coverage. For example, the RFX2400 and SBX boards are good candidates. They cover the **I**ndustrial **S**cientific **M**edical (ISM) frequency band of 2.4 GHz for applications of wireless sensor networks. Figure 4.8 depicts graphics, which sums up the frequency coverage of these daughter boards. Even though the manufacturer has announced these frequency bands, the maximum output power remains unknown under each band.

Some recent results dealing with performance evaluations of the N 210 USRP have been published in [91]. In this work, the frequency stability and the phase differences have been measured between two SBX daughter boards carried by USRP N210. The authors concluded that the carrier frequency is sufficiently stable and that the phase error can be a neglected. Other performance evaluations are given in [56], where the USRPs 2 were used at the data packet level. Through this work a measure of communication delay, jitter and throughput of wireless transmissions is achieved. In [92], the authors objective is to evaluate the cooperative diversity in a cellular network via USRP experimental. Their results showed the problem

of output-power variation from one USRP to another. As a solution, the authors proposed to calibrate manually the transmission power, each USPR needing to be attached to a spectrum analyzer. Neither the USRP series, nor the daughter board was specified. The lack of details makes the reusability of their results difficult.

Our published work [2] is the first work dealing with frequency bandwidths characterization of USRP daughter boards. Our RF measurements published in [2] and [3] are organized into four sections:

- RFX2400 daughter board
- RFX900 daughter board
- SBX daughter board
- **M**ultiple-**I**nput and **M**ultiple-**O**utput (MIMO) USRP B 210

3 Experimental approach

Our experiments are based on an empirical method that arbitrates between several hypotheses. The objective is to verify the hypotheses announced by the daughter boards' manufacturer, *i.e.* Ettus Research, particularly the frequency bandwidth and output power parameters of some daughter boards. Furthermore, we analyze the linearity of the output power depending on amplifier values of a flow graph.

3.1 Hardware Setup

As mentioned in Section 3, the USRP N210 provides higher-bandwidth and higher-dynamic processing range capabilities than the other USRP versions. The USRP architecture includes a Spartan 3A-DSP 3400 FPGA from Xilinx, 100 **M**ega **S**amples **P**er **S**econd (MSPS) dual ADC, 400 MSPS dual DAC and Gigabit Ethernet connectivity to stream data to and from host processors. A modular design allows the USRP N210 to operate from DC to 6 GHz, while an expansion port allows multiple USRP N210 series devices to be synchronized and used in an MIMO [37] configuration. An optional **GPS D**isciplined **O**scillator (GPSDO) module can also be used to discipline the USRP N210 reference clock within 0:01 parts per million (ppm) of the worldwide GPS standard. The USRP N210 can stream up to 50 MSPS, to and from host applications. Users can implement custom functions in the FPGA, or in the on-board 32-bit RISC softcore. The USRP N210 provides a larger FPGA than other USRP series, being able to run applications demanding additional logic, memory and DSP resources. The FPGA also offers the potential to process up to 100 MSPS in both transmit and receive directions. The FPGA firmware can be reloaded through the Gigabit Ethernet interface.

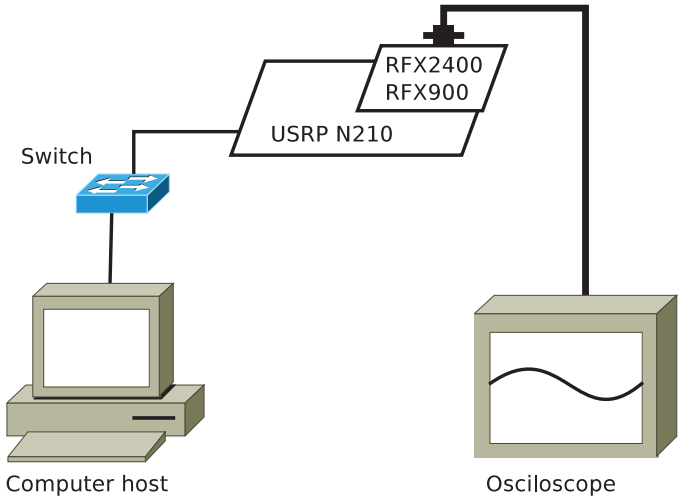


Figure 4.9: Experimental setup for the measurements

In Figure 4.9, we depict the hardware setup used in our measurements. A host computer is connected to a USPR N210 SDR via a Gigabit Ethernet Switch. The host computer uses a Linux distribution (UBUNTU 12.04 LTS)OS (Operating System) powered by an Intel Core i5-2400 CPU, clocked at 3,10 GHz and an RAM memory of 8 GB. The oscilloscope (spectrum analyzer) used in our measurements was a LeCroy 640 Zi [93] series oscilloscope, having an input bandwidth from 400 to 4000 MHz and having a sampling rate of up to 40 GS/s. The oscilloscope was directly connected to a TX/RX terminal of daughter boards through a high-frequency coaxial cable from ATEM 404-0202-xxxxA [94].

3.2 Software Setup

Two flow graphs have been implemented on GNU Radio toolkit: BPSK and sine-wave generator (see Figure 4.21). The BPSK is considered as a black box. The flow graph of this modulator was already shown in Section 2.1. The second flow graph is a source of a sinusoidal signal. For both flow graphs and before the **USRP H**ardware **D**river (UHD) sink, a complex stream is fed through an amplifier block called `gr.multiply_const_cc` (see Figure 4.21). This block multiplies the two components of a complex input by an amplifier DAC parameter. The latter can be represented in floating point for more precision in the interval $[0; 1]$. In Figure 4.21, only one of the two first blocks, *i.e.* Sinusoidal signal source and BPSK modulator, is activated at one time, to generate the baseband signal. BPSK modulator is launched in order to analyze the difference between a modulated and an unmodulated signal output. In this case, the transmitter generates a scrambled bit stream of 0 and 1 with a bit

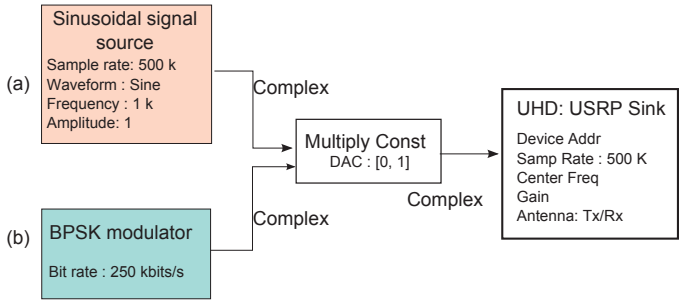


Figure 4.10: A simplified sinusoidal and BPSK transmitter flow graph

rate of 250 Kbits/s

The data brought after the measurements was subsequently processed and displayed using the Matlab R2012a software package from Mathworks Inc [90]. We focused on two parameters of interest of the daughter boards: their output RF bandwidth and the average output power versus the DAC value.

3.2.1 The expected DAC vs ouput power relationship

This DAC value parameter from GNU Radio is supposed to control the signal output amplitude of the device. Therefore, we expect the following relation:

$$\langle P_{out} \rangle = \mathrm{DAC}^2 \cdot \mathcal{P}_0 \tag{4.6}$$

where $\langle \cdot \rangle$ signifies the statistical average of the output signal (since it is supposed to be ergodic) and $\mathcal{P}_0$ is a reference power level. In the new version of the GNU Radio, we have $0 \leq \mathrm{DAC} \leq 1$. The central carrier frequency of each daughter board (f_0) was tuned in software according to their advertised frequency range.

4 Spectrum Analyzer measurements

4.1 RFX2400 Daughter board

RFX2400 is a daughter board transceiver designed to operate in the 2.4 GHz ISM band. It can be connected with a wide range of USRP devices. The RFX2400 provides a typical power output of 50 mW, and noise figure[1] of 8 dB [37]. It uses independent local oscillators for the transmit and receive chains and is MIMO capable. The advertised operational bandwidth is from 2300 MHz to 2900 MHz in both Tx/Rx modes [37].

The output of RFX2400 daughter board was measured thoroughly for frequencies in the range of 2350-2550 MHz. We also measured the frequency intervals 2300-2350

[1]Figure Noise: It is a numerical characteristic of a radio receiver that indicates how much its sensitivity to an incoming signal is decreased by the effects of its internal noise

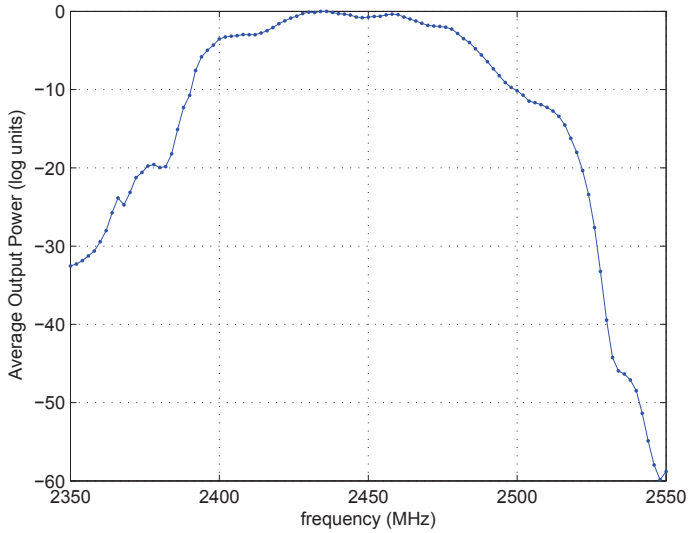


Figure 4.11: Measured bandwidth of the RFX2400 daughter board. The reference level of 0 dB was taken at 2434 MHz. Units are logarithmic (dB).

and 2550-2600 MHz. Nonetheless, the output levels were extremely low, therefore we focused only on the frequency interval discussed in this section.

4.1.1 Frequency bandwidth

The bandwidths of the daughter boards were measured by generating, with GNU Radio, a sinusoidal non-modulated carrier. Through the flow graph of Figure 4.21, the Sinusoidal signal source is activated and generates a carrier at central frequency f_0. Daughter board's Tx terminal receives the signal in its input from GNU Radio and transmits it via Tx antenna output. A high-frequency coaxial cable then carries signal to the spectrum analyzer.

The measurements are performed with a fixed DAC value of 1 *i.e.* the maximum power output. The obtained results swept through the advertised bandwidth, recording enough values in order to completely characterize the device's bandwidth. In Figure 4.11 the output power curves (in logarithmic units), setting the reference power to the maximum measured value, *i.e.* the maximum output power is set to 0 dB and all the other values from our graphics are negative in dB. We found the maximum value to be at 2434 MHz.

The measured -3 dB bandwidth is $B_{3dB} = 2480.5 - 2407.6 = 72.9$ MHz. If we accept a -10 dB attenuation, the bandwidth is $2390 - 2499$ MHz (109 MHz) and for a -30 dB attenuation we have a bandwidth of $2359 - 2527$ (168 MHz).

Furthermore, for the case of DAC $= 0.5$ the found -3 dB bandwidth is equal to $B_{3dB} = 2481.24 - 2399.52 = 81.72$ MHz. This bandwidth is slightly bigger than our result for DAC $= 1.0$. Thus, the advertised bandwidth of 600 MHz ($2300 - 2900$

MHz) is largely overstated.

4.1.2 Output power versus DAC value

The average power versus the DAC value was computed via two methods. In the first one, we created a custom function on the oscilloscope that computes the average of the squared amplitudes of the input samples *i.e.* $P_t \sim \langle v_i^2 \rangle_t$ where v_i represents the i^{th} input sample and $\langle \cdot \rangle_t$ represents time average. In order to have stable values, a supplementary averaging over 1000 oscilloscope sweeps was implemented. We used our flow graph shown in Figure 4.21 to generate a sinusoidal non-modulated signal and swept the DAC value from 0.1 to 1.0 in increments of 0.05 (we assumed that for DAC = 0.0, the residual output power is small enough to be ignored), each time recording the output power in the daughteroard Tx terminal.

The second method was based on the Spectrum Analyser software package installed on the LeCroy WaveRunner 640 Zi oscilloscope. The recorded spectral measurements were converted from dBm to linear power units and summed over a bandwidth of 2 MHz around the central frequency in the case of an unmodulated carrier and over a bandwidth of 5 MHz in the case of BPSK modulation. Therefore, the second estimator can be written as $P_f = \sum_f 10^{P_{dBm}(f)/10}$, where $P_{dBm}(f)$ is the vector of spectral power samples given by LeCroy's Spectrum Analyzer and f sweeps all the samples from the measured bandwidth. We performed the same measurements over the bandwidths of 10, 15 and 30 MHz and found no significant differences, which was the expected results since our signal is either a very peaked Dirac-like function (when we generate an unmodulated carrier) or has a bandwidth under 1 MHz (when using the BPSK benchmark).

In Figure 4.12 we depict the average power versus the DAC value for 6 different frequencies. The expected $\langle P_{out} \rangle \sim \text{DAC}^2$ law is quite valid throughout the whole range of the DAC parameter. Considering again the spectral power graphic results (Figure 4.11) we note that the spectral amplitudes for the 6 frequencies from Figure 4.12 are, in decreasing order, as follows: 2460, 2450, 2470, 2480, 2490 and 2500 MHz. This order can be found in Figure 4.12, where the curve corresponding to 2460 MHz is the top one while the one corresponding to 2500 MHz is the bottom one.

A major change is noted when using a BPSK modulation instead of the unmodulated carrier frequency. In Figure 4.13 we plot on the same graphic the average powers for both unmodulated and BPSK modulated signals. For a DAC value around 0.7, the $\langle P_{out} \rangle \sim \text{DAC}^2$ law breaks down for all BPSK curves. We can explain this phenomenon by the large peak-to-average ratio of the BPSK modulation [86].

4.2 RFX900 Daughter board

This transceiver daughter board is intended to operate in the 900 MHz band. It connects to all USRP SDR devices, making it a popular choice. It has a typical power output of 200 mW, and noise figure of 8 dB. The cellular and $902 - 928$ MHz

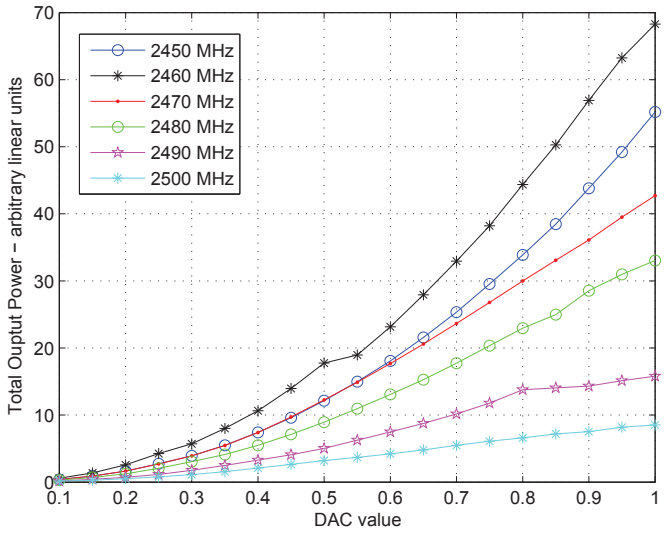


Figure 4.12: The average output power of the RFX2400 versus the DAC value for 6 frequencies (unmodulated carrier). The $\langle P_{out} \rangle \sim \mathrm{DAC}^2$ law is closely followed.

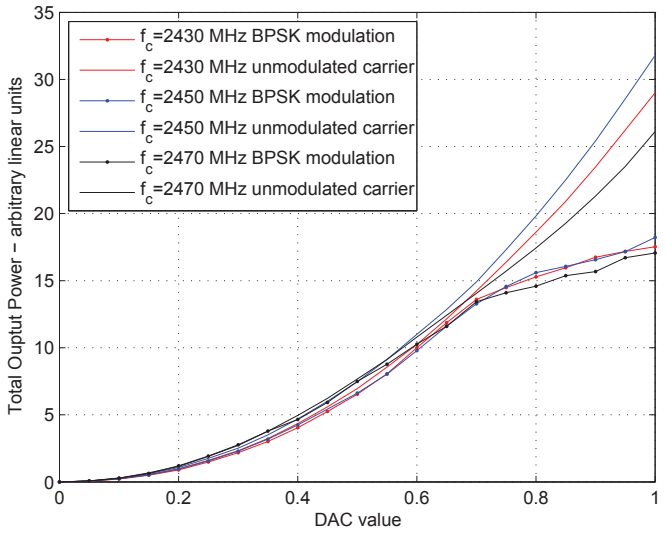


Figure 4.13: The average output power of the RFX2400 versus the DAC value for 3 frequencies in the cases: unmodulated carrier and BPSK transmission. The $\langle P_{out} \rangle \sim \mathrm{DAC}^2$ law breaks down for BPSK at DAC $= 0.7$.

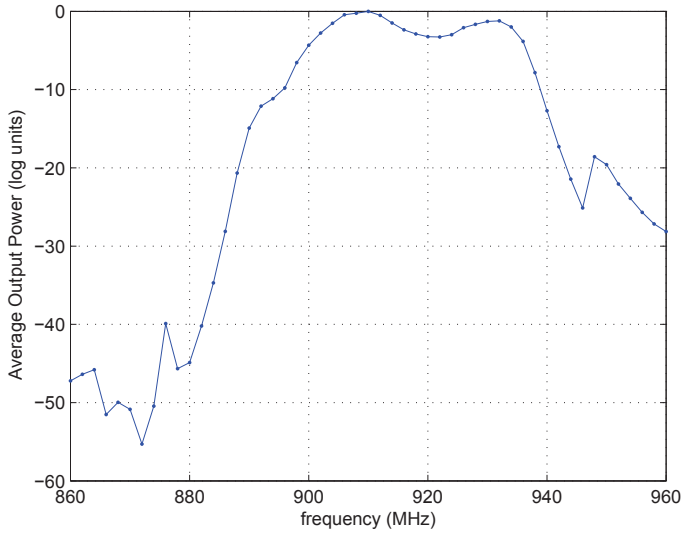


Figure 4.14: Measured bandwidth of the RFX900 daughter board. The reference level of 0 dB was taken at 910 MHz. Units are logarithmic (dB).

ISM bands are specified by Ettus research as possible applications of this board. The advertised operational bandwidth is $750 - 1050$ MHz in both transmit/receive (Tx/Rx) modes [37].

The output power of the RFX900 daughter board was thoroughly measured in the frequency band of $860 - 960$ MHz. We summarize the obtained results below.

4.2.1 Frequency bandwidth

The bandwidth of the RFX900 daughter board has been measured using the same method explained previously. The DAC parameter has been fixed at a value of 1.0 throughout the measurements. The results are depicted in Figure 4.14. The maximum measured value of the output was at $f = 910$ MHz, where we set our reference of 0 dB. We found the -3 dB bandwidth[2] of the RFX900 daughter board to be $B_{3dB} = 935.09 - 901.7 = 33.39$ MHz and its -10 dB bandwidth $B_{10dB} = 938.87 - 895.5 = 43.37$ MHz.

We conclude that the measured bandwidth of our RFX900 daughter board is far smaller than the advertised.

4.2.2 Output power versus DAC value

For this measurement we used an unmodulated carrier frequency on four frequencies and the DAC value was modified, as explained earlier.

[2]As can be seen in Figure 4.14, there is a small dip below the -3 dB level in the center of the graphic, from 918.5 to 924 MHz. Nonetheless, this dip has a maximum depth of -3.285 dB at 928 MHz, therefore we will disregard it and consider the RFX900's B_{3dB} as discussed in the main text.

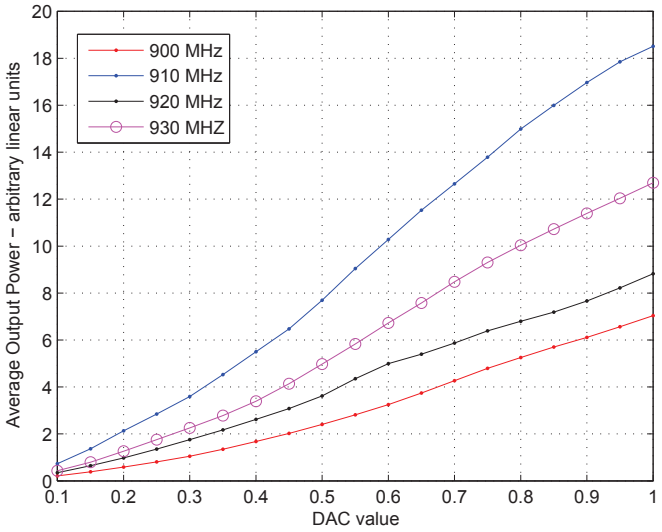


Figure 4.15: The average output power of RFX900 versus the DAC value at 900 MHz (unmodulated carrier).

The average output power versus the DAC value is depicted in Figure 4.15. The expected $\langle P_{out} \rangle \sim \text{DAC}^2$ law is valid only for DAC ≤ 0.5. If $DAC > 0.5$, the average output power shows a non-quadratic behavior.

According to Figure 4.14, the highest power is expected for a carrier on 910 MHz, a result confirmed in Figure 4.15.

4.3 SBX Daughter board

The technical data of the SBX daughter board is given by the manufacturer in [37] and [95]. The ADL5375 [96] is used as broadband quadrature modulator, the chip being designed for operation from 400 MHz to 6 GHz. The Daughter board provides a wide frequency range from 400 to 4400 MHz and features an output power up to 100 mW (20 dBm) with a typical noise figure equal to 5 dB. The output gain flatness varies no more than 1 dB from 450 MHz to 3.5 GHz [96].

Given the wide operating bandwidth, the SBX is ideal for applications requiring access to a variety of different frequency bands like ISM frequency band applications, Radar, Satellite and **G**lobal **P**ositioning **S**ystem (GPS) receivers etc. The local oscillators for the receive and transmit chains operate independently with a dual-band operation. The SBX daughter board is also MIMO capable and provides 40 MHz of bandwidth.

The output power can also be increased using an UHD gain (or UHD_G) parameter. The UHD driver of an USRP sink block controls this gain. It was already examined in the previous Chapter in Section 3.5.

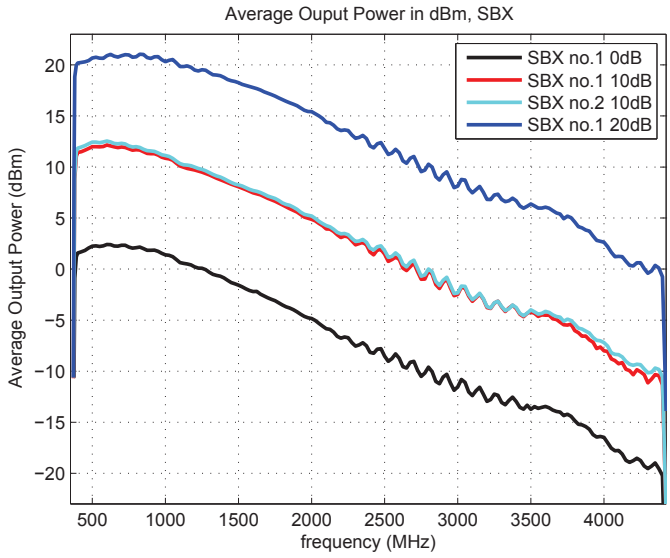


Figure 4.16: Measured bandwidth of the SBX daughter boards for UHD_G of 0, 10, 20 dB. The small oscillation between 2.2 and 3.4 GHz is probably due to a slight mis-adaptation in the SBX board.

4.3.1 Frequency bandwidth

The hardware and the software setup presented in Section 3.1 and Section 3.2 are reproduced in order to measure the frequency bandwidth on two SBX daughter boards. A sinusoidal signal was generated at a given carrier frequency f_1 and all important peaks in the SBX output spectral power were measured by the spectrum analyzer. The bandwidth measurements where performed by varying the central frequency f_1 in steps of 50 MHz, except at 400 MHz and 4400 MHz, where smaller steps were used in order to quantify the sharp drop in the output power. An extra gain parameter UHD_G was investigated over the entire bandwidth. The effect of the UHD_G was evaluated throughout the whole SBX bandwidth for UHD_G 0, 10 and 20 dB. The DAC value was kept constant at a value of 1 in these measurements.

Our results are depicted in Figure 4.16, where the found bandwidth size matches with the announced specifications, nonetheless the gain throughout (rather wide) passband was found to decrease as the frequency increases. For example, if the UHD_G is set to 0 dB, we found $P_1 = 1.84$ dBm (1.52 mW) at 900 MHz, $P_1 = -8.15$ dBm (0.15 mW) at 2400 MHz and $P_1 = -16.49$ dBm (0.02 mW) at 4000 MHz.

From 0 to 20 dB, we indeed found that this gain is reflected in the output power of the SBX board, as can be seen from Figure 4.16.

4.3.2 Total Harmonic Distortion (THD)

The supplementary parameter called UHD_G boosts the output power and can potentially distorts the output signal, causing massive out-of-band RF emissions. In order

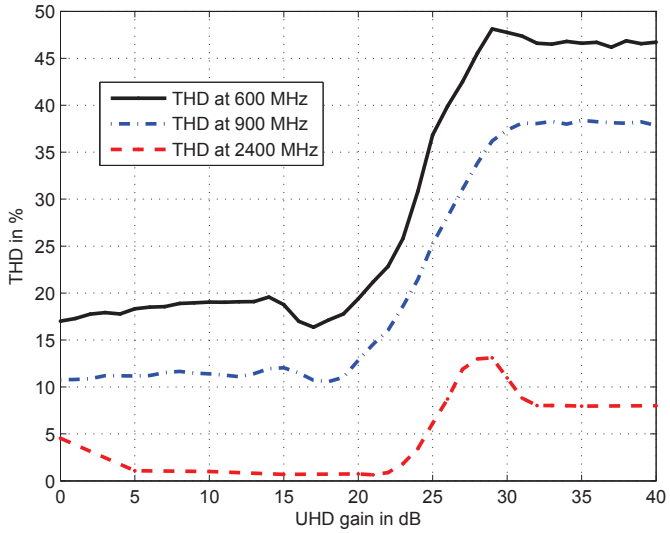


Figure 4.17: THD measured at carrier frequencies $f_1 = 600$ MHz, 900 MHz and 2400 MHz.

to quantify this distortion, we also measured the **T**otal **H**armonic **D**istortion (THD) [97] [98], a widely used parameter.

The distortion introduced by this UHD_G parameter was evaluated at 600, 900 and 2400 MHz. We continuously varied the UHD gain from 0 to 40 dB in steps of 1 dB and measured all output power peaks. The DAC value was constant and equal to 1. The Total Harmonic Distortion (THD) was calculated in order to quantify the level of unwanted harmonics in our sinusoidal waveform using the formula

$$\text{THD[in \%]} = 100 \cdot \sqrt{\frac{P_2 + P_3 + \cdots + P_N}{P_1}} \tag{4.7}$$

where $P_2, \ldots P_N$ are the **R**oot **M**ean **S**quare (RMS) powers measured on the $2^{nd} \ldots N^{th}$ harmonic and P_1 is the RMS power on the fundamental frequency f_1.

According to Figure 4.17, going beyond value of UHD_G equal to 20 for frequencies below 1.5 GHz causes a sharp increase in the THD as can be seen . This phenomenon can be explained by the saturation of the carrier frequency as seen in Figure 4.18. The board reaches its advertised maximum power of 100 mW (20 dBm) and the extra power demanded by the UHD_G parameter "leaks" into the higher frequency harmonics. For values of UHD_G above 25 dB, the heavily distorted waveform practically does not change anymore. It becomes obvious that for frequencies below 1 GHz, UHD_G values above 20 dB *should never be used*.

We also performed power measurements on all harmonics of interest for $f_1 = 600$ MHz and $f_1 = 900$ MHz. Our results are depicted in Figures 4.19.1 and 4.19.2. They improve our understanding of the sudden increase of the THD for UHD_G values between 20 and 25 dB. Indeed, mainly the second and fourth harmonics are

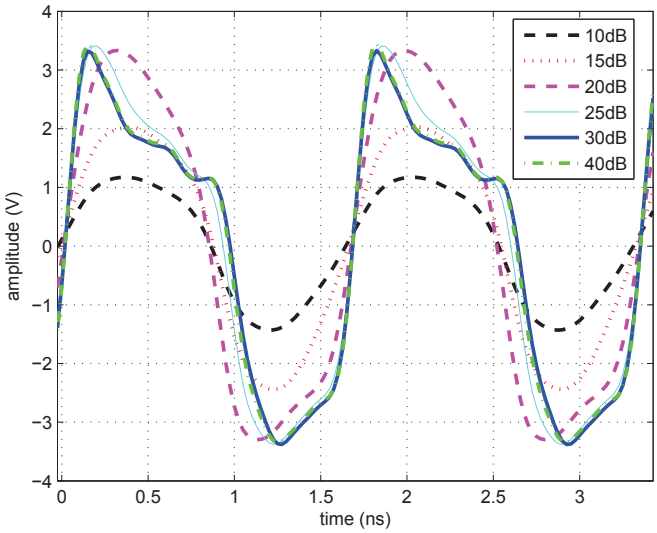


Figure 4.18: Time domain waveform at 600 MHz for UHD_G from 10 to 40 dB. Beyond the UHD gain value of 20 dB, the waveform becomes heavily distorted.

Freq.	**UHD_G value set in software (in dB)**						
(GHz)	**10**	**15**	**20**	**25**	**30**	**35**	**40**
2.4	1.68	6.77	12.05	17.14	**19.15**	**19.1**	19.1
3	-2.3	2.97	8.25	13.59	17.6	**18.35**	**18.34**
3.5	-4.05	1.26	6.53	11.42	16	**17.25**	**17.25**
4	-7.5	-2.37	2.92	7.86	12.79	**14.27**	**14.26**
4.4	-11.12	-6.17	-0.71	4.19	9.43	**10.88**	**10.89**
	Measured average output power (in dBm)						

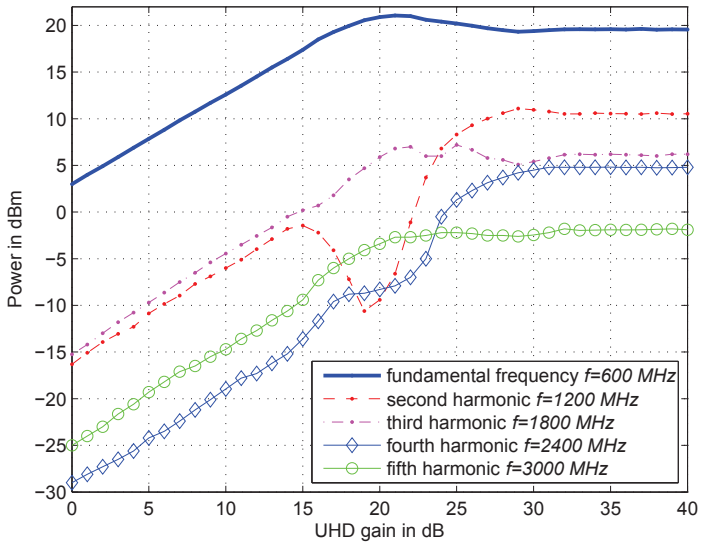
Table 4.1: Output Power at high frequencies and high UHD_G values

responsible of this sharp increase in the THD. In addition, the saturation of the power on the carrier frequency beyond UHD gains of 20 dB can also be clearly seen. The third and fifth harmonics show no sudden increases, and are quite similar to the carrier (fundamental) frequency they tend to saturate.

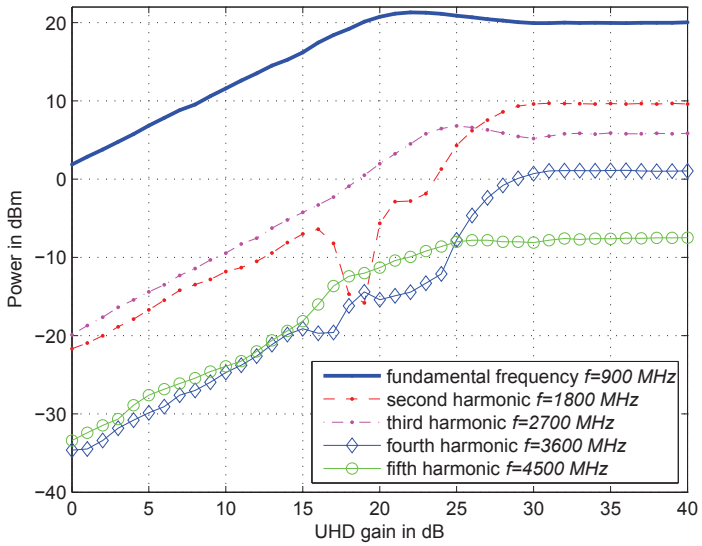
In Table 4.1, we summarize the effect of the UHD_G at some frequencies above 2 GHz. We can conclude that the UHD_G range $20 - 30$ dB is quite useful and should be used whenever needed. However, going beyond the UHD value of 30 dB brings no significant power increase, as the power on the carrier tends to saturate.

4.3.3 Output power versus DAC value

In Figure 4.20, we depict the "leakage of the output power" phenomenon. For an UHD_G of 15 dB the square law from Equation 4.6 is closely obeyed. However, for an UHD_G of 20 dB and for DAC ≥ 0.7, the square law is no longer followed, and



4.19.1: $f_1 = 600$ MHz. The second harmonic at $f_2 = 1800$ shows a sharp increase between 20 and 30 dB, partially explaining the increase in the THD.



4.19.2: $f_1 = 900$ MHz.

Figure 4.19: The measured output power for a carrier frequency $f_1 = 600$ MHz and $f_1 = 900$ MHz versus the UHD_G.

the total power (*i.e* on the fundamental frequency *plus the unwanted harmonics*) increases while the power on the carrier frequency saturates.

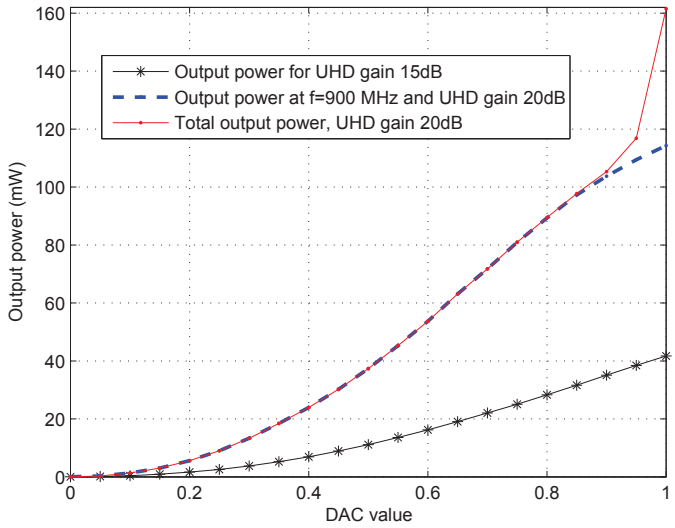


Figure 4.20: The output power on the carrier frequency and the total output power versus the DAC value for $\text{UHD}_G = 15$ and $\text{UHD}_G = 20$ dB at $f_1 = 900$ MHz. The square law from Equation 4.6 is closely followed for a UHD_G of 15 dB, however it breaks down for a gain of 20 dB and for DAC ≥ 0.7.

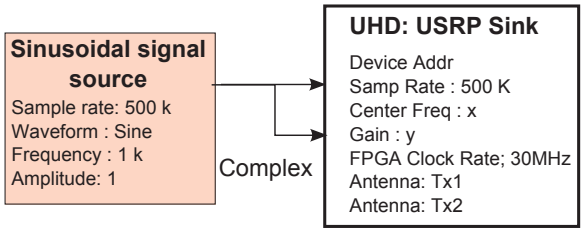


Figure 4.21: A simplified sinusoidal MIMO flow graph

4.4 MIMO USRP B210

This board is a fully integrated or single-board USRP platform with a large frequency coverage from 70 MHz to 6 GHz. It is able to stream up to 56 MHz of real-time RF bandwidth. The B210 integrates both signal-processing chains of the AD9361 transceiver [99], providing coherent MIMO capability (two transmitters and two receivers). The onboard signal processing and control of this transceiver are performed by Spartan6 FPGA connected to a host PC using SuperSpeed USB 3.0.

Obviously, across the frequency bandwidth, numerous applications can be handled such as **L**ong **T**erm **E**volution-**A**dvanced (LTE-A), TV broadcast, cellular, GPS, Wifi, ISM, etc. In the literature, this board has been used in wireless communication testbeds to evaluate the performance without precise information about its RF behavior. In [100], the **O**rthogonal **F**requency-**D**ivision **M**ultiplexing (OFDM) modulation has been evaluated under large Doppler spreads with USRP B210.

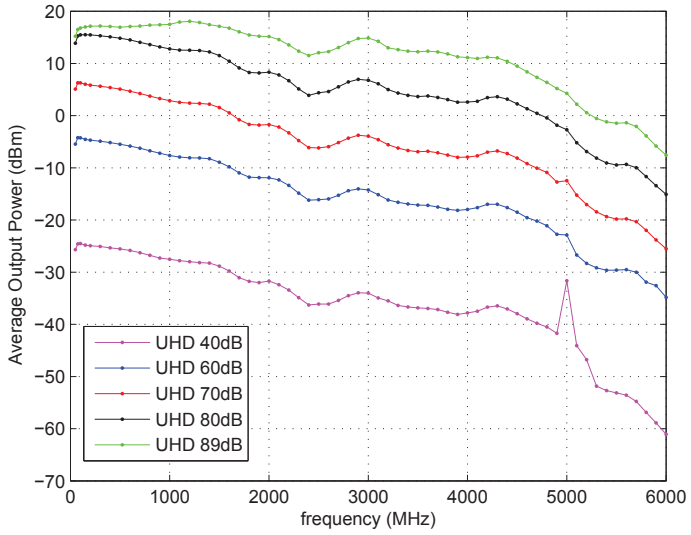


Figure 4.22: The measured frequency bandwidth of the MIMO B210 USRP board for different UHD_G values

4.4.1 Frequency bandwidth

Figure 4.22 shows the obtained bandwidth through different output power levels measured by increasing the UHD_G parameter, for values equal to 40 dB, 60 dB, 70 dB, 80 dB and 89 dB. The sinusoidal signal at each carrier frequency has been generated using an MIMO flow graph. The signal has been generated from the source block into two channels of the USRP sink (see Figure 4.22). The clock rate of the USRP's FPGA must be adjusted less than the threshold of 32 MHz. In our flow graph, the clock rate was equal to 30 MHz.

We have seen the same behavior of the software amplifier at the two MIMO Transmitter (Tx) outputs of the USRP. The measured values give the same bandwidth in the announced interval. Otherwise, the output leakage is proportional to the growth of the output frequency. In fact, a low output power is obtained with high central frequencies. The effective output power increases when the UHD gain takes high values. We can also notice that the signal saturation occurred when UHD gain was higher than 90 dB.

5 Measurements through flow graphs

The daughter boards output power has been analyzed previously via a spectrum analyzer. Consequently, this analysis remains valid only in output. At the receiver side, the daughter board's or GNU Radio entrance needs to be characterized. However, we perform some complementary measurements within GNU Radio and USRP SDR. We reuse the same transmitter's flow graph of the previous section with the sinusoidal

signal source. At the receiver side, a new flow graph has been created. It calculates a relative output power based on an FFT block. The latter is based on the energy detector to estimate the output of a time-averaged Power Spectral Density (PSD). More details about this block will be presented in Chapter 6.

As shown in Section 4.1.2 and 4.2.2, the RFX 2400 and RFX 900 are subject of output power leakage. Thereafter, these daughter boards daughter boards have not been used to transmit output signal. The SBX daughter board was the transmitter in this part of experimental tests, since its output power leakage is observed only in a small frequency band. Obviously, each daughter board has been handled by one USRP, and each one is connected to another through a coaxial cable.

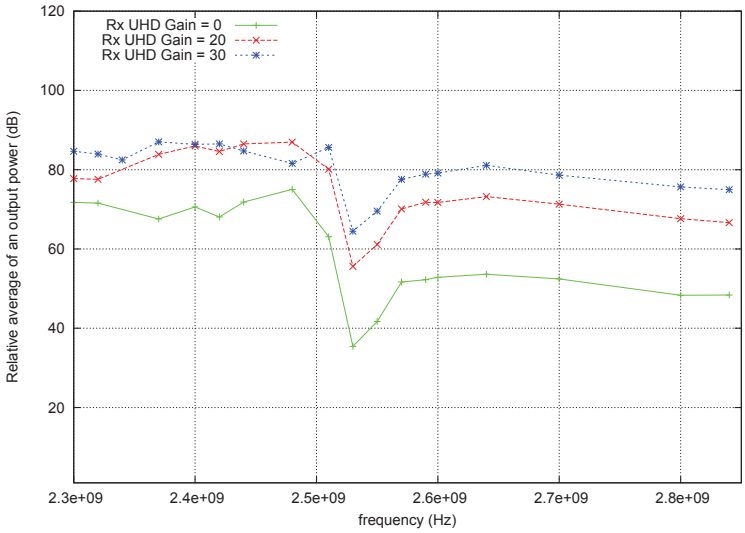
5.1 RFX and SBX Daughter boards

The hardware setup is simple with two USRP 1 connected to a host computer. It allows experiments to be performed following two cases: with two SBX daughter boards or with one SBX and another RFX daughter board. Each USRP 1 handles a given daughter board separately. For each condition, only the SBX daughter board generates the sinusoidal signal. The flow graph transmitter sets DAC to a maximum value, *i.e.* DAC=1, and the UHD gain to 40 dB throughout all experiments. Similarly to the transmitter, the flow graph receiver keeps the same UHD gain for each experiment. However, the relative output power is measured subject to a number of central frequencies. For 10 reported values, the receiver calculates the average output power.

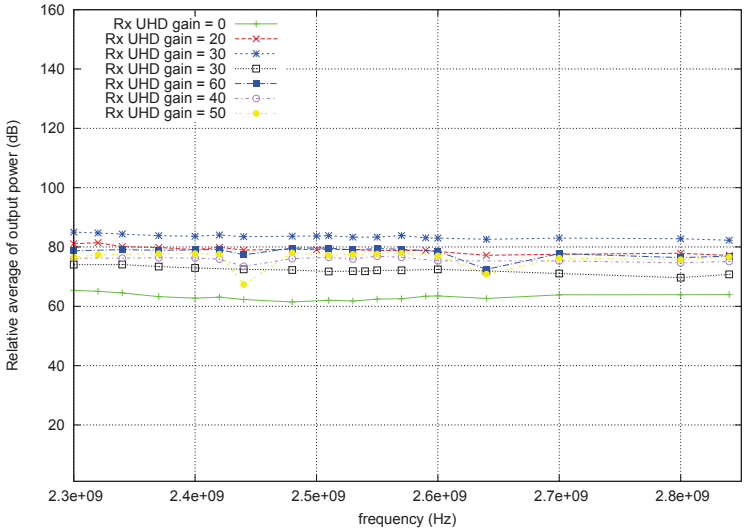
The RFX 2400 daughter board was controlled by a flow graph transmitter. The latter changes its UHD gain to 0 dB, 20 dB and 30 dB. The transmitter sweeps a frequency band from 2300 MHz to 2800 MHz with the same UHD gain value. Since more applications can be implemented from 2400 MHz to 2500 MHz, several central frequencies have been selected in this band. Figure 4.23.1 highlights results of experiments. The curve shows a sudden attenuation of the output power near 2525 MHz frequency. It indicates a leakage output power although the receiver increases its UHD gain. On another hand, a saturation is caused in the band from 2400 MHz to 2500 MHz. In fact, the output power with an UHD gain = 20 dB can be greater than that when UHD gain has a value equal to 30 dB.

In the second case where an SBX daughter board is a receiver, the average of the output power over the band from 2300 MHz to 2800 MHz is relatively steady (see Figure 4.23.2). This result has been similar for each UHD gain of the receiver raised by a step of 10 dB from 0 dB to 50 dB. However, the software amplifier of the daughter boards at UHD gain value equal to 30 dB with a measured output power around 85 dB. The output power remains around 85 dB or decreases lower even if UHD gain is higher than 30 dB.

The results from these experiments allow us to define a threshold of the UHD gain amplifier for SBX and RFX daughter boards. Commonly, UHD gain = 30 dB is



4.23.1: Relative average of output power measured with RFX daughter board.



4.23.2: Relative average of output power obtained with SBX daughter boards.

Figure 4.23: Output power results measured over SBX and RFX 2400 daughter boards within the GNU Radio USRP SDR

the maximum and useful UHD gain and beyond it, the amplifier saturates. Extended experiments can be carried out on RFX 900 daughter board.

6 Empirical model for SBX daughter boards

We consider the output-power behavior of the SBX daughter boards as the steady one. Thus, we extend the radio-frequency measurement to propose an empirical

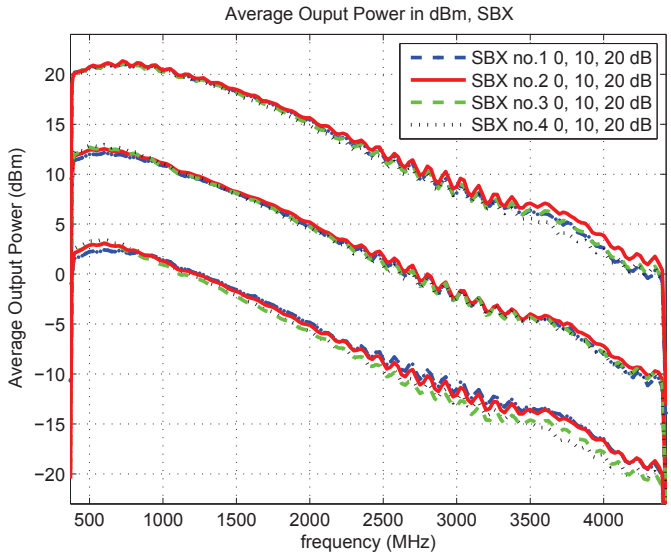


Figure 4.24: Average output power versus carrier frequency for UHD_G gains of 0, 10, 20 dB for the four measured SBX boards.

model. Thereafter, we use four SBX boards. As shown by Figure 4.24, the spread of the output power is fairly small among different boards. The small oscillation between 2.2 and 3.4 GHz is probably due to a slight mis-adaptation in the SBX board. In all measurements DAC = 1. This fact allows us to take the average value among the four boards and use it in order to find a best fit for an empirical model. All these results motivated us in the search of a simple, empirical model that can provide a fast and relatively accurate average output power of the SBX board as a function of the DAC value, the UHD_G and the output frequency.

Our first observation was a near perfect DAC dependence of the average output power, *i.e.* $P_{out}[\text{mW}] \sim \text{DAC}^2$, when UHD_G gain values are below 20 dB. The UHD_G gain was also found to behave as expected. However, the output power versus frequency is far from being flat. We found a roughly flat behavior only in the frequency band 400 MHz – 1 GHz. For frequencies above 1 GHz, the average output power is constantly falling with an almost linear drop in dB.

The objective of our model is to estimate reasonably the average output power for a software radio N210 with a SBX daughter board. We first take into account the contribution to the output power given by the DAC and UHD_G parameters. We have

$$P_1[\text{dB}] = 20 \log_{10}(\text{DAC}) + \text{UHD}_G \tag{4.8}$$

and P_1 models the partial contribution of these two parameters.

The most simple model for the average output power versus the frequency is a linear fit in dB, yielding

$$P_{out}[\text{dBm}] = P_1 + \alpha_0 + \beta_0 \cdot f[\text{MHz}] \tag{4.9}$$

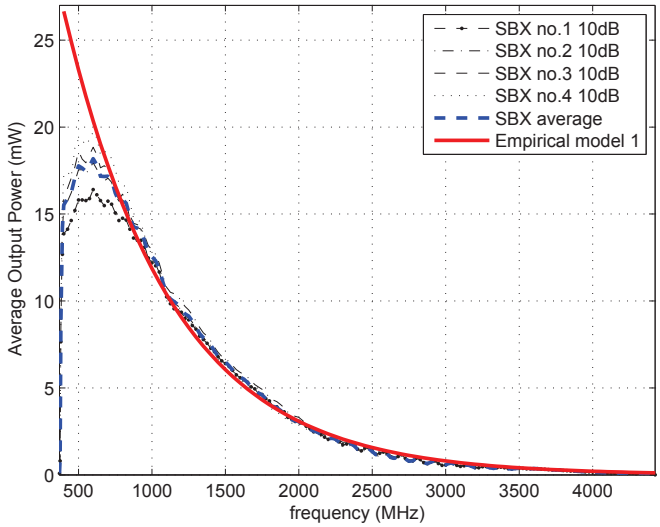


Figure 4.25: The average output (in mW) for the four measured SBX boards and the average output power (thick line) versus the empirical model given by Equation (4.10).

or, in normal units (mW) we have

$$P_{out}[\text{mW}] = \chi \text{DAC}^2 10^{\text{UHD}_G/10} 10^{\beta_0 \cdot f[\text{MHz}]} \tag{4.10}$$

where $\chi = 10^{\alpha_0/10} = 4.57$ and $\beta_0 = -5.856 \cdot 10^{-3}$.

In Figure 4.25, for $\text{UHD}_G = 10$ dB we plot the measurements results for each individual SBX board, the average value over the four measured boards (in thick line), together with the empirical model (4.10). For frequencies above 900 MHz the match is almost perfect. However, this simple model is not accurate for lower frequencies as below 600 MHz.

Therefore, we improve this empirical model with a more elaborate one, defined on two different spectral regions. For frequencies in the range $400 - 1000$ MHz the SBX board has a quadratic behavior, with a maximum output power at a frequency of 600 MHz. For frequencies above 1 GHz, the linear fit in dB is quite accurate. Therefore, we only adjust its parameters in order to yield a better fit. We end up with the following model (in dBm):

$$P_{out} = \begin{cases} P_1 + \alpha_1 + \beta_1 \cdot f + \gamma_1 \cdot f^2 & \text{if } 400 \leq f \leq 1000 \\ P_1 + \alpha_2 + \beta_2 \cdot f & \text{if } 1000 < f < 4400 \end{cases} \tag{4.11}$$

with the coefficients $\alpha_1 = -2.25$, $\beta_1 = 1.49 \cdot 10^{-2}$, $\gamma_1 = -1.167 \cdot 10^{-5}$, $\alpha_2 = 7.5$ and $\beta_2 = -6.4 \cdot 10^{-3}$ and f is expressed in MHz.

The model is compared to the measurement results in Figure 4.26. As it can be seen, throughout the whole frequency range of the SBX daughter board, the match

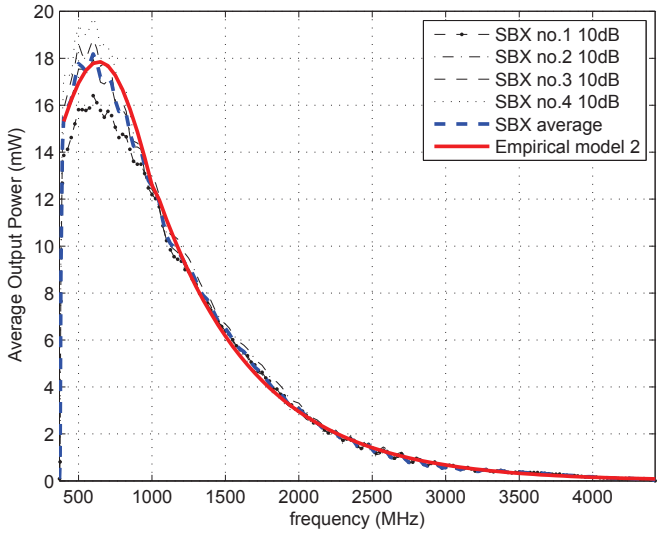


Figure 4.26: The average output (in mW) for the four measured SBX boards versus the empirical model Equation (4.11) (converted to mW).

between the average measured output power (over the four boards) and the empirical model given in Equation (4.11) is almost perfect throughout the whole frequency range. The too optimistic predictions of the empirical model for $\text{UHD}_G = 20$ dB and $f < 1$ GHz are due to the non-linear regime of the daughter board.

We also compare this model in Figure 4.27 with the actual measured output of the SBX board for three values of UHD_G gain, namely $\text{UHD}_G = 0$, 10 and 20 dB. The match is nearly perfect for $\text{UHD}_G = 0$, 10 dB but a little optimistic for $\text{UHD}_G = 20$ dB if $f < 1$ GHz. This result can be explained by the saturation of the power RF amplifier. Indeed, further measurements showed that for output powers above 20 dBm the THD reaches rather high values and the output power on the fundamental frequency is accompanied with a very strong emission on the second harmonic *i.e.* on a frequency $2f$.

Both empirical models introduced in this section remain valid even for bigger values of the UHD_G parameter, however, the total average output power has to be below 20 dB in order to avoid nonlinearities. For example we measured the output power for DAC = 1 and $\text{UHD}_G = 30$ dB at a frequency $f = 3.5$ GHz. We found $P_{out} = 15.7$ dBm. The model from Equation (4.11) yields a value of $P_{out} = 15.1$ dBm.

7 Summary

In this chapter, some unexpected behavior of USRP daughter boards has been proven via a real wireless communication of a bit stream. BPSK software modulator/de-

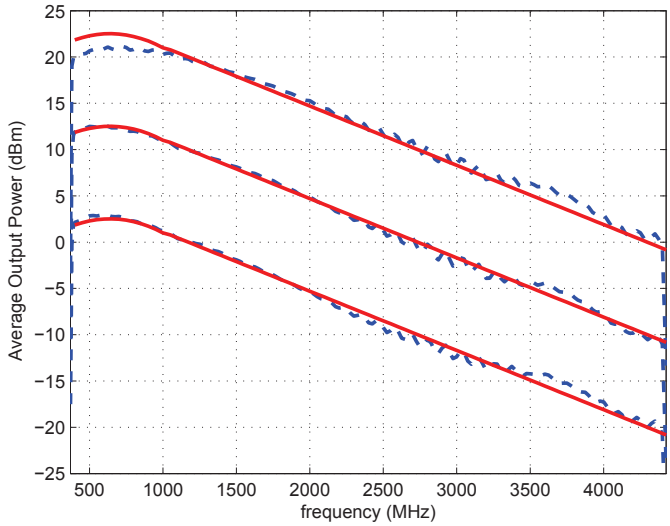


Figure 4.27: The average output power (in dB) versus the empirical model from Equation (4.11) for $\text{UHD}_G = 0$ (lower curve), $\text{UHD}_G = 10$ (moddle curve) and $\text{UHD}_G = 20$ (upper curve).

modulator has been implemented. Its evaluation has been performed analyzing SNR and BER parameters. Four SNR estimators have been explored throughout simulation and real transmissions. The obtained results over simulations confirm the usefulness of the "Simple" SNR estimator. Nevertheless, this estimator should also be used with care. It gives SNR values higher than the expected (real) ones in a lower range of SNR values. By contrast for real-time communications, the obtained curve of the Simple estimator was not linear. However, a relation between a software amplifier and an estimated SNR cannot be featured. These results lead us to follow an experimental approach and to arbitrate between manufacturer hypothesis.

Throughout experiments, two measurements processes have been performed with a spectrum analyzer and within the GNU Radio itself. We started by analyzing performances of daughter boards using a spectrum analyzer. The first boards were the RFX2400 and the RFX900. The overall output power linearity of the RFX2400 was better than that of the RFX900. For RFX900 daughter board, the quadratic relationship between the DAC value and the average output power breaks down if the DAC parameter has values above 0.5. The found frequency bandwidths are 24% and 18% smaller than the advertised bandwidths for RFX2400 and RFX900, respectively. Accurately, for RFX900, the measured -30 dB bandwidth is 72 MHz contrary to the 300 MHz advertised value. Furthermore, for RFX2400, the bandwidth is 168 MHz rather than 600 MHz. In contrast, the SBX daughter boards and USRP B210 bandwidth were confirmed. Nevertheless, the output power was found to decrease with increasing carrier frequency. The UHD_G was found to be a valuable parameter, its increase has to be done with care. We were able to show that beyond a given

threshold for the UHD_G, the measured THD quickly increases. The THD indicates that the output power occurred on the unwanted harmonic frequencies. Furthermore, an empirical model was introduced to accurately predict the average output power of an SBX daughter board. The simplicity of the model makes it potentially useful in SDR applications. The true values of the output power are known a priori and they can be used for specific application. For example, we can estimate the maximum transmission range of the USRP and GNU Radio SDR.

In the second measurements we used a flow graph as a spectrum analyzer. The results show a threshold at receiver UHD_G for SBX and RFX daughter boards. The maximum and useful UHD_G is equal to 30 dB and beyond it, the amplifier saturates.

Other complementary experiments can be carried out on RFX 900 daughter board.

These measurements allow us to be well informed about the behavior of the hardware driven by the GNU Radio flow graphs. Nevertheless, this information remains usable only in an output Tx side. It might be more effective to add another part of measurements dealing with the input behavior.

After performing these measurements and obtaining the characterization of some USRP daughter boards, we would exhibit SDR implementations for the IEEE 802.15.4 standard.

Chapter 5

SDR implementations for IEEE 802.15.4-based WSN

Talk is cheap. Show me the code.

Linus Torvalds

Contents

1 Introduction 114

2 Problem statement 114

3 Related works 116

4 IEEE 802.15.4 PHY layers 118

4.1 Common specifications for 868/915 MHz and 2450 MHz PHY layers 118

4.2 2450 MHz specifications 119

4.3 868/915 MHz specifications 120

5 Software Implementations 121

5.1 Software transmitter/receiver for 2450 MHz PHY 121

5.2 Software transmitter/receiver for 868/915 MHz PHY 124

6 SDR communications for 2450 MHz 131

6.1 Communications between two SDRs 131

6.2 Communications between sensor motes and SDRs 132

7 SDR communications for 868/915 MHz 133

8 Summary 136

1 Introduction

We have seen in the previous chapter that real-time wireless communications can be implemented on GNU Radio USRP SDR. IEEE 802.15.4 standard specifications of **L**ow-rate **W**ireless **P**ersonal **A**rea **N**etwork (LR-WPAN) can be implemented by an SDR. The standard defines Physical (PHY) and Link layers (LNK) of low-power wireless networks or WSN. The PHY layer (Physical layer) is available as off-the-shelf hardware transceiver. For example, we can find CC2420 and AT86RF231 transceivers from Texas Instrument and Atmel, respectively. The Transmitter (Tx) and Receiver (Rx) flow graphs can replace these hardware transceivers. To prove the possible usage of software transceivers, real-world communications can be performed, and a proof of concept validates the prototyping before its manufacturing.

In this chapter, we extract knowledge from a reverse engineering process of an existing SDR for 2450 MHz band of IEEE 802.15.4 standard. We bring more details than in the literature about the first SDR prototype [38]. Then, we propose a new implementation for 868/915 MHz ISM band for the same standard. This frequency band has been dedicated for the IEEE 802.15.4 standard but without an existing SDR prototype based on GNU Radio USRP SDR. We present our Tx and Rx flow graphs with some tests and performance evaluations.

2 Problem statement

PHY (MHz)	**Frequency Band** (MHz)	**Propagation parameters**		**Parameters of data transmission**		
		Chips rate (k chips/s)	**Modulation**	**Bits rate** (kb/s)	**Symbols rate** (k symbols/s)	**Symbols**
780	779-787	1000	O-QPSK	250	62.5	16-ary orthogonal
		1000	MPSK	250	62.5	16-ary orthogonal
868/915	868-868.6	300	DSSS + BPSK	20	20	Binary
		400	DSSS + O-QPSK[1]	100	25	16-ary orthogonal
		400	PSSS + BPSK et ASK	250	12.5	20 bits PSSS
	902-928	600	BPSK	40	40	Binary
		1000	DSSS + O-QPSK	250	62.5	16-ary orthogonal
		1600	PSSS + BPSK et ASK	250	50	5 bits PSSS
950	950-956	—	GFSK	100	100	Binary
		300	BPSK	100	100	Binary
2450 (DSSS)	2400-2483.5	2000	O-QPSK	250	62.5	16 ary-orhogonal

[1] **O**ffset-**Q**uadtrature **P**hase **S**hift **K**eying (O-QPSK)

UWB sub gigahertz	250-750					
2450 (CSS)	2400-2483.5	–	CSS + DQPSK	250	62.5	8-ary
				1M	166.667	16-ary bi-orthogonal
UWB low band	3244-4742	Depends on environment conditions				
UWB high band	5944-10234	Depends on environment conditions				

Table 5.1: Synthesized specifications of IEEE 802.15.4 [19], [20]

Wireless sensor network (WSN) is a set of sensor nodes, which communicate throughout radio frequency links. This simplified definition primarily involves three lowest layers of OSI (Open Systems Interconnection) model. These layers are the Physical, the MAC and the Network layer. Current research works, in most cases, deal with each layer issue separately. It is the case of PHY layer problems such as radio-frequency (or channel) interference and spectrum sensing. The solutions are often performed and evaluated via a simulation. However, SDR based testbeds present a potential interest for a real world proof of concept [101]. In addition, Cognitive Radio Sensor Network can be thought as an application of an SDR [102].

IEEE 802.15.4 is standard specifications of Physical (PHY) and Media Access Control (MAC) layers of LR-WPAN. It is emerged as the de-facto standard for WSN and supports higher applications like WirelessHart, Zigbee and 6 LowPan. The IEEE 802.15.4e of 2012 [20] is an enhanced version of that published in 2003 [19]. New specifications have been included throughout the proposed versions. In that of 2012, the data rate of IEEE 802.15.4e network can reach 1 Mb/s instead of 250 kb/s. In addition, high-frequency band of **U**ltra **W**ide **B**and (UWB) can be used above 3 GHz. Thus, each specification depends on the frequency bands and digital modulations. Table 5.1 shows all possible frequency bands and modulations for each PHY layer. Note that, all these specifications have not been implemented by sensor mote's manufacturer.

Standard specifications bring to manufacturers of nodes' transceivers an opportunity to ensure the scalability and the interoperability with other Off-the-shelf products. Nevertheless, the standard delays the evolutivity and the maintainability of these transceivers. However, to test a little modification, the manufacturer must go through manufacturing of a hardware prototype. In most cases, this process is slow and needs more resources. Moreover, the applications of wireless networks, particularly WSN, are diverse, and each application has its specific constraints. Thus, software transceivers are an opportunity to overcome these constraints, since a software source code can easily be modified and adapted to application's context.

Some WSNs applications need to be implemented in specific frequency bands, and also with completely different wireless technologies (see Table 5.1). In IEEE 802.15.4 standard, a hardware transceiver contains only one modulation such as O-QPSK (Offset Quadrature Phase-Shift Keying), GFSK (Gaussian Frequency-Shift Keying) or BPSK (Binary Phase-Shift Keying), and one spectrum that ranges application from narrowband to wideband systems. Furthermore, since WSNs are strongly application specific, it is interesting to measure performance parameters of wireless communications. For that the Signal to Noise Ratio (SNR) is not the only representative evaluator. A Packet Success Rate (PSR) or Packet Error Rate (PER) can also be used to evaluate a PHY layer.

The real nodes are limited to the information that the transceiver chip provides about their environment. Even though the nodes are able to investigate network layer metric such as PSR, they don't have the possibility to understand some radio effects. For example, it can be hard to determine if packet loss occurred due to an interference or due to a noise. Commonly, transceiver chips do not provide any information about packets that could not be decoded. Hence, the nodes cannot be used to investigate new PHY layer strategies. Moreover, some of the proposed extensions for IEEE 802.15.4 PHY layers are not yet available on actual devices.

A software transceiver or SDR allows a researcher and designer the possibility to replace real nodes with SDRs, signal processing done in software instead of being hidden inside a transceiver chip. As explained in previous Chapters 2 and 3, only RF front end and Intermediate Frequency parts are hardware. With such a re-programmable SDR, the software designer has full control overall baseband processing steps.

3 Related works

Hardware implementations of the IEEE 802.15.4 specifications have been performed by manufacturers or the research community on ASICs [41]. They are limited in flexibility and scalability as explained in the previous section 2. However, SDRs implementations in high level programming languages answer to these primary constraints.

In [38], Tomas Schmid introduced decoding blocks of an O-QPSK PHY layer of IEEE 802.15.4 within an old GNU Radio version and USRP 1. The worldwide ISM band of 2450 MHZ was the main motivation behind this work. The decoding blocks were connected to construct Tx and Rx flow graphs. In this work, real communications have been carried out with real motes (sensor nodes), where USRP 1 is the RF Front End of this SDR prototype. The author makes sure that prototype is well-running by receiving packets from CC2420 ChipCon transceiver [103]. Obviously, the digital modulation is an Offset Quadrature Phase Shift Keying (O-QPSK) technique. The

particularity is the non coherence [2] of the receiver, since it decodes a signal without recovering a carrier frequency. In fact, SDR designer cannot access directly to the carrier on the USRP daughter-board itself [38]. Indeed, we have experienced that prototype and some difficulties in packet reception have been noticed. Even though the receiver synchronization has not been established, the interoperability with MicaZ and TelosB motes has been ensured. The drawback of this work is the lack of information about setup parameters such as sampling rate, gain, signal amplitude and USRP 1 setup.

An extension of T. Schmid works [38] has been proposed in [39]. The particularity of this proposal is the compatibility of flow graphs with a USRP 2. It allows also the designer to evaluate PSR/PER with a Wireshark packet analyzer [104]. The authors implement a multichannel (multi-central frequency) reception. Five consecutive channels from 16 existing ones have been used in the 2450 MHz band. The authors explain that the number of channels could be higher than five if USRP resources are enough. Another extension has been published in [40]. It reuses flow graphs introduced by T. Schmid, but in an OSI stack form. From the physical up to the network layer and with an attached application layer. The flow graphs interacts with Contiki operating system, which is a state-of the-art operating system for research in WSNs [105]. This work contributes to simplify access to IEEE 802.15.4 blocks through GNU Radio companion GUI. The user can integrate a new application such as Rime (A Lightweight Layered Communication Stack for Sensor Networks) from Contiki. Furthermore, the stack includes PCAP (PaCket CAPture) to easily debug and monitor for WSN communications. All the previous works [38], [39], [40] were interested only in the 2450 MHz band.

The PHY layer of 868 MHz band is available in Europe, whereas the 915 MHz band is dedicated to North America. Low frequency bands are needed when low data rate and long distance transmissions are suitable. Indeed, both bands present a longer range than that of the 2450 MHz band for a given link budget. The distance of radio links with 868/915 can be around 100 km whereas this distance can be approximately 70 m in 2450 MHz [106]. Regarding these advantages, some solutions have been proposed by the research community. For example, the authors in [107] construct an FPGA based SDR for this band. They synthesized in VHDL a Simulink model of the developed system. The design has been fully functional at 128 MHz and not in 868 MHz or in 915 MHz. Furthermore, a number of hardware implementations of the IEEE 802.15.4 have been reported on ASICs or FPGAs [107], [41], but they do not allow us to control the flexibility and the functionalities of all software stack layers. Our work is different because it uses GPP base SDR with high level programming language within GNU Radio platform. The study and reverse engineering of the O-QPSK PHY layer have allowed us to introduce new blocks and construct D-BPSK PHY layer.

[2] Coherent transmission is obtained when sender and receiver either share the same clock, or the receiver decodes a synchronization signal from the carrier.

4 IEEE 802.15.4 PHY layers

As we have seen in the previous section, the IEEE 802.15.4 standard was defined for two layers: PHY and MAC layer. Our research and prototyping works focus only on PHY layer, for the two frequency bands 868/915 MHz and 2450 MHz. In this section, we summarize the primary specifications by separating the common definitions from the appropriate ones [108].

4.1 Common specifications for 868/915 MHz and 2450 MHz PHY layers

The two frequency bands are different by their transmission parameters such as frequency band, modulation and pseudo random sequence of a Direct Sequence Spread Spectrum (DSSS), etc. However, after analyzing specifications of each frequency band in [108], we find that common parameters of both 868/915 MHz and 2450 MHz PHY layers can be aggregated on the packet structure and the spread spectrum. A number of fields are packed to construct an IEEE 802.15.4 packet. Each field represents useful information for synchronization, error control and network signalization (packet type DATA or acknowledgment packet). Since the PHY layer is the last step before the radio front end, we need to present PHY Protocol Data Unit (PPDU) format, which contains the following fields (see Figure 5.1):

- **Preamble** is a part of Synchronization HeadeR (SHR) of a packet. Its length is 8 symbols (i.e., 4 bytes), and the bits in the Preamble field shall be binary zeros.
- **Start of Frame Delimiter (SFD)** is a field indicating the end of the SHR and the start of the packet data. The SFD is formatted as illustrated in hexadecimal by $0xA7$
- **PHY header (PHR)** is Frame Length field which specifies the total number of bytes contained in the Physical Service Data Unit (PSDU), *i.e.* PHY payload. It is a value between 0 and 127 Bytes.
- **PSDU field** carries the data of the PPDU with a maximum size of 127 Bytes. It contains a number of fields :
  - **Frame Control Field (FCF)** that contains information defining the frame type, addressing fields, and other control flags.
  - **Sequence Number field** specifies the sequence identifier for the frame. It allows to specify if the frame is a data, acknowledgment, or MAC command frame.
  - **Address information** can contain source/destination address and information for security protocol.

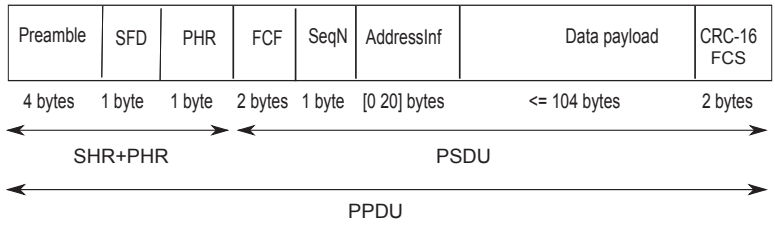


Figure 5.1: IEEE 802.15.4 packet structure and size

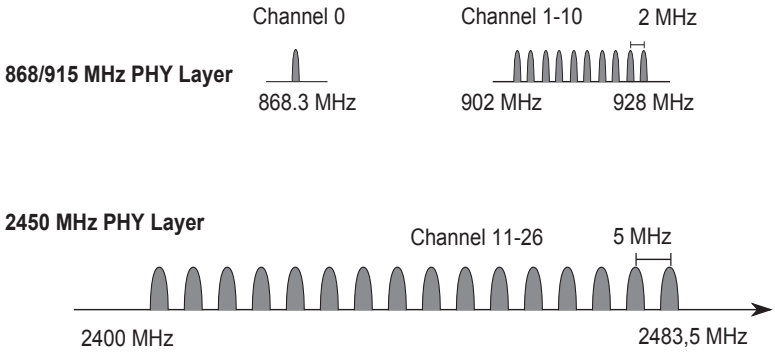


Figure 5.2: Channel allocation in 868/915 MHz and 2450 MHz

- **The payload** of a data frame shall contain the sequence of bytes that the next higher layer has requested the MAC PHY layer for transmission.
- **Frame Check Sequence (FCS)** field contains a 16-bit **C**yclic **R**edundancy **C**heck (CRC). The FCS is computed from the MAC header and payload parts, *i.e.* from FCF to data payload passing through Sequence Number and Address information of the frame.

Note that the size of a PPDU is up to 133 bytes with 6 bytes dedicated to packet header.

4.2 2450 MHz specifications

This frequency band is occupied with 16 channels. The center frequency F_c of these channels is defined as follows:

$$F_c = 2405 + 5(k - 11), k = 11, 12, \ldots, 26 \tag{5.1}$$

Where k is the channel number and F_c is central frequency in megahertz. Each channel occupies a bandwidth of 2 MHz and the time interval between two consecutive central frequencies is 5 MHz. Figure 5.2 shows channel allocation in this band.

This PHY layer is called O-QPSK PHY, since the signal is modulated/demodulated with O-QPSK. The latter employs 16-ary quasi-orthogonal modulation technique. During each data symbol period, four information bits are used to select 1 of 16 nearly orthogonal pseudo-random noise (PN) sequences to be transmitted. The

PN sequences for successive data symbols are concatenated, and the aggregate chip sequence is modulated onto the carrier using offset quadrature phase-shift keying (O-QPSK). The encoding on chips sequence gives a Direct Sequence Spread Spectrum (DSSS). Note that the modulator with half-sine pulse shape is equivalent to Minimum Shift-Keying (MSK).

The data rate of the O-QPSK PHY is 250 kbps. Each data symbol are mapped into a 32-chip PN sequence as specified in Table 5.2. The PN sequences are related to each other through cyclic shifts and/or conjugation (*i.e.*, inversion of odd-indexed chip values). During each symbol period, the least significant chip, c_0, is transmitted first.

Data symbol	**Chip values (c_0, c_1, ... ,c_{30}, c_{31})**	**Hexadecimal**
0	1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0	0xD9C3522E
1	1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0	0xED9C3522
2	0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0	0x2ED9C352
3	0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1	0x22ED9C35
4	0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1	0x522ED9C3
5	0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0	0x3522ED9C
6	1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1	0xC3522ED9
7	1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1	0x9C3522ED
8	1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1	0x8C96077B
9	1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1	0xB8C96077
10	0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1	0x7B8C9607
11	0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0	0x77B8C960
12	0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0	0x77B8C96
13	0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1	0x6077B8C9
14	1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0	0x96077B8C
15	1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0	0xC96077B8

Table 5.2: Symbol-to-chip mapping for the 2.4 GHz band

4.3 868/915 MHz specifications

The two bands of 868 MHz and 915 MHz are grouped and defined as one band, since they have a similar digital modulation.

The frequency band of 868 MHz contains only one channel in the first version [19] and three channels in E version [20], whereas the 915 MHz band handles 10 channels in the first version and 30 channels in E version [20]. We have chosen the first version in this thesis. The channel allocation is as follows:

$$F_c = 868.3, k = 0 \tag{5.2}$$

$$F_c = 906 + 2(k-1), k = 1, 2, ..., 10 \tag{5.3}$$

where k is the channel number. The distance between two channels is 2 MHz and their bandwidth are narrow-band signals compared to the spectrum of the ADC input signal (see Figure 5.2).

Similarly to the previous frequency band of 2450 MHz, the name of these specifications is BPSK PHY regarding the digital modulation. In fact, a direct sequence spread spectrum (DSSS) with BPSK are used for chip modulation. Each input is mapped into 15-chip PN sequence as specified in Table 5.3.

Input bits	**Chip values (c_0, c_1, ... ,c_{13}, c_{14})**	**Hexadecimal**
0	1 1 1 1 0 1 0 1 1 0 0 1 0 0 0	0x7AC8
1	0 0 0 0 1 0 1 0 0 1 1 0 1 1 1	0x537

Table 5.3: Symbol-to-chip mapping for the 868/915 MHz band

The BPSK modulator includes a differential encoding of data symbol. It performs the modulo-2 addition (exclusive or) of a raw data bit with its previous encoded bit. The data rate is lower than in OQPSK PHY. It is 20 kb/s when operating in the 868 MHz band and 40 kb/s when operating in the 915 MHz band.

5 Software Implementations

After considering the main properties of each PHY layer, it is time to translate them to software. Each PHY layer will be described separately. We start by 2450 MHz PHY layer, which is widely used, and we will finish by the 868/915 MHz PHY layer.

5.1 Software transmitter/receiver for 2450 MHz PHY

In this frequency band, our contribution in addition to that of [40] is to have performed reverse engineering of the implementation of Thomas Schmid [38]. Throughout flow graphs, we have studied each block and detailed their parameters. This study allowed us to understand the aim of each block and how to reuse them in the second implementations for the frequency band 868/915 MHz. In addition, programs have been adapted to the latest versions of GNU Radio and USRP hardware.

5.1.1 Tx flow graph

Figure 5.3 shows a schematic graph of the software Transmitter (Tx), where the blocks are connected via a Python code directly without GUI of gnuradio companion.

Transmitter Tx flow graph starts by a packet generator, which creates repetitively a flow of packets. Within this block, a set of parameters can be changed such as size of packets, data payload and inter packet time. Obviously, the structure of

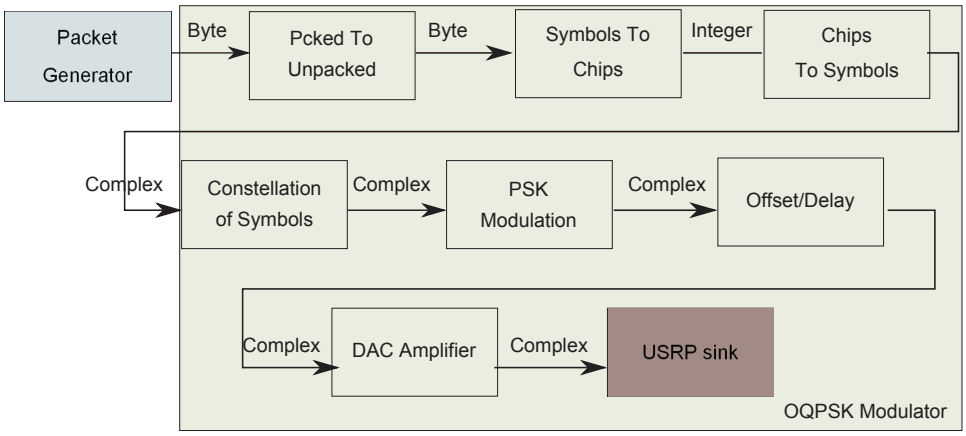


Figure 5.3: Transmitter (Tx) flow graph for 2450 PHY layer

packets has already been described in [19] and shown in Figure 5.1. The second block unpacks the stream of packets in *Byte* to symbols, which are a set of 4 bits. Thereafter, each symbol is transformed into a sequence of chips carried by an *Integer* sample and following Table 5.2. Every two chips are also grouped in symbols but with *complex* output when digital stream enters the Chips to Symbol block, which represents two chips by a complex sample. The latter are mapped on the complex axis (IQ axis) through symbol constellations and Phase Shift Keying (PSK) blocks. In fact, the 0, $\frac{\pi}{2}$, π and $\frac{3\pi}{2}$ phase variations represent chips sequences (0, 0), (0, 1), (1, 0) and (1, 1). Up to this step, the modulator is a QPSK. Thus, PSK modulator is connected to Offset/Delay block, which causes a delay of 0.5 ms, however, the obtained modulator is an OQPSK. Before the last block of USRP sink, the obtained baseband signal is amplified through DAC amplifier (the function of DAC parameter has been explained in Section 3.2.1 of the previous chapter).

5.1.2 Rx flow graph

In contrast to the transmitter flow graph, the Rx flow graph starts with an USRP block to receive baseband signal. Squelch block follows the USRP source block (see Figure 5.4). It stops a signal reception when a signal strength is lower than certain threshold of a received noise. An FM demodulator demodulates a band-limited, complex down-converted signal into an output float stream in the range of $[-1.0, +1.0]$. This stream enters an Infinite Impulse Response (IIR) filter with float input, float output and double taps. Then, the subtraction block subtracts from FM demodulated signal an IIR filtered signal. The result feds through a clock recovery block, which applies a Mueller and Müller (M&M) discrete-time error-tracking synchronizer [109]. After that, the synchronizer output is received by the packet sink. Each packet is constructed by slicing symbols to bits and by following symbol-to-chip mapping sequences as well as packet structure (see previous Sections 4.1 and 4.2). Finally, once the complete packet is found, it is added to the message

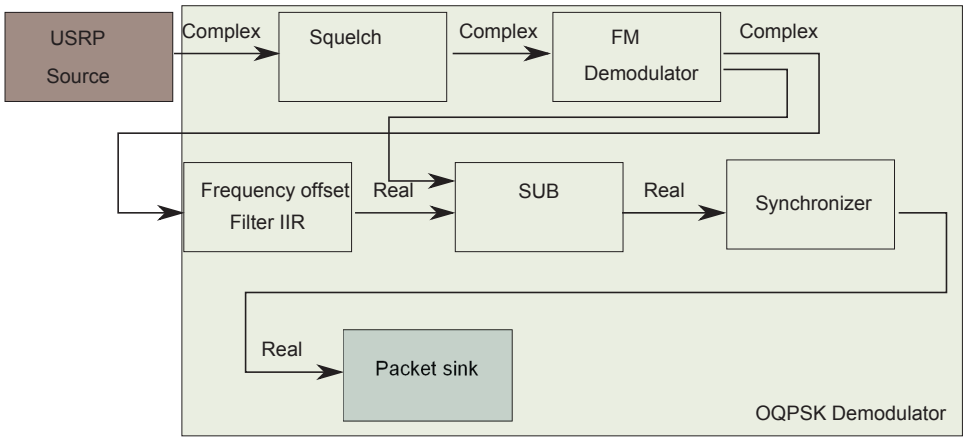


Figure 5.4: Receiver (Rx) flow graph for 2450 PHY layer

queue. The packet sink is implemented in C++ following an algorithm to decode the received packets.

The processing time taken by each block can be estimated to detect the cause of a delay in the receiver or transmitter flow graph. In Section 4.1 of Chapter 3 we have seen how to calculate this time using ControlPort block. In GNU Radio 2.6, ControlPort cannot be used, thus we found some difficulties to synchronize packet decoding. Our solution for this synchronization problem was to increase the size of generated packets. In Section 6, we will show how the PSR was improved by increasing the packet size.

5.1.3 Packet decoder

The packet sink block performs packet decoding in three steps: SYNC SEARCH, HEADER SEARCH and HAVE HEADER. These steps are cases of a switch control mechanism, and they are separated into three algorithms 1, 2 and 3. Each step performs a set of processing and tests on the input stream. The items flow from the synchronizer are float samples which are the input of the packet sink. Throughout the three steps, a while loop controls a stream processing (see Algorithm 1), and If/Else condition statements are placed to verify the byte pattern. Some test conditions should be verified to go from one step to another. If they are not verified the packet decoding is relaunched several times when needed.

Algorithm 1 shows the set of actions to detect the preamble and the SFD field. The first operation of SEARCH SYNC is to transform the input items (or samples) to chips. Each item is equivalent to a chip which takes 0 or 1 value. In addition, each bit is represented by 32 chips following pseudo noise sequence given in Table 5.2. The lines from **9** to **13** concatenates a sequence of 32 chips. Then, this sequence of chips is compared to that corresponding to the entrance number 0 in PN sequence's table. The decoder restarts the search if the sequence in the table entry doesn't match the constructed one, else it continues with next iterations until it finds all four **0x00**

preamble bytes. However, it is possible to carry out the preamble search allowing errors in preamble decoding. Even if the decoder finds less than four preamble bytes, the decoding process and the block will also detect the frame. This trick considerably improves the packet reception rate. After the preamble is found, the decoder checks if the next byte value is equal to **0x7A**, which correspond to the packet SFD field.

The HEADER SEARCH is the second step of the decoder. Its objective is to find the header byte or PHR. It starts by constructing a sequence of 32 chips representing four bits. Similarly to the SYNC SEARCH, in this step the bytes are constructed using two times four bits. Each pair of four bits in one byte is used as a key of a hash table defined in Table 5.2 to get corresponding chip value. The packet decoding can be restarted if and only if one matching fails. The decoded byte gives the packet length, which shall be less than or equal to the PSDU maximum length (128 bytes). Thereafter, the decoder goes to the next case to decode PSDU field.

After executing the two previous steps, the decoder launches the last steps of HAVE HEADER case. The objective in this case is to append bytes in one data structure. Bytes are collected until reaching a number of bytes equal to the packet length found in PHR field. Similarly to the HAVE SYNC algorithm, the byte matching tests are performed for each iteration. Thus, the packet decoding can be restarted if and only if one test fails. Once a complete PSDU is found, it is added to a packet queue. Meanwhile, an external python thread is observing this queue. The thread calls a function written in python to process the PSDU as soon as possible.

Note that the PSR of the packet decoder depends on the source code and on the algorithm efficiency. Each operation added to a source code needs an extra processing time, which delays the preamble decoding. Thereafter, a phase shift can be generated at the demodulator. This delay depends on the complexity of the block algorithm. The decoder algorithm has a complexity in $O(n^2)$ since two nested-loops are required by the decoder.

It is recommended to avoid as much as possible print instructions. The debugging of source code should be done before real wireless communications. For example, transmitter and receiver flow graphs can be connected in a loop back simulation (as seen in the previous Chapter). Hence, programming this part with GUI is not efficient.

5.2 Software transmitter/receiver for 868/915 MHz PHY

Our work published in [1] reports an implementation of the PHY specifications for the frequency band 868/915 MHz. This frequency band 868/915 MHz is recommended when low data rate is needed in particular applications. Furthermore, it presents a longer range than the 2450 MHz band for a given link budget.

The frequency band of 902–928 MHz is one of the Industrial, Scientific, and Medical bands in the US, commonly abbreviated as the 915 MHz ISM band. Furthermore, the frequency 868,6 MHz is a license-free band for Short-Range Device (SRD860).

Algorithm 1: Work function of packet sink for 2450 MHz band - SYNC SEARCH PART

```
Data: Items[], nbrItems, tab_PN_seq, data_packet_bytes
Result: PPDU of IEEE 802.15.4 for 2450 MHz
initialization;
cItems ← 0; data_packet_bytes ← 0; MAX_PKT_LEN ← 127 ;
tab_PN_seq[];
% Table of PN sequences
while cItems < nbrItems:
    switch dStatedo
        case SYNC_SEARCH:
            while cItems < nbrItems:
                cItems ++ ;
                if Items[cItems] ≥ 0:
                    chips_seq ← (chips_seq ≪ 1) | 1 ;
                else:
                    chips_seq ← chips_seq | 1 ;
                if find_preamble(chips_seq):
                    data_packet_bytes ← (data_packet_bytes ≪ 4) | 0x00 ;
                    find_preamble ← true ;
                if find_first_SFD_byte(chips_seq) and find_preamble:
                    % Checks if the concatenated chips (4 bits) exist in the table of PN
                    sequences and equal to 0x7.
                    preamble_found ← true ;
                    data_packet_bytes ← 0x7 ≪ 4;
                else:
                    break ;  % Wrong first byte of SFD and restart search
                if find_second_SFD_byte(chips_seq):
                    % Checks the second 32 chips sequence if it constructs 4 bits and
                    matches with the entry 0xA in PN table.
                    SFD_found ← true ;
                    data_packet_bytes ← data_packet_bytes | 0xA ;
                    goto : HAVE_SYNC ;
                else:
                    break  % Restart search ;
                break; ;
```

Algorithm 2: Work function of packet sink for 2450 MHz band - HEADER SEARCH PART

```
1  case HAVE_SYNC:
2      while cItems < nbrItems:
3          cItems ++ ;
4          if Items[cItems] ≥ 0:
5              chips_seq ← (chips_seq ≪ 1) | 1 ;
6          else:
7              chips_seq ← chips_seq | 1 ;
8          byte ← decode_chips_exist(chips_seq, tab_PN_seq[] ) ;
9          if byte_exist(byte):
10             data_packet_bytes ← data_packet_bytes | (byte ≪ 8) ;
11             pkt_length = decimal(byte) ;
12         else:
13             break; % Restart search
14         if pkt_length ≤ MAX_PKT_LEN:
15             PHR_found ← true ;
16             goto HAVE_HEADER ;
17         else:
18             break ;  % Restart search
```

Algorithm 3: Work function of packet sink for 2450 MHz band - HAVE HEADER PART

```
1  case HAVE_HEADER:
2      while cItems < nbrItems:
3          cItems ++ ;
4          if Items[cItems] ≥ 0:
5              chips_seq ← (chips_seq ≪ 1) | 1 ;
6          else:
7              chips_seq ← chips_seq | 1 ;
8          byte ← decode_chips_exist(chips_seq, tab_PN_seq[] ) ;
9          if byte_exist(byte)):
10             data_packet_bytes ← data_packet_bytes | (byte ≪ 8);
11             nbr_byte_data ++ ;
12             if nbr_byte_data = pkt_length:
13                 pkt_queue(data_packet_bytes) ;
14                 % insert packets to queue break; % Restart search ;
15         else:
16             break;  % Restart search
```

The latter is defined for Europe from 863 MHz to 870 MHz with Effective Radiated Power (ERP). The IEEE 802.15.4 standard remains useful even if the SRDs also share the same frequency band. It can be used, for example, as a backup solution of SRDs.

As seen in Section 4.3, the differential BPSK (or D-BPSK) modulation is the same for the two frequency bands 868 MHz and 915 MHz. The frequency agility of an SDR motivates an implementation for two frequency band with only adapted parameters, *i.e.* data rate in 868.6 MHz (respectively 915 MHz) is 20 kbps (respectively 40 kbps).

The next sections show details about our implementation in order to facilitate its re-usability. We keep the same names used in a python flow graph developed over GNU Radio version 3.6. In addition, technical information about USRP 1 setup is given.

5.2.1 Tx flow graph

The Tx flow graph for this frequency band was also implemented based on the IEEE 802.15.4 specifications. Transmitter was described by a flow graph of eight connected blocks (see Figure 5.5). This flow graph transforms messages from a packet format to a baseband signal through D-BPSK modulator. It starts by message source, which generates IEEE 802.15.4 packets. Each packet is divided into chunks of symbols by the `gr.packed_to_unpacked` block, where one symbol represents 1 bit. Since the C++ programming language does not allow us to have a data type of 1 bit, the bits in the bytes of an input stream are grouped into chunks of 1 byte. The MSB (Most Significant Bit) of 8 output bits represents the one bit from the input of `gr.map_bb`. After that, the differential encoder `gr.diff_encoder_bb` encodes a current symbol adding modulo-2 addition of the previous one. Then, the symbols are mapped by `gr.symbols_to_chips` into 15 Pseudo Number Sequence chip as specified in Table 5.3. The output of mapping is of short-type (16 bits carrying the 15 chips). With the same technique, the stream is unpacked to chunks of 16 bits representing chip stream. Through `gr.chunks_to_symbols_sc` block, each chip is represented by complex constellation points separated by an angle equal to π radians. The stream is then enters through a Root Raised Cosine `gr.interp_fir_filter_ccf` filter which up-samples the signal, after which it is sent from the host computer via USB to the transmitting USRP.

Similarly to the first block of 2450 MHz band's flow graph, the generated packets follow the format described in Section 4.1 through Figure 5.1). The packet size should be a multiple of 128 samples and a packet has a maximum size of 133 bytes and an USB 2 hardware interface . Therefore, the transmitter performs a white padding to adapt the packet size to hardware requirements. It appends zero bytes (or the x/00 character) at the end of each packet. The number of zero bytes depends on *Byte_Modulus* parameter, which also depends on the sampling rate and the number of bits per symbol. The *Byte_Modulus* is given by:

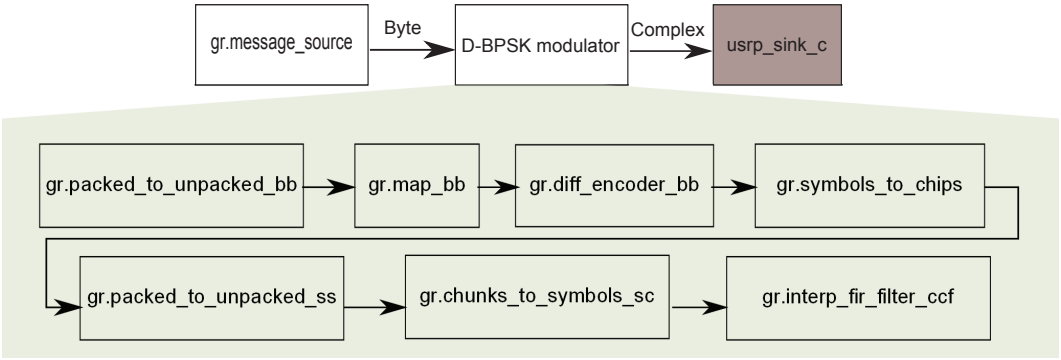


Figure 5.5: Transmitter (Tx) flow graph for 868/915 PHY layer

$$Byte\_Modulus = \text{LCM}\left(\frac{128\ \text{MSPS}}{8\ \text{MSPS}}, sps\right) \cdot \left(\frac{bps}{sps}\right) \tag{5.4}$$

where

- 128 MSPS – is the DAC sampling rate of the USRP1
- 8 MSPS – is the Sampling rate of the USB tunnel
- *sps* – is the Number of samples per symbol
- *bps* – is the Number of bits per symbol
- LCM – is the Lowest Common Multiple of 16 MSPS and *sps*

The white padding can be avoided if the packet size is set equal to 130 bytes. This size is obtained by reducing the address information field $AddressInf$. In addition, a 16-bit CRC (Cyclic Redundancy Check) is attached to the packet payload, allowing the receiver to calculate the PER (Packet Success Rate) and the PRR (Packet Received Rate).

5.2.2 Rx flow graph

The receiver objective is to demodulate a received baseband signal using a D-BPSK demodulation. Figure 5.6 shows the flow graph of receiver blocks. The flow graph begins with a USRP source connected to a squelch filter `gr.pwd_squelch`, which admits only signals with a certain dB strength. The squelch filter outputs 0 when the incoming signal is too weak. The stream results of the squelch is passed to the Automatic Gain Control `gr.agc_cc` (AGC) of the D-BPSK demodulator. The AGC regulates the gain in a way that does not have a large or small amplitude and to avoid distortions. The AGC output enters into two filters of `gr.interp_fir_filter_ccf`, which are Finite Impulse Response (FIR) and Root Raised Cosine (RRC). These two filters allow the receiver to process the change of the transmitted pulse and

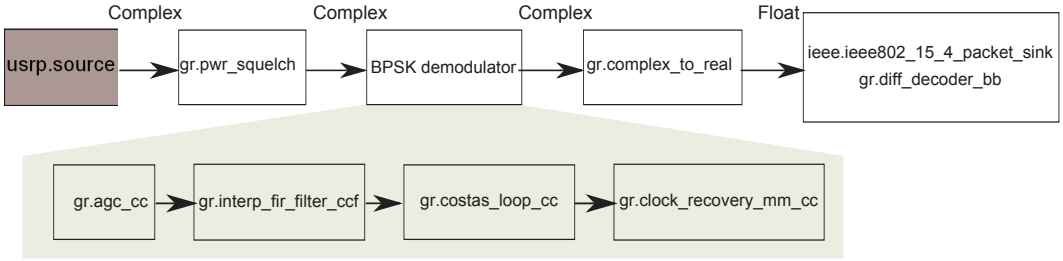


Figure 5.6: Receiver (Rx) flow graph for 868/915 PHY layer

to minimize symbol interferences. The RRC filter makes the correlation between the received signal and the expected one. It calculates an FIR filter coefficient or a tap weight. The demodulator synchronizer is built based on two blocks, a Costas Loop `gr.costas_loop_cc` (Phase Locked Loop) and the Mueller and Müller `gr.clock_recovery_mm_cc`. The Costas Loop recovers the carrier and improves the Bit Error Rate of BPSK demodulator [110]. Furthermore, the Mueller-Müller Timing recovery block tracks the symbol timing phase of the input signal [109]. After the demodulator, the stream is converted from complex to float in order to send it through `ieee.ieee802_15_4_packet_sink` block, which transforms the stream to packets and decodes them. The decoding algorithm will be detailed in the next section.

5.2.3 Packet decoder

The packet decoder for this band differs from that for 2450 MHz band in three things: the number of bits per symbol, the table of chips sequence and the differential decoding. The algorithm reuses the steps of the packet decoding given in Section 5.1.3, but by adjusting the quoted three things.

SEARCH SYNC algorithm (see Algorithm 1) has been adapted to the number of chips per bit. Each bit represents 15 chips following pseudo noise sequence given in Table 5.3. The preamble of the packet is found if the concatenated sequence matches with that of the first entry (or 0 entry) in Table 5.3. If a preamble is found, then the SFD field, *i.e.* **0x7A**, is compared to the next byte. Thereafter, this field is padded at the end of the decoded packet. Before branching the decoder to the HEADER SEARCH part, the differential decoding is applied to the preamble and the SFD fields (see Algorithm 4).

The HEADER SEARCH and HAVE HEADER algorithms are slightly different than those used by 2450 MHz receiver. They perform two additional operations: differential decoding and application of specific sequences of 15 chips. Thus, we limit our description to highlight these two different operations instead of showing all the algorithms.

Algorithm 4: Work function of packet sink for 868/915 MHz band- SYNC SEARCH PART

```
Data: Items[], nbrItems, tab_PN_seq, data_packet_bytes
Result: PPDU of IEEE 802.15.4 for 868/915 MHz
initialization;
cItems ← 0; data_packet_bytes ← 0; MAX_PKT_LEN ← 127 ;
tab_PN_seq[];
% Table of PN sequences
while cItems < nbrItems:
    switch dStatedo
        case SYNC_SEARCH:
            while cItems < nbrItems:
                cItems ++ ;
                if Items[cItems] ≥ 0:
                    chips_seq ← (chips_seq ≪ 1) | 1 ;
                else:
                    chips_seq ← chips_seq | 1 ;
                if find_preamble(chips_seq):
                    data_packet_bytes ← (data_packet_bytes ≪ 1) | 0x0 ;
                    preamble_found ← true ;
                if find_SFD_byte(chips_seq) and find_preamble:
                    % Checks if the concatenated chips sequence exists in the table of PN sequences
                    SFD_found ← true ;
                    data_packet_bytes ← (data_packet_bytes ≪ 8) | 0x7A ;
                    differential_decoding(data_packet_bytes) ;
                    goto : HAVE_SYNC ;
                else:
                    break; % Wrong first byte of SFD and restart search
                break ;
```

6 SDR communications for 2450 MHz

We carried out real wireless one-to-one communications using implemented SDRs for the 2450 MHz frequency band. These communications have been experimented through two setups with:

- Two SDRs based on Tx and Rx flow graphs
- Sensor motes [3] and SDRs based flow graphs

6.1 Communications between two SDRs

We started our experiment tests by wireless communications between transmitter and receiver SDRs. The objective was to measure the PSR versus the sizes of packets. The idea to perform these measurements was coming from some tests carried out when we programmed the packet decoder.

The transmitter and receiver were characterized by flow graphs. They were executed by a host computer which is connected to a USRP (USRP 1) with an RFX 2400 daughter board. The transmitter generates and sends bursts of 100 packets for each packet size. The packet header follows the pattern presented in Figure 5.1, whereas the packet payload is the result of padding "00" after the header. The packet size was increased by 10 bytes from 20 bytes to 100 bytes. The size modification was performed after a transmission of a burst of 100 packets. An inter packet time was fixed equal to 1 second for each burst. Other parameters of experiment tests are shown in Table 5.4.

	Parameter values
Distance between two USRP 1	**2 meters**
Carrier frequency	**2405 MHz**
Data rate	**1 Mbps**
Number of data packets	**100 packets**
Inter packet time	**1 second**
Sampling rate (Tx and Rx)	**4 MSps**
Number of samples per symbol (Tx and Rx)	**2 sps**
Gain radio (only at Rx)	**60 dB**

Table 5.4: Parameters of packet transmissions between two SDRs

Figure 5.7 shows the rate of successfully received packets (PSR) and the rate of all received packets. The latter counts decoded packets including those with a wrong CRC field. The PSR is higher when the packets size is increased to 100 bytes (see Figure 5.7). The PSR depends on the packet size based on our empirical observations

[3] A sensor node, also known as a mote, is a node in a sensor network

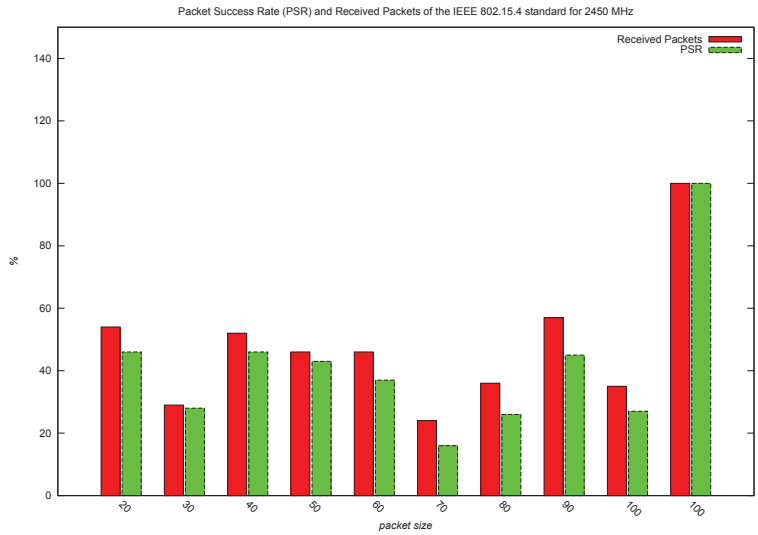


Figure 5.7: The PSR and received packet rates versus pocket size using SDR implementation for 2450 MHz

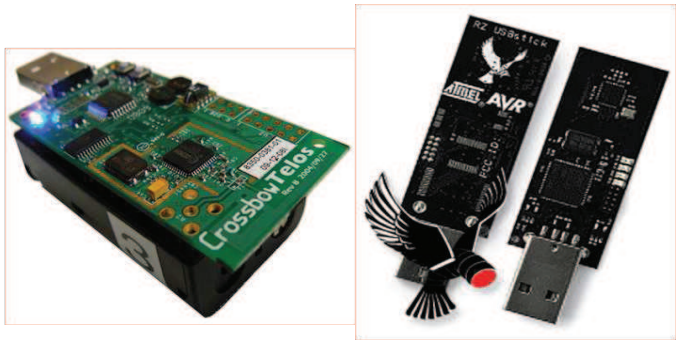
and our results. This relation cannot be described by an effective model, since the experiments were performed in indoor environment. These results can be interpreted by the nature of GNU Radio programs on a GPP based architecture.

6.2 Communications between sensor motes and SDRs

We have carried out real communications between Tx/Rx flow graphs and hardware transceivers (sensor motes) in an indoor environment. The Tx/Rx flow graphs are adapted to version 2.6 of GNU Radio. In the first step, we have replicated experiments proposed in [38]. Packets exchanges have been realized between software transmitter/receiver and hardware transceiver. In our case, we used CC2420 from Texas Instrument and AT86RF230 [111] from Atmel. For example, Telos B [112] sensor nodes handle CC2420 transceiver with an msp430 micro-controller (see Figure 5.8). Contiki micro OS [113] can be loaded on TelosB motes. Numerous applications can be performed by these motes. Unicast and Broadcast wireless communications are two available applications on these motes.

Packet reception via SDR Receiver (Rx) has been tested with a Telos B node as a transmitter. In our case, we selected a simple Broadcast application when each mote periodically broadcasts the same packet. These packets have been received by the SDR receiver (Rx).The packet success rate is difficult to calculate since the application was a broadcast.

In another experiment, the SDR Transmitter (Tx) has been launched, and the hardware receiver tried to receive the disseminated packets. Hardware interface has been analyzed by Wireshark software [104], which allowed us to see a packet traffic coming from the USB interface.



(a) (b)

Figure 5.8: TelosB and Raven sticks

7 SDR communications for 868/915 MHz

The packet communication for this band was performed with USRP 1. The experimental setup and their results have been published in [1]. In this section, we give detailed information on tests we have made and some obtained results.

Two USRP1 platforms have been coupled with RFX 900 daughterboards, covering a frequency range from 750 MHz to 1050 MHz. The GNU Radio software has been executed on a host computer having a one Core 2 Duo CPU running at 2.4 GHz and 2 GB of RAM. The distance between the two USRP 1 was 2 meters.

A USRP sink (see Figure 5.5) is characterized with a transmitter Interpolation I and receiver Decimation D. They are calculated according to a symbol rate r, DAC_s and ADC_s sampling rate, and a number of samples per symbol sps as follows :

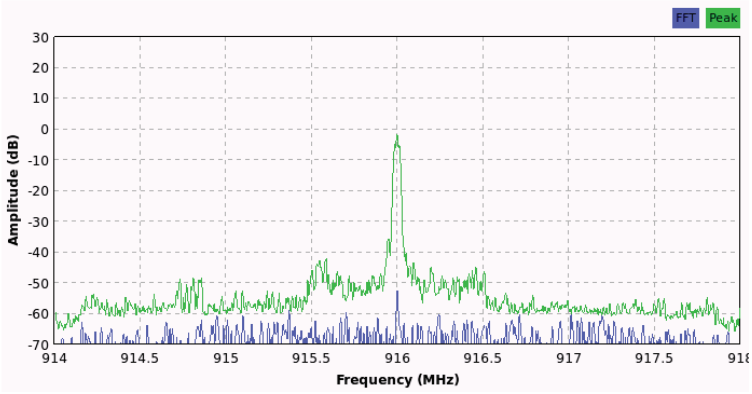
$$I = \frac{DAC\_s}{r \cdot sps}, \qquad D = \frac{ADC\_s}{r \cdot sps} \tag{5.5}$$

where :

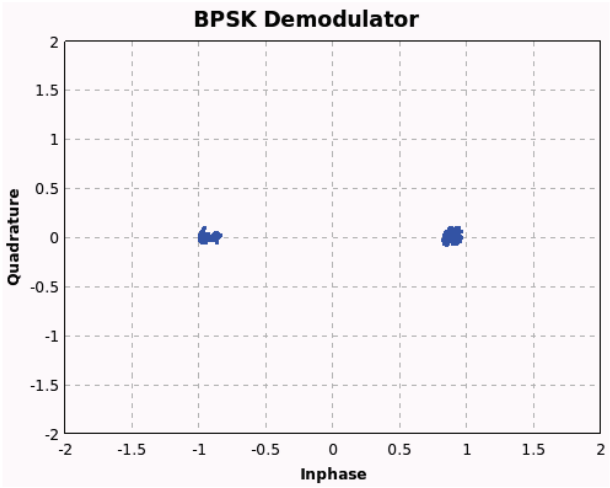
- $DAC\_s = 128$ MSPS
- $I \in [16,\ 20,\ 24, \ldots 508,\ 512]$
- $ADC\_s = 64$ MSPS
- $D \in [8,\ 10,\ 12, \ldots 254,\ 256]$

For 20 kbps, the transmitter and receiver parameters are respectively $I = 400$ and $D = 200$ with $sps = 16$. Otherwise, when the data bit rate is equal to 40 kbps, I and D take the same values but with $sps = 8$. The amplifier of signal (or DAC) amplitude is defined by a dimensionless scalar with values ranging from 0 to 32767.

Figure 5.9.1 depicts the received power spectrum of the transmitted signal. It corresponds to the output of the FFT spectrum-analyzer tool that is included in the GNU Radio framework. A peak is visible with our software receiver when we choose



5.9.1: Power spectrum of our software transceiver recorded with the USRP and drawn by FFT gnuradio plot.



5.9.2: Receiver symbol constellations.

a central frequency equal to 916 MHz. In addition, the sampling rate has a value equal to 35 samples per symbol in order obtain an intermediate frequency equal to 1.5 MHz. This value is in concordance with the values taken by the transmitted power spectral density of the IEEE 802.15.4 standard. Furthermore, frequencies at the edge of the main band are visible but strongly attenuated. These imperfections may be due to the roll-off characteristics of the interpolation filter in the up-conversion processing of the FPGA USRP.

Figure 5.9.2 shows a constellation of two symbols separated by an angle π. Obviously, this result confirms the demodulation process of D-BPSK signal.

The D-BPSK communication has been evaluated with two parameters: BER/SNR transmission and PER. To calculate BER parameter, we replaced at a transmitter side the blocks of a packet generator by a source of bits. Furthermore, we placed BER and SNR calculator block just after D-BPSK demodulator. Figure 5.9 illustrates

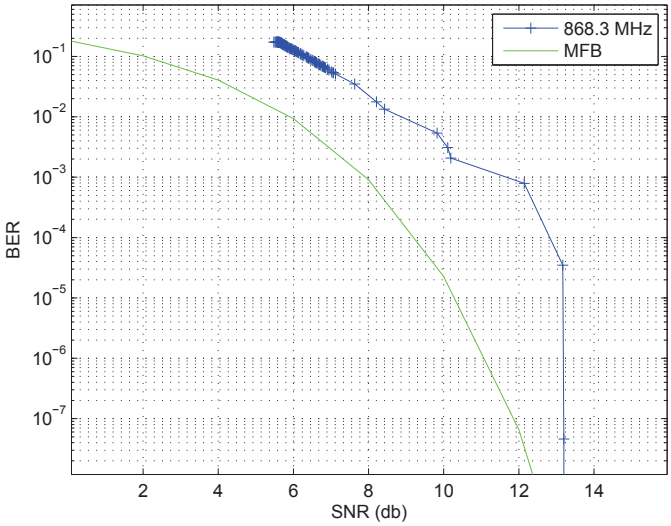


Figure 5.9: The BER versus received SNR for central frequency 868.3 MHz and for the MFB

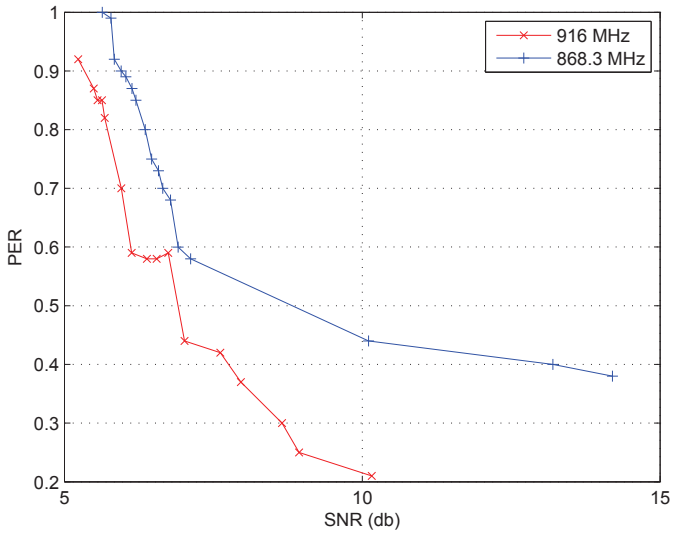


Figure 5.10: The PER versus SNR using two central frequencies 916 MHz and 868 MHz

the average BER versus the input SNR (dB) for the frequency 868.3 MHz and for the MFB Matched Filter Bound of D-BPSK modulation. The results have been computed by changing the amplifier DAC values from 1000 to 12000 with a step equal to 100 for a time period equal to 400 seconds. Despite noisy wireless environment, the results are in concordance with the theory, proving that the implementation is working.

To obtain the PER versus the SNR (dB), the packet generator and packet sink are connected respectively to transmitter and to receiver chains. The PER is calculated by the receiver for a burst of packets. In fact, the transmitter sends for a given amplitude 100 packets. The period time between two successive packets was 0.2 sec. The PER decreases when the amplifier amplitude increases. Thus, the obtained curve (see Figure 5.10) confirms that of the BER/SNR.

The PER depends on the synchronization between the transmitter and the receiver. Note that the synchronization has not been obtained for each burst of packets. This problem can be caused by two sources: the management of the USRP's buffer and inter-blocks delays within blocks of flow graphs.

8 Summary

In this chapter, we reported two SDR implementations of two possible PHY layers of WSNs based on the IEEE 802.15.4 standard. The first one for 2450 MHz frequency band and the second one for 868/915 MHz frequency band. We performed a reverse engineering process of the available SDR transmitter and receiver for 2450 MHz. The result of this process was depicted through flow graphs. A flow graph describe the PHY layer from data packets generation up to their transmission in a baseband signal and vice versa. The data packet reception was the most important step. However, the packet decoder of a receiver flow graph was described by pseudo-code algorithms.

Our new implementation of PHY layer for the 868/915 MHz frequency band was also described by flow graphs. It was inspired from that implemented for the 2450 MHz frequency band. Our contribution suggests the use of a new frequency band and an optional D-BPSK modulation/demodulation. It also contributes to GNU Radio platform by introducing a new packet decoder block. The latter integrates differential decoding and DSSS based on new PN sequences.

We tested the implemented flow graphs through real wireless communications. We were able to exchange IEEE 802.15.4 packets between two separated GNU Radio USRP SDRs. Of course, these flow graphs process a baseband signal generated/received by a host computer, where USRPs serve as a radio signals receiver and transmitter. In addition, we ensured wireless communication between these SDRs in one side and hardware transceiver in another side. The hardware transceivers are the AT86RF230 and the CC2420 used by some off-the-shelf sensor motes for the frequency band 2450 MHz.

Some conclusions can be drawn from our programming experience under the GNU Radio USRP SDR platform. A synchronization problem of SDR transmitter and SDR receiver can be encountered. This problem comes from both processing speed and packet size. Within a flow graph, some blocks can take more time than others, and a phase shift can be generated at the demodulator. This processing time depends on the complexity of the block algorithm. In our case, the algorithm of the packet decoder has a complexity in $O(n^2)$, which is greater than that of other blocks. The result of an unbalanced processing time generates a phase shift between a receiver and transmitter. Hence, the algorithms of each block need to be efficient by avoiding unnecessary control instructions. It is preferable to do the debugging in simulation mode, without intrusive print functions. Furthermore, the impact of changing packet size on PSR of SDR communications was experienced. The PSR depends on the packet size without a precise correlation.

With the presented fourth flow graphs, a reconfigurability of radio transmitters and receivers can be planed. The flow graphs can be loaded on-line (or during run time) of radio communications. They can also be combined to create a multi PHY layers transmitter and receiver. Each PHY layer can be selected according to application requirements, *i.e.* transmission range and data rate.

Other experiment test can be realized to communicate a new hardware transcievers, such as those for 868/915 band, with proposed flow graphs.

Chapter 6

Cognitive Wireless Sensor Network based on IEEE 802.15.4

What man-made machine will ever achieve the complete perfection of even the goose's wing?

Abbas Ibn Firnas

Contents

1 Introduction . . . 139

2 Problem statement . . . 140

3 Related works . . . 142

4 Dynamic spectrum access on GNU Radio USRP SDR . . . 143

4.1 Reconfigurable SDR settings . . . 144

4.2 Energy Detector . . . 146

4.3 Dynamic frequency selection . . . 146

5 Experiments and results . . . 148

6 Summary . . . 152

1 Introduction

The reconfigurability of the suggested flow graphs in Chapter 5 allow us to extend our research works to their usage for **D**ynamic **S**pectrum **A**ccess (DSA). DSA is one of the most important tasks towards developing Cognitive Radio Wireless Sensor Networks. This chapter deals with DSA for IEEE 802.15.4 networks. Some of its content has already been published in Wireless Innovation Forum SDR'15 [4].

The objective of DSA is to improve the IEEE 802.15.4 network reliability when coexisting with other networks, such as IEEE 802.11b/g/n. Thanks to the flow

graphs developed in Chapter 5, network nodes can communicate simultaneously in the two unlicensed frequency bands of 868/915 MHz and 2450 MHz. Using DSA, the network interference can be avoided, and packet success can be improved with DSA. To reach this purpose, the network nodes need a Spectrum Sensing (SS) technique and coordination algorithm to select the carrier frequency.

We start this chapter by motivating our proposal, after that, we outline the related works dealing with cognitive radio, DSA and implementations of the IEEE 802.15.4 standard. In the Section 4, we describe the SDR of our DSA solution with energy detector (or energy-sensor) and synchronization algorithms. We end this chapter by a presentation of experiments results and discussions on the obtained results.

2 Problem statement

Spectrum scarcity issue in wireless communications is a main consequence of spectrum regulation and rigidity of telecommunication standards. Regulation authorities of telecommunication, such as ITU (International Telecommunication Union) or FCC (Federal Communications Commission), define unlicensed spectrum bands for numerous applications in ISM bands. As seen in the previous chapter, IEEE 802.15.4 based WSN uses these bands [20]. Under 2450 MHz band, a WSN shares the unlicensed spectrum with other networks such as IEEE 802.11 and IEEE 802.15.1. Sharing the spectrum without coordination is the source of interference and unreliable transmissions. Figure 6.1 shows the overlapped channels of the IEEE 802.15.4 with those of IEEE 802.11b/g. Although the new version IEEE 802.15.4e [20] introduces a new mechanism of frequency hopping, the values of each frequency remain static (see Section 3.0.1).

868/915 MHz frequency band is an alternative band that depends on geographical region, 868 MHz for Europe and 915 MHz for North America. Thus, this additional frequency band can be used conjointly with that of 2450 MHz. The objective is to avoid disturbed frequency bands and transmit on the good ones. The estimation of the channel quality and channel selection should be performed on-line and transparently for the application constraints.

Cognitive Radio (CR) is a system which senses the electromagnetic environment and dynamically adjusts the radio parameters to improve radio performances. The carrier frequency is one main parameter, adjusted by CR, in order to avoid interference and to efficiently use a spectrum. According to the definition of Haykin in [15], there are three main tasks for CR: Radio scene analysis, Channel identification and Transmit power and frequency control (see Figure 6.2). These tasks can be accomplished by an SDR. However, we need transmitters and receivers based SDR. The implementations shown in Chapter 5 can be reused for a cognitive WSN.

The crowded state of 2450 MHz band can be addressed with DSA or Dynamic

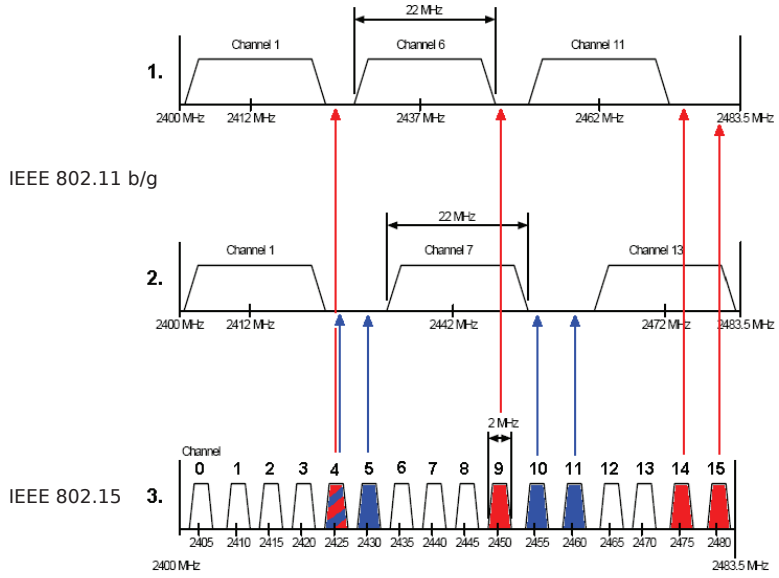


Figure 6.1: Overlapping of IEEE 802.15.4 channels with that of IEEE 802.11b/g in 2450 MHz band

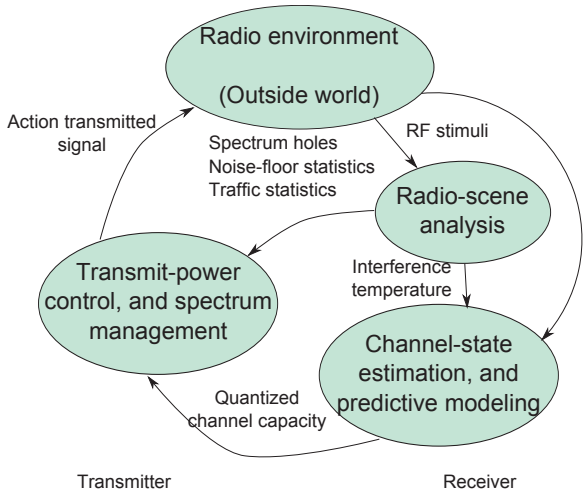


Figure 6.2: Basic cognitive cycle [15]

Spectrum Sharing (DSS). Contrary to static spectrum access, dynamic spectrum users can adjust the carrier frequency dynamically. DSA can be considered as a sub-system of CR dealing with spectrum access. Similarly, Spectrum Sensing (SS) is a DSA sub-system. It provides information about spectrum state. For example, signal strength (power) for each carrier frequency is an information returned by energy detector based on spectrum sensing. We are therefore interested in estimating the quality of a central frequency and in detecting spectrum holes. In addition, we investigate the problem of selecting the spectrum sensing node, *i.e.* the receiver or transmitter.

Currently, nodes in WSNs cannot be SDRs due to a high-power consumption of SDRs. However, GPP based software transceiver can emulate functions of wireless sensor nodes. The two software implementations of IEEE 802.15.4 standard presented in the previous chapter can be reused. Since the two frequency bands of 2450 MHz and 868/915 MHz are far from each other, the particular RF front end setup is needed. Furthermore, the packet transmission requires a coordination of the transmitter and the receiver.

3 Related works

In the literature, the first research work dealing with cognitive radio for WSNs is that published by Ozgur B. Akan et.*al* in [114]. This work motivates the development of Cognitive Radio Sensor Networks (CRSNs) instead of classical WSN, especially when these networks overlap in the same frequency band. However, it doesn't give any details about how to realize the network node coordination for dynamic frequency change. Classification of spectrum sensing for WSNs has been proposed in [102]. The authors in [102] consider energy resources of sensor nodes as a criterion to select an adapted spectrum sensing technique. The energy estimator (or energy detector) was identified as simple to be implemented by sensor nodes. Nevertheless, this work needs to be completed by proof of concepts and experiments.

Since we are interested in IEEE 802.15.4 standard, we focus on works connected to the IEEE 802.15.4 standard and dealing with channel overlapping. In the following two sub sections, we briefly describe the related specifications and implementations.

3.0.1 Related specifications

The main new contribution of IEEE 802.15.4e is an access mode based on Time Slotted Channel Hopping (TSCH) mode [20]. TSCH was introduced for network capacity increase, high reliability and predictable latency. It handles multichannels based on channel (frequency) hopping. In the 2450 MHz band, the hopping among 16 channels is a function of time slots and on the number of available channels. Thus, a frequency is selected based on previous chosen channels and on the number of available channels. The channel allocation shows the possibility to dedicate different

channels to each couple of wireless nodes. However, this allocation only depends on the time slots and not on the link quality. Link Quality Indicator (LQI) [20] informs on the energy strength and the quality of received data frames in a selected channel. Although the LQI is measured, the selected carrier frequency is predefined. In addition, changing dynamically channels in TSCH is not possible without MAC protocol coordination. These drawbacks motivate the use of spectrum sensing before channel selection.

3.0.2 Related implementations

Several research works have been proposed on IEEE 802.15.4 standard, using GNU Radio and USRP. The first SDR has already been detailed in Section 5.1 in Chapter 5. As we have seen, it reproduces the OQPSK layer in the 2450 MHz frequency band. The state of the art on further development was presented in the same chapter [39] [40]. In addition, the BPSK layer was implemented for the 868/915 MHZ frequency band [1]. The SDR for two bands could be combined to implement a multi bands and multi specifications SDR. The frequency bands and standard specifications changing can be based on specific criteria. For example, spectrum sensing can be used to formulate one criterion.

Surveys of DSA and SS techniques have respectively been proposed in [115] and [116]. The DSA techniques have been classified in three classes: Dynamic Exclusive Use Model, Open Sharing Model and Hierarchical Access [115]. The Open Sharing Model employs open sharing among peer users as the basis for managing unlicensed spectral bands. Spectrum sensing techniques have been grouped with three main classes: Energy-detector based sensing, Cyclostationarity-Based Sensing and Matched-Filtering. Energy-detection based spectrum sensing is a simple SS to implement, and the only one found on GNU Radio. It is proposed in [117], based on time averaged Power Spectral Density. This detector is used as a general dynamic spectrum access in [118]. In [119], this energy detector is evaluated according to a probability of detecting wireless activity for cognitive radio. The works [118] and [119] are not specified for a particular network. However, the spectrum sensing can be used by DSA with IEEE 802.15.4-based network.

The reminder of this chapter presents technical details about the proposed DSA and the performed experiments.

4 Dynamic spectrum access on GNU Radio USRP SDR

Our DSA follows an open sharing model (or spectrum commons). It is a DSA strategy where each network has equal rights in an unlicensed frequency bands. We consider the IEEE 802.15.4 2450 MHz and the 868/915 MHz bands, where this network is

considered as SU and other unlicensed users are PUs. For each band, IEEE 802.15.4 Tx/Rx chains are implemented with GNU Radio and can be reused as black boxes. In addition, we dedicate the spectrum sensing and the frequency selection only to the SU receiver. A spectrum sensor measures the energy (power) strength in a given frequency band, and according to a threshold, a carrier frequency is selected. Notice that, PUs could be based an Orthogonal Frequency-Division Multiplexing (OFDM) transmitter in these two bands.

4.1 Reconfigurable SDR settings

In our SDR we assemble a number of transmission chains, as several chains can be handled in one GNU Radio program. For SU receiver (Rx), we implement five chains. The two firsts are IEEE 802.15.4 Receivers (Rx) for two bands, the 2450 MHz and 868/815 MHz. The third and forth chains are Transmitter (Tx) and Rx of **G**aussian **M**inimum **S**hift **K**eying (GMSK) packets. Finally, the fifth chain is an energy detector Rx. Each chain is selected to transmit or receive information according to a synchronization algorithm. The particularity of our work is to be able to perform real wireless transmissions of packets and to online reconfigure transmission chains online.

The two components of the SDR are the USRP 1 front end and the GNU Radio software. The USRP 1 has been chosen regarding its less sampling rate compared to newest versions *e.g.* USRP N210 [37]. The sampling rates are sufficient to build an IEEE 802.15.4 communication and to experiment DSA. In addition, USRP 1 can hold two daughter boards. They contain two antennas, the first for Transmission/Reception (TX/RX) and the second for Reception (RX) only. An SBX daughter board is used since it covers a large frequency band at radio front end, *i.e.* from 400 MHz to 4000 MHz, the boards cover the two 802.15.4 frequency bands of 868/915 MHz and 2400 MHz. In Section 5, SDR setup will be discussed. SDR chains are flow graphs built on GNU Radio toolkit. One flow graph represents a chain of software blocks written in C++ and connected through Python script.

Tx and Rx of SU and PU are featured by a set of GNU Radio chains. Figure 6.3 shows chains needed by a SU receiver to sense a spectrum, to coordinate a frequency selection and to receive IEEE 802.15.4 packets (data). Two receivers of 802.15.4 packets in two frequency bands 868/915 MHz and 2450 MHz are based on [1] [38]. Tx and Rx chains of GMSK packets are connected to SU receiver. In fact, through several tests, GMSK packets exchange was found reliable, *i.e* every time when Tx transmits GMSK packets, Rx succeeds in packet reception without a phase synchronization problem. Hence, to coordinate a frequency selection, the acknowledgment GMSK packets are exchanged. The spectrum sensing is handled by an energy detector chain (see next Section 4.2).

SU Tx chains are shown by Figure 6.4. Two sub transmitters are implemented for each frequency band. Similarly to the SU receiver, a frequency selection is

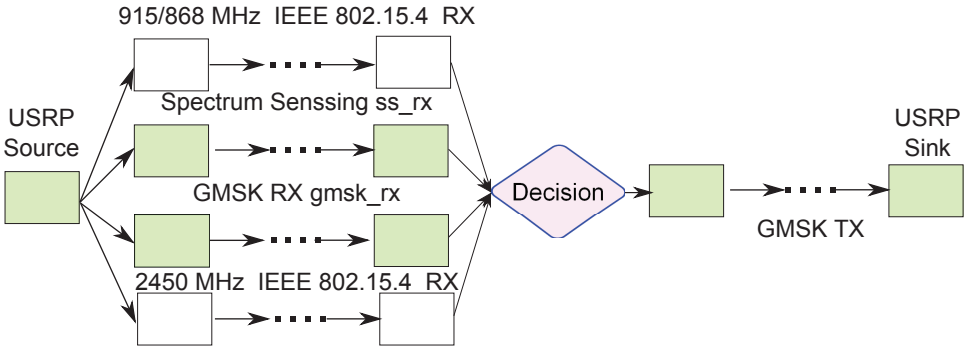


Figure 6.3: Software chain of SU receiver (Rx)

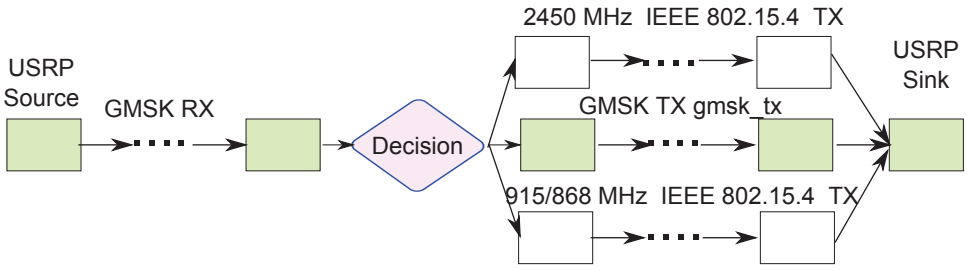


Figure 6.4: Software chain of SU transmitter (Tx)

coordinated through GMSK acknowledgment exchange. Figure 6.5 highlights an SDR chain of PU Tx, which generates a random data stream and modulates it via OFDM modulator. This modulation is chosen since it is the one specified for the IEEE 802.11 standard of Wifi network.

To separate SDR chains of SU, two daughter boards are used by the USRP module. Each daughter board is dedicated to Tx or Rx chains of SU (Tx SU/ Rx SU). In addition, one daughter board has two possible antennas outputs: Tx/Rx or Rx. Hence, for SU Rx, the GMSK Tx and Rx are carried out by the first daughter board through Tx/Rx and Rx antennas, respectively. The second daughter board supports the energy detector and the IEEE 802.15.4 Rx chains. Separated antennas allow the energy detector, GMSK Tx and Rx to be carried out continually. Furthermore, the SU Rx is similar to the USRP of SU Tx, and it contains two daughter boards, and each one supporting the GMSK Tx/Rx and the IEEE 802.15.4 Rx chains.

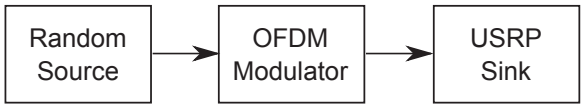


Figure 6.5: Flow graph of PU transmitter (Tx)

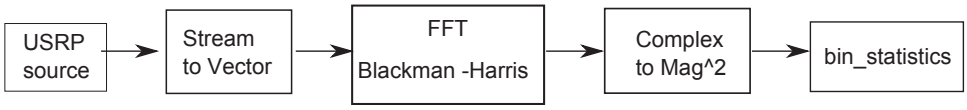


Figure 6.6: Flow graph of our energy detector based spectrum sensing

4.2 Energy Detector

Spectrum sensor or energy detector (see Figure 6.6) estimates the output of a time-averaged Power Spectral Density (PSD). For this purpose, the flow graph starts by receiving the baseband stream from USRP source. The stream is adapted to the capacity of the USB host. Since this stream is continuous, Stream to Vector block packs a group of samples to form vectors of complex samples. Then, under a Fast Fourier Transform (FFT) block, a Blackman-Harris window is used for single tone measurement, applied to each 512 sample vector. In the next block, the modulus squared is calculated averaging the magnitudes of each bin (carrier frequency) over many samples. The last block *bin_statistics* deals with USRP 1 constraints.

The average energy at a given carrier frequency is calculated using the following model:

$$E = \frac{1}{2N}\left[\sum_{n=-N}^{N} |s(n)|^2\right] \tag{6.1}$$

where N is the number of samples and $s(n)$ is the sample values as a function of the sample number n.

In fact, an RF bandwidth from and to host computer is limited regarding USB 2 capacity(8 MHz). Consequently, the frequency bandwidth to examine is divided into chunks of 8 MHz. Since a central frequency is changed via GNU Radio program, the effective change takes an extra delay on the local oscillator. During this delay or *tune_delay*, the received samples are considered wrong and dropped. As explained in the previous section, only the receiver carried out the energy detector.

4.3 Dynamic frequency selection

The proposed algorithm is a message-based algorithm. It is defined in two parts algorithms 5 and 6 performed at SU Rx and SU Tx, respectively. The algorithm allows the nodes of SU to decide on which carrier frequency should be selected. It also reconfigures the communication chains by loading online adapted ones to the selected frequency.

In order to select a carrier frequency, the Rx triggers a coordination process. It starts by a spectrum sensing in a given frequency band. Then, it selects a carrier frequency which has minimum energy power. Thus, the GMSK acknowledgment messages are exchanged to ensure the effective change of the carrier frequency.

Algorithm 5: Receiver (Rx)

```
1  initialization();
2  while energy > threshold do
3  |  spectrum_sensing(ss_rx);
4  end while
5  while not receive_freq_ack(gmsk_rx) do
6  |  send_new_freq(gmsk_tx);
7  |  if time > timeout then
8  |  |  break;
9  |  end if
10 |  ;
11 end while
12 while time ≤ timeout do
13 |  send_clear-to-receive(gmsk_tx);
14 end while
15 start_rx_802.15.4(802_15_4_rx);
```

Algorithm 5 enumerates actions of SU Rx, which senses a given frequency band and selects a carrier frequency when a sensed energy for that frequency is less than a fixed threshold (**(2)** to **(4)** in Algorithm 5). This threshold is taken empirically based on previous experiments. Only 8 MHz FFT window was swept by the energy detector. The desired frequency band is covered by shifting this window across the frequency band. The energy detection is the output of the flow graph *ss_rx* (see Figure 6.3 and Figure 6.6). Thus, the new carrier frequency is selected and forwarded to 802.15.4 Tx via the *gmsk_tx* (see Figure 6.3 and see also Algorithm 5 from **(5)** to **(7)**). As explained above in Section 4.1, one antenna is dedicated to the GMSK exchange. Since *gmsk_rx* demodulation is launched simultaneously with *gmsk_tx*, the forwarding of this frequency is repeated until the reception of an acknowledgment from the SU Tx. After that, during a *timeout*, the SU Rx confirms to the SU Tx that it is clear to receive data packets (from **(8)** to **(10)** in Algorithm 5).

The SU Tx starts data transmission only after receiving a new carrier frequency and verifying if the SU Rx is clear to receive (from **(5)** to **(7)** in Algorithm 6). An acknowledgment is transmitted using *gmsk_tx* to confirm the reception of a new frequency. As compared with the receiver SU Rx, the SU Tx continually resends acknowledgments during a *timeout* until it receives a clear-to-receive message. From line **(8)** to **(12)** of Algorithm 5, the SU Tx sends data packets only if the clear-to-receive message is received, else the reception is failed.

Algorithm 6: Transmitter (Tx)

```
initialization();
while not new_freq_received do
    receive_new_frequency(gmsk_rx);
end while
while (not clear-to-receive(gmsk_rx)) and (time ≤ timeout) do
    send_freq_ack(gmsk_tx);
end while
if clear-to-receive(gmsk_rx) then
    start_tx_802.15.4(802_15_4_tx);
else
    transmitter failed to receive clear-to-receive;
end if
```

5 Experiments and results

In our experiments, three USRP 1 devices are connected to a laptop computer in an office environment. Two devices represent SU transmitter (Tx) and receiver (Rx), whereas PU transmitter is the third one. The 868/915 MHz and 2450 MHz bands are covered by SBX daughter boards, which are plugged into a USRP 1. In GNU Radio part of the SDR, each USRP 1 is controlled via set of chains as shown in Figure 6.3 and Figure 6.4 of the previous Section 4.1 Each chain has its parameters to initialize before and during SDR execution.

Table 6.1 shows offline and online parameters of spectrum sensor. The offline parameters are the sample rate and the channel bandwidth. They are initialized in the source code program before its execution. The online parameters are the bandwidth of spectrum chunks, the window's FFT, and the number of bins. They are calculated based on the offline parameters. The size of an FFT window is defined by a number of bins. It is given by Equation 6.2. The frequency bandwidth recovered at software level depends on the USB port's permeability, this bandwidth is bounded upper by 8 MHz. Thus, bin_start and bin_stop variables are introduced to reduce the size of one FFT window by 1/8 (see Equations 6.3 and 6.4). In fact, in our experiments 80 bins are discarded at the beginning and the end of an FFT window. Thus, the energy detector deals with a chunk of frequency bandwidth defined by a number of bins (or carrier frequency) spaced by a channel of 6250 Hz. For each frequency, the energy sensed is the average of the magnitudes of each bin over 512 samples (see Equation 6.1). For example, the frequency band from 2405 to 2480 is divided into bandwidth chunks of 3 MHz, where the energy is calculated for each carrier frequency spaced by 6250 Hz.

$$\text{fft\_size} = \left\lceil \frac{\text{usrp\_rate}}{\text{channel\_bandwidth}} \right\rceil \tag{6.2}$$

Table 6.1: Parameters of energy detector

USRP sample rate	**channel bandwidth**	**chunk of bandwidth**	**number of bins**	**FFT window**
4 MS/s	6250 Hz	3 MHz	480	640

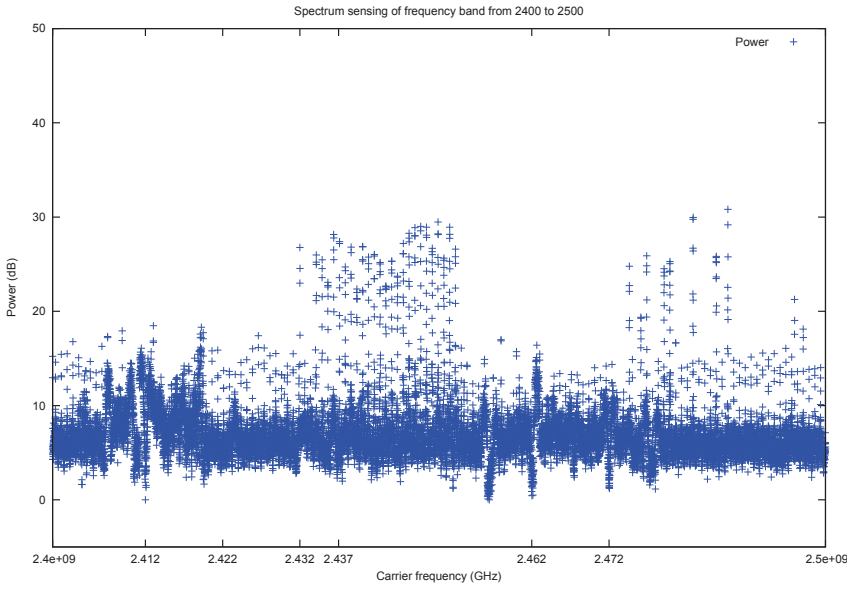


Figure 6.7: Spectrum Sensing of frequency band 2.4 GHz to 2.5 GHz

$$\text{bin\_start} = \left\lceil \frac{\text{fft\_size}}{8} \right\rceil \tag{6.3}$$

$$\text{bin\_stop} = \text{fft\_size} - \text{bin\_start} \tag{6.4}$$

Since the experiments are performed in an office environment, the two targeted frequency bands of 868/915 and 2450 MHz have been sensed to get the energy power. In addition, the WiFi board of the laptop computer has detected the presence of seven IEEE 802.11 networks. Figure 6.7 shows the obtained Power Spectrum Density (PSD) for each carrier frequency from 2400 MHz to 2500 MHz using the energy detector. Two high-power zones have been observed in the interval from 2400 MHz to 2500 MHz. In fact, the energy power is in the order of 30 dB in intervals [2430 MHz, 2450 MHz] and [2475 MHz, 2490 MHz]. The seven detected networks have a small impact on the spectrum. In the second frequency band from 850 MHz to 950 MHz, the energy level is lower than 25 dB (see Figure 6.8). Hence, the detected radio-frequency activities cannot significantly disturb our experimental scenarios.

After characterizing of the radio frequency environment, we consider two scenarios, with and without the DSA. In the first scenario, the SUs are disturbed by OFDM transmitter, *i.e* PU. The SUs exchange 7519 packets following the IEEE 802.15.4

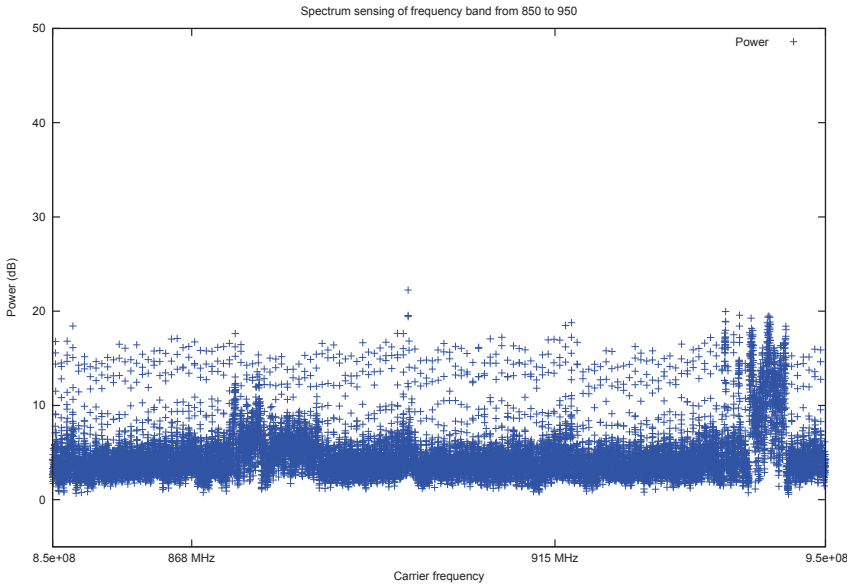


Figure 6.8: Spectrum Sensing of frequency band 850 MHz to 950 MHz

standard, *i.e* data packets, where 50 ms is the inter-packet generation time. A disturbance is triggered at different frequencies that are around the carrier frequency of SUs. In the second scenario, the SU performs a dynamic frequency selection. Thus, the robustness of the dynamic frequency selection is measured using **P**acket **S**uccess **R**ate (PSR) and **P**acket **R**eceived **R**ate (PRR) parameters.

We consider in the first scenario that the couple of SU communicates in channel 26 (carrier frequency is 2480 MHz). The Tx generates a data stream of 1 MB and splits it into packets. Each packet has a size of 133 bytes. The data rate between Tx, and Rx is fixed to 250 kb/s (note that this rate is the same as the OQPSK PHY). In addition, we placed the transmitter USRP near the USRP receiver. However, software amplifier DAC and software gain UHD_G have low values equal to 0.4 and 40 dB, respectively. For amplifying the baseband signal, the constant float value DAC is multiplied by the two signal components: In and Quadrature-phase. The UHD_G is a relative gain fixed in the block of a USRP sink.

The disturbance (or the OFDM PU) generates an OFDM data stream in frequencies close to that of the SU. In fact, in an interval of 2 MHz, from 2479 MHz to 2481 MHz, the OFDM signal is triggered and sweeps this interval by a step of 0.1 MHz. Figure 6.9 shows the obtained Packet Success Rate (PSR) and Packet Received Rate (PRR) calculated using the Cyclic Redundancy Check (CRC). The PRR is calculated in the case when packets are received but with a wrong CRC. Obviously, the PSR drop to 0 when the spectral distance between PU and SU is lower than 0.3 MHz. Indeed, the PSR and the PRR are low since the SU Tx cannot detect the PU Tx.

The second scenario proceeds like the first one but the SU adopts DSA to avoid PU disturbance. DSA is started by SU Rx, which continually senses frequency band from 2400 MHz to 2480 MHz and the central frequency 868 MHz. Each carrier

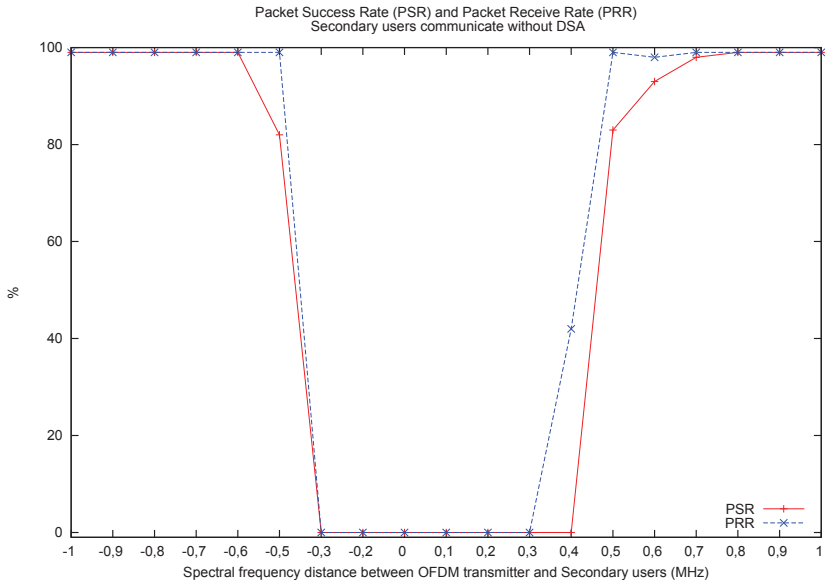


Figure 6.9: Packet Success Rate (PSR) and Packet Received Rate (PRR) function of spectrum distance between PU and SU without DSA.

frequency is characterized by an energy level. Thus, the selected one is that with a minimum energy level. It is communicated to SU Tx following algorithms 5 and 6. After that, SU Tx starts data transmission. The OFDM disturbance (or PU role) is triggered for the selected frequency, meanwhile the SU Rx continually senses new bandwidth chunks. The size of each chunk is equal to 3 MHz for the two frequency bands 2450 MHz and 868/915 MHz. Then, the SU Rx selects a new carrier frequency according to the lower energy sensed (see Section 4.2). In the experiment, time period needed for every chunk is 1800 ms. Thus, a number of data packets are dropped during spectrum sensing.

Figure 6.10 shows that PSR and PRR fall approximately by 20%, when PU is at spectral distance of 0.3 MHz. In fact, this packet loss results in an extra time required for spectrum sensing and for frequency selection. Obviously, this extra time depends on the spectrum sensing parameters (see Table 6.1). The time to sense a band of 1 MHz is around 600 ms using previous parameters. In addition, when SU Rx selects 868 MHz, the modulation change to BPSK and bit rate decreases from 250 kbps to 20/40 kbps.

With DSA, SU improves the PSR by 80% compared to a classical transmission over a static channel. This result depends on the spectrum sensor and the SU experimental parameters. In fact, this percentage can be improved if SU Tx increases inter-packet generation time, and SU Rx reduces the time to sense spectrum bandwidth.

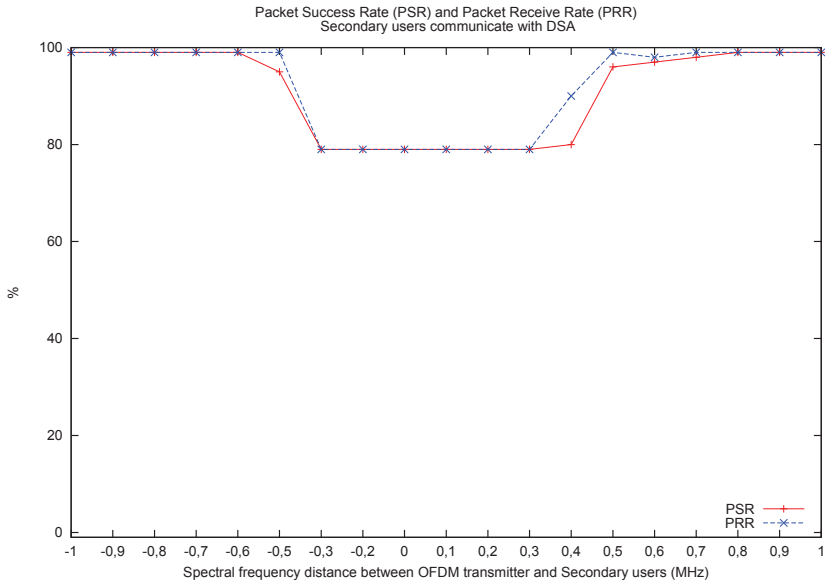


Figure 6.10: Packet Success Rate (PSR) and Packet Received Rate (PRR) function of spectrum distance between PU and SU with DSA.

6 Summary

In this chapter we have proposed a new dynamic spectrum access using an energy-detector based spectrum sensing. Implemented on the GNU Radio, the DSA has been performed throughout two frequency bands 868/915 MHz and 2450 MHz of the IEEE 802.15.4 standard. The two realized SDR presented in the previous chapter have been combined. In fact, communication chains of BPSK, OQPSK, GMSK and energ detector receiver/transmitter have been assembled in one SDR. A message based algorithm has been developed to synchronize frequency selection. It coordinates the reconfigurability of transmitter/receiver chains. It loads online a corresponding chain to the frequency band of a selected carrier frequency.

Under a real packet transmission and in indoor environment conditions, we showed the usefulness of the developed DSA. We improved the PSR by 80% when using DSA rather than a static frequency selection, despite the extra time needed for spectrum sensing and carrier frequency selection.

Chapter 7

Conclusions and Future Work

The duty of the man who investigates the writings of scientists, if learning the truth is his goal, is to make himself an enemy of all that he reads, and, applying his mind to the core and margins of its content, attack it from every side. He should also suspect himself as he performs his critical examination of it, so that he may avoid falling into either prejudice or leniency

Ibn al-Haytham "Alhazen"

Contents

1 **Conclusions** 153

2 **Future Work** 156

1 Conclusions

WSNs are an emergent technology with their diverse applications such as in smart metering and environment monitoring. Some industrial companies invest to replace wired sensors by wireless ones. Such replacement require reliable WSNs. This reliability is influenced by the coexistence of other wireless technologies in the same frequency band within the same geographical area. Commonly, these frequency bands are defined by standard organizations. It is the case of the IEEE 802.15.4 standard. The latter defines a number of the PHY layers corresponding to dedicated frequency bands, nevertheless, its implementation in Hardware is not reconfigurable. Thanks to SDR, it is possible to implement these PHY layers in software.

The SDR is a new way to define and manipulate radio communication components in software. The reconfigurability is the most important feature of an SDR. It allows engineer and researcher to implement radio standards and wireless technologies with a maximum flexibility. It is possible to easily update radio parameters. The frequency band and the modulation are the parameters of the WSNs PHY layer we want to

address in this thesis. Cognitive radio is based on SDR. It deals with frequency scarcity problem using essentially DSA and spectrum sensing.

This thesis mainly focused on a SDR solution for WSN based on the IEEE 802.15.4 standard. The main contribution presented in this thesis is: the characterization of an SDR platform that implements both standardized PHY layers and cognitive radio for WSNs.

All of our works tried to answer to three main challenges:

- If we select GNU Radio and USRP platform to implement standardized PHY layer, what is its characteristics in software and hardware?
- How to implement the two PHY layers for the two frequency bands 2450 MHz and 868/915 MHz.
- How to realize cognitive WSN based on the IEEE 802.15.4 standard?

The first challenge led us to a theoretical study of different architectures, standards and platforms of SDR (see Chapter 2). We focused on existing proposed SDR for embedded devices. We suggested two classifications of existing architectures: based on GPP and based on reconfigurable hardware. We found in the state of the art that GPP based architecture brings more facility compared to reconfigurable hardware architectures. An SDR platform based on GPP allows a radio designer to quickly implement an SDR. These implementations are also cheaper using open-source high level programming languages. These advantages were our arguments for selecting the GNU Radio and USRP SDR platform.

The difficulty discovered in the GNU Radio is the lack of rigor in program source-code and frequent upgrading of software design. The software architecture was analyzed and described in Chapter 3. An implementation (or prototyping) based on GNU Radio is a model of flow graph of blocks. The blocks are primarily developed in C++ and they are connected through Python script. Although the design of the GNU Radio is object oriented, the designer needs more time to understand the class organization. Some other properties of flow graph execution were shown, especially the scheduler and SIMD programming. We identified the presence of buffers between OS kernel and executed flow graphs. These buffers can be a source of delayed processing time. It is important to estimate processing time of each block via ControlPort and performance counters tools.

The USRP hardware was detailed through a chosen general architecture for different USRP versions. FPGA of a given USRP is irreplaceable. It accomplishes high-speed general-purpose operations. Daughter boards are also important since they define the radio-frequency front end. We explored the GNU Radio USRP throughout real wireless communications and simulations. We discovered the difficulty to estimate effectively an SNR of a BPSK modulation/demodulation (see Chapter 4). The obtained SNR values in SDR communications were not coherent with the

expected ones. This result adds to the lack of effective information about daughter boards led us to perform a deep analysis of their performances.

Our analysis of the output power and radio-frequency bandwidth of daughter boards was the first research work found in the literature (see Chapter 4). These parameters were characterized through a thorough experimental approach. The analyzed daughter boards were the RFX 2400, RFX 900, SBX and the MIMO USRP B210. The frequency bandwidths of the RFX 2400 and RFX 900 cover, respectively, only 24% and 18 % compared to the advertised bandwidths. The overall output power of these two boards behaves linear with a better linearity for RFX 2400. In contrast, SBX daughter boards and B210 effectively cover the announced frequency bandwidth by Ettus Research [37]. Nevertheless, the output power of these two boards is found to decrease with increasing carrier frequency. Another result of our experiments was the characterization of the impact of signal amplifiers used in GNU Radio flow graphs. The UHD_G and DAC parameters are valuable parameters and their modification has to be done with care. A simple empirical model was introduced to accurately predict the average of output power of an SBX daughter board. These results are useful for developing new implementations of SDRs based on GNU Radio USRP.

We implemented a new possible IEEE 802.15.4 PHY layer for the frequency band 868/915 MHz. It was a result of a performed reverse engineering process of another implementation for the frequency band of 2450 MHz. We thoroughly described the PHY layer flow graphs from data packets generation up to their transmission in a baseband signal and vice versa. The D-BPSK modulation and demodulation were constructed using a set of signal-processing blocks (see Chapter 5). A new block for packet decoding was included to the GNU Radio platform. This block includes DSSS based on new PN sequences compared to those for the 2450 MHz. The implemented flow graphs were tested through real wireless communications. Furthermore, we ensured wireless packet exchange between these flow graphs and sensor nodes. We measured two different parameters for one-to-one SDR communications: the PSR and BER. We found that the PSR depends on the packet size for the PHY layer of the 2450 MHz frequency band. The BER was obtained in relation with the input SNR (dB) for the new PHY layer. Some synchronization problems were encountered to decode packets. The cause of this problem was identified as the unbalanced processing time of blocks and the size of transmitted packets. To avoid such problem, the programming should be similar to that on embedded devices by avoiding unnecessary instructions. Our solution was based on increasing the packets' size. Despite this synchronization problem, the two PHY layers were functional to be combined and to be reconfigured for cognitive radio.

We proposed new Dynamic Spectrum Access based on implemented IEEE 802.15.4 PHY layers (see Chapter 6). This proposal is one of the most important step towards developing Cognitive Wireless Sensor Netwrok. The WSN was defined as SU of the 2450 MHz and 868 MHz frequency bands. The wireless technologies sharing these

bands were considered as PUs. We assembled the two implemented PHY layers to cover their corresponding frequency bands. We also added two other flow graphs for: spectrum sensing based energy detector and GMSK modulator/demodulator. The energy detector was performed only by the receiver. It sweeps the two frequency bands before the selection of a central frequency. A frequency is selected if it receives minimum energy power among all examined frequencies. The synchronization of DSA was ensured by a message based algorithm. The latter also coordinate the reconfigurability of transmitter and receiver flow graphs. Our DSA was experienced through real packet transmissions within indoor environment conditions. We found that despite an extra needed time for sensing, our DSA improves the PSR by 80% compared to the obtained one by a static frequency selection,

2 Future Work

The obtained results of RF measurements on USRPs are useful for future implementations on GNU Radio USRP SDR. Other applications can use the characterized daughter boards. For some applications, the output power for each central frequency can be used to develop effective localization algorithms. The proposed empirical model for SBX daughter boards can be implemented within GNU Radio block. Flow graphs can use this block to calibrate in software the output power from USRPs. The measurement results can also be a benchmark for the research community. In particular, when researchers work on a remote USRP platform, such as CortexLab [1]. Furthermore, our work can be extended to characterize the behavior of USRPs using as an input a wide-band signal, *e.g.* an OFDM signal.

The IEEE 802.15.4 standard is still the de-facto standard for WSNs. Future improvements could be done on the proposed implementations. We draw up three precise extensions:

- It is interesting to estimate the processing time of each flow graph's block. This estimation allows us to synchronize a transmitter and receiver with less difficulty.

- For the proposed PHY layer (868/915 MHz), the tests can be extended by exchanging packets with hardware transceivers, such as ZigBit transceivers [120].

- Other PHY layers can be explored (see Table 5.1 in Chapter 5). The Differential QPSK modulation with a Chirp Spread Spectrum (CSS) can be good candidates to implement, especially when we know that the possible data rate is equal to 1Mb/s .

[1] http://www.cortexlab.fr/

Our DSA can be extended by improving spectrum sensing and the frequency selection condition. The ideal extension of our DSA is to use a wide range of a frequency band. In our research works, we were able to cover with one USRP 1 two different frequency bands. It was possible thanks to SBX daughter boards. One important difficulty comes from antenna levels using two different transmitters. Furthermore, spectrum sensing could be distributed by sensing in each transmitter/receiver as well as sharing information about spectrum state. The frequency can also be selected using networks requirements, such as data rate and data priority.

In the literature the concept of Cognitive Radio is active with several theoretical solutions. It is more efficient to think about the real realization of these solutions before their theoretical formulation. Our experience through SDR implementations brought us more comprehension of wireless networks and digital signal processing particularities. For further works on cognitive radio, we are able to use the accumulate knowledge of our thesis experience.

Finally, the Internet of Things (IoT) is an interesting application of WSN. The IEEE 802.11 ah [121] is a new standard for low power wireless networks for IoT. It addresses the network lifetime which is the most important limitation of WSN. It also promotes a high data rate compared to the last release of IEEE 802.15.4 (version 802.15.4e). The ISM frequency band from 902 MHz to 928 MHz was specified again. Thus, it also overlaps with other radio technologies. We think that Software Defined Radios might be useful for IoT communications under the scarcity of the spectrum.

Glossaries

Acronyms

ADC

***A**nalog to **D**igital **C**onverter*. 48, 49, 53, 61

ADRES

***A**rchitecture for **D**ynamically **R**econfigurable **E**mbedded **S**ystems*. 58

ADS

***A**ccès **D**ynamique au **S**pectre*. 5–7, 16–19

ADS-B

***A**utomatic **D**ependent **S**urveillance-**B**roadcast*. 62, 84

AMD

***A**dvanced **M**icro **D**evices*. 54

ARM

***A**cron **R**ISC **M**achine*. 57, 58

ARQ

***A**utomatique **R**epeat re**Q**uest*. 16

ASIC

***A**pplication **S**pecific **I**ntegrated **C**ircuit*. 7, 48, 53, 54, 59, 60

ASIP

***A**pplication-**S**pecific **I**nstruction-set **P**rocessor*. 58–60

AWGN

***A**dditive **W**hite **G**aussian **N**oise*. 86

BER

***B**it **E**rror **R**ate*. 11, 12, 23, 84, 85, 88, 89

BPSK

***B**inary **P**hase-**S**hift **K**eying*. 11, 12, 15, 39, 84, 85, 92, 95

CORBA

***C**ommon **O**bject **R**equest **B**roker **A**rchitecture*. 8, 52–54, 166, —
Glossary: CORBA

CR

***C**ognitive **R**adio*. 35

CRC

***C**yclic **R**edundancy **C**heck*. 119

CSS

***C**hirp **S**pread **S**pectrum*. 19

D-BPSK

***D**ifferential-**P**hase **S**hift **K**eying*. 14, 15

D-QPSK

***D**ifferential-**Q**uadtrature **P**hase **S**hift **K**eying* . 19

DAC

***D**igital to **A**nalog **C**onverter*. 48, 50, 53, 61

DDC

***D**igital **D**own **C**onversion*. 49, 61

DSA

***D**ynamic **S**pectrum **A**ccess*. 28, 139, 140, 142–144, 149–152

DSP

***D**igital **S**ignal **P**rocessor*. 7, 8, 29, 41, 48, 49, 52–54, 56–61

DSSS

***D**irect **S**equence **S**pread **S**pectrum*. 14

DUC

***D**igital **U**p **C**onverter*. 50

ETSI

***E**uropean **T**elecommunication **S**tandards **I**nstitute*. 54, 55

FCC

***F**ederal **C**ommunication **C**ommission*. 35

FFT

***F**ast **F**ourier **T**ransform*. 16, 17, 67, 70, 84

FI

***F**réquence **I**ntermédiaire* . 7–9

FIR

***F**inite **I**nput **R**esponse*. 49, 70,
—
Glossary: FIR

FPGA

***F**ield **P**rogrammable **G**ate **A**rray*. 7, 8, 10, 11, 48, 49, 52–54, 56–60, 67

git

Git. 67,
—
Glossary: git

GMSK

***G**aussian **M**inimum **S**hift **K**eying*. 16–18, 144–147, 152

GPL

***G**eneral **P**ublic **L**icense*. 40

GPMC

***G**roup **P**olicy **M**anagment **C**onsole*. 73

GPP

***G**eneral **P**urpose **P**rocessor*. 6–11, 18, 41, 48, 52, 53, 56, 57, 59, 61, 66

GPS

***G**lobal **P**ositioning **S**ystem*. 98

GPSDO

***GPS D**isciplined **O**scillator*. 91

GUI

***G**raphical **U**ser **I**nterface*. 70

HDTV

***H**ight **D**efinition **T**elevision*. 62

Hw

Hardware. 88

IEEE

***I**nstitute of **E**lectrical and **E**lectronics **E**ngineers*. 32, 37

IETF

***I**nternet **E**ngineering **T**ask **F**orce*. 32, 37

IF

***I**ntermediate **F**requency*. 48–50, 52, 53

IP

***I**ntellectual **P**ropriety*. 57

ISM

***I**ndustrial **S**cientific **M**edical*. 6, 12, 15, 39, 42, 90, 93, 97, 98

ITU

***I**nternational **T**elecommunication **U**nion*. 32, 35, 37

JTRS

***J**oint **T**actical **R**adio **S**ystems*. 40, 53,
—
Glossary: JTRS

LR-WPAN

***L**ow-rate **W**ireless **P**ersonal **A**rea **N**etwork*. 37, 114, 115

LTE-A

***L**ong **T**erm **E**volution-**A**dvanced* . 103

MAC

***M**edium **A**ccess **C**ontrol*. 37, 42

MFB

***M**atched **F**ilter **B**ound*. 89, 90

MIMO

***M**ultiple-**I**nput and **M**ultiple-**O**utput*. 91

MPSoC

***M**ulti **P**rocessors **S**ystem **o**n **C**hips*. 56

MSPS

***M**ega **S**amples **P**er **S**econd*. 62, 91

O-QPSK

***O**ffset-**Q**uadtrature **P**hase **S**hift **K**eying*. 6, 14, 114, 116

OFDM

***O**rthogonal **F**requency-**D**ivision **M**ultiplexing*. 16, 17, 103

OMAP

***O**pen **M**ultimedia **A**pplications **P**latform*. 73

OMG

***O**bject **M**anagement **G**roup*. 53, 166

OS

***O**perating **S**ystem*. 53

OSI

***O**pen **S**ystems **I**nterconnection*. 37

OSSIE

***O**pen **S**ource **S**CA **I**mplementation::**E**mbedded*. 22, 54

PAs

***P**seudo **A**léatoires*. 14

PCIe

***P**eripheral **C**omponent **I**nterconnect **e**xpress*. 73, 74

PHY

***PHY**sique*. 5–7, 14, 18, 19

ppm

parts per million. 91

PRR

***P**acket **R**eceived **R**ate*. 17, 150

PSR

***P**acket **S**uccess **R**ate* . 15–17, 19, 150

QAM

***Q**uadrature **A**mplitude **M**odulation*. 39

RAM

***R**andom-**A**ccess **M**emory*. 53

RCSFs

***R**éseaux de **C**apteurs **S**ans **F**ils*. 5

RF

***R**adio **F**requency*. 35

RISC

***R**educed **I**nstruction **S**et **C**omputing*. 58

RL

***R**adio **L**ogicielle*. 5–9

RMS

***R**oot **M**ean **S**quare*. 100

RRS

***R**econfigurable **R**adio **S**ystem*. 8, 54, 55

RSSI

***R**eceived **S**ignal **S**trength **I**ndication*. 38

Rx

Receiver. 21, 47–49, 114

SCA

***S**oftware **C**ommunication **A**rchitecture*. 8, 51, 52, 54

SDR

***S**oftware **D**efined **R**adio*. 29, 33, 35, 48, 51–57, 59, 61, 63

SIMD

***S**ingle **I**nstruction **M**ultiple **D**ata*. 10, 18, 60

SNR

***S**ignal-to-**N**oise **R**atio*. 11, 23, 84–86, 88–90

SoC

***S**ystem-**o**n-**C**hip*. 48

SWIG

***S**implified **W**rapper and **I**nterface **G**enerator*. 67

THD

***T**otal **H**armonic **D**istortion*. 13, 100, 109, 111

Tx

Transmitter. 48, 104, 111, 114

UHD

***USRP H**ardware **D**river*. 11, 92, 100, 110, 111

UPs

***U**tilisateurs **P**rimaires*. 16

US

***U**tilisateur **S**econdaire*. 16

USRP

***U**niversal **S**oftware **R**adio **P**eripheral*. 6–18, 41, 52, 55, 56, 61, 62, 66, 67, 73, 80

UWB

***U**ltra **W**ide **B**and*. 115

VHDL

***V**ery high speed integrated circuit **H**ardware **D**escription **L**anguage*. 8, 57, 58, 60

VOLK

***V**ector **O**ptimazed **L**ibraray of **K**ernels*. 10, 11, 18

WSN

***W**ireless **S**ensor **N**etwork*. 37, 42, 52

XML

*E**X**tensible **M**arkup **L**anguage*. 67

Glossary

CORBA

Common **O**bject **R**equest **B**roker **A**rchitecture (CORBA) is an **O**bject **M**anagement **G**roup (OMG) standard designed to facilitate the communication of systems that are deployed on diverse platforms, where a several software collaborate on different operating systems. This standard has many of the same design goals as object-oriented programming: encapsulation and reuse. 160

FIR

Digital filter that have an impulse response which reaches zero in a finite number of steps. An FIR filter can be implemented non-recursively by convolving its impulse response with the time data sequence it is filtering. 161

git

Git is a distributed version control system with data integrity, and support for distributed, non-linear programming workflows. 161

JTRS

USA Defence Project launched to gather a several military and civilian radio communications technologies in the same device. 162

Bibliography

[1] **Rafik Zitouni**, Stefan Ataman, Marie Mathian, and Laurent George. "IEEE 802.15.4 transceiver for the 868/915 MHz band using Software Defined Radio". In: *Proceedings of Wireless Innovation Forum SDR'12* abs/1304.8028 (2012). (Cited on pages 3, 124, 133, 143, 144).

[2] **Rafik Zitouni**, Stefan Ataman, and Laurent George. "RF Measurements of the RFX 900 and RFX 2400 Daughter Boards with the USRP N210 Driven by the GNU Radio Software". In: *Proceedings of 5ths International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. IEEE, Oct. 2013, pages 490–494. ISBN: 978-0-7695-5106-7. (Cited on pages 3, 91).

[3] **Rafik Zitouni** and Stefan Ataman. "An empirical model of the SBX daughter board output power driven by USRP N210 and GNU Radio based Software Defined Radio". In: *Proceedings of 12th Multi-Conference on Systems, Signals and Devices SSD'15*. IEEE, Mar. 2015. (Cited on pages 3, 91).

[4] **Rafik Zitouni**, Laurent George, and Yacine Abouda. "A Dynamic Spectrum Access on SDR for IEEE 802.15.4e networks". In: *Proceedings of Wireless Innovation Forum SDR'15* (Mar. 2015). (Cited on pages 3, 139).

[5] **Rafik Zitouni**, Stefan Ataman, Mathian Marie, and Laurent George. "Radio Frequency Measurements on a SBX Daughter Board using GNU Radio and USRP N-210". In: *Proceedings of 3rd IEEE International Workshop on Measurements and Networking 2015*. IEEE, Oct. 2015. (Cited on page 3).

[6] **Rafik Zitouni**, Stefan Ataman, and Laurent George. "Radio Frequency Measurements on SBX Daughter Boards using GNU Radio/USRP N-210". In: *GNU Radio 2015 conference* (Aug. 2015). (Cited on page 3).

[7] J. C. Maxwell. "A Dynamical Theory of the Electromagnetic Field". In: *Philosophical Transactions of the Royal Society of London* 155 (Jan. 1865), pages 459–512. ISSN: 0261-0523. (Cited on page 32).

[8] Claude E. Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27 (1948), pages 379–423, 623–. URL: http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf.
(Cited on page 32).

[9] *Ntia2015*. URL: http://www.ntia.doc.gov/ (visited on 04/15/07).
(Cited on page 34).

[10] J. Mitola. "Software radios: Survey, critical evaluation and future directions". In: *IEEE Aerospace and Electronic Systems Magazine* 8.4 (Apr. 1993), pages 25–36. ISSN: 0885-8985.
(Cited on pages 35, 44, 48, 66).

[11] J. Mitola. "The software radio architecture". In: *IEEE Communications Magazine* 33.5 (1995). ISSN: 0163-6804.
(Cited on page 35).

[12] Joseph Mitola. "Software Radio". In: *Wiley Encyclopedia of Telecommunications*. John Wiley & Sons, Inc., 2003. ISBN: 9780471219286.
(Cited on page 35).

[13] *Definitions of Software Defined Radio (SDR) and Cognitive Radio System (CRS)*. 2009. URL: http://www.itu.int/pub/R-REP-SM.2152-2009 (visited on 04/14/30).
(Cited on page 35).

[14] *Topic 8: Cognitive Radio for Public Safety*. 2003. URL: http://transition.fcc.gov/pshs/techtopics/techtopic8.html (visited on 04/14/30).
(Cited on page 35).

[15] S. Haykin. "Cognitive radio: brain-empowered wireless communications". In: *IEEE Journal on Selected Areas in Communications* 23 (2005). ISSN: 0733-8716.
(Cited on pages 35, 140, 141).

[16] H. Harada, Y.D. Alemseged, and O. Holland. *IEEE Dynamic Spectrum Access Networks (DYSPAN) standards committee*. 2011.
(Cited on page 36).

[17] I Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. "Wireless sensor networks: a survey". In: *Computer Networks* 38 (2002), pages 393–422. ISSN: 13891286. arXiv: 1004.3164.

(Cited on page 37).

[18] Andrew Tanenbaum. "Computer Networks". In: (Aug. 2002). URL: http://dl.acm.org/citation.cfm?id=572404.

(Cited on page 37).

[19] IEEE. *IEEE Standard for Local and metropolitan area networks, Part 15.4: Low-Rate Wireless Personal Area Networks*. 2003.

(Cited on pages 37, 115, 120, 122).

[20] IEEE. *IEEE Standard for Local and metropolitan area networks, Part 15.4: Low-Rate Wireless Personal Area Networks*. 2011. ISBN: 9780738166834.

(Cited on pages 37, 115, 120, 140, 142, 143).

[21] Myung Lee et al. "IEEE 802.15.5 WPAN mesh standard-low rate part: Meshing the wireless sensor networks". English. In: *IEEE Journal on Selected Areas in Communications* 28.7 (Sept. 2010), pages 973–983. ISSN: 0733-8716. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5555922.

(Cited on page 37).

[22] Deji Chen, Mark Nixon, and Aloysius Mok. *WirelessHART™*. Boston, MA: Springer US, 2010. ISBN: 978-1-4419-6046-7. URL: http://www.springerlink.com/index/10.1007/978-1-4419-6047-4.

(Cited on page 37).

[23] *ISA | The International Society of Automation*. URL: https://www.isa.org/ (visited on 05/15/17).

(Cited on page 37).

[24] Zach Shelby and Carsten Bormann. *6LoWPAN: The Wireless Embedded Internet*. John Wiley & Sons, 2011, page 244. ISBN: 1119965349. URL: https://books.google.com/books?hl=en\&lr=\&id=3Nm7ZCxscMQC\&pgis=1.

(Cited on pages 37, 41).

[25] Nikolaos A Pantazis, Stefanos A Nikolidakis, Dimitrios D Vergados, and Senior Member. "Energy-Ef fi cient Routing Protocols in Wireless Sensor Networks : A Survey". In: *IEEE Communications Surveys & Tutorials* 15.2 (2013), pages 551–591. ISSN: 1553-877X.

(Cited on page 38).

[26] Mert Bal and Hamada Ghenniwa. "Localization in cooperative Wireless Sensor Networks: A review". English. In: *2009 13th International Conference on Computer Supported Cooperative Work in Design*. IEEE, 2009, pages 438–443. ISBN: 978-1-4244-3534-0. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=4968098.

(Cited on page 38).

[27] Reinhard German Falko Dressler Feng Chen Isabel Dietrich. "An Energy Model for Simulation Studies of Wireless Sensor Networks using OMNeT++". In: ().

(Cited on page 38).

[28] Nouha Sghaier et al. "Wireless Sensor Networks for medical care services". In: *2011 7th International Wireless Communications and Mobile Computing Conference*. IEEE, July 2011, pages 571–576. ISBN: 978-1-4244-9539-9. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5982596.

(Cited on page 38).

[29] Fotis Foukalas, Vangelis Gazis, and Nancy Alonistioti. "Cross-layer design proposals for wireless mobile networks: A survey and taxonomy". English. In: *IEEE Communications Surveys & Tutorials* 10.1 (2008), pages 70–85. ISSN: 1553-877X. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=4483671.

(Cited on page 38).

[30] Cris dos Remedios. *The Value Of Fundamental Research*. Technical report. Sydney: The University of Sydney, 2006. URL: http://iupab.org/publications/value-of-fundamental-research/.

(Cited on page 39).

[31] *Why we need standards*. URL: http://www.etsi.org/standards/why-we-need-standards (visited on 06/14/19).

(Cited on page 39).

[32] Gnuradio.org. *Gnu Radio*. URL: http://gnuradio.org/redmine/projects/gnuradio/wiki (visited on 2011).

(Cited on pages 40, 62).

[33] *Premium Market Research Reports and Industry Analysis | ASDReports*. URL: https://www.asdreports.com/ (visited on 05/15/18).

(Cited on page 40).

[34] "First Steps in Receiving Digital Information with RDS/TMC". en. In: (). URL: https://fosdem.org/2015/schedule/event/sdr_rds_tmc/.
(Cited on page 40).

[35] *Free Software on the final frontier: GNU Radio controls the ISEE-3 Spacecraft — Free Software Foundation — working together for free software.* URL: https://www.fsf.org/blogs/community/free-software-in-space-gnu-radio-and-the-isee-3-spacecraft (visited on 05/15/26).
(Cited on page 40).

[36] Guy Gugliotta. "An Air-Traffic Upgrade to Improve Travel by Plane". In: *The New York Times* (2009). URL: http://www.nytimes.com/2009/11/17/science/17air.html?ref=science\&pagewanted=all.
(Cited on page 40).

[37] ettus.com. *Ettus Research - Home*. 2010. URL: http://home.ettus.com/ (visited on 06/14/16).
(Cited on pages 41, 42, 52, 56, 58, 62, 63, 72, 74, 75, 76, 90, 91, 93, 97, 98, 144, 155).

[38] Thomas Schmid. "Gnu radio 802.15. 4 en-and decoding". In: *Networked & Embedded Systems Laboratory, UCLA, ...* (). URL: http://citeseerx.ist.psu.edu/viewdoc/download?year={2006}.
(Cited on pages 42, 45, 114, 116, 117, 121, 132, 144).

[39] Leslie Choong. *Multi-Channel IEEE 802.15.4 Packet Capture Using Software Defined Radio*. Technical report. University of California, Los Angeles, 2009.
(Cited on pages 42, 117, 143).

[40] Bastian Bloessl, Christoph Leitner, Falko Dressler, and Christoph Sommer. "A GNU Radio-based IEEE 802.15.4 Testbed". In: *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*. Cottbus, Germany, 2013, pages 37–40.
(Cited on pages 42, 117, 121, 143).

[41] Nam-Jin Oh, Sang-Gug Lee, and Jinho Ko. "A CMOS 868/915 MHz direct conversion. ZigBee single-chip radio". English. In: *IEEE Communications Magazine* 43.12 (Dec. 2005), pages 100–109. ISSN: 0163-6804. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=1561914.
(Cited on pages 42, 116, 117).

[42] C. J. Chiang, Y. M. Gottlieb, and R. Chadha. "GNU Radio-Based Digital Communications: Computational Analysis of a GMSK Transceiver". English. In: *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*. IEEE, Dec. 2011, pages 1–6. ISBN: 978-1-4244-9268-8. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6133692.
(Cited on pages 44, 70).

[43] Thomas W. Rondeau, Timothy O'Shea, and Nathan Goergen. "Inspecting GNU radio applications with controlport and performance counters". In: *Proceedings of the second workshop on Software radio implementation forum - SRIF '13*. New York, New York, USA: ACM Press, Aug. 2013, page 65. ISBN: 9781450321815. URL: http://dl.acm.org/citation.cfm?id=2491246.2491259.
(Cited on pages 45, 78, 79).

[44] Muhammad Imran Taj. "Network on chip based multiprocessor system on chip for wireless software defined cognitive radio". PhD thesis. Sept. 2011. URL: http://www.theses.fr/2011PEST1050.
(Cited on pages 48, 51, 53, 54).

[45] Rodger H. Hosking. *Software Defined Radio Handbook*. Pentek. 2012. URL: http://www.pentek.com.
(Cited on pages 49, 50, 57).

[46] *Software Communications Architecture Specifications*. Technical report. JTRS Standard, 2006. URL: http://jtrs.spawar.navy.mil/sca.
(Cited on pages 51, 52, 53).

[47] Matthew D. Sunderland. "SOFTWARE-DEFINED RADIO INTEROPERABILITY WITH FREQUENCY HOPPING WAVEFORMS". PhD thesis. The Pennsylvania State University, 2010.
(Cited on page 51).

[48] Bruce A. Fette and Bruce Fette. "Cognitive Radio Technology (Communications Engineering)". In: (Aug. 2006). URL: http://dl.acm.org/citation.cfm?id=1211272.
(Cited on page 53).

[49] *Software Communications Architecture Specifications*. Technical report. JTRS Standard, 2010. URL: http://jtrs.spawar.navy.mil/sca.
(Cited on page 54).

[50] Jason Snyder. "OSSIE: an open source software defined radio (SDR) toolset for education and research (abstract only)." In: *SIGCSE*. Edited by Laurie A. Smith King, David R. Musicant, Tracy Camp, and Paul T. Tymann. ACM, 2012, page 672. ISBN: 978-1-4503-1098-7.
(Cited on page 54).

[51] Markus Mueck et al. "ETSI reconfigurable radio systems: Status and future directions on software defined radio and cognitive radio standards". In: *IEEE Communications Magazine* 48 (2010), pages 78–86. ISSN: 01636804.
(Cited on pages 54, 55).

[52] Kun Tan et al. "Sora". In: *Communications of the ACM* 54.1 (Jan. 2011), page 99. ISSN: 00010782. URL: http://dl.acm.org/ft_gateway.cfm?id=1866760\&type=html.
(Cited on page 56).

[53] Yuan Lin, Manjunath Kudlur, Scott Mahlke, and Trevor Mudge. "Hierarchical coarse-grained stream compilation for software defined radio". In: *Proceedings of the 2007 international conference on Compilers, architecture, and synthesis for embedded systems - CASES '07*. New York, New York, USA: ACM Press, Sept. 2007, page 115. ISBN: 9781595938268.
(Cited on page 56).

[54] *Creating High-performance SDR Architectures*. URL: http://www.microwavejournal.com/articles/7198-creating-high-performance-sdr-architectures (visited on 06/14/19).
(Cited on page 56).

[55] *CCCP: Coarse-Grained Reconfigurable Architecture*. URL: http://cccp.eecs.umich.edu/research/cgra.php (visited on 06/14/19).
(Cited on page 57).

[56] Omer Anjum et al. *State of the art baseband DSP platforms for Software Defined Radio: A survey*. 2011.
(Cited on pages 57, 59, 60, 90).

[57] Mickaël Dardaillon, Kevin Marquet, Tanguy Risset, and Antoine Scherrer. "Software Defined Radio Architecture Survey for Cognitive Testbeds". In: (Sept. 2013). arXiv: 1309.6466.
(Cited on page 57).

[58] *QuickSilver*. URL: http://www.srl-llc.com/ (visited on 06/14/27).
(Cited on page 58).

[59] *Sora: High Performance Software Radio Using General Purpose Multi-core Processors - Microsoft Research*. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=79927 (visited on 06/14/23).
(Cited on pages 58, 63).

[60] *rtl-sdr.com - A blog about software defined radio and in particular the RTL2832U RTL-SDR radio*. URL: http://www.rtl-sdr.com/ (visited on 07/14/09).
(Cited on page 58).

[61] *SDR4All | Alcatel-Lucent Chair on Flexible Radio*. URL: http://www.flexible-radio.com/sdr4all (visited on 06/14/23).
(Cited on page 58).

[62] Dake Liu, Anders Nilsson, Eric Tell, Di Wu, and Johan Eilert. "Bridging dream and reality: Programmable baseband processors for software-defined radio". In: *IEEE Communications Magazine* 47 (2009), pages 134–140. ISSN: 01636804.
(Cited on page 59).

[63] Yuan Lin et al. "SODA: A high-performance DSP architecture for software-defined radio". In: *IEEE Micro* 27 (2007), pages 114–123. ISSN: 02721732.
(Cited on page 59).

[64] *Great Scott Gadgets HackRF One*. URL: http://greatscottgadgets.com/hackrf/ (visited on 07/14/15).
(Cited on pages 62, 63).

[65] *ZeptoSDR | Nutaq*. URL: http://nutaq.com/en/products/zeptosdr (visited on 07/14/15).
(Cited on page 62).

[66] *SDR4All | Alcatel-Lucent Chair on Flexible Radio*. URL: http://www.flexible-radio.com/sdr4all (visited on 06/14/23).
(Cited on page 63).

[67] *Front | iQuadLabs, LLC*. URL: https://iquadlabs.com/index.php (visited on 06/14/23).
(Cited on page 63).

[68] *RDP-100*. URL: http://www.solenet.net/products/software-defined-radio-platform.html (visited on 06/14/23).
(Cited on page 63).

[69] *Nuand | bladeRF Software Defined Radio*. URL: http://www.nuand.com/ (visited on 06/14/23).
(Cited on page 63).

[70] *Bitshark Express RX | Epiq Solutions*. URL: http://www.epiqsolutions.com/express-rx/ (visited on 06/14/23).
(Cited on page 63).

[71] *FlexRadio Systems*. URL: http://www.flexradio.com/ (visited on 06/14/23).
(Cited on page 63).

[72] *Matchstiq | Epiq Solutions*. URL: http://www.epiqsolutions.com/matchstiq/ (visited on 06/14/23).
(Cited on page 63).

[73] *Radio définie par logiciel NI FlexRIO - National Instruments*. URL: http://sine.ni.com/nips/cds/view/p/lang/fr/nid/211407 (visited on 06/14/23).
(Cited on page 63).

[74] Eric Blossom. "GNU radio: tools for exploring the radio frequency spectrum". In: *Linux journal* 2004 (2004), page 4. ISSN: 1075-3583.
(Cited on page 66).

[75] *GNU Radio 3.7.1 C++ API: QT Graphical User Interface*. URL: http://gnuradio.org/doc/doxygen-3.7.1/page_qtgui.html (visited on 12/14/02).
(Cited on page 67).

[76] *GNU Radio Manual and C++ API Reference: Class Hierarchy*. URL: http://gnuradio.org/doc/doxygen/inherits.html (visited on 12/14/03).
(Cited on page 68).

[77] *Zynq - GNU Radio - gnuradio.org*. URL: http://gnuradio.org/redmine/projects/gnuradio/wiki/Zynq (visited on 12/14/03).
(Cited on page 70).

[78] Jesper M. Kristensen. *GNU Radio An introduction*. Technical report. 2007. (Cited on page 70).

[79] Nicholas MacCarthy Thomas W. Rondeau and Timothy O'Shea. "SIMD Programming in GNU Radio: Maintainable and User-Friendly Algorithm Optimization with VOLK". In: *SDR-WInnComm*. Edited by SDR Forum. Washington, DC, USA, 2013. (Cited on pages 71, 72).

[80] Minseok KIM. "Prototyping and Evaluation of Software Defined Radio using GNU Radio-USRP". In: *IEICE technical report* 110.153 (July 2010), pages 51–58. ISSN: 09135685. URL: http://ci.nii.ac.jp/naid/110007889866/en/. (Cited on page 75).

[81] Nguyen B. Truong, Young-Joo Suh, and Chansu Yu. "Latency Analysis in GNU Radio/USRP-Based Software Radio Platforms". English. In: *MILCOM 2013 - 2013 IEEE Military Communications Conference*. IEEE, Nov. 2013, pages 305–310. ISBN: 978-0-7695-5124-1. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6735640. (Cited on page 76).

[82] *Configure Latency in GNU Radio | Nutaq*. URL: http://nutaq.com/en/blog/configure-latency-gnu-radio (visited on 12/14/06). (Cited on page 77).

[83] M Tahir, H Mohamad, N Ramli, and S P W Jarot. "Experimental implementation of dynamic spectrum access for video transmission using USRP". In: *International Conference on Computer and Communication Engineering (ICCCE), 2012*. 2012, pages 228–233. ISBN: 9781467304795. (Cited on page 84).

[84] Dien Nguyen Van et al. "A real-time COFDM transmission system based on the GNU radio". In: *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication - ICUIMC '14*. New York, New York, USA: ACM Press, Jan. 2014, pages 1–5. ISBN: 9781450326445. URL: http://dl.acm.org/citation.cfm?id=2557977.2558050. (Cited on page 84).

[85] T S Rappaport. *Wireless Communications: Principles and Practice*. Volume 207. 2002, pages 1–707. ISBN: 0130422320. (Cited on page 85).

[86] Simon Haykin and Michael Moher. *Introduction to Analog and Digital Communications*. 2nd edition. Hoboken, NJ: Wiley, 2007. ISBN: 0471432229. URL: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0471432229.

(Cited on pages 85, 86, 95).

[87] Richard Lyons. *Understanding Digital Signal Processing (3rd Edition)*. Prentice Hall, 2010. ISBN: 0137027419.

(Cited on page 86).

[88] David R. Pauluzzi and Norman C. Beaulieu. "A comparison of snr estimation techniques for the awgn channel". In: *IEEE Transactions on Communications* 48 (2000), pages 1681–1691. ISSN: 00906778.

(Cited on pages 86, 89).

[89] *GNU Radio - Blog*. URL: http://www.trondeau.com/blog/tag/snr (visited on 09/14/09).

(Cited on pages 86, 89).

[90] Matlab. *Matlab*. Technical report. Matlab. URL: http://fr.mathworks.com/products/matlab/.

(Cited on pages 89, 93).

[91] Steven Corum, Jason D. Bonior, Robert C. Qiu, Nan Guo, and Zhen Hu. "Evaluation of phase error in a software-defined radio network testbed". English. In: *2012 Proceedings of IEEE Southeastcon*. IEEE, Mar. 2012, pages 1–4. ISBN: 978-1-4673-1375-9.

(Cited on page 90).

[92] Glenn J. Bradford and J. Nicholas Laneman. "An experimental framework for the evaluation of cooperative diversity". In: *2009 43rd Annual Conference on Information Sciences and Systems*. IEEE, Mar. 2009, pages 641–645. ISBN: 978-1-4244-2733-8. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5054797.

(Cited on page 90).

[93] *Teledyne LeCroy - Oscilloscope*. URL: http://teledynelecroy.com/oscilloscope/ (visited on 09/14/01).

(Cited on page 92).

[94] *Atem*. URL: http://www.atem.com/ (visited on 09/14/01).

(Cited on page 92).

[95] *USRP Hardware Driver and USRP Manual*. URL: http://files.ettus.com/manual/ (visited on 09/14/04).
(Cited on page 98).

[96] Analog Devices. *ADL5375 Data sheet*. Technical report. Analog Devices, 2011.
(Cited on page 98).

[97] *IEEE Standard Definitions for the Measurement of Electric Power Quantities Under Sinusoidal, Nonsinusoidal, Balanced, or Unbalanced Conditions*. 2010.
(Cited on page 100).

[98] Alexander Eigeles Emanuel. "Power definitions and the physical mechanism of power flow". In: (2010). URL: http://cds.cern.ch/record/1611482.
(Cited on page 100).

[99] *ANALOG DEVICES -RF Agile Transceiver AD9361*. URL: http://www.analog.com/static/imported-files/data_sheets/AD9361.pdf (visited on 09/14/08).
(Cited on page 103).

[100] Jose Rodriguez-Pineiro et al. "Experimental validation of ICI-Aware OFDM receivers under time-varying conditions". English. In: *2014 IEEE 8th Sensor Array and Multichannel Signal Processing Workshop (SAM)*. IEEE, June 2014, pages 341–344. ISBN: 978-1-4799-1481-4.
(Cited on page 103).

[101] Umar Saif Kay Römer Adam Dunkels Koen Langendoen Joseph Polastre Zartash Afzal Uzmi Muneeb Ali Thiemo Voigt. "Medium access control issues in sensor networks". In: ().
(Cited on page 115).

[102] Gyanendra Prasad Joshi, Seung Yeob Nam, and Sung Won Kim. *Cognitive radio wireless sensor networks: applications, challenges and research trends.* Volume 13. 2013, pages 11196–228. ISBN: 8253810474.
(Cited on pages 115, 142).

[103] Chipcon. "Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee(TM) Ready RF Transceiver". In: (2006).
(Cited on page 116).

[104] *FrontPage - The Wireshark Wiki*. URL: https://wiki.wireshark.org/ (visited on 04/15/07).

(Cited on pages 117, 132).

[105] A. Dunkels, B. Gronvall, and T. Voigt. "Contiki - a lightweight and flexible operating system for tiny networked sensors". English. In: *29th Annual IEEE International Conference on Local Computer Networks*. IEEE (Comput. Soc.), pages 455–462. ISBN: 0-7695-2260-2. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=1367266.

(Cited on page 117).

[106] *900 MHz versus 2.4 GHz in long distance radio links -www.afar.net*. URL: http://www.afar.net/tutorials/900-mhz-versus-2.4-ghz/ (visited on 06/29/2015).

(Cited on page 117).

[107] J Sabater, J M Gomez, and M Lopez. "Towards an IEEE 802.15.4 SDR transceiver". In: *Icecs*. Ieee, 2010, pages 323–326.

(Cited on page 117).

[108] *IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*. 2011.

(Cited on page 118).

[109] G.R. Danesfahani and T.G. Jeans. *Optimisation of modified Mueller and Müller algorithm*. 1995.

(Cited on pages 122, 129).

[110] M. Simon and W. Lindsey. "Optimum Performance of Suppressed Carrier Receivers with Costas Loop Tracking". In: *IEEE Transactions on Communications* 25.2 (1977). ISSN: 0090-6778.

(Cited on page 129).

[111] *AT86RF230*. URL: http://www.atmel.com/devices/at86rf230.aspx (visited on 04/15/07).

(Cited on page 132).

[112] *TelosB Data Sheet*. http://www.sentilla.com/files/pdf/eol/tmoteskydatasheet. pdf. Sentilla, 2010.

(Cited on page 132).

[113] *Contiki: The Open Source Operating System for the Internet of Things*. URL: http://www.contiki-os.org/ (visited on 04/15/09).
(Cited on page 132).

[114] O. Akan, O. Karli, and O. Ergul. "Cognitive radio sensor networks". In: *IEEE Network* 23 (2009). ISSN: 0890-8044.
(Cited on page 142).

[115] Qing Zhao and B M Sadler. "A Survey of Dynamic Spectrum Access". In: *Signal Processing Magazine, IEEE* 24 (2007), pages 79–89. ISSN: 1053-5888.
(Cited on page 143).

[116] Tevfik Yucek and Huseyin Arslan. "A survey of spectrum sensing algorithms for cognitive radio applications". English. In: *IEEE Communications Surveys & Tutorials* 11.1 (2009), pages 116–130. ISSN: 1553-877X. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=4796930.
(Cited on page 143).

[117] H. Ebeid T. O'Shea, T. Clancy. "Practical Signal Detection and Classification in GNUradio". In: *SDR Forum Technical Conference*. 2007.
(Cited on page 143).

[118] Mutsawashe Gahadza, Minseok Kim, and Jun-ichi Takada. *Implementation of a Channel Sounder using GNU Radio Opensource SDR Platform*. Technical report. IEICE, 2009.
(Cited on page 143).

[119] Rozeha A. Rashid et al. "Spectrum Sensing Measurement using GNU Radio and USRP Software Radio Platform". In: *ICWMC 2011, The Seventh International Conference on Wireless and Mobile Communications*. June 2011, pages 237–242. ISBN: 978-1-61208-140-3. URL: http://www.thinkmind.org/index.php?view=article\&articleid=icwmc_2011_11_20_20268.
(Cited on page 143).

[120] Atmel. *ZigBit™ 700/800/900 MHz Wireless Modules*. Technical report. Atmel. URL: http://www.atmel.com/Images/doc8227.pdf.
(Cited on page 156).

[121] Evgeny Khorov, Andrey Lyakhov, Alexander Krotov, and Andrey Guschin. "A survey on IEEE 802.11ah: An enabling networking technology for smart cities". In: *Computer Communications* 58 (Mar. 2015), pages 53–69. ISSN: 01403664. URL: http://www.sciencedirect.com/science/article/pii/S0140366414002989.

(Cited on page 157).