ARM11 과 Linux 기반의 무선 가스 센서 데이터 전송용 플랫폼 구현

선희갑* · 김영길**

A Study on Implementation for Wireless Gas Sensor Data Transmission Platform using ARM11 and Linux

Hee-Gab Sun* · Young-Kil Kim**

요 약

최근 화두가 되는 유비쿼터스(Ubiquitous)는 "언제, 어디서나 존재하는"을 의미하는 라틴어로 사용자가 네트워크나 컴퓨터를 의식하지 않고 장소에 상관없이 자유롭게 네트워크에 접속할 수 있는 정보통신 환경을 말한다. 본 논문에서 구현한 시스템은 크게 두 부분으로 나누어진다. 한 부분은 환경정보를 감지하여 무선으로 대상 플랫폼에 데이터를 전송이 가능한 센서부로 Gas Sensor와 저전력 Zigbee Module을 사용하여 구현하였으며 센서 노드의 역할을 수행한다. 나머지 한 부분은 센서부로부터 얻은 데이터를 수신하여 LCD에 표시하는 플랫폼으로 Sink 노드의 기능을 수행한다. 본 논문에서 구현한 플랫폼은 ARM11기반의 프로세서에 공개소스 기반의 OS(Operating System)인 리눅스를 포팅하여 구현하였다. 또한, 리눅스상의 윈도우 매니저인 Qtopia를 포팅하여 사용 함으로써 사용자 중심의 응용프로그램을 작성하기에 유연하게 하였다.

ABSTRACT

What Ubiquitous means "being or existing anywhere, anytime"in Latin, which is, in other words, the users are able to access the network no matter where they are, what kind of network or computer terminals they use. This paper focuses on the implementation of hardware system. The first part of the sytem is the sensor node which transmits the sensor data from node to ARM11 platform through the Zigbee network wirelessly. The other part of the system is the ARM11 platform which receives and displays the sensor data. ARM11 platform is sink node. The ARM11 platform is based on ARM11 architecture and ported with Linux OS. Qtopia is used as Window Manager in order to make applications. The highly efficient ARM11 processor, S3C6400 MPC is the main part of the ARM11 platform.

키워드

ARM11, Embedded Linux, Zigbee,Ubiquitous computing

* 아주대학교 NCW공학과 박사과정
** 아주대 전자공학과 교수

접수일자 2008. 12. 17
심사완료일자 2009. 02. 25

Ⅰ. 서 론

미국 캘리포니아주에서는 2007년 한 해에만 800평방 마일 이상의 산림이 화재로 소실되어 환경에 막대한 손상을 입혔으며 생태계 중 상당 부분이 유실되었다. 산림은 오실 가스로서 지구 온난화의 원인인 이산화탄소(CO_2)대기를 없애주는 지구의 주요 공기 필터 가운데 하나이다. 화재는 자연적 원인, 담뱃불, 모닥불, 농작 방법 또는 그 밖의 어떤 이유로 발생하든 막대한 산림지 손실을 가져오므로, 이제는 기술을 이용하여 자연 산림지의 보존을 도와야 한다.

현재 적외선 스캐너와 위성 사진 같은 최신 기술들이 화재 감지에 사용되고 있지만, 저마다 그 나름의 단점을 갖고 있다. 위성 기술은 복잡하고 비용이 매우 많이 들며, 적외선 스캐너는 초목 밀집 지역에서는 가시성이 차단된다는 문제가 있다. 오래된 "감시 탑"방식이 세계도처에서 아직 사용되고는 있지만, 이를 위해서는 항상 경계 상태를 유지해야 하므로 다수의 인력 투자가 필요하다. 그러나 저렴한 최신 기술인 저전력 무선 센서 네트워크를 사용하면 화재 발생시 거의 즉각적으로 이를 감지할 수 있다. 산불의 경우에는 작은 불도 바람에 의해 급속히 대형 화재가 될 수 있으므로 시간이 중요한 요소이다. 센서 네트워크는 가능한 빠른 시간 안에 화재발생 지역에 소방 자원을 집중시킬 수 있도록 조기 경보를 제공한다. 또한, Zigbee(IEEE 802.15.4)프로토콜은 각 센서노드에 MAC주소를 지정하므로 적용지역의 위치를 판단할 수 있으며 배터리 교체 없이 수개월 또는 수년간 작동할 수 있으므로 배터리 수명이 문제가 되지 않는다. 추가 전력이 필요할 때는 태양열 전지를 사용하여 설치할 수도 있다.

본 논문에서는 구현한 플랫폼에서 사용된 ARM11프로세서와 리눅스, Zigbee등의 기반 기술에 대해 알아보고 이 기반 기술을 바탕으로 구현한 화재 경보용 시스템에 대하여 알아본다.

Ⅱ. ARM11 코어의 개요 및 특성

ARM은 Advanced RISC Machine의 ARM의 약자로 임베디드 시스템에서 가장 범용적으로 쓰이는 Mobile용 RISC 코어로 저 전력사용 저가의 고효율 core이다. 이 제품군은 상당시간 여러 업체나 연구를 통해 검정되어 온 솔루션으로 사용범위가 계속 확장되는 추세이다. 여러 회사와 제휴하여 OEM방식으로 IP를 제공하며 한국에서는 삼성과 현대 등이 제휴하고 있다. 또 각 회사 모델에 따라 여러 주변 장치를 하나의 CPU 에서 제공하고 있으며 시대적 요구에 의한 기술의 발전에 따라 빠르게 변화를 거쳐 오고 있다.

ARM11 마이크로 아키텍처는 시스템 성능을 대폭 향상시킨다. 최초의 코어 타깃 성능은 350~500MHz 범위이며, 로드맵은 1GHz 이상까지 제시하고 있다. ARM11 마이크로 아키텍처에서는 효율적으로 성능을 발휘시켜 개발자가 구체적인 어플리케이션에 맞춰 성능과 소비전력의 트레이드오프(trade-off)를 이행할 수 있다. 개발자들은 클록 주파수와 공급전압을 모두 변화시킴으로써 소비전력과 성능을 조절할 수 있다. ARM11 마이크로 아키텍처를 기반으로 한 프로세서의 경우, 0.13m 공정 기술을 사용하고 1.2V에서 0.4mW/MHz 미만의 소비전력을 달성했다. 다음 그림 [1]에 ARM1136JF의 블록도를 나타내었다.

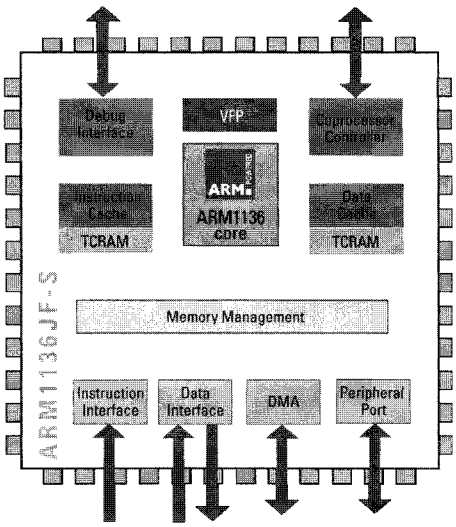

그림 1. ARM1136JF 의 블록 다이어그램
Fig 1. Block Diagram of ARM1136JF

ARM11 마이크로아키텍처는 효율적으로 고성능을 제공하도록 설계되어 코어 클럭의 향상은 물론 전체적인 성능 향상을 가져왔다. 그리고 여러 장치에 대한 컨트롤러를 내장하여 시스템의 확장성을 향상시켰다. ARM11 마이크로아키텍처 는 종전의 ARM10의 6단계 파이프라인에 비해 8단계 파이프라인 명령어 파이프라인으로 바뀌었다. ARM11 마이크로아키텍처 의 8 단계 파이프라인은 종래의 ARM 코어에 비해 40%의 처리 성능 향상을 실현시켰다.

그러나 단계수가 많은 파이프라인 구조는 시스템에 지나친 지연 혹은 대기시간을 초래하기 때문에 효율성이 떨어질 수 있다. 일반적으로, 단계수가 많은 파이프라인에서는 일부의 명령이 앞의 명령의 결과에 의존하기 때문에 지연될 수 있다. ARM11의 파이프라인은 포워딩 기능을 완전히 활용함으로써, 이런 지연을 막을 수 있다. 또한, 단계수가 많은 파이프라인을 가진 프로세서에서는 파이프라인에서 처리되는 명령의 원활한 흐름에 인터럽트가 발생할 경우, 예를 들면 분기 명령이 발생했을 때에도 성능이 떨어질 수 있다. ARM11의 파이프라인은 명령의 흐름을 예상하는 분기예측을 사용하여 이 지연을 막는다. 포워딩과 분기예측은 파이프라인의 Stall을 줄여 효율성을 유지한다. Stall 이란 다음 명령어가 아직 파이프라인을 통과중이기 때문에 프로세서가 기다리지 않으면 안 되는 상황을 가리킨다. 이상과 같은 파이프라인의 최적화에 의해, 유효 대기시간은 ARM9TM 마이크로아키텍처 의 5단계 파이프라인과 같으면서 훨씬 높은 처리 성능을 실현한다.

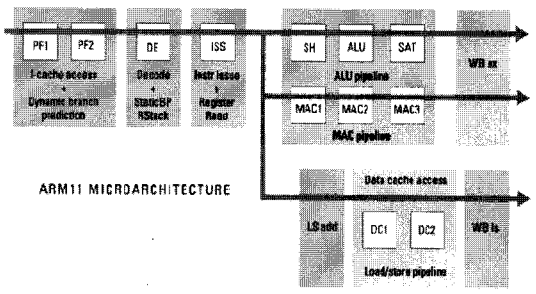

그림 2. ARM11 파이프라인 구조
Fig 2. ARM11 Pipeline Architecture

ARM11 아키텍처 변화면서 ARM은 그 동안 사용해 왔던 32/16bit RISC 아키텍처인 ARMv4T의 명령어 구조에서 성능이 향상된 Powerful ARMv6 명령어셋 구조로 바뀌었다. 이는 기본 명령어 구조에 미디어에 관한 명령어 구조를 통합시킨 것이다. ARMv6 구조는 특히 차세대 소비 기기, 무선통신, 네트워킹, 자동차 제품의 요구를 염두에 두고 개발되었다.

표 1. ARM 제품군의 명령어 체계
Table 1. ARM Family Feature set

Architecture	Feature Set			
	Thumb	DSP	Jazzelle	Media
V4T	✓			
V5TE	✓	✓		
V5TEJ	✓	✓	✓	
V6	✓	✓	✓	✓

Ⅲ. 리눅스 개요

리눅스는 1991년 핀란드에 위치한 헬싱키 대학에서 공부하던 리누스 토발즈라는 학생이 타넨바움 교수가 개발한 미닉스와 유사한 공개 유닉스 커널을 만들기로 결심하면서부터 시작되었다. 리누스 토발즈는 공개 유닉스인 만큼 일반인들에게 커널을 공개하고 사람들이 토발즈에게 피드백을 주며 그것을 토대로 발전하여 왔다. 리눅스 커널은 단일 커널을 사용한다. 하지만 리눅스는 적재 가능한 동적 모듈을 사용해 확장성을 유지한다. 적재 가능한 동적 모듈을 사용하면 불필요한 재시동 없이도 여러 기능을 추가할 수 있다. 리눅스는 커널을 모듈화하여 이용하기 때문에 운영체제의 크기를 원하는 만큼 작게 만들 수 있고 하드웨어 구성요소의 기능을 완전히 활용하기 때문에 성능이 떨어지는 시스템에도 네트워크 서버를 구축하는 것이 가능하다. 그리고 하드웨어에 의존적인 소스코드와 하드웨어에 무관한 소스코드를 명확하게 구분하여 원하는 플랫폼에 쉽게 적용이 가능하다.

리눅스의 가장 큰 장점은 상용 운영체제가 아니라는 것이다. 리눅스는 GPL(GNU General Public License)을 따르는데 GPL은 대표적 공개 소스 라이센스로써 자유롭게 복사하고 배포하며 수정하고 소스코드에 접근할 수 있는 권리를 부여한다. 다만 다음의 제약 조건을 갖는다.

- GPL을 따르는 소프트웨어 소스 코드 일부를 사용해 만든 소프트웨어는 GPL을 따라야 한다.
- GPL을 따르는 소프트웨어 소스 코드를 수정해 개인적으로 사용할 수 없다. 반드시 소프트웨어를 개발한 원작자나 공동체에 환원해야 한다.

Ⅳ. 리눅스 커널 구조

커널이란 운영체제의 핵심을 이루는 요소로서 컴퓨터내의 자원을 사용자 프로그램이 사용할 수 있도록 관리하는 프로그램이다. 커널에서 제공하는 기능은 크게 프로세스의 관리, 파일 시스템, 메모리 관리 네트워크가 있으며, 사용자 프로그램은 이러한 기능들을 정해진 규칙에 따라서 커널에 요구하게 되며, 커널은 이러한 요구를 만족시켜 주어야 한다. 그림 3은 리눅스 커널의 기본 구조를 보여준다.

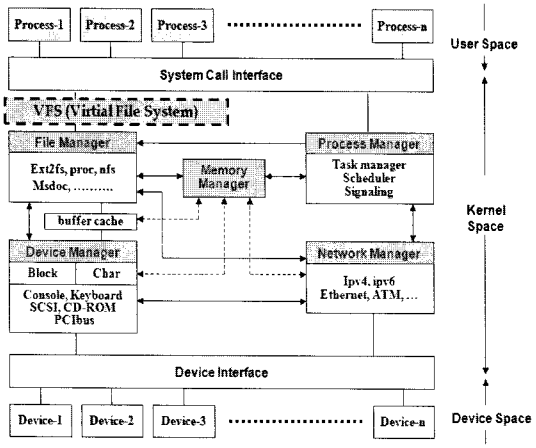

그림 3. 리눅스 커널 구조
Fig 3. Structure of Linux Kernel

- 프로세스 관리

커널은 프로세스를 생성, 제거하고 외부와 통신(입출력)하는 책임을 가진다. (시그널, 파이프, IPC 프리미티브를 통해서) 각각의 프로세스 사이의 통신은 전체 시스템 기능의 바탕이 되며 마찬가지로 커널이 관리한다. 게다가 운영체제 전체에서 가장 중요한 루틴인 스케쥴러도 프로세스 관리의 일부분이다. 일반적으로 커널의 프로세스 관리 활동은 단일 CPU위에 몇 개의 프로세스 추상화를 구현하는 것이다.

- 메모리 관리

컴퓨터의 메모리는 가장 중요한 자원이며 이것을 다루는 정책은 시스템 성능을 결정하는 중요한 요소이다. 커널은 제한적인 가용 자원을 가지고 모든 프로세스에게 가상 주소 공간을 제공해 준다. 커널의 다른 부분은 간단한 malloc/free 루틴에서부터 훨씬도 복잡한 기능을 가진 일련의 Fuction Call을 이용해서 메모리 서브 시스템과 상호 작용을 한다.

- 파일 시스템

리눅스는 유닉스의 파일 시스템 개념에 기반하고 있다. 리눅스에서는 디바이스나 기억장치 상의 데이터를 파일이라고 하는 형식을 통해 액세스한다. 커널은 파일 안의 데이터 형태를 인식하지 않으며, 해당 파일을 이용하는 각각의 프로그램에서 데이터 형태를 인식한다. 모든 파일은 root(/)에서 시작되고 1개의 트리 구조로 관리된다.

- 디바이스 제어

대부분의 작업은 물리적 디바이스를 연결하는 것이다. 프로세서, 메모리, 그리고 극히 일부의 다른 자원을 제외하고는 모든 디바이스 컨트롤 작업은 지정된 디바이스에 따른 코드에 의해서 수행된다. 이 코드를 디바이스 드라이버라고 부른다. 커널은 하드 드라이버에서 키보드와 테이프 장치에 이르기까지 시스템에 있는 모든 주변 장치를 위해 각각의 디바이스 드라이버를 가지고 있어야 한다.

- 네트워크

네트워크는 반드시 운영체제가 관리해야 하는데, 그 이유는 대부분의 네트워크 작업은 하나의 프로세스에 국한되지 않기 때문이다. 들어온 패킷은 비동기적 사건이다. 이 패킷은 프로세스가 이들과 관련된 작업을 시작하기 전에 반드시 모으고 구별하며 발송해야 한다. 시스템은 프로그램과 네트워크 인터페이스 사이에서 데이터 패킷을 전달할 책임이 있으며 네트워크로부터 데이터가 전달되기를 기다리는 프로그램을 정확히 Sleep 상태로 만들고 깨울 수 있어야 한다. 게다가 모든 라우팅과 주소 분석에 관한 일도 커널 안에 구현되어 있다.

V. Zigbee(IEEE 802.15.4)

Zigbee란 IEEE 802.15.4기반으로 저전력과 저가격을 목표로 하는 저속 근거리 개인 무선 통신 국제 표준 스펙이다. Zigbee는 전력소모가 적고 칩 가격이 저렴하고 통신의 안정성이 높아 최근 급속한 발전을 하고 있는 기술 중 하나이다.

Zigbee는 IEEE 802.15.4 표준의 물리 계층(Physical Layer)과 매체접근제어 계층(Medium Access Control Layer)위에 상위 계층으로 네트워크 계층(Network Layer), 응용지원 계층(Application Support Layer), 응용계층(Application Layer)로 구성된다. 또한, 반경 75m정도의 근거리에서 27개의 주파수 중에 하나를 선택하여 250Kbps로 데이터를 전송하며 최대 216=65,536개의 노드를 연결할 수 있다. IEEE 802.15.4표준은 다른 표준과 비교해 볼 때 전력 소모 특성이 우수하고 안정적이며 보안이 제공되며 Mesh형태의 네트워크까지도 지원된다. 그리고 많은 노드들을 연결할 수 있어서 대규모의 센서 네트워크에 적합하다.

Zigbee 기기는 성능에 따라 FFD(Full Function Device), RFD(Reduced Function Device) 두가지로 나눌 수 있다. FFD는 Cordinator, Router, End Terminal중에 어떤 기기로도 사용될 수 있다. 네트워크로는 Star형, Peer to Peer, Cluster-Tree네트워크 형태 모두를 지원한다. 반면에, RFD는 End Terminal로 동작되며 네트워크가 Star 망으로 제한적이고, PAN Cordinator나 Router의 역할은 할 수 없다. RFD의 경우 4Kbyte정도의 작은 메모리로 구현이 가능하므로 FFD와 RFD를 구분함으로써 비용을 줄일 수 있다.

표 2. 지그비 사양
Table 2. Zigbee Specification

구분	내용
Frequency Band	868Mhz, 915Mhz, 2.4Ghz
Modulation	DSSS(Direct Sequence Spread Spectrum
Channel Access	CSMA/CA(Carrier Sense Multiple Access with Collision Avoidance)
Data Rate	20Kbps(868Mhz) 40Kbps(915Mhz) 250Kbps(2.4Ghz)
Range	10~75m
Tx Current	25~35mA(WLAN:400mA Bluetooth: 40mA)
Stanby Current	3uA (WLAN:20mA Bluetooth: 200uA)

VI. 시스템의 구현

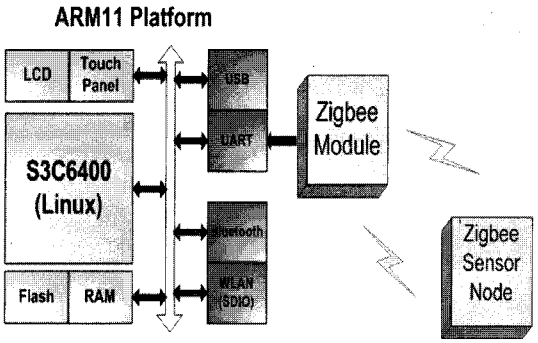

그림 4. 구현 시스템의 블록도
Fig 4. Block Diagram of the Implemented System

구현 시스템은 ARM11기반의 S3C6400 프로세서가 장착되어 있는 저전력 고성능의 모바일 플랫폼으로 임베디드 리눅스를 포팅하여 구현 하였다. 또한, 여러 가지의 유비쿼터스 환경에 맞추어 통합 단말기로의 발전 가능하도록 여러 가지의 유·무선 인터페이스를 제공하며 64MB의 Nand 플래쉬 메모리와 128MB SDRAM을 장착하고 있다. 유선 인터페이스로는 시리얼 통신과 USB OTG 2.0, USB Host 1.1을 지원하며 무선 인터페이스로는 WLAN, 블루투스를 지원한다. Zigbee 센서 노드는 가스 센서, Atmega128, Zigbee 모듈로 구성된다. 가스 센서는 연기 감지 또는 유독 가스 감지를 하여 아날로그 데이터 값을 Atmega128로 전송한다. Atmega128은 센서로부터 입력된 아날로그 값을 A/D 변환하여 Zigbee 모듈로 시리얼 통신 인터페이스를 사용하여 전송하며 Zigbee 모듈을 초기화하고 제어하는 역할을 한다. Zigbee 모듈은 Atmega128로부터 들어온 통신 데이터를 무선 채널을 통해 대상으로 전송한다.

표 3. 시스템 사양
Table 3. the Specification of the System

Part	Model
CPU	S3C6400 ARM11 Processor OS: Embedded Linux (Kernel v2.6.21)
Memory	128MB SDRAM
	64MB Nand Flash(Nand Booting Support)
Data 전송 I/F	USB : Linux Kernel Download UART : Zigbee Module Interface, 각종 제어 및 디버깅
Display	4.8 inch TFT LCD
Debugging	OPENice A1000
Zigbee module	ZBS-200
Gas Sensor	DS-SD-003

Ⅶ. 결 론

본 논문에서 구현한 ARM11 기반 센서 시스템에는 향후 USN 통합단말로의 발전을 위해 멀티 태스킹을 지원하기 위하여 임베디드 운영체제인 리눅스를 포팅하였다. 리눅스를 포팅 하는 절차는 부트로더의 포팅으로 시작된다. 부트로더를 포팅하기 위해서는 구조에 종속적인 소스코드의 수정과 부트로더를 적재할 메모리 관련 소스 코드의 수정이 필요하다. 부트로더는 시스템을 초기화하고 메모리를 초기화하며 부팅 과정을 수행하며 부팅과정이 완료되면 커널에게 시스템의 제어권을 넘겨준다. 커널은 디바이스 관리, 메모리 관리, 파일시스템 관리, 네트워크 관리 등의 기능을 수행한다. 리눅스 시스템은 Windows CE와 달리 커널과 사용자 인터페이스가 분리되어 있다. Windows CE의 경우 커널을 포팅 하였을 경우 바탕 화면이 표시되어 GUI가 구현되지만 리눅스의 경우 윈도우 매니저를 포팅한 이후에야 GUI가 구현된다. 윈도우 매니저에는 X 윈도우, 마이크로 윈도우, PicoGUI, QT/Embedded가 있는데, 본 논문의 구현 시스템에는 QT/Embedded를 포팅하였다. QT/Embedded는 ARM계열의 프로세서를 지원하며 개발 역사가 길어 다른 윈도우 매니저에 비하여 호환성이 좋다.

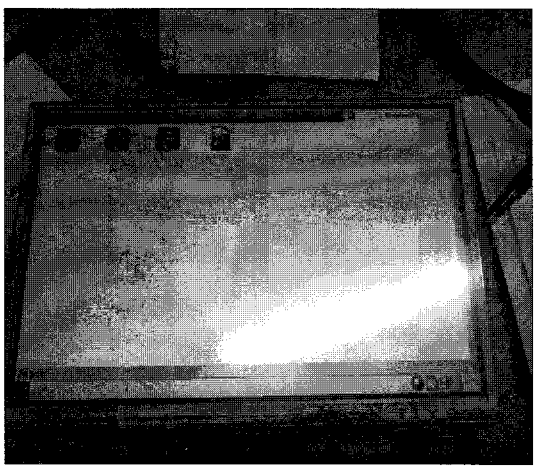

그림 5. 리눅스 포팅
Fig 5. Embedded Linux Porting

본 시스템에는 Zigbee 센서 노드로부터 들어오는 데이터를 사용자가 쉽게 인식하기 위하여 QT기반 GUI 어플리케이션 프로그램을 사용 하였다. 어플리케이션 프로그램은 Zigbee 센서 노드로부터 수신되는 Hex 코드 형태의 데이터 패킷(Packet)을 십진수 형태로 변환시켜 표시해 주는 기능을 한다. Zigbee 센서 노드로부터 송신되는 데이터는 두 개의 패킷으로 구성되어 있으며 각 패킷은 1바이트의 센서 데이터를 포함하고 있다. 이는, ADC에 의해 생성되는 데이터가 2바이트이기 때문이다. 다음 그림6은 본 시스템에서 수신된 데이터를 어플리케이션 프로그램을 통하여 표시하는 사진이다.

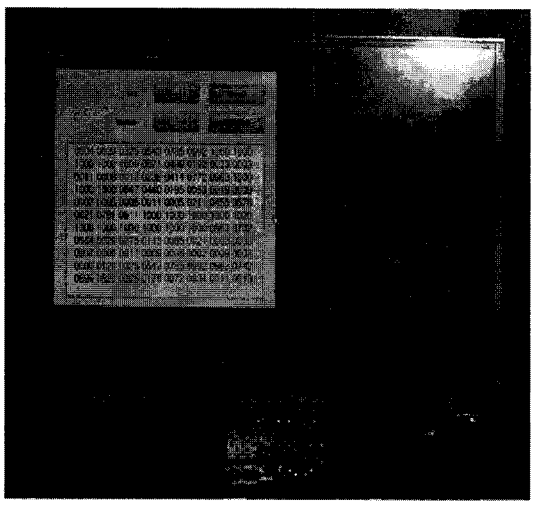

그림 6. 데이터 표시
Fig 6. Data Display

참고문헌

[1] 김종덕, "ARM9 기반의 Ubiquitous Data Assistanct 플랫폼 구현에 관한 연구" , 아주대 석사논문, 2006

[2] 유호준 "ARM11기반의 SAW Sensor 리더 플랫폼에 관한 연구:, 아주대 석사논문, 2008

[3] Callaway, "Wireless Sensor networks: Architecture and Protocols", AUERBACH, Page 293 - 299, 2003

[4] LAN/MAN Standards Committee, "MAC/PHY Specifications for LR-WPANs ", IEEE, page 13~54, 2003

[5] Zigbee Alliance Board of Directors, "Zigbee Specification v1.0", Zigbee Alliance, 2005

[6] Samsung S3C6400X User's Manual Preliminary Rev0.2", 한국, 2008

[7] ATMEL staff, ATmega128/128L, ATMEL, 미국, 2003

[8] Steve Furber, "ARM System-on-Chip Architecture", Addison-Wesley, 2000

[9] ARM DEVELOPER Suite Compilers AND LIBRARIES GRFIDe, 2001

[10] David cormie, " The ARM11 Microarchitecture", ARM Ltd, 2002

[11] 보베이, 체사티 저, 심마로, 이호 역, "리눅스 커널의 이해", 한빛미디어, 한국, 2004

[12] Christopher Hallinan, "임베디드 리눅스", 정보문화사, 한국, 2008

[13] 타카하시 히로카즈 외 2명 "리눅스 커널 2.6 구조와 원리", 한빛미디어, 한국, 2007

[14] 야크무르 저, 김태석 역, "임베디드 리눅스 시스템 구축하기", 한빛미디어, 한국, 2004

[15] www.arm.com

[16] http://www.kldp.net

[17] http://www.ieee.org

[16] http://www.kernel.org

저자소개

선희갑(Hee-Gab Sun)

해군사관학교 전자공학과 학사
연세대학교 전자공학과 석사
아주대학교 NCW공학과 -
박사과정

※관심분야 : Network Centric Warfare, Embedded system, 통신 네트워크,

김영길(Young-kil Kim)

고려대 전자공학과 학사
한국과학기술원 석사
ENST(프랑스) 박사
아주대 전자공학과 교수(현재)

※관심분야 : RFID Platform , Embedded system, 초음파 의료기기, Mobile 의료정보 시스템