CMCIS 시스템을 위한 MDEC 모듈의 설계 및 구현

정현철[O], 김태군*, 강동원**,김재영***,이세진****, 조성칠*****, 김홍삼[•], 이용희[••], 서광석[•••], 박명철[••••]
*[O](주)메디칼써프라이, **(주)메디아나, ***에스디바이오센서(주), ****그루솔루션(주), *****(주)하이버스,
[•](재)원주의료기기테크노밸리, [••]한라대학교 컴퓨터공학부, [•••](주)메디게이트, [••••]송호대학교
e-mail: {jhc, tkkim}@medicalsupply.co.kr*[O], dwkang@mediana.co.kr**, ybme@naver.com***,
sejin@grusolution.co.kr****, jce@hybus.net***** , redginseng@wmit.or.kr[•], yhlee@halla.ac.kr[••],
whiteseo@medigate.biz[•••], africa@songho.ac.kr[••••]

Design and Implementation of MDEC module for CMCIS System

Hyon-Chel Jung[O], Tea-Koon Kim*, Dong-Won Kang**, Jea-Young Kim***, Se-jim Kim****,
Sung-chil Cho*****, Hong-Sam Kim[•], Yong-Hee Lee[••], Kwang-Suk Seo[•••], Myeong-Chul Park[••••]
*OMEDICAL SUPPLY CO., LTD. **MEDIANA CO,. LTD. ***SD BIOSensor CO., LTD
****GRU SOLUTION CO., LTD, *****HyBus CO., LTD.
[•]Wonju Medical Industry Technovalley(WMIT), [••]Halla University,
[•••]Medigate CO., LTD, [••••]Songho College

● 요 약 ●

본 논문에서는 다중 생체계측 장비를 CMCIS(Central Monitoring and Cardiology Information System) 기반의 통합 시스템에 연동하기 위한 MDEC(Medical Device Exchange Communication)의 설계 및 구현을 소개한다. MDEC은 6종의 계층장비의 신호를 받아 각 서버로 데이터를 전송할 수 있도록 설계하고 Linux OS을 포팅 하였다. 각 장비별 프로토콜에 대해 최소화된 기본 프로토콜을 정의하고 통신방법에 따라 UART 포트 혹은 USB 포트를 생체정보의 물리적 입력장치로 사용하였다. 호스트 서버와는 유무선 LAN을 통해 정보를 전달할 수 있게 하드웨어를 설계하였다. 본 연구의 결과는 CMCIS 기반의 통합 시스템을 구축하는 기반기술로 향후 효율적 의료서비스에 기여할 것으로 사료된다.

키워드: CMCIS(Central Monitoring and Cardiology Information System), MDEC(Medical Device Exchange Communication)

I. Introduction

최근 급속히 증가하고 있는 병원내의 의료정보시장의 요구에 발맞춰 생체정보 계측기기의 데이터를 저장, 관리하는 CMS(Central Monitoring System)와 DMS(Data Management System)를 통합할 수 있는 CMCIS 기반의 통합 시스템 개발의 요구가 높아지고 있다[1]. 심혈관계 질환의 경우 및 사망자의 지속적인 증가로 인한 경제적 손실이 해마다 증가하고 있는 추세이며 이와 관련하여 질병의 조기 진단과 효율적 의료서비스를 위해 CMCIS 개발은 필수적인 요소이다 [2,3]. 본 논문에서는 CMCIS 시스템 구축을 위한 기반 기술이라 할 수 있는 계측 장비용 MDEC 모듈의 설계와 구현을 소개한다. 개발되는 MDEC은 외장형으로 6종(재세동기, 진단용 심전계, 환자 감시 장치, 바이탈사인 모니터, 산소포화도 측정기, 당화혈색소 측정기)의 계측장비의 신호를 받아 정의된 장비별 프로토콜에 다라 해당 계측정보를 각 서버에 전송하는 역할을 하게 된다. 의료장비와 서버간의 물리적 흐름을 갖는 전체적인 MDEC의 역할은 Fig. 1과 같다. 본 연구의 결과는 향후, CMCIS 기반의 통합 시스템의 통신 체계를 위한 기반 기술로 활용될 것이다.

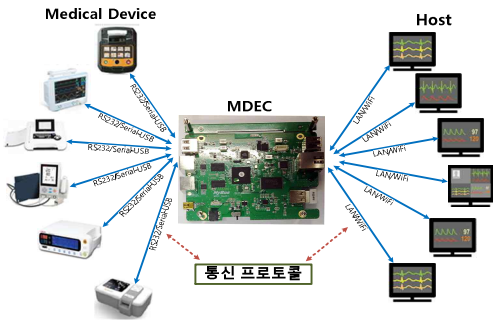

Fig. 1. MDEC : Communication Module of Medical Device and Host

II. Design and Implementation

1. MDEC Hardware

각 기관(Medical Device 업체)별 장비를 조사하여 현재 사용되고 있는 Local Protocol을 기준으로 각 Medical Device별, 기관별 Protocol에 대해 최소화된 기본 Protocol의 정의를 진행했고, 필요에 따라 정의된 Protocol을 사용하거나 기관이 요구하는 별도의 통신방법에 따라 UART Port 혹은 USB Port를 생체정보의 물리적 입력장치로 사용하고 Host Server와는 Wired LAN 및 Wireless LAN을 통해 취득된 정보를 전달할 수 있는 MDEC System을 개발하기 위해 하기의 Fig. 2의 블록도와 같이 의료장비와 Host Server간의 물리적 흐름을 갖는 MDEC 하드웨어 설계를 진행 하였다.

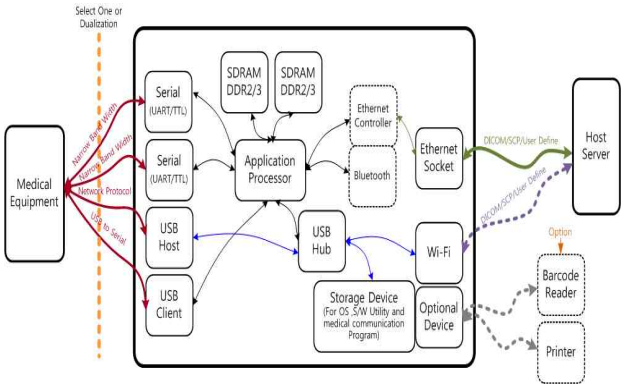

Fig. 2. Block diagram of MDEC

Fig. 2의 블록다이어그램 기준으로 회로 설계를 진행했으며 수집된 생체 신호 데이터에 대한 Server로 전송을 하기 위한 통신수단으로 Wireless LAN 및 Ethernet은 기본사양으로 구성했고 Input Port는 기본은 UART를 사용하는 Serial Port와 USB Host Port를 통한 NDIS지원 장치 혹은 USB to Serial 장치의 지원을 고려해서 구성했다. BT는 Optional 사양으로 구성하고 향후 외부장치에서의 관련 정보의 Input Port로 사용 될수 있도록 했으며 내부에 수집된 정보를 별도로 저장이 가능하게 할 수 있도록 기본이 되는 OS 및 필수 Filesystem을 사용할 수 있는 메모리 이외의 Storage Device로 mSD를 구성했다. MDEC의 외형에 대한 기구개발은 설계초기부터 PCB와 부품의 크기 등을 기준으로 설계했다. 기구제작을 간소화하기 위해 Button을 Status를 나타내는 LED와 동시의 1개의 부품으로 제작될 수 있도록 설계했다. 정보의 흐름에 따른 Input Side와 Output Side에 대한 각각의 상태표시를 고려해 설계했다. 설계에 따라 개발된 보드와 완성품은 Fig. 3과 같다.

Fig. 2. Board and Final Product of MDEC

2. MDEC Software

각 기관별 Medical Device 마다 정의된 Local 프로토콜을 적용하고 서버간 통신을 할 수 있도록 하고 프레임웍 기반에서 개발의 유연성을 고려하여 QT 기반의 어플리케이션을 작성하였다. 그리고 MDEC 어플은 Table 1과 같은 사항들을 포함 하여 개발 되었다.

Description
Dicom 및 SCP 관련 Library 혹은 System Program
Option기능에 대한 지원 Library
Wave Form전송을 위한 Graphic Library
프로토콜의 형태에 따라 필요한 Network Socket Program
Configuration관련 System Program

Table 1. Application of MDEC

Medical device와 MDEC 그리고 의료 server와의 통신은 다음과 같은 절차로 이루어진다. 우선 Medical Device와 MDEC은 의료정보를 UART/USB Interface를 통해 송수신 한다. 그리고 MDEC 소프트웨어 내부의 UART Interface에서는 Medical Device에서 들어오는 데이터 중 MDEC 정보를 분석해서 MDEC Protocol Manager로 전달하는 역할을 한다. 또한 MDEC Protocol Manger에서 데이터를 받아 Medical Device로 전달하는 역할을 한다. MDEC Protocol Manager는 MDEC protocol 관련 데이터를 처리하는 역할을 한다. Connection Pool은 MDEC Protocol Manager 및 Network Interface에서 들어오는 데이터를 처리한다. Network Interface의 경우는 LAN 및 WiFi를 통해 Host와 데이터를 송수신하는 역할을 한다. (LAN 및 WiFi 둘 중 하나만 동작함. 동시에 작동하지 않음) Host는 Medical Device의 정보를 화면에 나타내고 필요 데이터를 Medical Device로 전송하는 역할을 한다. Fig. 4는 Local 프로토콜의 적용 및 서버간 통신, 각종 라이브러리를 포함한 전체 어플리케이션의 구조도를 나타낸 것이다.

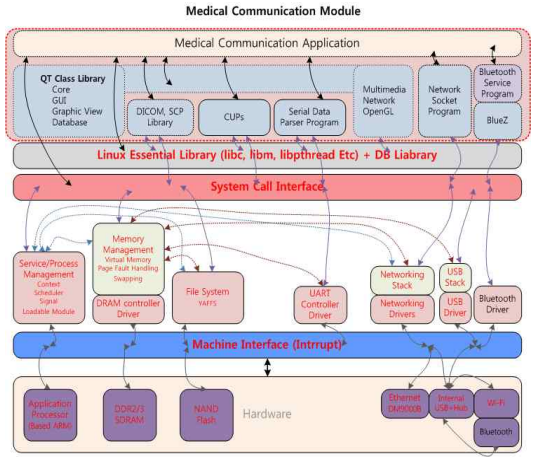

Fig. 4. Structure of MDEC S/W

MDEC application 개발 환경은 linux기반에서 QT Creator IDE 2.4.1을 사용하여 GUI 및 프로그램을 작성 했으며 프레임웍으로 사용된 QT Library는 4.8.2버전을 사용하여 개발 되었다. 또한 wireless 검색 및 적용을 위해 wpa_supplicant 툴을 이용하여 개발 하였다.

Item	Description
Platform	Linux(Core1 Quad)
OS	Ubuntu 12.04 LTS
Tool	QT Creator IDE 2.4.1 QT Library 4.8.2 wpa_supplicant
Network	Ethernet 10/100M

Table 2. Development Environment of MDEC App.

III. Verification

연구 결과의 시험을 위한 하드웨어 환경은 Intel Core i5 호스트 PC에서 Target Board는 MDEC(Samsung S5PV210), 계측장비는 오실로스코프와 Power Supply를 사용하였다 소프트웨어 환경은 컴파일러와 디버거는 arm-linux-gcc, kernel 2.6.35를 사용하고 S/W Tool은 Dragin.exe, DNW.exe, Terminal Program(Tera term)을 사용하였다. Table 3은 테스트 S/W를 보인 것이다.

Item	Program	Description
System Fusing & Booting Test	Dragin.exe, DNW.exe	linux console and PC
USB Test(Storage) USB Test(USB to Serial)	hdparam rs232_test	linux console
NAND Speed Test	hdparam	linux console
Network Test(Wired) Network Tset(Wireless)	iperf	linux console and PC
RS-232 Test(4 pin) RS-232 Test(6 pin)	rs232_test	linux console

Table 3. Test S/W

시스템 퓨징과 부팅테스트 결과는 Fig. 5와 같다. 주요절차는 먼저, Dragin의 Download Progress를 확인하였고 DNW의 OS Write Progress를 확인하였다. 그리고 다운로드 완료 후 MDEC Platform을 확인하여 OS가 정상적으로 부팅되었다. 전원인가 후 부팅시간을 확인한 결과 30초 이내로 요구사항을 만족하였다.

```
. . . . . .
login[64]: root login on 'ttySAC2'
[    22.250309]                          # <--------- boot 완료
. . . . . .
MDEC v0.00.07

UART /dev/ttySAC0 (115200) open.
UART /dev/ttySAC3 (115200) open.
Failed to UART /dev/ttyUSB0 (115200) open.
killall: wpa_supplicant: no process killed
130101  9:00:24 [Note] Event Scheduler: Loaded 0 events
130101  9:00:24 [Note] /usr/mysql/libexec/mysqld: ready for connections.
Version: '5.1.28-rc'  socket: '/tmp/mysql.sock'  port: 3306  Source distribution
[    28.348267] read data = 0xf1          # <--------- 시스템 구동 완료
```

Fig. 5. System Fusing & Booting Test

USB storage 테스트는 USB에 Storage를 연결하고 dmesg 로 인식 여부를 확인한다. USB Storage read timing이 18MB/sec 이상의 요구사항을 확인한 결과 Fig. 6과 같이 만족하였다.

```
[root@HyBus ~]# dmesg
. . . . . .
[  189.633814] sd 0:0:0:0: [sda] Assuming drive cache: write through
[  189.638449]  sda: sda1
. . . . . .

[root@HyBus ~]# hdparm -t -T /dev/sda1

/dev/sda1:
 Timing buffer-cache reads:   1662 MB in 3.00 seconds = 553.69 MB/sec
 Timing buffered disk reads:   71 MB in 3.03 seconds = 23.47 MB/sec
```

Fig. 6. USB Test(Storage)

NAND 속도 테스트는 df를 이용하여 Nand 메모리의 상태를 확인하여 Filesystem이 /dev/mtdblock2에 정상적으로 마운트 되어 있는지 확인하고 NAND read 속도가 2MB/sec 이상이고 NAND write 속도가 1.5MB/sec 이상인지를 확인한다. 확인결과는 Fig. 7과 같이 요구사항을 만족하였다.

```
[root@HyBus ~]# hdparm -t -T /dev/mtdblock2

/dev/mtdblock2:
 Timing buffer-cache reads:   1662 MB in 3.00 seconds = 553.69 MB/sec
 Timing buffered disk reads:   8 MB in 3.26 seconds = 2.45 MB/sec

[root@HyBus ~]# time dd if=/dev/zero of=/root/tmp bs=1M count=10
10+0 records in
10+0 records out

real    0m5.198s
user    0m0.000s
sys     0m5.191s                      # 10MB/5.19 = 약 1.92MB/s
```

Fig. 7. NAND Speed Test

네트워크 테스트는 유무선 동일하게 ping test로 네트워크 연결 상태를 확인하고 패킷 손실 없이 ack가 오는 것을 확인하였다. 그리고

네트워크 속도가 30Mbit/sec이상인지를 확인한 결과 Fig. 8과 Fig. 9와 같이 요구사항을 만족하였다.

```
[root@HyBus ~]# ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2): 56 data bytes
64 bytes from 192.168.0.2: seq=0 ttl=128 time=1.214 ms
64 bytes from 192.168.0.2: seq=1 ttl=128 time=0.320 ms
64 bytes from 192.168.0.2: seq=2 ttl=128 time=0.321 ms

--- 192.168.0.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.320/0.618/1.214 ms

[root@HyBus ~]# ./iperf -s -P 0 -i 1 -p 5001 -l 8K -f m -t 5
------------------------------------------------
Server listening on TCP port 5001
TCP window size: 0.08 MByte (default)
------------------------------------------------
[  4] local 192.168.0.10 port 5001 connected with 192.168.0.2 port 63998
[  4]  0.0- 1.0 sec  3.27 MBytes  27.4 Mbits/sec
[  4]  1.0- 2.0 sec  4.54 MBytes  38.1 Mbits/sec
[  4]  2.0- 3.0 sec  4.25 MBytes  35.7 Mbits/sec
[  4]  3.0- 4.0 sec  4.31 MBytes  36.2 Mbits/sec
[  4]  4.0- 5.0 sec  4.30 MBytes  36.0 Mbits/sec
```

Fig. 8. Network Test(Wired)

```
[root@HyBus ~]# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1): 56 data bytes
64 bytes from 192.168.0.1: seq=0 ttl=64 time=9.431 ms
64 bytes from 192.168.0.1: seq=1 ttl=64 time=6.287 ms
64 bytes from 192.168.0.1: seq=2 ttl=64 time=5.716 ms

--- 192.168.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 5.716/7.027/9.431 ms

[root@HyBus ~]# ./iperf -s -P 0 -i 1 -p 5001 -l 8K -f m -t 5
------------------------------------------------
Server listening on TCP port 5001
TCP window size: 0.08 MByte (default)
------------------------------------------------
[  4] local 192.168.0.229 port 5001 connected with 192.168.0.2 port 51044
[  4]  0.0- 1.0 sec  3.75 MBytes  31.4 Mbits/sec
[  4]  1.0- 2.0 sec  4.09 MBytes  34.3 Mbits/sec
[  4]  2.0- 3.0 sec  3.91 MBytes  32.8 Mbits/sec
[  4]  3.0- 4.0 sec  4.00 MBytes  33.6 Mbits/sec
```

Fig. 9. Network Test(Wireless)

RS-232 테스트는 MDEC에서 rs232_test를 실행시키고 PC 터미널 프로그램에서 정상적으로 통신되는지 확인하여 MDEC의 write count와 PC의 read count가 동시에 증가하는 지을 확인하여 통신 가능 여부를 판단하였다. 결과는 Fig. 10과 같이 정상적으로 동작함을 확인 하였다.

```
[root@HyBus ~]# ./rs232_test

====================
== MDEC RS232 Test APP ==
====================

BAUDRATE : 230400
RS232 Test : [4PIN]
Write count : 0
Write count : 1
Write count : 2
Write count : 3
Write count : 4
Write count : 5
Write count : 6
Write count : 7
Write count : 8
Write count : 9
```

```
====================
== MDEC RS232 READ TEST ==
====================
Read count : 0
Read count : 1
Read count : 2
Read count : 3
Read count : 4
Read count : 5
Read count : 6
Read count : 7
Read count : 8
Read count : 9
```

<MDEC Terminal> < PC>

Fig. 10. RS-232 Test(4 pin)

IV. Conclusions

본 논문에서는 다중 생체계측 장비를 CMCIS(Central Monitoring and Cardiology Information System) 기반의 통합 시스템에 연동하기 위한 MDEC(Medical Device Exchange Communication)을 설계 및 구현하였다. 구현된 결과는 요구사항에 따른 시험결과에서 모두 만족되어 CMCIS 기반의 통합 시스템을 구축하는 기반기술로 향후 효율적 의료서비스에 기여할 것으로 사료된다.

Acknowledgment

본 연구는 2014년도 정부(산업통상자원부, KIAT, 강원지역사업평가원) 재원의 경제협력권산업육성사업으로 수행된 연구임. (2014-R0002890)

References

[1] S-I. Kang, and A-S. Oh, "A Design and Implementation of Mobile Healthcare Syste based on Smart Gateway," Journal of The Korea Institute of Information and Communication Engineering, Vol. 16, No. 9, pp. 1970-1976, September 2012.

[2] S-M. Chun, J-W Nah, J-T Park, "Design and Implementation of IEEE 11073/HL7 Translation Gateway Based on U-Healthcare Application Service for M2M," Journal of The Korean Institute of Communications and Information Sciences, Vol. 36, No. 3, pp. 275-286, March 2011.

[3] Mohamed Fezari et al. “Ambulatory Health Monitoring System Using Wireless Sensors Node,“ Procedia Computer Science 65 pp. 86-94, 2015.