JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 23, No. 12: 1650~1658, Dec. 2019

소프트웨어 정의 라디오 기반 스펙트럼 센싱 시스템 설계를 위한 단일 보드 컴퓨터 내 연산 분석 및 측정 연구

김준영*

Computational Analysis and Measurement for SDR-based Spectrum Sensing System Design on Single Board Computer

Joon Young Kim*

*Senior Research Engineer, Hyundai Motor Company, Uiwang, 16082 Korea

요 약

최근 IoT 기기 및 플랫폼들의 발전 및 확장과 더불어 IoT 관련 기기 내 연산 성능도 지속해서 향상하고 있다. 그러나 기기 향상과는 별개로 IoT 기기, 특히 소형 단일 보드 컴퓨터의 제한적인 크기 및 연산 자원은 해당 기기 내 통신 시스템 구현 설계를 위한 중요 고려사항 중 하나이다. 현재 다양한 무선 통신 시스템을 활용할 수 있게끔 소프트웨어 정의 라디오 (SDR) 기술을 IoT 기기에 적용 시 소형 단일 보드 컴퓨터의 하드웨어 제한 사항으로 인한 열화 가능성으로 인하여 실제 시스템의 원활한 적용을 위해 해당 컴퓨터 성능에 대한 분석 및 조사가 필요하다. 본 논문에서는 소형 단일 보드 컴퓨터 내 SDR 적용 스펙트럼 센싱 시스템 디자인을 위한 시스템 연산 분석 및 실험을 진행한다. 먼저 단일 보드 컴퓨터를 위한 SDR 기반 스펙트럼 센싱 시스템을 설계하고 시스템 성능에 영향을 줄 수 있는 다양한 요소를 실험을 통해 조사하며. 이를 통한 중요 고려사항 및 디자인 가이드 절차를 도출한다.

ABSTRACT

In recent years, IoT device and platform become widely popular and the computing performance and capabilities of IoT devices are also getting improved. However, the size and computing resources of IoT devices, especially small single board computer, are limited in a way that the design and implementation of the system should be carefully considered to operate on the devices. Recently, SDR technologies are adapting in IoT devices and can perform various radio systems. Thorough analysis and investigation of computer performances on small single board computer are necessary for its usage. In this paper, we present the results of computing resources measurement and analysis on small single-board computers. At first, we consider to design SDR based spectrum sensing for single board computer, investigate various key factors and propose a design procedure that can affect performance of the system with experiments.

키워드 : 소프트웨어 정의 라디오, 중앙처리 유닛, 프로세서, 라즈베리 파이, 시스템

Keywords : SDR, CPU, Processor, RPI, System

Received 28 September 2019, **Revised** 17 October 2019, **Accepted** 12 November 2019

*** Corresponding Author** Joon Young Kim (E-mail:jkim@hyundai.com, Tel:+82-31-596-9904)
Senior Research Engineer, Hyundai Motor Company, Uiwang, 16082, Korea

Open Access http://doi.org/10.6109/jkiice.2019.23.12.1650 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

4차 산업 시대를 맞아 다양한 기기의 연결성 및 끊김 없는 서비스 제공이 요구되는 시점에서 다양한 사물인터넷 (Internet of Things: IoT) 기기 및 관련 플랫폼들에 대한 수요가 증가하고 있다. 앞으로 지속해서 비즈니스 IoT 기술 적용 및 IoT 연결 기기가 증가할 것으로 보고 있으며 [1] 국내 제조사, ICT 기업과 통신사들도 IoT 관련 플랫폼 및 연관 생태계 구축을 위한 오픈 플랫폼 출시를 진행하고 있다. 삼성의 경우 2015년도에 인수한 스마트싱스 (SmartThings) 플랫폼을 확장하여 다양한 연결성을 아우르는 플랫폼으로 탈바꿈시켰으며 현재 전용 허브를 포함한 모든 기기의 연결도 스마트싱스 플랫폼으로 집중하고 있으며, LG도 ThinQ Home이라는 IoT 플랫폼으로 기기 연동성을 확장해 가는 중이다. [2] 현재 ICT 기업과 통신사 경우는 개별적으로 구축된 인공지능 플랫폼과의 연동도 같이 진행 중이다. 네이버 클로버(Clova), 카카오 아이 (Kakao i), SKT 누구 플랫폼 (NUGU), KT 기가지니 플랫폼 (Gigagenie)가 대표적인 예라고 할 수 있다.

이러한 다양하게 확장되어가고 있는 플랫폼과의 연동을 위한 IoT 기기들의 연결성도 다양화되는 추세이다. SmartThings도 기존 hub 연결을 위한 무선표준인 Zigbee 혹은 Z-wave뿐만이 아닌 WiFi 및 Bluetooth 기기까지도 연동 지원하고 있으며 [3] 각 음성 스피커 제조사도 WiFi뿐만이 아닌 블루투스까지 확장하고 있다.

위와 같이 다양한 무선표준을 유연하게 적용해줄 수 있는 모뎀 구성이 필요하다. 현재 기기들의 경우 하드웨어 칩으로 개별 최적화되어 있는 칩 스택을 쓰는 문제로 인하여 채널 변경 및 재전송 등 제한적인 변화만 적용할 수 있으며 하드웨어 고정문제로 인해 실제 역동적인 무선 통신 적용을 위한 내부 모듈 변경 자체가 어려운 경우가 대부분이다. 이를 보완하기 위한 소프트웨어로 모뎀 내부 구성을 바꿀 수 있는 소프트웨어 정의 라디오 (Software Defined Radio: SDR)는 역동적인 환경에서 모뎀 내부 구성을 변경할 수 있게 함으로써 내부 주요 요소 등의 변경에 대한 유연성을 가지게 할 수 있다. [4] 이러한 유연성을 기반으로 한 SDR 기술을 IoT 기기에 적용할 수 있다면 모뎀 내부 시스템 변경 비용 절감 및 기기 무선 연결의 유연성을 보장할 수 있을 것이다.

다만 IoT 기기의 소형화에 따른 내부 프로세스나 메모리의 한계로 인한 성능 제한이 필연적으로 존재하며 적용 가능 시스템에 대한 한계도 분명히 존재한다. 현재 소형기기 기반 SDR 연구 및 분석의 경우도 대부분 시스템 구축 및 구현 연구 중심으로 맞추어져 있다. 각기 다른 프로세서 적용 시에 대한 실제 성능 비교 및 성능별로 인한 디자인 기준에 관한 구체적인 연구가 필요하며 이를 통한 시사점 도출이 필요하다.

본 논문은 소형기기 내 SDR 스펙트럼 센싱 구축을 통한 소형기기별 성능 차이를 분석한다. 본 논문에서는 먼저 일반적인 SDR 스펙트럼 센싱 시스템을 제시하고 실제 소형기기별로 작동 시 중앙처리유닛(Central Processing Unit: CPU) 및 메모리 사용량에 대한 측정을 진행한다. 이를 통해 실제 소형기기에 SDR 시스템의 작동 시 유의점 및 실험을 통한 주요 디자인 고려사항을 도출한다. 추가로 소형기기별로 가지고 있는 프로세서에 관한 내용을 기술하고 실제 실험값과 비교하며 프로세서별 차이점 및 고려사항에 관해서 기술한다.

본 논문 구성은 다음과 같다. 2장에서는 소형기기 내 SDR 기반 시스템 구현 관련 문헌들을 기술한다. 3장에서는 일반적 시스템 디자인 작성 및 시스템 내 연산량 계산을 기술한다. 4장에서는 프로세스별 구조 비교 및 중요 고려사항에 관해서 기술한다. 5장에서는 실험 구성, 실험 결과 도출 및 디자인 가이드에 대해서 제공하며 6장에서는 결론을 기술한다.

Ⅱ. 관련 문헌

소형기기 내 SDR 구현 관련 문헌들은 전반적으로 소형기기를 활용한 SDR 구현 문헌들이 대부분이며 RTL-SDR 혹은 싱글 보드 컴퓨터를 활용한 구현논문들이 주를 이룬다. 대부분 소형 싱글 보드 컴퓨터 중 하나인 라즈베리 파이(Raspberry Pi)를 활용하여 연구를 진행하였다.

RTL-SDR 활용하여 SDR의 기본 구조 설명 및 연동 진행 시 내부 구조에 대한 일반화 분석을 진행한 문헌의 경우 [5] RTL-SDR 활용했던 사례들 및 문헌별 적용 소프트웨어, Application 및 장단점 들을 기술하였다. Wireless Sensor Network (WSN) 구성 논문의 경우 [6] 저비용 Cognitive radio를 프로토타이핑하는데 있어 2.4GHz 대역 내 WSN 구성을 위한 센싱 기반 시스템을 SDR 기기 중 하나인 Universal Software Radio Peripheral

(USRP)으로 구현하였다. 간단한 RTL-SDR 동글 기반 구현 논문의 경우 [7] 스펙트럼 분석기, 아날로그 라디오 수신기 구현 및 실험을 진행하였으며 스펙트럼 분석기 기기와의 데이터 비교를 통한 RTL-SDR 성능도 같이 분석하였다.

Fast-Fourier Transform (FFT) 계산의 고속화 논문의 경우 [8] 싱글 보드 컴퓨터 내 그래픽 처리 유닛 (Graphics Processing Unit: GPU)를 활용한 방법을 제시하였으며 CPU와 일반 PC 내 Intel-Core 5와 성능 비교를 진행하였다. 센싱 시스템 구현 논문의 경우 [9] 특이하게 USRP와 라즈베리 파이를 연동하여 구현하여 CPU 로드까지 측정하였다. 라즈베리 파이 기반 통신 송수신기 구현 논문의 경우 [10] RTL-SDR 기반 Sensing 시스템과 General Purpose Input/Output (GPIO) 기반 FM 송신기 두 개를 하나의 기기안에 구현하였다.

Ⅲ. 센싱 시스템 구조 디자인

소형기기 기반 SDR 시스템을 위해서 우선 먼저 기본적인 구조로도 작동이 가능한 스펙트럼 센싱 구조가 가장 적합하다. 해당 부문에서는 먼저 소형기기 내 작동할 스펙트럼 센싱 시스템 구조도를 설명한다. 이후 관련 시스템의 연산량을 계산함으로써 복잡성 측면에서 시스템이 영향을 받을 가능성에 대한 분석을 진행한다.

3.1. 스펙트럼 센싱 구조도

기본적인 스펙트럼 센싱 구조도는 그림 1을 참고한다. 그림 1과 같이 실제 주파수 Down conversion, 저소음 증폭기(Low-noise Amplifier: LNA), 아날로그-디지털 신호 변환기 (ADC/DAC) 등 기본적으로 하드웨어 내에서 이루어지는 역할들은 SDR 기기 내에서 이루어지며 이후 일련의 과정들은 Host 기기 내에서 이루어진다. 그림 1에 기술된 SDR 기기 (SDR Hardware)와 Host 기기 내 모듈별 설명을 진행하면 다음과 같다.

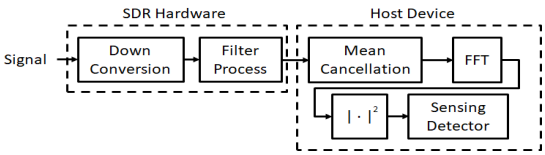

Fig. 1 General Block Diagram of Spectrum Sensing on SDR Device

- Down Conversion & Filter Process: SDR Hardware에서 받은 Passband 신호를 중심 주파수를 이용하여 Baseband 레벨로 변환시키며 노이즈 및 간섭 신호 제거를 위한 Filter process를 거친다. 필터 종류 및 필터 Coefficient 등은 정해진 바에 따라 쓰인다.
- Mean Cancellation: SDR 기기 내 Frequency Tunning을 위해 사용된 Local Oscillator에서 발생하는 Leakage를 제거하기 위해서 사용되는 모듈이다. 대처로 DC Offset으로 인하여 생기는 Peak이기 때문에 Mean Cancellation으로 제거할 수 있다. [11] SDR 기기 내에서 driver 세팅 조절을 통해서 일부 제거가 가능하기도 하나 해당 시스템 내에서 쓰이는 SDR 기기와 같이 Driver 레벨 조정이 불가능한 기기 경우 해당 모듈을 통한 제거가 유일한 방법이다. 실제 제거 전후의 차이를 보여주는 그림 2의 경우 실제 주파수 영역대에서 White Space 부분을 관찰시 실제 Local Oscillator로 인해 잔존하는 DC Offset 부분을 Mean Cancellation을 통해 제거는 모습을 보여주고 있다.
- Fast-Fourier Transform (FFT): Fourier Transform을 진행하는 모듈로써 해당 시스템 내에서는 주파수 대역 내에서 변화되는 신호 변동을 보기 위해서 쓰이고 크기 변화에 따른 주파수 대역의 Resolution도 다르다. 또한, FFT 변환 시 필터 Coefficient 적용으로 Filter 프로세스도 진행한다.
- Power Calculation: 주파수 대역의 신호 파워 세기를 계산하기 위해 있는 모듈이며 각 FFT 지점별로 파워 계산을 진행하는 모듈이다.
- Sensing Detector: Power 계산된 대역별로 일정 Threshold를 넘기면 감지 Flag를 Enable 시키는 모듈이다. 해당 시스템 내에서는 Threshold를 고정하며 진행하나 적응형으로도 적용이 가능하다.

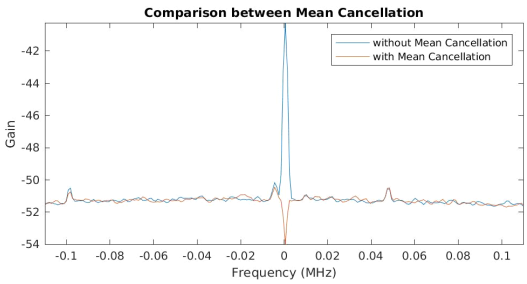

Fig. 2 Comparison Results of Frequency Spectrum with and without Mean Cancellation

3.2. 구조도 내 복잡성 연산량 작성

모듈별 데이터 전송 시 전송 지연이 없다고 가정한다. 또한, Data Type Conversion을 통해 발생하는 추가적인 Delay는 없다고 가정한다. 이때 복잡성 연산량의 확인이 필요한 모듈은 Mean Cancellation이다.

각 SDR 기기로부터 N개의 수신된 신호를 $x[n]$이라고 정의하고 $(n = 0,1,\cdots,N-1 \in R)$ $x[n]$의 평균값을 제하는 Mean Cancellation을 $x_{mc}[n]$으로 정의 시식 (1) 와 같다.

$$x_{mc}[n] = x[n] - \frac{1}{N}\sum_{n=0}^{N-1} x[n] \tag{1}$$

식 (1) 에 기반하여 $x_{mc}[n]$의 평균값 X_{mc} 로 정의한다면 식 (2) 와 같이 도출할 수 있다.

$$\begin{aligned} X_{mc} &= \frac{1}{N}\sum_{m=0}^{N-1}(x[m] - \frac{1}{N}\sum_{n=0}^{N-1} x[n]) \\ &= \frac{1}{N}\sum_{m=0}^{N-1} x[m] - \frac{1}{N}\sum_{m=0}^{N-1}\frac{1}{N}\sum_{n=0}^{N-1} x[n] \end{aligned} \tag{2}$$

식 (2) 의 복잡성을 계산하기 위해 Big O Notation 기반 복잡성 계산을 활용시 [12] 식 (3) 과 같다.

$$\begin{aligned} O(X_{mc}) &= N - N(N-1) \\ &= 2N - N^2 \ll cN^2 \,(\text{if } N \to \infty) \end{aligned} \tag{3}$$

식 (3) 이후에 적용되는 FFT의 경우 기본적으로 $O(X_{FFT}) = N \cdot \log_2 N$이며 [13] 파워 계산 및 Detector 도 N에 수렴된다. 이때 전체 시스템 자체의 복잡성은 $n \to \infty$ 일 시 $O(n^2)$에 가깝게 수렴하게 된다. 복잡성 측면에서는 Mean Cancellation 모듈을 제외하고는 복잡성이 높은 모듈들이 없는 시스템인지라 복잡성 측면에서 큰 문제가 될 여지는 낮은 상태이다. 특히 Matched Filter 혹은 correlation과 같은 기본적인 통신 구조 내 복잡성이 높은 모듈들이 시스템 내에 없다. 복잡성으로 인한 자원 할당 문제에 있어서 현 시스템은 기존 일반적인 송수신기보다 자유로운 시스템이다.

Ⅳ. 소형기기 비교 및 구현 고려사항

4.1 소형기기 내 프로세서 및 구성 비교

테스트 예정인 라즈베리 파이(Raspberry Pi: RPI) 1/2/3의 프로세서를 차례대로 열거한 내용은 표 1을 참고한다. 참고로 세 기기 전부 ARM 프로세서 기반으로 작동하며 ARM11의 경우 버전으로 고려 시 Cortex-A7과 Cortex-A53과 비교하면 이전 세대이다.

Table. 1 Comparison among different small single board computers

Raspberry Pi B+ [14]	Raspberry Pi 2 B v1.1[15]	Raspberry Pi 3 B [16]
ARM1176JZF-S 700 MHz	Cortex-A7 900 MHz	Cortex-A53 1.2 GHz
1 Core	4 Cores	4 Cores

표 1과 같이 위 세 가지 프로세서를 비교할 시 RPI 1의 경우 기본적인 프로세서 속도문제와 더불어 Single Core로 인하여 병렬처리 자체가 불가능하다. 반면에 RPI 2와 3의 경우 Cortex-A 시리즈를 기반으로 한 병렬처리가 가능하며 특히 Raspberry Pi 3의 경우 A53으로 높은 사양이 필요한 통신 시스템의 처리가 가능한 점도 유의미하다.

표 1 내 RPI 1, 2, 3의 기본 프로세서 성능 비교 차이를 위해 동일 Raspbian OS 기반 idle 상태에서의 1시간 CPU 사용량 평균을 측정하였다. 또한, SW 적인 차이 제거를 위해 동일 OS 내 같은 SW를 설치하고 동일시점에서 SW 업데이트 이후 실험을 진행하였다. 실험 결과는 그림 3과 같이 idle 상태에서 CPU 성능에 따른 사용량의 상당한 차이를 확인할 수 있다. RPI 1의 경우 사용량이 RPI 2, 3보다 4배 이상 더 발생하는 것을 확인할 수 있다.

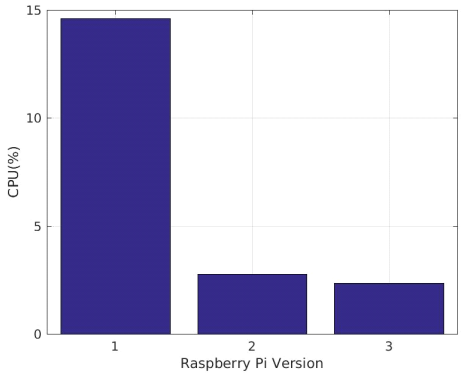

Fig. 3 Comparison of CPU Load among different raspberry pi devices

4.2. 기본 HW 구조기반 시스템 디자인 시 고려사항

그림 1과 같은 기본 구조로써 SDR 시스템 구현 시 고려사항은 싱글 보드 컴퓨터 측면에서 i) CPU 연산 처리, ii) 인터페이스 수율, iii) 메모리 사용량이다.

- CPU 처리: CPU 연산 성능에 따라 실제 처리할 수 있는 성능에 차이가 있으며 일정 시간 동안 작동을 진행하더라도 동일 세팅 하에서도 성능에 따른 데이터 처리 및 정상 기능 수행에 대한 지연으로 Sensing 기능에 대한 수행의 저하가 일어날 수 있다. 실제 무작동 상태인 idle 상에서의 측정상황에서도 RPI 1이 2/3보다 CPU 사용량이 높은 것을 확인할 수 있다. 실제 시스템 운영 시 성능에 따른 대폭 지연이 불가피하다.
- 인터페이스 수율: 인터페이스에 따라 실제 가용 가능한 데이터 수율에 의미 있는 차이가 있다. Host 기기 및 SDR 기기 처리 성능을 떠나 순수 인터페이스 측면만 보면 USB 2.0 경우 최대 수율이 480Mbps (bit per second)이며 실제 USB 장착한 RTL2832U의 경우 하드웨어 성능상 최대 샘플링인 3.2MHz에 기반 시 각 In-phase, Quadrature 신호 별 8 bits ADC를 고려하고 [17] Complex 샘플당 2 bytes로 정하고 최대 데이터 전송 필요량 계산 시 최대 51.2Mbps이며 이는 USB 최대량보다 낮다. 반면 1Gbps까지 가능한 Ethernet이 적용된 USRP 경우 높은 전송이 가능하여 RTL2832U의 최대 샘플링 주파수보다 더 높은 주파수를 적용해도 데이터 전송이 가능하다.
- 메모리 사용량: 메모리 사용량으로 인하여 타 기능들의 영향을 줄 수 있다. 이는 CPU 경우와 마찬가지로 실제 시스템 작동 상에서의 대폭 지연 및 치명적인 셧다운 발생의 가능성을 높인다.

Ⅴ. 실험 환경 및 결과

해당 장에서는 먼저 스펙트럼 센싱 구조기반 실험 모듈 구조도 및 환경 세팅에 대한 설명을 진행하며 이와 관련된 실험 결과를 분석한다. 이후 해당 실험과 연관하여 실제 시스템 내에서 싱글 보드 하드웨어에 최적화된 환경 세팅을 가능하게 해줄 수 있는 과정 설명 및 해당 예시를 제시함으로써 실제 시스템 구축 시 필요한 부문에 대해서 덧붙인다.

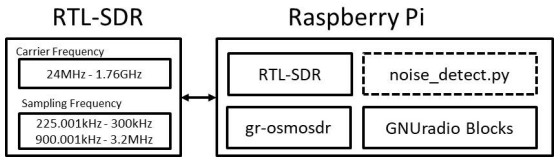

Fig. 4 Environmental Setup for SDR Device Testing

5.1. 실험 환경

실험 환경의 전반적인 구조도는 그림 4를 참고한다. 그림 4에 해당하는 RTL-SDR의 경우 R820T Tuner를 탑재한 RTL2832U USB 동글 SDR이며 [17] 소형 컴퓨터 기기의 경우 버전 별 RPI를 기반으로 테스트 되었다. 테스트 진행 전 필수 설치해야 하는 소프트웨어는 다음과 같다. 이 외 전반적인 구성은 표 2를 참고한다.

- rtl-sdr: RTL-SDR 기기 연결을 위한 드라이버로서 설치 이후에 동글 작동이 진행된다.
- gr-osmosdr: osmosdr의 경우 USB를 통한 rtl-sdr연결을 가능하게 하는 gnuradio 기반 별도 블록이며 이를 통해 data source/sink가 가능하다.
- gnuradio: 3.7.13 version으로 설치된 gnuradio이며 debian 계열인 raspbian내 default로 설치할 수 있다.
- noise_detect.py: 그림 1과 동일한 구조의 스펙트럼 센싱 시스템을 작동시키는 python 파일이다.

Table. 2 Environment Setup

Software	
gnuradio	3.7.13
gr-osmosdr	Built in June 27. 2018
rtl-sdr	0.6
Hardware RTL2832U	
Interface	USB 2.0
Tuner	R820T2
Center frequency range	24MHz ~ 1.76 GHz
Sampling frequency range	225KHz~300KHz 900KHz ~ 3.2MHz

위 모듈 사항들을 토대로 실험을 진행하며 실제 연산 성능의 평가를 위해 linux 내 Top 명령어를 사용하여 CPU 및 Memory 사용량을 측정하였으며 측정 시 전체 CPU, 메모리 사용량을 측정하였다. 측정 시 실제 noise_detect.py이 실행되어 3초간의 스펙트럼 활동 기록 및 완료까지의 작동 시간 동안 연산 성능 기록을 수

집하였으며 이외 상태일 때는 자동으로 수집을 정지하였다. 각 RPI 별로 측정 시 동일 SW 및 파라미터 조건으로 실행하였다.

5.2. 실험 결과

실험 결과는 크게 두 종류로써 i) HW별 성능 비교 ii) 파라미터별 성능 차이점에 대해서 분석한다. HW별 idle 시와 gnuradio 작동 시 CPU 및 Memory 사용량 비교는 그림 5와 6과 같다. 참고로 기본값으로 중심 주파수 1.1GHz, 샘플링 주파수 1MHz, 평균 계산 표본 수 10, FFT 크기는 1024이다.

그림 5와 같이 프로세서의 차이에 따른 기본적인 idle 상태에서도 차이가 있음이 확인되며 실제 Spectrum sensing 프로그램 작동 시 CPU 성능에 의미 있는 차이점을 발견할 수 있다. 특히 RPI 1과 2를 비교할 시 Single Core로써 발생하는 로드인 49%를 Quad Cores 내 4개의 Core로 분산시킴으로써 RPI 1과 비교 시 전체 CPU 성능의 절반만으로 처리 가능함을 확인할 수 있다. RPI 3의 경우는 프로세서 Clock의 성능과 더불어 내부 스케줄링의 유연함도 같이 더불어서 처리하기 때문에 RPI 2와 비교 시 10% 성능만으로도 처리할 수 있다.

메모리의 경우 그림 6과 같이 3개 기기 간의 메모리 할당량이 대략 최저 20MiB, 최고 30MiB 정도의 차이점만 보인다. 특히 Quad Core 및 비슷한 Clock 속도를 가지고 있는 RPI 2와 3간을 비교할 때 RPI 3의 고속처리 속도로 인하여 실제 메모리 할당량이 RPI 2와 비교 시 적은 것으로 관찰되며 RPI 1 과 RPI 2 비교 시 CPU 성능 문제로 인한 핸들링 가능한 데이터 문제로 인해 RPI 1의 할당량이 높은 것을 관찰할 수 있다.

위 내용을 기반으로 하여 변화가 상대적으로 큰 CPU 성능을 바탕으로 하드웨어와 Host 기기 내 중심 주파수, FFT 크기, 평균 산출 표본 수, 샘플링 주파수 파라미터 조절 통한 성능 변화 실험도 같이 진행하였으며 변경하는 해당 파라미터를 제외한 모든 값은 기본으로 중심 주파수 1.1GHz, 샘플링 주파수 1MHz, 평균 계산 표본 수 10, FFT 크기는 1024로 설정한다. 실험 결과는 그림 7, 8, 9, 10을 참고한다.

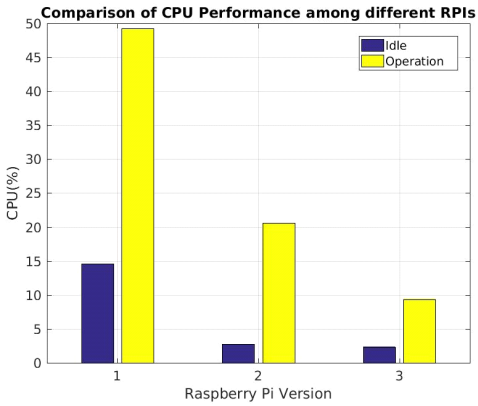

Fig. 5 Comparison of CPU Load among three Raspberry Pis

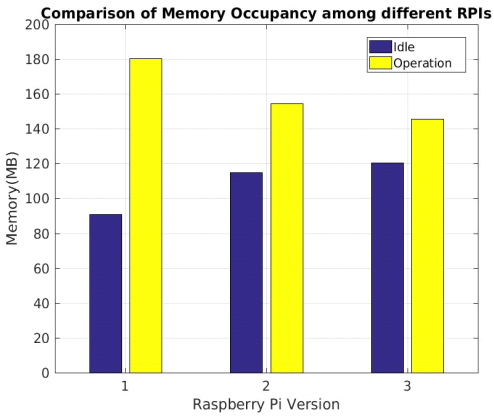

Fig. 6 Comparison of Memory Allocation among three Raspberry Pis

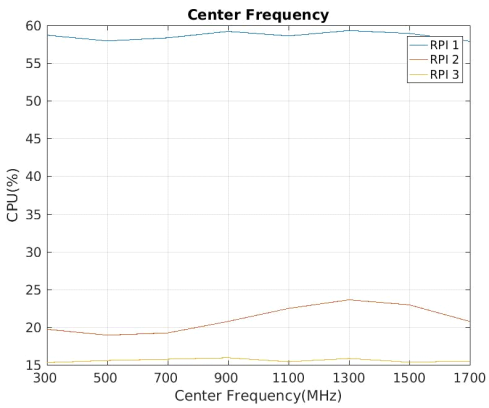

Fig. 7 Comparison of CPU Load over Center Frequency

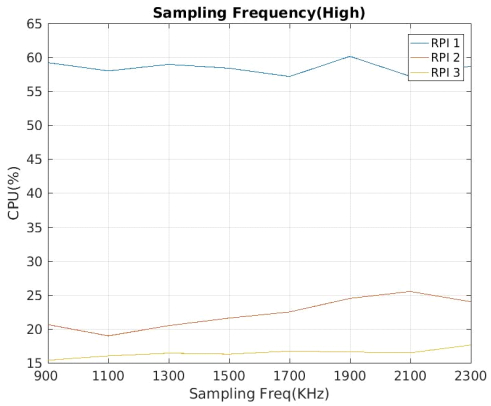

Fig. 8 Comparison of CPU Load over Sampling Frequency

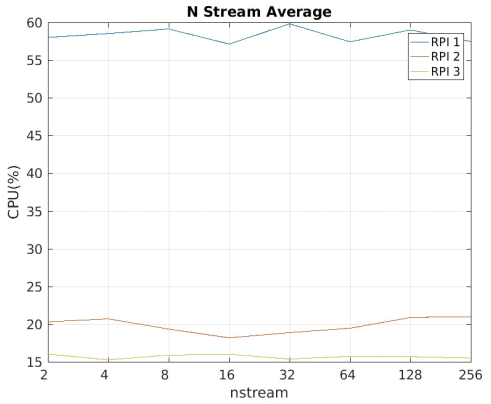

Fig. 9 Comparison of CPU Load over the Number of Average Samples

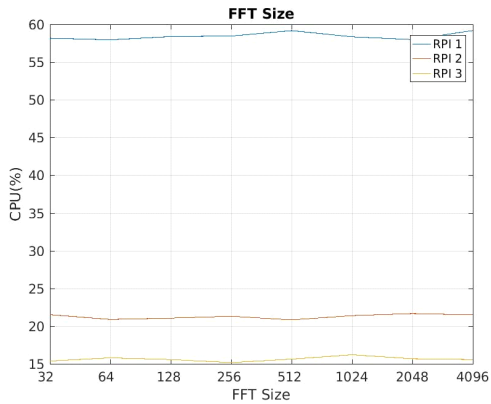

Fig. 10 Comparison of CPU Load over the FFT Size

그림 7과 같이 중심 주파수에 따른 큰 변화는 RPI 2를 제외하고는 없다. RTL2832U 내 Local Oscillator와 HW 기기 내 SW 모듈 간의 동기화 문제로 인해 추가적인 CPU 사용이 가능하며 RPI 2의 변화는 이를 보여준다. 그러나 전체적인 상황에서 보면 기존 파라미터가 고정된 상태에서 중심 주파수 변화만 두고는 사용량의 상당한 변화는 기대하기 어렵다. 그림 10도 마찬가지로 FFT 크기 변화만으로는 사용량에 큰 차이를 보이지 않았다. 그러나 그림 8, 9과 같이 샘플링 주파수, 평균 표본 수 변경으로 인해 일부 변화를 관찰할 수 있다. 특히 그림 8과 같이 샘플링 주파수가 높아질수록 RPI 2의 CPU 로드가 25% 이상까지 상승하는 상당한 변화를 관찰할 수 있다. 이는 샘플링 주파수의 상승으로 인한 프로세스의 부하가 점차 높아짐을 확인할 수 있는 그림이다. 특이점이 보이는 일부 주파수 점을 제외하고는 전반적으로 낮은 샘플링 주파수 대역이 CPU 로드를 적게 소모하는 것을 볼 수 있다. 그림 9도 특정 평균 표본 수에 따라 RPI 1과 RPI 2 사용량의 일부 변화를 관찰할 수 있다.

5.3. 실험 결과 기반 시스템 디자인 고려사항

본 논문에서 테스트한 스펙트럼 센싱시스템에서 연산적인 측면 상 우선 고려할 파라미터들은 실험 결과와 같이 샘플링 주파수, 평균 표본 수이다. 5.2의 과정을 거쳐서 얻은 실험 결과값을 기반으로 하여 실제 사용하고자하는 파라미터 세팅 값의 경계점을 파악하고 세팅하여 사용할 수 있다. 특히 샘플링 주파수의 영향을 받는 스펙트럼 센싱 시스템의 경우 관찰하고 싶은 대역폭 크기와 CPU 로드 간의 최적점을 위와 같은 방법으로 찾아서 디자인하는 것이 적합하다. 예를 들어 RPI 3에서 최대 17% CPU 로드만 허락될 시 테스트 결과인 그림 8에 기반하여 주파수 대역의 0.9MHz 이상에서 2.1 MHz 이하 샘플링 주파수를 세팅하면 되며 RPI 2의 경우 최대 25%까지 CPU 로드가 가능하다고 가정 시 그림 8과 같이 최대 1.9MHz까지의 샘플링 주파수 대역을 사용할 수 있다. 평균 표본 수도 위와 같은 방식으로 실험 결과에 기반한 시스템 내 파라미터 세팅이 가능할 것이다.

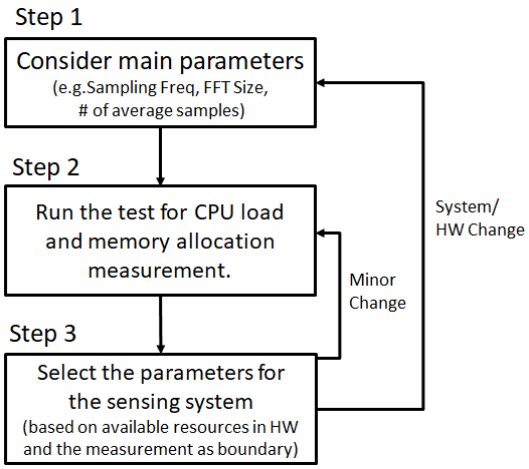

Fig. 11 System Design Procedure from Measurements

상기 내용을 기반으로 하여 싱글 보드 컴퓨터 내 통신 관련 시스템 디자인 시 및 파라미터 세팅을 그림 11 와 같이 3단계 방식으로 진행할 수 있다. 그림 11에서 1단계에서는 디자인된 시스템 내에서 영향을 미칠 샘플링 주파수, 평균 표본 수 등의 주요 파라미터들을 선정한다. 2단계에서는 1단계에서 선정된 주요 파라미터들을 대상으로 테스트를 진행하며 CPU 및 메모리 사용량을 측정한다. 3단계에는 측정된 결과들을 활용하여 사용 가능한 각 파라미터의 범위를 도출하고 도출된 범위 내에서 파라미터들을 세팅한다.

세팅 완료 이후 시스템 수정으로 인한 파라미터 관련 변경 시 크게 “Minor Change”와 “System/HW Change”로 나누어서 진행할 수 있다. “Minor Change”의 경우 SW 내 모듈 단에서의 소규모 수정 사항이 있을 때 변경이 필요한 상황을 뜻하며 이때는 SW 내 모듈 혹은 파라미터의 단순 변경으로 인하여 연산량의 변화만 있으므로 그림 11과 같이 2단계로 재진입해서 테스트 및 CPU, 메모리 사용량 재측정 후 파라미터들을 세팅할 수 있다. “System/HW Change”의 경우는 라디오 시스템 자체의 큰 변화이나 하드웨어의 수정 사항이 있을 때 변경이 필요한 상황을 뜻한다. 이때는 단순 연산량을 넘어선 큰 변화이기 때문에 주요 파라미터들의 세팅뿐만이 아닌 선정도 필요하다. 따라서 이 경우에는 1단계부터 다시 시작하여 파라미터를 선정해야 한다.

VI. 결 론

본 논문에서는 기본적인 스펙트럼 센싱 시스템에 기반한 싱글 보드 컴퓨터의 성능 및 메모리 할당량에 대한 분석을 진행하였다. 각 컴퓨터 내 프로세스의 성능 파악 이후 성능 결과를 도출하였으며 도출된 결과에 기반한 시스템 디자인 고려 예시도 같이 제시하였다.

앞으로 IoT 기반한 SDR 기술도 다양하게 나오는 만큼 이와 관련된 소프트웨어 아키텍처의 최적화 방안이 도출이 필요하다. 이를 위해 싱글 보드 컴퓨터 기반 SDR SW 프레임워크 사례 조사 및 설계 고려 요소들을 도출하는 것이 필요하다. 추가로 싱글 보드 컴퓨터 전용 SDR과 클라우드 기반한 다양한 연구들이 진행되고 있는 만큼 [18] 클라우드 기반 다중 변조 기반 통신 시스템 적용 구현 시 하드웨어와 연산 요구조건에 대해 도출할 수 있는 추가 분석도 동반해야 할 것이다.

References

[1] F. Dahlqvist, M. Patel, A. Rajko, and J. Shulman, “Growing opportunities in the Internet of Things”, *McKinsey & Company*, pp. 1-6, July 2019.

[2] S. Y. Oh, “[IFA 2019] Samsung's 5G and LG's Artificial Intelligence Ahead of Europe's Largest Consumer Electronics Event” [Internet] Available: http://m.elec4.co.kr/article/articleView.asp?idx=24002

[3] SmartThings Developer Documentation [Internet] Available: https://smartthings.developer.samsung.com

[4] J. R. Machado-Fernández, . “Software defined radio: Basic principles and applications.”, *Facultad de Ingeniería*, vol. 24, no. 38, pp.79-96., 2015

[5] M. Mishra, A. Potnis, P. Dwivedy, S. K. Meena, “Software defined radio based receivers using RTLSDR: A review.”, in *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*. IEEE, 2017.

[6] S. E. Barrak, A. Lyhyaoui, A Puilafito, S. Serrano, “Implementation of a low cost SDR-Based Spectrum sensing Prototype Using USRP and Rasbperry Pi Board.” in *Proceedings of Engineering and Technology-PET 20*, pp. 54-58, 2017

[7] E. G. Sierra, and G. A. R. Arroyave. “Low cost SDR spectrum analyzer and analog radio receiver using GNU

radio, raspberry Pi2 and SDR-RTL dongle." in *2015 7th IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE, 2015.

[8] S. Sabarinath, R. Shyam, C. Aneesh, R. Gandhraj, K. P. Soman, "Accelerated FFT computation for GNU radio using GPU of raspberry Pi." in *Computational Intelligence in Data Mining-Volume 2.*, pp. 657-664, Springer, New Delhi, 2015.

[9] H. J. Park, G. M. Lee, S. H. Shin, B. H. Roh, J. M. Oh, "Implementation of Multi-Hop Cognitive Radio Testbed using Raspberry Pi and USRP." *International Journal of Interdisciplinary Telecommunications and Networking (IJITN)*, vol. 9, no. 4, pp. 37-48, 2017

[10] S. Mohammed, K. A. Hatim, "Raspberry Pi and RTL-SDR for Spectrum Sensing based on FM Real Signals." in *2018 6th International Conference on Multimedia Computing and Systems (ICMCS)*. IEEE, 2018.

[11] J. Y. Kim, , A. C. Marcum, A. D. Balmos, A. W. Layton, S. G. Larew, J. V. Krogmeier, and D. J. Love. "Implementation and analysis of energy detection-based sensing using USRP/SBX platform." In *2014 IEEE Military Communications Conference*, pp. 1504-1509. IEEE, 2014.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009

[13] M. T. Heath, *Scientific computing: an introductory survey*. Vol. 80. SIAM, 2018.

[14] BCM2835 - Raspberry Pi Documentation [Internet]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/README.md

[15] BCM2836 - Raspberry Pi Documentation [Internet]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2836/README.md

[16] BCM2837 - Raspberry Pi Documentation [Internet]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2837/README.md

[17] RTL-SDR Blog V3 Datasheet [Internet]. Available: https://www.electrokit.com/uploads/productfile/41015/RTL-SDR-Blog-V3-Datasheet.pdf

[18] H. T. Thien, R. Tendeng, H. Vu-Van, and I. Koo. "Implementation of Spectrum-Sensing for Cognitive Radio Using USRP with GNU Radio and a Cloud Server." *Journal of information and communication convergence engineering*, vol 16, no. 1, pp. 23-30, 2018

김준영(Joon Young Kim)

퍼듀대학교 전기컴퓨터공학과 공학학사 (2010)
퍼듀대학교 전기컴퓨터공학과 공학박사 (2015)
현대자동차 책임연구원 (2015 ~ 현재)
※관심분야 : 스펙트럼 센싱, 간섭 전파 분류 및 감지, 소프트웨어 정의 라디오 구현, IoT 시스템 및 플랫폼