Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 16(1): 23-30, Mar. 2018 Regular paper

Implementation of Spectrum-Sensing for Cognitive Radio Using USRP with GNU Radio and a Cloud Server

Huynh Thanh Thien, Rene Tendeng, Hiep Vu-Van, and Insoo Koo[*], *Member*, *KIICE*

School of Electrical Engineering, University of Ulsan, Ulsan 44610, Korea

Abstract

In cognitive radio (CR), spectrum sensing is an essential function since secondary users (SUs) must determine whether the primary user (PU) is utilizing the channel or not, and furthermore, SUs opportunistically access the licensed channel when the PU is absent. In this paper, spectrum sensing is implemented by energy detection, and a software-defined radio testbed is built to evaluate sensing performance by energy detection in a real environment. In particular, the testbed was built based on the GNU's Not Unix (GNU) Radio software platform and Universal Software Radio Peripheral National Instruments 2900 devices. More specifically, a new block of energy detection is developed by using an out-of-tree module from GNU Radio. To successfully integrate CR into the cloud computing paradigm, we also implement cloud computing-based spectrum sensing by utilizing a cloud server with ThingSpeak, such that we can store, process, and share the sensing information more efficiently in a centralized way in the cloud server.

Index Terms: Cognitive radio, GNU radio, Spectrum sensing, ThingSpeak, USRP

I. INTRODUCTION

Wireless communications systems have multiplied significantly over the last two decades. However, the radio spectrum used for wireless communications is a finite resource. Therefore, it is necessary to find other systems that can use spectrum efficiently. Currently, technologists are interested in cognitive radio (CR) because with CR overall efficiency of spectrum utilization can increase significantly. CR enables the usage of temporarily unused frequency bands, commonly known as spectrum holes [1]. Spectrum holes are generally categorized into temporal and spatial spectrum holes. A temporal spectrum hole, which is unoccupied by the primary user (PU) during certain times, can be used by secondary user (SUs) in the unused time slots. A spatial spectrum hole is a band that is unoccupied by the PU in some geographic areas. Therefore, it can be used by SUs [2–4].

In CR, spectrum sensing is essential, since SUs must determine whether PUs are utilizing the channel or not, and furthermore, SUs opportunistically access the licensed channel when the PU is absent. So far, various approaches have been proposed for spectrum sensing, such as the matched filter, energy detection (ED), feature detection, and more recently, wavelet detection [5]. In these methods, ED uses the energy of the received signal at the SU to determine the presence of signals from the PU. This simple method can gather spectrum occupancy information quickly. It does not require prior information about the behavior (modulation) of the received signal or excessive analog-to-digital converter (ADC) and signal-processing capabilities of the matched filter and cyclostationary detector, respectively. Besides, spectrum sensing via ED in CR is included in the IEEE 802.22 standard. Hence, an energy detector is an optimal choice when information about the channel is not available.

Received 16 October 2017, **Revised** 21 March 2018, **Accepted** 22 March 2018
***Corresponding Author** Insoo Koo (E-mail: iskoo@ulsan.ac.kr, Tel: +82-52-259-1249)
School of Electrical Engineering, University of Ulsan, 93 Daehak-ro, Nam-gu, Ulsan 44610, Korea.

Open Access https://doi.org/10.6109/jicce.2018.16.1.23 print ISSN: 2234-8255 online ISSN: 2234-8883

In this paper, we implement a CR system by using Universal Software Radio Peripheral (USRP) National Instruments (NI) 2900 devices, and we evaluate the sensing performance of energy detection in a real environment. The contribution of this paper is measuring sensing parameters, such as the sensing threshold, probability of detection, and probability of false alarm, in a real environment. To successfully integrate CRs into the cloud computing paradigm, we also implement cloud computing-based spectrum sensing by utilizing a cloud server with ThingSpeak, such that we can store, process, and share the sensed information more efficiently in a centralized way. In some cases, users do not have the ability to sense the channel, or users may need sensing information for performance analysis. To do this, the sensing information should be saved where everyone can easily access spectrum sensing results to further utilize them. The cloud server is the solution to this problem. A cloud server allows users to store data online, which they can access from any location via the internet. So, the sensed information can be shared with other users. From that, users can use the sensed information that is uploaded to the cloud server to estimate the statistics of the primary network, such as the probability of absence and the transition probability for the PU. Then, users can analyze and improve the performance of local sensing. Besides that, we can construct a radio environment map (REM) for the considered area. The REM acts as a cognition engine by building long-term information via processing measurements collected from the local node to estimate the state of locations without any measurement data. Due to the dynamics of the primary network, if the time for delivering sensing information to the cloud server involves a long delay, the immediate sensed information/global decision may be meaningless. Therefore, in order to utilize the sensed information/global decisions that are shared in the cloud, the delay time should be very short in comparison with the whole time slot. Because of the limit in the time delay until sensing information is delivered to the cloud server, in this paper, we mainly focus on ways to implement local spectrum sensing on the user side to upload to, and share the local spectrum sensing results from, the cloud server and to estimate the performance of local sensing based on absence and transition probabilities.

This paper is organized as follows. Section II presents related works. In Section III, we describe the spectrum sensing model based on ED, hardware, and software in the implemented CR system, and the sensing performance of the implemented CR system in terms of detection and false alarm probabilities. In Section IV, we provide a conclusion.

II. RELATED WORKS

In cognitive radio, spectrum-sensing based on ED has been proposed and widely used because it does not require transmitted signal properties, channel information, or even the type of modulation. Abdulsattar and Hussein [1] presented a survey of energy detectors over additive white Gaussian noise (AWGN) on different fading channels for spectrum-sensing methodologies in CR. Theoretical analysis of time domain ED and threshold setting was investigated.

Ma et al. [2] described the fundamental signal-processing aspects involved in developing a fully functional CR network, including basic techniques of spectrum-sensing such as energy, cyclostationary, pilot-based coherent, covariance-based, and wavelet-based detection, that can effectively improve the overall detection capability of a CR network. Besides that, spectrum sculpting was also discussed for avoiding interference with the PUs.

Cabric et al. [5] explored the new field of CR with a special emphasis on one unique aspect: spectrum sensing. Three digital signal processing techniques (matched filtering, ED, and cyclostationary feature detection) were investigated, which identified two key issues related to the CR front end: dynamic range reduction and wideband frequency agility.

Sowmiya and Sangeetha [6] analyzed a CR system based on ED with modulation by using USRP NI 2920 devices. In that experiment, the quadrature phase-shift keying (QPSK) modulated signal was received and the energy analyzed to detect the presence of PUs.

Sensing performance in a real-time testbed of a GNU's Not Unix (GNU) Radio and USRP software-defined radio (SDR) communications platform was implemented [7]. Rashid et al. [7] investigated two main performance metrics for spectrum-sensing: probability of false alarm (P_{fa}) and probability of detection (P_d). P_{fa} is used to determine the threshold of the CR dynamic spectrum access (DSA) system, while P_d is used to determine how many samples are needed by CR DSA to meet the desired performance.

The ThingSpeak application programming interface (API) is an open source interface that listens to incoming data, timestamps it, and outputs it for both human users (through visual graphs) and machines (through easily parseable code) [8]. The cloud brings everything together and allows us to interact with things and, even more interestingly, allows things to interact with other things. For connection to the cloud server, we focus on the ThingSpeak API. The interface for ThingSpeak capabilities communicates with objects, as well as interesting additional applications. Moreover, ThingSpeak allows us to collect data using sensors and display them in graphs. It offers near-real-time data collection, data processing, and simple visualizations for users. Data are stored in so-called channels, which provide the user with a list of features [9]. Each channel can store up to eight fields of data, using up to 255 alphanumeric characters each. All incoming data are time- and date-stamped, and receive sequential IDs. When a channel has been created, data can

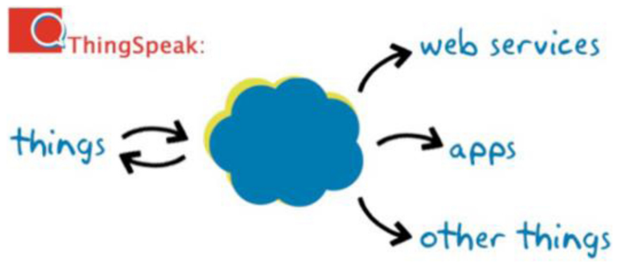


Fig. 1. ThingSpeak with a cloud interface.

be published by accessing the ThingSpeak API with a *write key*. Consequently, a *read key* is used to access channel data when it is set to keep the data private (the default setting). Channels can change to public status, in which case no read key is required. The first steps to setting up ThingSpeak are always the same. After signing up for a new user account, we can log in and create new channels [10]. After that, we can create a new channel by selecting Channels > My Channels and then Create New Channel. The channel has a unique identifier key that is used to identify the channel when reading or uploading data.

According to the ThingSpeak website, the API works as noted in Fig. 1 [8]. *Things* are objects, such as data, that are collected from sensors. Data are sent and received via simple Hypertext Transfer Protocol (HTTP) POSTs, much like going to a web page and filling out a form. This communication is implemented through plain text, JavaScript Object Notation (JSON) or Extensible Markup Language (XML). The data are then uploaded to the cloud and, from there, can be used for a variety of purposes.

III. SYSTEM DESIGN

In this work, we implement cloud-based spectrum sensing where a USRP NI 2900 device with GNU is used to collect sensing information and make local sensing decisions for local/immediate usage. The raw sensing information and the temp-sensing decision will be uploaded to the cloud. After that, the cloud will process the received information to make a global decision in order to share it with in-demand users on the network.

Energy detection is used for spectrum sensing in the USRP device, in which two hypotheses about the PU signal—absence (H_0) and presence (H_1)—are considered [6]. The received signal, $x(n)$, can be expressed as:

$$x(n) = \begin{cases} w(n) & H_0 \\ s(n)+w(n) & H_1 \end{cases} \quad (1)$$

where $n = 1, \ldots, N$, and N is the number of samples; $w(n)$ is AWGN with zero mean and variance σ_w^2, and $s(n)$ is the transmitted signal.

Based on received signal $x(n)$, the total power over the N collected samples can be given as $P_N = \sum_{k=0}^{N-1} |x(n)|^2$. Furthermore, the presence of the PU signal can be determined as:

$$D_N = \begin{cases} H_0, & if\ P_N \le \gamma \\ H_1, & otherwise\ P_N > \gamma \end{cases} \quad (2)$$

The performance of the energy detector is characterized by two metrics: probability of detection, P_d, which occurs when decision D_N is H_1, and the PU signal is practically present (H_1); and probability of false alarm, P_{fa}, which corresponds to the decision H_1 but the PU signal is absent (H_0) [6].

$$P_{fa} = P(P_N > \gamma | H_0) \quad (3)$$

$$P_d = P(P_N > \gamma | H_1) \quad (4)$$

If the noise term is assumed to be a circularly symmetric complex Gaussian, using Gaussian distribution approximation for the probability density function of P_N, it can be derived from (3) and (4) [11]:

$$P_{fa} = Q\left(\frac{\gamma - 2N}{\sqrt{4N}}\right) \quad (5)$$

$$P_\mathrm{d} = Q\left(\frac{\gamma - 2N(SNR+1)}{\sqrt{4N(2SNR+1)}}\right) \quad (6)$$

where SNR and $Q(\ )$ represent signal-to-noise ratio and the Q-function, respectively. The challenge in local spectrum-sensing is to reliably decide on the two hypotheses to achieve high P_d for good protection of the PU, and low P_{fa} to provide satisfactory access for SUs.

A. Hardware and Software

Performance evaluation of the energy detector was conducted by Ettus Research with experiments realized by means of USRP boards. The USRP platform is a low-cost and high-quality realization of SDR. It allows users various functionalities to achieve efficient, real-time realization of very complicated wireless systems that operate in the radio frequency (RF) band. The USRP platform converts the digital baseband signal delivered from the computer to an analog signal in the RF band. USRP includes two main boards: the daughterboard and the motherboard [10]. A programmable USB 2.0 controller communicates between USRP and GNU Radio and a field programmable gate array (FPGA) for implementing four digital down converters (DDCs) and high-rate signal processing. In our experiments, two USRP boards were used: the PU signal was a generated signal for the first board, whereas the second was used for spectrum-sensing purposes, and acted as the SU. The USRP NI 2900 device is an RF transceiver capable of transmitting and receiving signals from 70 MHz to 6 GHz, which means that many applications can be available by utilizing the frequency band from FM radio through to Wi-Fi bands. The USRP NI 2900 has two connectors: RX2 (input terminal for the RF signal) and TX1 RX1 (input and output terminals for the RF signal). These connectors go to an RF switch on board, which allows transmit and receive operations to occur on the same shared antenna.

Unlike other off-the-shelf options, the USRP family provides a complete and versatile solution with software that accelerates development. All software processing was realized in the open-source GNU Radio environment [12]. These libraries, together with the appropriate drivers for manipulating the USRP boards and graphical programming environment, allow efficient and accurate implementation of the selected spectrum-sensing algorithms. All signal processing was realized in the GNU Radio environment with the graphical tool called GNU Radio Companion (GRC), where the whole system is built from blocks.

B. Energy Detection Implementation

A schematic diagram and dedicated photographs of the experiment are shown in Figs. 2 and 3, respectively. The experimental system consists of two USRP devices, one laptop, and one PC.

The laptop with USRP A acts as the transmitter while the PC with USRP B acts as the receiver. There is interference at neighboring access points, which can affect the USRP frequencies. Therefore, the USRP center frequency was set at 2.48 GHz to avoid interference and jamming with the said access point operating in the 2.4 GHz band. The energy detector was implemented using 1024-point fast Fourier transform (FFT). Each block of FFT output was averaged and stored inside a buffer. N averaged FFT blocks were collected and then averaged again to acquire the final result used to make the decision on the presence of the PU.

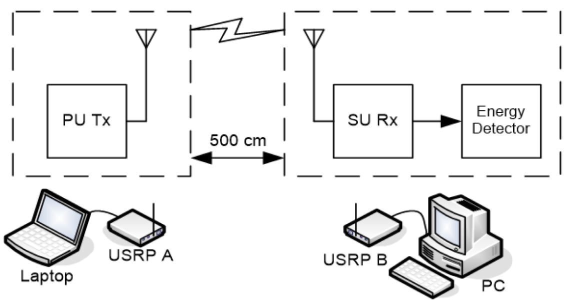


Fig. 2. Schematic diagram of the system.

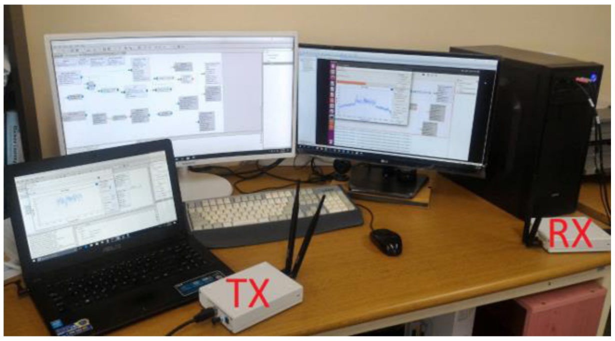
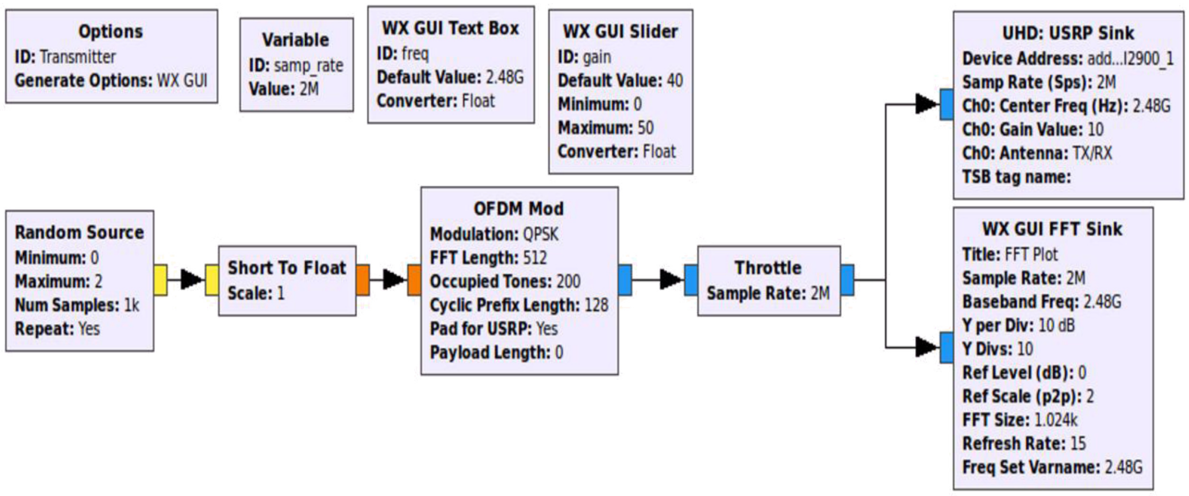


Fig. 3. PU transmitter and SU receiver with USRP hardware and GNU Radio software.

On the transmitter side, the spectrum of the multicarrier signal was tested in this research work—the orthogonal frequency division multiplexing (OFDM) symbol with N_{OFDM} = 512 subcarriers. The PU transmitter side with OFDM modulation in the GRC is shown in Fig. 4. The signal source block (*Random Source*) generated repeatedly random data, which were mapped to QPSK symbols and were then subject to the OFDM modulation block (realized in *OFDM Mod*). Finally, after power was adjusted properly, the signal was sent to the local spectrum analyzer block (*FFT Sink*) and to the USRP block (*USRP Sink*) responsible for sending data. Notice that the complex sampling frequency is set to 2 MHz, the antenna gain is set to 40 dB, and the center frequency is set to 2.48 GHz. Analogous to the PU transmitter, the SU receiver was also realized in the GNU Radio environment. A schematic diagram of the SU receiver is shown in Fig. 5. The parameters at the receiver were also set to the same center frequency, sample rate, gain, and FFT size. In addition, the receiver included parameters like the given P_{fa} (0.05) and the number of sensing samples (300) for local spectrum sensing. Data from the RF spectrum were collected by the *USRP Source* block. It operates at the center frequency equal to 2.48 GHz. Afterwards, the signal goes through chains with ED. We can observe the received signal through the *FFT sink* block.

Next, we focus on the ED algorithm. The signal is transformed to the frequency domain by the *FFT* block. After that, these data can be analyzed and stored in the *Detect upload* block, which has two main functions: (i) processing and storing data into a data file, and then, performance of the ED algorithm is estimated by using MATLAB; and (ii) providing a connection between the GNU system and ThingSpeak to upload P_d to ThingSpeak.

The *Detect upload* block not only implements the ED algorithm but also connects the ED system and ThingSpeak. This block will use a *write API key* (“VLZAO7DVD8Z9AKEN”)



Options
ID: Transmitter
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 2M

WX GUI Text Box
ID: freq
Default Value: 2.48G
Converter: Float

WX GUI Slider
ID: gain
Default Value: 40
Minimum: 0
Maximum: 50
Converter: Float

Random Source
Minimum: 0
Maximum: 2
Num Samples: 1k
Repeat: Yes

Short To Float
Scale: 1

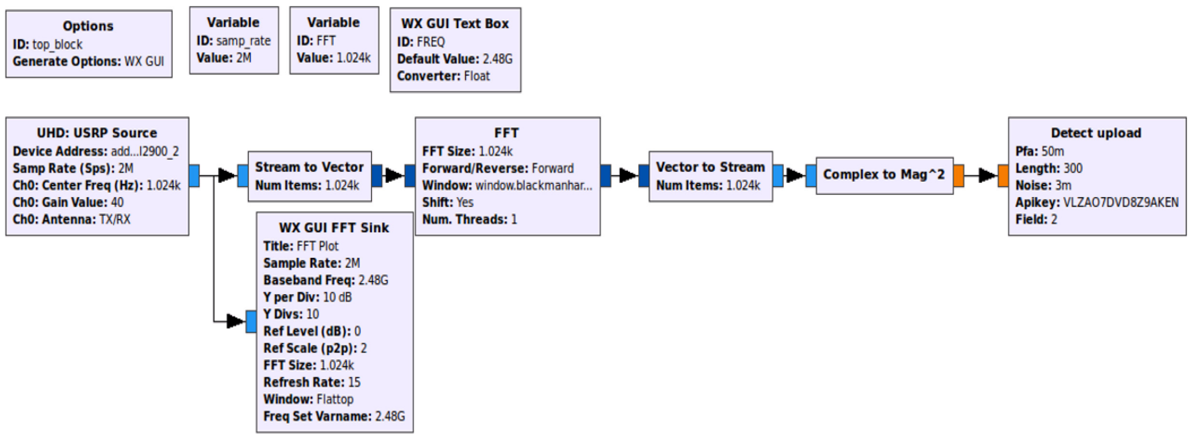
OFDM Mod
Modulation: QPSK
FFT Length: 512
Occupied Tones: 200
Cyclic Prefix Length: 128
Pad for USRP: Yes
Payload Length: 0

Throttle
Sample Rate: 2M

UHD: USRP Sink
Device Address: add...l2900_1
Samp Rate (Sps): 2M
Ch0: Center Freq (Hz): 2.48G
Ch0: Gain Value: 10
Ch0: Antenna: TX/RX
TSB tag name:

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 2M
Baseband Freq: 2.48G
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Freq Set Varname: 2.48G

Fig. 4. The PU transmitter with OFDM modulation in the GNU Radio Companion (GRC).



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 2M

Variable
ID: FFT
Value: 1.024k

WX GUI Text Box
ID: FREQ
Default Value: 2.48G
Converter: Float

UHD: USRP Source
Device Address: add...l2900_2
Samp Rate (Sps): 2M
Ch0: Center Freq (Hz): 1.024k
Ch0: Gain Value: 40
Ch0: Antenna: TX/RX

Stream to Vector
Num Items: 1.024k

FFT
FFT Size: 1.024k
Forward/Reverse: Forward
Window: window.blackmanhar...
Shift: Yes
Num. Threads: 1

Vector to Stream
Num Items: 1.024k

Complex to Mag^2

Detect upload
Pfa: 50m
Length: 300
Noise: 3m
Apikey: VLZAO7DVD8Z9AKEN
Field: 2

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 2M
Baseband Freq: 2.48G
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Window: Flattop
Freq Set Varname: 2.48G

Fig. 5. The SU receiver with energy detection and data uploading function blocks in the GNU Radio Companion (GRC).

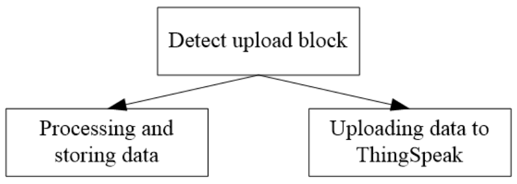


Fig. 6. The *Detect upload* function block.

to connect to ThingSpeak. All the collected data are stored in text files, which are integrated into the *Detect upload* block and will be uploaded to ThingSpeak. Transmitter and receiver parameters are shown in Table 1. The *Detect upload* block is shown in Fig. 6.

C. Experiment Results

We tested a QPSK signal centered on the 2.48 GHz carrier frequency. From the experiment, power transmission is shown in Fig. 7. On the receiver side, the sensing information, when there is no signal transmission from the PU, is shown in Fig. 8. Fig. 9 shows the SU signal on the receiver side when the PU transmitter utilizes the channel. A thresh-

Table 1. Parameters for the transmitter and receiver

Parameter	Transmitter	Receiver
Center frequency (GHz)	2.48	2.48
Sample rate (MHz)	2	2
Gain (dB)	40	40
FFT	1024	1024
Number of sensing samples		300
Given probability of false alarm		0.05
Noise (dB)		0.003
Write API key		VLZAO7DVD8Z9AKEN

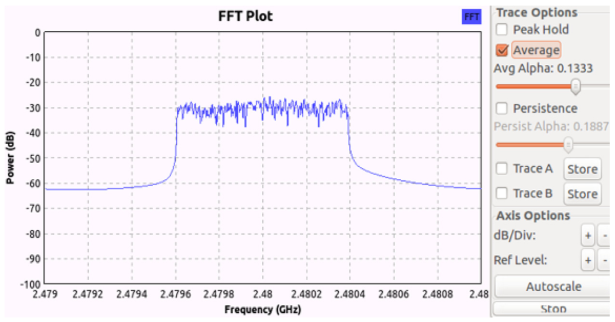


Fig. 7. Transmitted power at the PU transmitter.

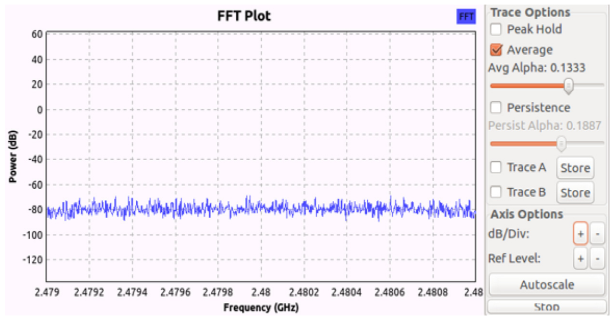


Fig. 8. The received signal power at center frequency f_c = 2.48 GHz when the signal of the PU is absent.

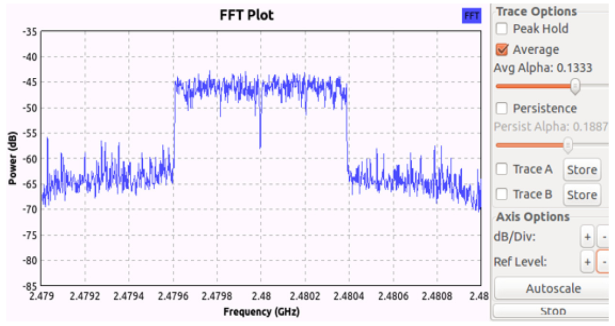


Fig. 9. The received signal power at center frequency f_c = 2.48 GHz when the signal of the PU is present.

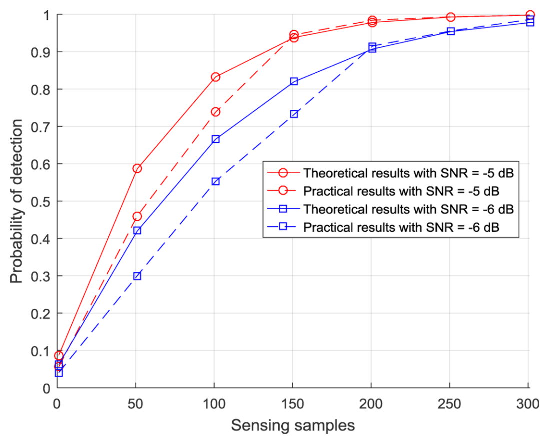


Fig. 10. The probability of detection according to the number of sensing samples when P_{fa} = 0.05.

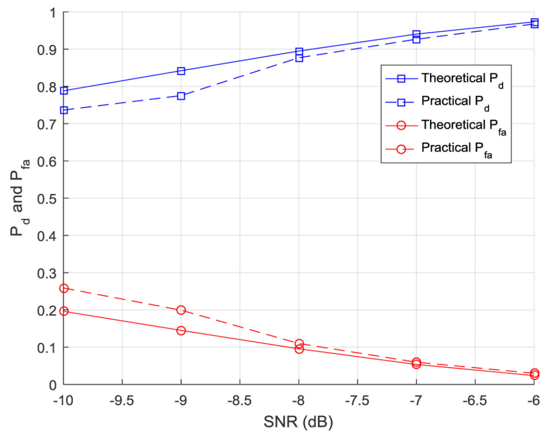


Fig. 11. The probabilities of detection and false alarm based on SNR.

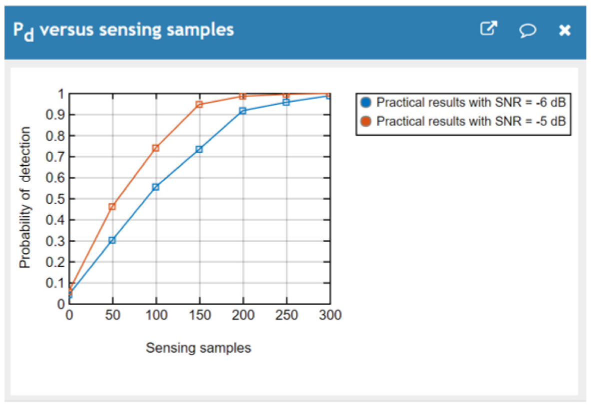


Fig. 12. The probability of detection uploaded via ThingSpeak to the cloud. From top to bottom: the first line is for SNR = -5 dB and the second line is for SNR = -6 dB.

old is calculated based on Eq. (5) and is used for the ED algorithm.

In GNU Radio, we can observe received signals based on the *FFT sink* block. Moreover, the processed signal at the SU is also stored in a data file that is integrated into the *Detect upload* block, and then, we used this data file for analysis in MATLAB. Fig. 10 shows the probability of detection according to the number of sensing samples. It shows that if we use more samples, the performance of the energy detector is better. Fig. 10 also shows that practical and theoretical results are almost similar. Fig. 11 shows the probabilities of detection and false alarm according to SNR. When SNR is higher, the performance in probability of detection and probability of false alarm improves more.

In the next step, using ThingSpeak, we integrated data uploading assigned to the *Detect upload* block with ED. As a

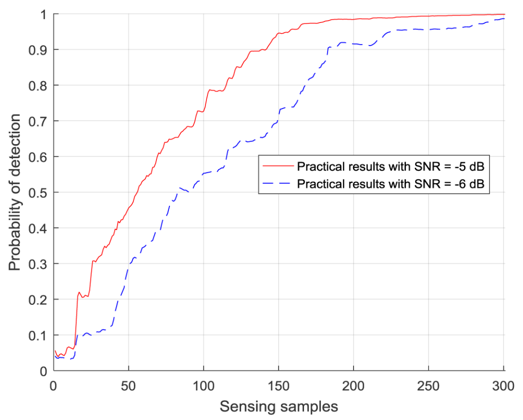


Fig. 13. The spectrum sensing data received from the cloud by the end user.

result of the experiment, the probability of detection versus the number of sensing samples was uploaded to ThingSpeak for saving and sharing via the cloud, as shown in Fig. 12. Data that can be downloaded from the cloud by the end user are shown in Fig. 13.

IV. CONCLUSION

In this paper, we implemented a spectrum-sensing system with ED in a USRP NI 2900 and GNU Radio environment. Two main performance metrics for spectrum sensing were measured in real environments: the probability of false alarm (P_{fa}) and the probability of detection (P_d). P_{fa} is used to determine the sensing threshold, whereas P_d is used to determine the desired performance. We observed that ED-based spectrum sensing can achieve higher performance when the number of sensing samples is larger. In addition, the results showed that evaluation of the probability of detection and the probability of false alarm can improve more when SNR is higher. Besides, in this paper, ThingSpeak was also utilized to upload data to the cloud, which is measured in a local CR node, such that we can store, process, and share the sensed information more efficiently in a centralized way.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (Grant No. NRF-2015R1D1A1A09057077, 2016K2A9A1A01950711, and 2018R1A2B6001714).

REFERENCES

[1] M. A. Abdulsattar and Z. A. Hussein, “Energy detection technique for spectrum sensing in cognitive radio: a survey,” *International Journal of Computer Networks & Communications*, vol. 4, no. 5, pp. 223–242, 2012.

[2] J. Ma, G. Y. Li, and B. H. Juang, “Signal processing in cognitive radio,” *Proceedings of the IEEE*, vol. 97, no. 5, pp. 805–823, 2009. DOI: 10.1109/JPROC.2009.2015707.

[3] M. B. H. Weiss, “Spatio-temporal spectrum holes and the secondary user,” in *Proceedings of IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Aachen, Germany, pp. 216–222, 2011. DOI: 10.1109/DYSPAN.2011.5936209.

[4] J. C. Na, “Optimization in cooperative spectrum sensing,” *Asia-Pacific Journal of Convergent Research Interchange*, vol. 3, no. 1, pp. 17–27, 2017. DOI: 10.21742/APJCRI.2017.03.02.

[5] D. Cabric, S. M. Mishra, and R. W. Brodersen, “Implementation issues in spectrum sensing for cognitive radios,” in *Proceedings of Conference Record of the 38th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, pp. 772–776, 2004. DOI: 10.1109/ACSSC.2004.1399240.

[6] M. Sowmiya and M. Sangeetha, “Energy detection using NI USRP 2920,” *International Journal of Science and Research*, vol. 5, no. 5, pp. 597–603, 2006. DOI: 10.21275/v5i5.nov163414.

[7] R. A. Rashid, M. A. Sarijari, N. Fisal, A. Lo, S. Yusof, S. Kamilah, and N. H. Mahalin, “Spectrum sensing measurement using GNU Radio and USRP software radio platform,” in *Proceedings of the 7th International Conference on Wireless and Mobile Communications*, Luxembourg, pp. 237–242, 2011.

[8] M. A. G. Maureira, D. Oldenhof, and L. Teernstra, “ThingSpeak: an API and web service for the Internet of Things,” 2014 [Internet], Available: http://mediatechnology.leiden.edu/images/uploads/docs/wt2014_thingspeak.pdf.

[9] ThingSpeak Community, “Introduction to the “Internet of Things” and ThingSpeak,” [Internet], Available: https://community.thingspeak.com/tutorials/introduction-to-the-%E2%80%9Cinternet-of-things%E2%80%9D-and-thingspeak/.

[10] ThingSpeak Community, “ThingSpeak Channels,” [Internet], Available: http://community.thingspeak.com/tutorials/thingspeak-channels/.

[11] H. V. Poor, *An Introduction to Signal Detection and Estimation*, 2nd ed. New York, NY: Springer-Verlag, 1994.

[12] GNU Radio: the free and open software radio ecosystem [Internet], Available: http://gnuradio.org/.

Huynh Thanh Thien
received a B.Eng. in electronics and telecommunications engineering from Ton Duc Thang University, Vietnam, in 2012, and an M.Sc. in mechatronics technology from the University of Chinese Culture University, Taiwan, in 2014. He is currently a Ph.D. student at the University of Ulsan, Korea. His current research interests include cognitive radio and next-generation wireless communications systems.

Rene Tendeng
received a B.Eng. in telecommunications and networks engineering from Universite Libre de Dakar, Senegal, in 2009. He is currently a Master's student with the University of Ulsan, Korea. His current research interests include cognitive radio and next-generation wireless communications systems.

Hiep Vu-Van
received a B.Eng. in electronics and telecommunications engineering from Ton Duc Thang University, Vietnam, in 2005, a Bachelor of Business Administration from the University of Economy, Ho Chi Minh City, Vietnam, in 2007, and a Ph.D. in electrical engineering from the University of Ulsan, Korea, in 2013. He is currently a Research Fellow with the University of Ulsan. His current research interests include cognitive radio and next-generation wireless communications systems.

Insoo Koo
received his B.Eng. from Konkuk University, Seoul, Korea, in 1996, and his M.Sc. and Ph.D. from Gwangju Institute of Science and Technology (GIST), Gwangju, Korea, in 1998 and 2002, respectively. From 2002 to 2004, he worked with the Ultrafast Fiber-Optic Networks (UFON) Research Center at GIST as a research professor. For one year from September 2003, he was a visiting scholar at the Royal Institute of Science and Technology, Sweden. In 2005, he joined the University of Ulsan, where he is now a full professor. His research interests include next-generation wireless communications systems and wireless sensor networks.