

유비쿼터스 컴퓨팅을 위한 사용자 친화적 지능형 공간 구현

최 종 무[†]·백 창 우^{††}·구 자 경^{†††}·최 용 석^{††††}·조 성 제^{†††††}

요 약

본 논문에서는 유비쿼터스 컴퓨팅을 위한 지능형 공간 제어 시스템을 제안한다. 이 시스템은 전등, TV, 오디오, 전자 열쇠 등을 제어하는 일종의 홈/사무실 자동 제어 시스템으로 기존의 시스템에 비해 다음의 4가지 특징을 갖는다. 첫째, 사용자는 언제 어디서나 이 시스템을 사용할 수 있다. 구체적으로 제안된 시스템은 웹 서버의 기능을 제공하고 있으며 따라서 사용자는 인터넷에 유무선으로 연결된 어떠한 컴퓨터의 브라우저로도 접근할 수 있으며, 또한 휴대폰으로 접근할 수도 있다. 둘째, 이 시스템은 음성 인식 기능을 지원한다. 따라서 기존의 컴퓨터 인터페이스에 익숙하지 않은 사용자들도 보다 인간 중심적인 음성 인터페이스를 통해 시스템을 제어할 수 있다. 셋째, 시스템은 사용자의 요청에 반응하는 수동적인 서비스 뿐만 아니라, 사용자 행동의 규칙성을 기반으로 미래를 예측하고 이에 따라 적극적인 서비스도 제공한다. 넷째, 이 시스템은 최근 내장형 기술을 적용하여 구현되었다. 제안된 시스템의 하드웨어는 206MHz로 동작하는 StrongARM CPU, 32MB SDRAM, 16MB 플래시 메모리, 그리고 가전제품의 전원 공급을 제어하는 릴레이 박스(Relay box) 등으로 구성된다. 이러한 하드웨어 플랫폼 상에 내장형 리눅스가 동작하고 있으며, 음성 인식 도구, 내장형 시스템을 위한 웹 서버, 릴레이 박스를 구동하는 GPIO driver 등의 소프트웨어 컴포넌트들이 유기적으로 협력하여 지능형 공간을 제공한다.

Implementation of User-friendly Intelligent Space for Ubiquitous Computing

Jongmoo Choi[†] · Changwoo Baek^{††} · Jakyoungh Koo^{†††}
Yongsuk Choi^{††††} · Seongje Cho^{†††††}

ABSTRACT

The paper presents an intelligent space management system for ubiquitous computing. The system is basically a home/office automation system that could control light, electronic key, and home appliances such as TV and audio. On top of these basic capabilities, there are four elegant features in the system. First, we can access the system using either a cellular phone or using a browser on the PC connected to the Internet, so that we control the system at any time and any place. Second, to provide more human-oriented interface, we integrate voice recognition functionalities into the system. Third, the system supports not only reactive services but also proactive services, based on the regularities of user behavior. Finally, by exploiting embedded technologies, the system could be run on the hardware that has less-processing power and storage. We have implemented the system on the embedded board consisting of StrongARM CPU with 206MHz, 32MB SDRAM, 16MB NOR-type flash memory, and Relay box. Under these hardware platforms, software components such as embedded Linux, HTK voice recognition tools, GoAhead Web Server, and GPIO driver are cooperated to support user-friendly intelligent space.

키워드: 유비쿼터스 컴퓨팅(Ubiquitous Computing), 음성 인식(Voice Recognition), 능동 서비스(Proactive Service), 내장 시스템(Embedded System), 리눅스(Linux)

1. 서 론

유비쿼터스 컴퓨팅(ubiquitous computing)은 1988년 미국 제록스사 팔로알토(Palo Alto) 연구소의 마크 와이저(Mark Weiser)에 의해 처음으로 제창되었다[1, 2]. 유비쿼

터스는 '어느 곳이나 존재한다'라는 의미의 라틴어로 모든 사물에 컴퓨터가 내장되어 있어 장소나 시간에 구애받지 않고 항상 컴퓨터를 사용할 수 있는 환경의 제공을 목표로 한다. 컴퓨팅의 진화 과정 관점에서 보면 유비쿼터스 컴퓨팅은 메인 프레임 기반 컴퓨팅, PC 기반 컴퓨팅에 이어서 제 3세대 컴퓨팅 환경에 해당한다.

* 이 연구는 2003학년도 단국대학교 대학연구비의 지원으로 연구되었음.

† 종신회원 : 단국대학교 정보컴퓨터학부 교수

†† 준 회원 : 동국대학교 컴퓨터공학과

††† 준 회원 : 단국대학교 경영정보전산학과

†††† 준 회원 : 단국대학교 정보컴퓨터학부

††††† 정 회원 : 단국대학교 정보컴퓨터학부 교수

논문접수 : 2003년 9월 2일, 심사완료 : 2004년 1월 6일

유비쿼터스 컴퓨팅을 보다 명확히 파악하기 위해 마크 와이저가 정의한 4가지 속성을 살펴보면 다음과 같다[2].

① 네트워크에 연결되지 않으면 유비쿼터스 컴퓨팅이 아

니다.

- ② 사용자 인터페이스는 편하고 자연스러우며 필요할 때만 눈에 띄어야 한다.
- ③ 사용자 상황(장소, 시간, 날씨, 사용자 특성 등)에 따라 서비스 내용이 변해야 한다.
- ④ 사물에 컴퓨터가 내장되어 있어 언제 어디서나 사용이 가능해야 한다.

본 논문에서는 위의 속성들을 만족하는 사용자 친화적 지능형 공간 제어 시스템을 구현하였다. 이 시스템은 기본적인 기능면에서 기존의 홈 자동화나 사무실 자동화 시스템과 동일하다. 즉 전등, 오디오, TV, 전자 열쇠 등과 같은 가전 제품을 통합하여 제어하는 시스템이다. 하지만 본 논문에서 제안된 시스템은 유비쿼터스 컴퓨팅에서 요구하는 연결성(connectivity), 사용자 친화적 인터페이스(calm interface), 사용자의 상태에 파악하고 이를 기반으로 변하는 능동 서비스(proactive service), 그리고 내장성(embedding)이라는 특징이 있다.

위의 4가지 특징은 본 논문에서 다음과 같이 구체화되었다. 첫째, 연결성을 제공하기 위해 본 논문에서는 최근 주목 받고 있는 웹 프로그래밍 모델을 채택하였다. 즉 제안된 시스템은 항상 네트워크에 연결되어 있으며, 웹 서버의 기능을 제공한다. 따라서 유무선으로 인터넷에 연결된 컴퓨터에서 익스플로러(explorer)나 넷스케이프(netscape), 같은 범용 브라우저를 이용해 시스템에 접근할 수 있으며, 함께 제공되는 기본 웹 페이지를 통해 시스템에 연결된 가전 제품들을 제어할 수 있다. 뿐만 아니라 널리 보급되어 있는 휴대 전화기를 이용하여 시스템에 접근할 수 있으며, 따라서 사용자는 최소한 휴대폰만 있으면 시간과 장소에 상관없이 시스템을 사용할 수 있다.

둘째, 사용자 친화적인 인터페이스를 위해 본 논문에서는 음성 인식(Voice Recognition)을 사용하였다. 음성은 사용자가 쓰기 쉬운 자연스러운(natural) 인터페이스이며 또한 사용자가 인터페이스를 필요로 하지 않을 때에는 인터페이스가 사용자에게 보이지 않는다(invisible). 본 연구에서는 캠브리지 대학에서 개발한 HTK(Hidden Markov Model Toolkit)라는[20] 음성 인식 도구를 내장형 시스템에 맞도록 수정하여 시스템에 이식하였다.

셋째, 제안된 시스템은 단지 사용자의 요청에 반응하여 서비스를 제공하는 수동적인 시스템(reactive system)이 아닌, 사용자의 요구를 미리 예측하고 자동으로 서비스를 제공하는 능동적인 시스템(proactive system)이다. 본 논문에서 설계한 능동 서비스(proactive service)는 사용자 행위의 규칙성에 기반을 둔다. 즉 많은 사용자는 일상생활에서 상당한 규칙성을 보이며, 이것은 시스템이 사용자 행위를 예측하여 지능적으로 동작할 수 있는 근거가 된다. 예를 들어 어떤 사용자가 월요일부터 금요일까지 일정한 시간에

지능형 공간에 입장하여 에어컨을 켜다면, 이러한 수집 자료를 기반으로 시스템이 사용자 입장 시간 5분 전에 자동으로 미리 에어컨을 켤 수도 있다. 능동 서비스는 사용자의 특성과 상태에 따라 변화하는 서비스(context aware service)를 가능하게 하며, 결국 본 연구에서 제안한 시스템을 지능적인 시스템으로 만들게 된다.

넷째, 제안된 시스템은 자원 제약이 있는 내장형 시스템에서 동작할 수 있도록 설계되었다. 실제로 본 연구에서는 저가의 PDA에서 사용하는 206MHz StrongARM 처리기와 16MB NOR 유형 플래시 메모리, 그리고 주변 장치 제어 컨트롤러가 내장된 보드 상에서 시스템을 구현하였다. 이러한 하드웨어 플랫폼 상에 내장형 리눅스(embedded linux)가 기본 시스템 소프트웨어로 동작하고 있으며, 그 위에 웹 서버, 음성 인식 도구, 능동 서비스 관리자(proactive service manager), GPIO 드라이버 등의 소프트웨어들이 협동하여 지능형 공간을 제공한다.

제안된 지능형 공간 제어 시스템의 전체적인 시스템 프로토타입(prototype)은 구현되어 동작 중이며, 현재 전등, 오디오, TV, 그리고 전자 열쇠 등을 제어할 수 있다. 실험 결과 본 연구에서 제안한 특징을 제대로 지원하고 있음을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문과 유사한 지능형 공간을 제공하려는 기존의 연구를 조사한다. 3장에서는 사용자 친화적 지능형 공간을 제공하기 위한 시스템 전체 구조와 설계 내용을 기술하며, 구체적인 구현 방법을 4장에서 설명한다. 5장에서는 실험 결과를 보이며, 마지막으로 6장에서 결론을 맺고 향후 연구 과제를 논의한다.

2. 관련 연구

최근 전 세계적으로 유비쿼터스 컴퓨팅에 대한 연구가 활발하게 진행되고 있다. 미국의 NSF와 DARPA가 주도하는 노매딕(Nomadic)과 퍼베이시브(Pervasive) 컴퓨팅[3-5], 유럽 정보화사회기술계획(IST)의 일환으로 미래기술계획(FET)이 자금을 지원하는 사라지는(Disappearing) 컴퓨팅[6], 일본 총무성이 주도하는 유비쿼터스 네트워크 포럼의 정보기반 컴퓨팅과 TRON 프로젝트[7] 등이 대표적인 예이다. 국내에서도 정부의 u-Korea 정책 하에 관련 연구가 ETRI와 대학들을 중심으로 급속히 증가하고 있다[8].

유비쿼터스 컴퓨팅에 대한 연구는 TTT(Things That Think)와 입을 컴퓨터(Wearable Computer), Smart-Its 등 차세대 단말기에 대한 연구, WPAN(Wireless Personnel Area Network)과 임시망(Ad Hoc Network) 등 통신 프로토콜에 대한 연구, DCOM과 JINI, HAVi 등 미들웨어에 대한 연구, RTOS와 지능형 에이전트(Intelligent Agent) 등 소프트웨어에 대한 연구, MEMS와 SoC, 생체 인식과

유비쿼터스 건강관리(ubiquitous health care)에 대한 연구 등 다양한 주제로 진행된다.

이 중에서 본 논문의 주제인 지능형 공간과 관련된 연구에는 MIT의 Oxygen 프로젝트[9], CMU의 Aura 프로젝트[10, 11], HP의 CoolTown[12, 13], Berkeley의 Endearvor 프로젝트[14], Georgia Tech.의 AwareHome 프로젝트[15, 16], UW의 Portolano 프로젝트[17], MS의 EasyLiving 프로젝트[18], ETRI 중심의 스마트 홈/스마트 타운 프로젝트[8] 등이 있다. 이 중에서 본 논문과 밀접하게 관련된 3가지 프로젝트를 심도 깊게 분석하면 다음과 같다.

MIT에서 진행하고 있는 Oxygen 프로젝트는 마치 산소와 같이 편하고 자연스럽게 사용할 수 있는 인간 중심의 연산과 통신 환경을 제공하려는 연구이다[9]. 이 연구는 H21(Handheld Devices), E21(Environmental Devices), N21(Network Technologies), O21(Software Technologies) 등의 세부 분야로 진행되고 있으며, 이 중에서 E21의 대표적인 프로토타입이 지능형 방(iRoom : intelligent Room)이다. E21은 말이나 행동, 그림 등의 인터페이스로 사람과 대화하며 이를 위해 많은 스피커와 카메라, 지능형 보드, Cricket 위치 파악 시스템, 그리고 커튼과 전등 등 실제 사물을 제어하는 시스템으로 구성된다. 또한 E21은 네트워크에 연결되어 있으며 이 공간에 존재하지 않는 사용자는 휴대폰이나 PDA 등 다양한 H21을 통해 E21의 기능을 사용할 수 있다. Georgia Tech.의 AwareHome 프로젝트[15, 16]와 MS의 EasyLivingch[18] Oxygen과 유사한 목적을 갖으며 다양한 기법을 시도하는 프로젝트이다.

한편, CMU에서 진행 중인 Aura 프로젝트에서는 향후 컴퓨팅 환경에서 중요한 자원이 CPU나 메모리 또는 네트워크 대역폭(bandwidth)이 아니라 인간의 주의(attention)라고 파악한다[10, 11]. 따라서 인간의 주의력이 흩어지지 않는(distraction-free) 컴퓨팅 환경 구현이 프로젝트의 주요 목표이며, 이것은 능동성(proactivity)과 자체 튜닝(self-tuning)이라는 핵심 기술로 구체화되고 있다. 능동성이란 시스템이 외부 이벤트에 대해 단순히 반응하는 것이 아니라 미래를 예측하고 적극적으로 대응하는 능력이며, 자체 튜닝은 예측에 따라 시스템의 상태를 적절하게 동적으로 재구성하는 능력이다. 이러한 Aura의 능력은 소프트웨어 계층 구조(layer architecture)를 통해 구현된다. 예를 들어 시스템 층에 존재하는 Coda라는 모듈은 파일에 대한 비연결 연산(disconnected operation)을 지원하며, 응용과 사용자 층에 존재하는 Prism이라는 모듈은 사용자의 의도를 파악하고 이에 따라 소프트웨어 컴포넌트를 재구성하는 기능을 제공한다. 결국 Aura는 시스템 전체적인 분야에서 동적 재구성과 문맥 인식이 가능한 지능형 에이전트 기술을 이용하여 적극적이고 능동적인 서비스 제공 기법을 연구하고 있으며, Berkeley의 Endearvor 프로젝트[14]와 UW의 Portolano 프로젝트[17] 등도 이와 유사한 철학을 갖는 프

로젝트이다.

마지막으로 HP에서 진행 중인 쿨타운(CoolTown) 프로젝트를 살펴보자[12]. 이 프로젝트에서는 현실 세계를 사람, 사물, 장소라는 3가지 객체로 모델링하고, 각 객체에 웹 표현(web presence)을 연결시켜 실세계 객체에 디지털 정보를 증강(Augment)시킨다[13]. 예를 들면 장소는 장소 관리자(PlaceManager)라는 도구를 이용해 자신의 웹 페이지를 갖는 웹 서버로 동작하며, 사용자는 이 웹 표현을 통해 장소가 소유하고 있는 사물과 사람에 대한 정보를 파악하고 제어하게 된다. 또한 사물과 사람도 자신의 정보를 웹 표현으로 제공하며, 따라서 내용 중심(contents oriented) 컴퓨팅 환경을 제공하게 된다. 한편, 각 객체에 연결된 웹 표현의 URL은 사용자가 해당 객체에 접근했을 때 IR Beacon이나 RFID tag를 이용한 자동 센싱 방법으로 투명하게 얻을 수 있다. 따라서 사용자는 장소를 이동함에도 불구하고 현재 위치에 적합한 문맥 인식 서비스를 자동으로 얻을 수 있게 된다. 지능형 공간이 웹 서버의 기능을 수행하도록 하는 본 논문의 방법도 객체에 웹 표현을 연결하여 증강된 정보를 제공하는 쿨타운 연구자들과 같은 접근 방법을 사용한 것이며, 최근 브라우징이 가능한 많은 단말기의 출시를 고려해 볼 때 현실화 가능성이 매우 높은 접근 방법이다.

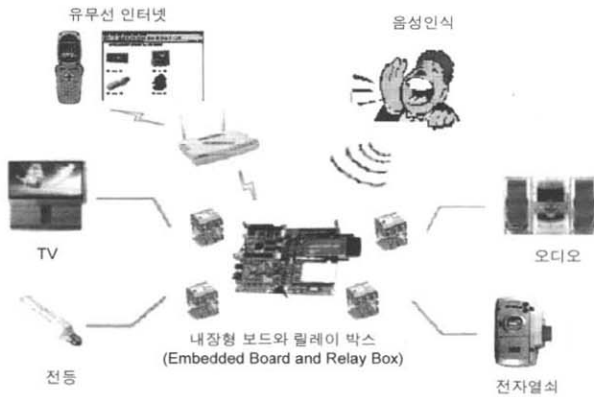
지금까지 사용자 친화적인 지능형 공간을 제공하려는 기존의 프로젝트들을 살펴보았다. 본 연구도 이러한 연구 중의 하나이며 언제 어디서나 접근 가능, 음성 인식 기반 사용자 친화적 인터페이스 제공, 사용자의 규칙성에 따른 능동 서비스 제공, 자원 제약이 있는 내장형 환경에 효과적인 수행 가능 등의 특징을 갖는다. 특히 능동 서비스(proactive service)는 리눅스의 데몬(daemon) 기능을 이용하여 구현되었으며, 사용자 행위의 규칙성을 파악하는 방법과 설정 파일 형식은 본 연구에서 처음 제안된 것이다. 능동 서비스는 사용자에게 서비스를 자동으로 제공하여 편의성을 높일 수 있으며, 또한 각 사용자에게 적합한 서로 다른 서비스를 지능적으로 제공할 수 있다는 장점을 갖는다.

3. 시스템 설계

(그림 1)은 본 논문에서 제안한 시스템의 전체적인 구조를 개념적으로 도시한 것이다. 전체 시스템에서 핵심적인 기능을 담당하는 것은 그림의 가운데 존재하는 내장형 보드이다. 이 보드는 StrongARM CPU, SDRAM, 플래시 메모리, 주변 장치 제어기, 그리고 릴레이 박스(Relay box)로 구성되며, 그 위에 내장형 리눅스가 기본 시스템 소프트웨어로 동작한다.

사용자는 음성 인터페이스 또는 인터넷으로 연결된 PC나 휴대폰을 이용해 내장형 보드에 접근할 수 있다. 이를 위해 본 연구진은 음성 인식 도구, 웹 서버, 능동 서비스(proactive service) 관리자 등의 소프트웨어 컴포넌트를

이식/구현하였다. 음성 인식 도구는 캠브리지 대학에서 개발한 HTK라는 패키지를 사용하였으며[20], 웹 서버는 내장형 웹 서버를 위해 작은 크기로 개발된 GoAhead라는 모듈을 사용하였다[19]. 능동 서비스 관리자는 사용자의 요청 시간, 요청 내용 등의 로그를 수집하고, 이를 기반으로 적극적인 지능형 서비스를 제공한다.



(그림 1) 지능형 공간 관리 시스템 전체적인 구조

한편, 내장형 보드는 릴레이 박스(Relay box)를 통해 전등 및 가전 제품을 제어한다. 릴레이 박스란 전원의 흐름을 개폐하는 일종의 전자 스위치로, 보드의 GPIO 포트에 연결되어 있다. 따라서 보드에서 GPIO의 특정 레지스터를 조절하여 핀에 3.3V의 전원을 흐르게 하면, 릴레이 박스 내부의 전자석이 접지되어 가전제품에 220V의 전원을 흐르게 한다. GPIO를 제어하기 위해 본 연구진은 GPIO 드라이버를 커널 내부에 구현하였다. 응용 프로그램은 /dev/GPIO라는 특수 장치 파일을 통해 드라이버에 접근할 수 있다.

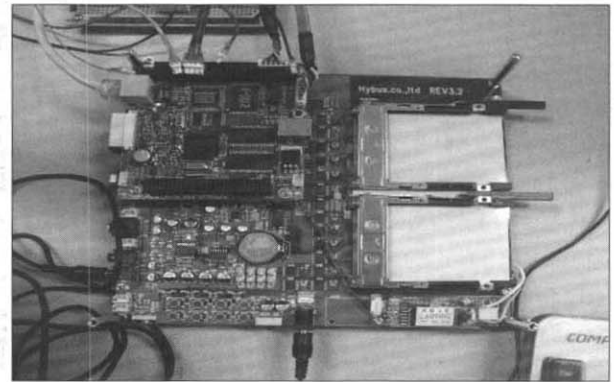
4. 구 현

4.1 하드웨어 구성 요소

(그림 2)는 본 연구에서 사용한 내장형 보드 사진이다. 이 보드는 206MHz로 동작하는 인텔의 StrongARM 처리기, 32MB 크기의 삼성 SDRAM, 16MB 크기의 인텔 Strata NOR 유형 플래시 메모리가 탑재되어 있다[21]. 또한 네트워크 카드, LCD 제어기, UART, 인터럽트 제어기, JTAG 인터페이스 등이 보드에 통합 내장되어 있다. 이 보드는 휴대형 기기나 내장형 기기에 적합하도록 저전력 관리 기능을 제공하고 있으며, USB, Serial, GPIO, JTAG 등 다양한 포트를 제공하여 유연한 개발 환경 구현을 가능하게 한다. <표 1>은 보드의 하드웨어 구성 요소를 요약한 것이다.

(그림 3)의 내장형 보드 위에 있는 하드웨어 모듈은 본 연구진이 제작한 릴레이 박스이다. 이 모듈은 릴레이(Relay)와 트랜지스터(Transistor) 그리고 외부 전원으로

구성되며, GPIO 핀을 통해 보드와 연결된다. 릴레이의 내부에는 높은 전기의 흐름을(AC 220V) 단속할 수 있는 물리적인 스위치가 내장되어 있다. 그리고 이 전기적 접점은 릴레이의 전원 단자에 연결되어 있는 전자석의 힘에 의해 개폐된다. 따라서 GPIO 핀을 통해 전류가 흐르면, 이것이 릴레이의 전자석을 구동해 스위치의 동작을 야기하게 되고 결국 전등이나 가전 제품의 전원을 제어하게 된다. 그런데, 본 연구진이 사용한 보드에서 출력되는 전류는 0.3A로, 릴레이가 요구하는 1A에 비해 너무 적어, 실제 구동에 어려움이 있었다. 이 문제를 해결하기 위해 트랜지스터를 릴레이 박스에 추가하여 제작하였다. 결국 GPIO 레지스터의 특정 비트가 설정되면 대응되는 GPIO 핀을 통해 전원이 트랜지스터의 base에 흐르게 되며, 이것이 릴레이의 스위치를 개폐시켜 외부에서 들어오는 가전 제품의 전원을 제어하게 되는 것이다.



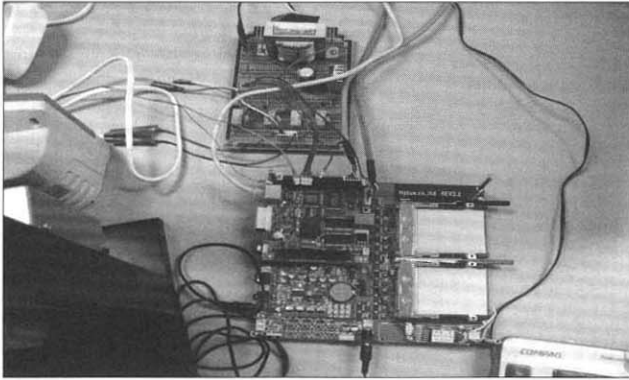
(그림 2) 내장형 보드

<표 1> 보드 구성 요소 (Hardware Components)

구성 요소 (components)	사 양 (specification)
CPU	Intel StrongARM SA1110(206MHz)
주 메모리	Samsung SDRAM 32M
플래시 메모리	Intel Strata Flash 16M
네트워크 카드	CS8900A 10BaseT
UART	RS232, IrDA 연결 가능
USB	Slave - B type
Serial connector	5핀, 2.5mm Housing Header, Serial port 2
LCD Connector	22핀, 1.0mm FPC Connector, STN Color LCD 연결
Touch Connector	4 핀, 1mm Housing Header, Right Angle
PCMCIA	2 Slot
Stereo Audio	UDA1341
LED	GPIO 기능사용(GPIO1-27)
TFT LCD Interface	LCD Interface(GPIO 2-9를 통해 확장)
Push Button	GPIO 기능사용(GPIO 1-16)
JTAG	JTAG Port

릴레이 박스는 GPIO 핀을 통해 보드와 연결되며, StrongARM 처리기 기반 보드는 28개의 GPIO를 지원한다. 본 연구에서는 GPIO2, GPIO3, GPIO4, GPIO5번을 사용하였다. GPIO

에 관계된 레지스터의 주소와 기능을 정리하면 <표 2>와 같다. 구체적인 GPIO 프로그래밍 방법은 <표 2>의 GPDR에서 사용하려는 GPIO 핀에 대응하는 비트를 1로 설정하여 방향을 출력으로 전환하고, GPSR의 동일한 비트를 1로 설정하면 해당 GPIO 핀으로 3.3V의 전원이 흐르게 된다.



(그림 3) 릴레이 박스

<표 2> GPIO 레지스터 주소와 기능

물리 주소	레지스터 이름	기능
0x90040000	GPLR	GPIO pin level register
0x90040004	GPDR	GPIO pin direction register
0x90040008	GPSR	GPIO pin output set register
0x9004000C	GPCR	GPIO pin output clear register
0x90040010	GRER	GPIO rising-edge register
0x90040014	GFER	GPIO falling-edge register
0x90040018	GEDR	GPIO edge detect status register
0x9004001C	GAFR	GPIO alternate function register

4.2 소프트웨어 구성 요소

본 논문에서 사용한 리눅스 커널은 버전 2.45로 StrongARM 처리기에 최적화된 것이다. 이것은 PC용 일반 커널 소스인 kernel-2.4.5를 기본적으로 설치하고, ARM 패치인 rmk7과 SA1110 패치인 np2, 그리고 보드 종속적인 패치인 hyper104를 추가하면 만들 수 있다[21]. 또한 본 연구진은 커널의 크기를 줄이기 위해 많은 부분을 모듈(module)로 생성하여 보드에 존재하지 않은 장치를 위한 드라이버는 커널 구성에서 제외하였다. 또한 커널에 내장되어 있지만 수행 상에 필요 없는 부분도 - 예를 들어 기존의 COFF 바이너리 형식을 지원하기 위한 커널 소스 부분이나 플로피 블록 장치 드라이버 그리고 내장형 보드에서 사용하지 않는 통신 프로토콜 등 - 커널을 구성할 때 제거하였다. 이 결과 생성된 커널 크기는 702KB이며, 이 크기는 PC 상에서 동작하는 일반적인 커널 크기의 57% 정도이다.

이 커널은 JFFS2(Journaling Flash File System 2)와 Cramfs 같은 내장형 시스템을 위한 파일 시스템과 작은 크기의 윈도우 시스템인 TinyX를 지원한다. 현재 지원되는 디바이스 드라이버에는 이더넷(Ethernet), USB, UART, 적외

선 통신(IrDA), 터치스크린, LCD를 위한 프레임 버퍼(Frame-Buffer) 등이 있다. 이러한 커널 위에 제안된 시스템의 특징을 제공하는 소프트웨어 컴포넌트들이 동작한다. 주요한 컴포넌트에는 음성 인식 도구, 웹 서버, 능동 서비스 관리자, GPIO 드라이버가 있다. 이제부터 각 컴포넌트를 자세히 살펴보자.

4.2.1 음성 인식 도구

음성 인식을 위한 도구로 소스가 공개되어 있는 것에는 HTK, JULIUS, CMU의 Sphinx, Mississippi 대학의 ISIP 등이 있으며, 이중 내장형 시스템에 가능한 도구로는 HTK와 Sphinx가 있다. 본 연구에서는 HTK를 이식하여 사용하였으며, 이는 HTK가 DARPA의 벤치마크 테스트에서 세계 최고 수준의 인식률을 인정받았기 때문이다[22].

HTK는 영국 캠브리지 대학에서 개발한 음성 인식 도구로 음성 파일(wav file)로 구성된 음성 데이터베이스를 생성하는 부분, HMM(Hidden Markov Model)을 생성하는 부분, 학습 부분, 그리고 인식 부분 등으로 구성된다[20]. 이 중에서 HMM은 음성 데이터베이스로부터 음원 특성 요소를 추출하여 계산한 가우시안(Gaussian) 확률 분포 함수의 평균, 분포 등으로 구성된다. 그리고 학습 부분에서는 인식을 위한 문법을 정의하게 된다. 이 과정을 통해 얻어진 HMM과 문법은 이후 인식 단계에서 이용된다. 즉 음성 인식은 HMM을 기반으로 새로 들어온 음성의 각 단위를 파악하고, 이들이 문법을 따르는지 확인하는 과정으로 이루어진다. HTK는 화자 종속적인 음성 인식과 독립적인 음성 인식을 모두 지원하며, 한글 인식도 가능하다는 특징이 있다. 또한 성능 향상을 위해 다중 쓰레드를 지원하며, 토큰 전달 방식에 의한 탐색기를 사용한다.

4.2.2 웹 서버

제안된 시스템이 범용 브라우저를 이용해 접근될 수 있도록 하기 위해 본 연구진은 내장형 보드에 웹 서버를 구성하였다. 웹 서버를 구성할 때 가장 일반적으로 사용되는 패키지는 아파치(Apache)이다. 하지만 아파치는 libutil.so와 libpam.so 같은 많은 공유 라이브러리와 아파치 종속적인 모듈들을 사용하기 때문에 자원 제약이 있는 내장형 시스템에서 사용하기에는 성능상에 문제가 있다. 따라서 본 연구진은 내장형 시스템을 위한 대표적 웹 서버인 GoAhead와 Boa 등을 조사하였으며, CGI와 보안 기능을 지원하는 GoAhead를 선택하였다.

GoAhead사에서 공개한 GoAhead 2.1 웹 서버는 HTTP 1.0 프로토콜을 지원한다[19]. 또한 이식성이 뛰어나 디렉터리 등 약간의 환경 변수만 적절히 설정하면 제안된 시스템에 그대로 이식할 수 있다. 한편, 웹 서버는 인터넷에 연결된 다양한 사용자들이 접근할 수 있기 때문에 보안 기능이 매우 중요하다. 따라서 본 연구진은 SSL(Secure Socket Layer)과 DAA(Digest Access Authentication)를 이용하

여 패킷 암호화와 사용자 인증 기능을 사용할 수 있도록 구성하였다. 또한 페이지 단위 또는 하위 트리 단위 별로 차별된 접근 권한을 주거나, 세션시간 제한(session-time limits) 기법으로 일정 시간 이후 다시 인증을 요구하는 등의 부가 기능도 지원할 수 있도록 하였다.

4.2.3 능동 서비스 관리자

능동 서비스 관리자(proactive service manager)는 사용자 행동의 규칙성을 기반으로 자동 서비스를 제공하는 소프트웨어 컴포넌트로 자료 수집 함수, 자료 분석 함수, 그리고 자동 서비스 수행 함수로 구성된다. 자료 수집 함수는 사용자의 요청(음성 요청 또는 웹 서버를 통한 요청)이 발생할 때마다 호출된다. 그리고 요청 대상, 요청 내용, 요청 시간 등의 정보를 기록한다.

한편, 자동 서비스 수행 함수는 주기적으로 수행되는 데몬(daemon) 형태로 구현되었다. 이것은 설정 파일인 /etc/ispac.conf의 내용에 의해 수행 방식이 결정된다. 설정 파일에는 대상 장치, 행위, 능동 서비스 사용 여부, 수집 데이터 구분 기준, 능동 서비스 제공 기준인 최소 횟수와 점유율 등의 정보가 기록된다. (그림 4)는 /etc/ispac.conf의 내용을 예시한 것이다.



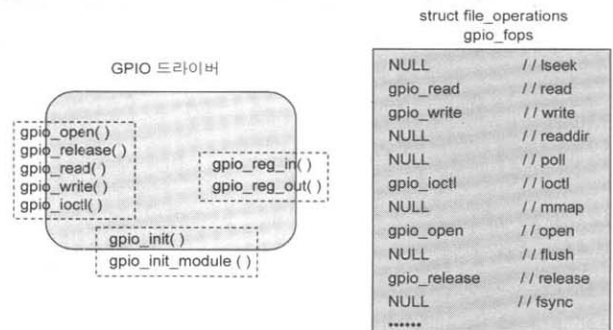
(그림 4) /etc/ispac.conf 파일 내용

(그림 4)에 예시된 /etc/ispac.conf 파일 내용을 해석하면 다음과 같다. 첫 번째 엔트리의 3번째 라인을 보면 전등의 불을 켜는 요청에 대한 능동 서비스가 허용되어 있음을 알 수 있다. 그리고 24시간을 주기로, 0.25시간(15분)을 기본 시간 단위로 하여 요청이 발생한 횟수를 분석하도록 설정되어 있다. 또한, 요청의 전체 횟수가 10 이상이고 특정 기본 시간 단위에 발생한 요청의 점유율이 80% 이상이라면, 그 시간에 요청을 자동으로 발생시키도록 되어 있다. 따라서 어떤 사용자가 10번 이상 전등 켜는 행위(action)를 요청했고 그 중에 80% 이상의 요청이 오전 8시 45분~9시 사이에 발생했다면, 이제부터 능동 서비스 관리자는 그 시간이 되었을 때 자동으로 전등 켜는 요청을 발생시키게 된다.

현재 구현에서는 자동 서비스 수행 함수가 5분을 주기로 호출되도록 하였다. 자동 서비스 수행 함수는 우선 서비스가 허용되어 있는 엔트리에 대하여 자료 분석 함수에게 현재 시점에 능동 서비스를 수행해야 하는지 문의한다. 자료 분석 함수는 이미 존재하는 통계 정보를 기반으로 예/아니오 응답을 리턴 한다. 주기적으로 수행되는 자동 서비스 수행 함수의 계산 부하를 줄이기 위해, 실제로 통계 정보를 분석하는 작업은 자동 서비스 수행 함수가 수행될 때마다 이루어지는 것이 아니라 자료 수집 함수에서 새로운 자료가 수집될 때 이루어지도록 하였다. 또한 매번 설정 파일을 읽는 것이 아니라 파일의 내용이 수정되었을 때만 읽도록 구현하였다.

4.2.4 GPIO 디바이스 드라이버

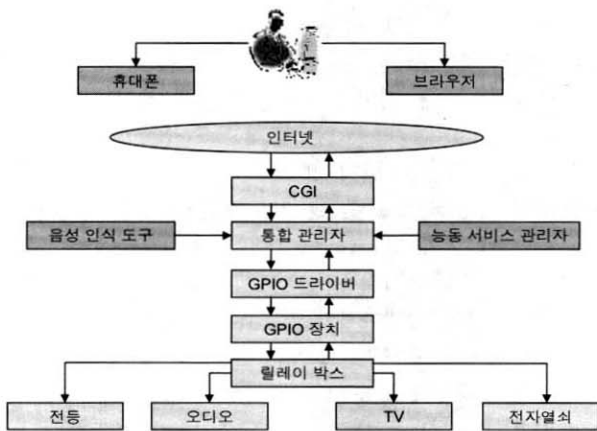
GPIO 디바이스 드라이버는 릴레이 박스를 제어하는 기능을 담당한다. (그림 5)는 GPIO 드라이버 구조를 인터페이스 중심으로 도시한 것이다. GPIO 드라이버는 초기화를 담당하는 부분, 리눅스 커널과 통신하는 부분, 하드웨어를 제어하는 부분 등 크게 3부분으로 구성된다. 드라이버 초기화는 커널에서 직접 호출 또는 리눅스의 모듈 기능을[23] 이용하여 모듈로 적재될 때 호출되도록 하는 2가지 방법이 가능한데, 전자는 gpio_init 함수가 후자는 gpio_init_module 함수가 담당한다. 한편, 커널과 통신하는 부분은 gpio_open, gpio_release, gpio_read, gpio_write, gpio_ioctl로 구성되며, 리눅스 커널에 등록될 때에는 file_operations 자료 구조를 이용해 등록된다[23]. 이 함수들 중에 응용 프로그램이 GPIO 핀을 선택하거나 설정할 때 사용하는 함수는 gpio_ioctl이다. 마지막으로 GPIO 하드웨어를 제어하는 부분은 gpio_reg_in, gpio_reg_out이며, <표 2>의 GPDR이나 GPSR 레지스터의 내용을 읽거나 쓰는 작업을 수행한다. 이 함수들은 gpio_ioctl 내부에서 호출된다.



(그림 5) GPIO 드라이버 구조

본 연구진은 드라이버를 구현할 때 장치 주 번호(device major number)로 211번을 사용하였으며, 부 번호(minor number)로 0을 사용하였다. 그리고 /dev/GPIO라는 특수 장치 파일을 'mknod' 명령으로 생성하였으며, 결국 응용 프로그램은 이 특수 장치 파일을 통해 GPIO 드라이버에 접근하게 된다.

(그림 6)은 지금까지 설명한 각 소프트웨어 컴포넌트를 전체적으로 도시한 것이다. 사용자의 요청은 휴대폰이나 PC 상의 브라우저에서 만들어져 웹 서버를 통해 전달될 수도 있으며, 음성 인식이나 능동 서비스 관리자에 의해 만들어질 수도 있다. 생성된 요청은 관련된 명령과 인자로 해석되어 GPIO 드라이버로 전달된다. 한편 웹 서버나 음성 인식에서 생성된 요청은 능동 서비스 관리자에게 전달되어 통계 자료로 수집된다. 결국 요청은 관련된 각 소프트웨어 컴포넌트에 적절하게 전달되어야 하며, 본 연구진은 통합 관리자(Integrated Manager)라는 컴포넌트를 추가하여 이러한 기능을 일관되게 통합 관리하도록 하였다.



(그림 6) 사용자 요청의 전체적인 제어 흐름

5. 실험 결과

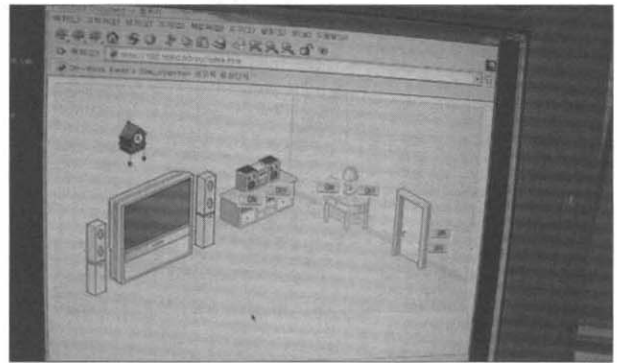
본 절에서는 구현된 시스템의 실험 결과를 기술한다. 실험 내용은 브라우저 및 휴대폰을 이용한 시스템 제어 실험, 음성 인식에 기반한 시스템 제어 실험, 그리고 능동 서비스 관리자에 의한 시스템 제어 실험으로 구성된다. 실험에서 사용한 시스템에는 전등, 오디오, 그리고 문을 제어하는 전자 열쇠가 연결되어 있다.

(그림 7)은 구현된 시스템의 전체 사진과 시스템을 제어하기 위한 브라우저 수행 예를 보여준다. (그림 7)(a)에 도시된 전체 시스템은 내장형 보드와 릴레이 박스, 그리고 제어되는 가전 제품인 전등과 오디오, 전자 열쇠로 구성되어 있다. 그리고 (그림 7)(b)에 도시된 브라우저의 웹 페이지는 가전 제품 이미지들과 전원 제어를 위한 인터페이스로 구성된다. 이 웹 페이지의 URL은 embedded.dankook.ac.kr/HNS/menu.html이며 컴퓨터에 익숙하지 않은 사용자라도 직관적으로 이해할 수 있도록 쉽게 구현하였다. 한편 사용자는 휴대폰으로도 시스템을 제어할 수 있으며, 이때는 embedded.dankook.ac.kr/HNS/html/index.html의 웹 페이지가 사용된다.

(그림 8)은 브라우저를 이용해 전자 열쇠를 제어하는 실험 결과를 보여준다. (그림 8)(a)는 (그림 7)(b)에 도시된



(a) 시스템 전체 사진



(b) 시스템 제어를 위한 브라우저 사진
(그림 7) 시스템 전체 사진과 브라우저 사진



(a) 전자 열쇠 잠금 해제 상태

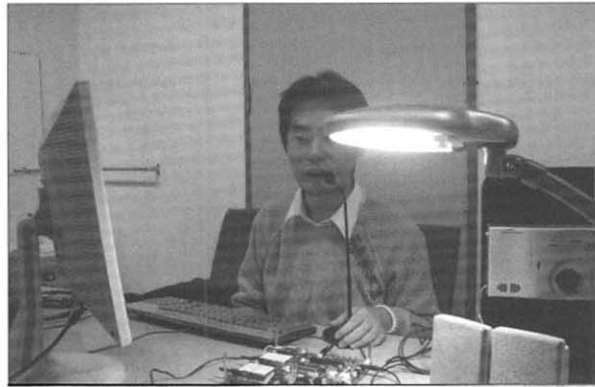


(b) 전자 열쇠 잠금 상태
(그림 8) 브라우저를 이용한 전자 열쇠 제어 실험

브라우저에서 문에 연결된 'off' 버튼을 누른 이후의 상태이다. 반면 (그림 8)(b)는 문에 연결된 'on' 버튼을 누른 이후의 상태이다. 우리는 이 실험을 통해 구현된 시스템이 브라우저에서 요청된 명령을 제대로 전달받고 있으며, 또한 GPIO 드라이버와 릴레이 박스를 통해 적절하게 전자 열쇠를 제어하고 있음을 알 수 있다((그림 8)에서 화살표 표시 부분).



(a) '전등 온' 요청 이전



(b) '전등 온' 요청 이후

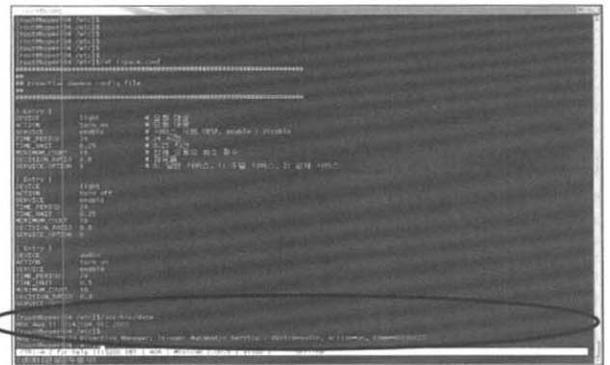
(그림 9) 음성 인식을 이용한 전등 제어 실험

(그림 9)는 음성 인식에 의한 시스템 제어 실험 결과를 보인 것이다. 본 연구진은 음성 인식 도구인 HTK를 제안된 시스템에 이식할 때 보드에 내장된 UDA1341 오디오 드라이버와 충돌이 발생하여 어려움을 겪었으며, 또한 70여 단계로 구성되는 학습 부분에서 발생한 예외와 다양한 선택 사항 때문에 많은 시행착오를 겪었다. 이식 결과 생성된 실행 파일의 크기는 음성 데이터베이스를 구축하는 HSLab 파일이 763KB, 학습을 담당하는 HEReset 파일이 451KB, HMM를 생성하는 HCopy 파일이 363KB, 인식을 담당하는 HVite 파일이 533KB이다. 이 중에서 HSLab, HEReset, HCopy는 문법과 HMM을 생성하기 위해 필요한 것이며, 이러한 정보들이 생성되면 실행 중에는 더 이상 사용되지 않는다. 따라서 실제 보드에는 음성 인식을 위한 HVite 파일만 존재하게 된다. 그리고 현재 구현에서는 '전등 온', '열

쇠 온', '오디오 오프' 등 20여 문장을 인식할 수 있도록 HMM을 구성하였으며, 이를 위해 생성된 HMM 저장 파일인 HMMdefs의 크기는 61KB이다.

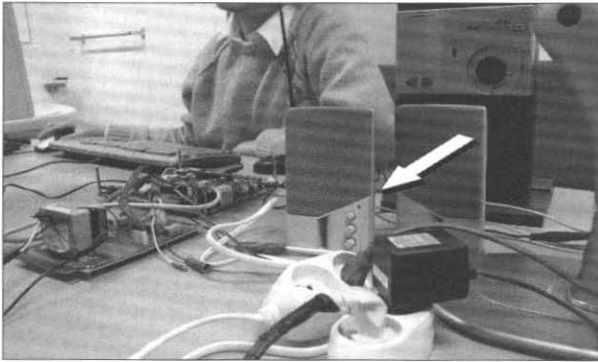
(그림 9)(a)는 '전등 온'이라는 음성 요청을 수행하기 이전 상태이며, (그림 9)(b)는 요청을 수행한 이후 상태이다. 이 실험은 구현된 시스템에서 사용자의 음성 요청을 제대로 인식하고 있음을 보여주며, 이것은 음성이라는 보다 인간 중심적인 인터페이스를 통해 가전 제품을 제어할 수 있음을 의미한다. 음성 인식 성능에 대한 정량적인 실험 결과로는 음성 인식률이 80% 정도이며, 음성 요청과 장치 제어 간에 지연 시간은 1~2초로 분석되었다. 하지만 음성 인식을 문제와 속도 문제는 HTK에 종속된 것으로 본 논문의 연구 범위를 벗어나며, 따라서 자세한 분석 결과를 기술하지는 않았다.

(그림 10)은 능동 서비스 관리자의 실험 결과이다. 이 실험을 하기 전에 본 연구진은 오전 8시~8시 15분 사이에 오디오 켜는 작업을 이미 10번 수행했다. 이 상태에서 우선 'cat' 명령을 이용해 /etc/ispac.conf 설정 파일의 내용을 확인하였으며, 'date' 명령을 이용해 현재 시간을 확인하였다(이때 시간은 7시 42분). 그 이후 시스템에 아무런 사용자의 요청이 발생하지 않았는데, 8시 0분 23초에 능동 서비스 관리자가 자동으로 오디오를 켜고 있음을 볼 수 있다(그림에서 타원으로 표시된 부분이 능동 서비스 관리자가 출력한 로그이다). 즉 설정 파일의 내용과 그동안 수집된 통계 정보에 따라 시스템이 자동으로 적극적인 서비스를 제공하는 것이다.

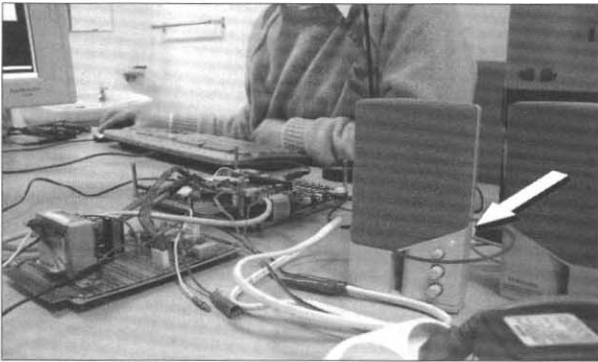


(그림 10) 능동 서비스 관리자에 의한 오디오 제어 실험 : 관리자 수행 로그

(그림 11)은 능동 서비스 관리자의 자동 수행 결과 오디오가 켜진 결과를 보여준다(그림 11)(b)에서 화살표와 타원으로 표시된 부분에 오디오가 동작 증임을 알리는 불이 켜져 있다). 이 실험을 통해 우리는 능동 서비스 관리자가 사용자의 행위로부터 규칙성을 파악하고 이를 기반으로 미래를 예측하여 서비스를 자동으로 제공하였음을 알 수 있다. 결국 지능화된 대리자(Intelligent Agent) 역할을 수행한 것이다.



(a) 능동 서비스 관리자 수행 이전



(b) 능동 서비스 관리자 수행 이후

(그림 11) 능동 서비스 관리자에 의한 오디오 제어 실험 : 오디오 제어 결과

한편 능동 서비스 관리자는 잘못된 예측에 대한 보완 작업을 할 수 있어야 한다. 예를 들어 전등을 자동으로 켜는데 실제로는 사용자가 그 공간에 들어오지 않았다면 다시 끌 수 있는 기능이 제공되어야 하는 것이다. 이것은 결국 사용자의 위치나 존재 여부를 파악할 수 있는 문제와 연관되며 현재 시스템에 구현되어 있지는 않다. 본 연구진은 향후 연구에서 이 내용을 확장할 것이다. 구체적으로 보드에 존재하는 적외선(IrDA)을 이용한 센싱을 통해 사용자가 방에 존재하는지를 파악할 수 있으며, 이를 기반으로 보완 작업을 수행할 수 있다. 한편, 능동 서비스 관리자의 또 다른 확장 방향에는 특정 시점에 해야 할 작업을 사용자에게 알리는(alert) 기능이 있다. 구체적으로 설정 파일인 /etc/ispac.conf에는 서비스 옵션(service option)이 있으며 여기에 강제 서비스(mandatory service)로 설정하고 특정 시간을 명시하면, 시스템은 LED나 진동 등의 방법으로 그 시간을 사용자에게 알릴 수 있다. 예를 들어 매일 오후 3시에 애완견 먹이를 주어야 한다면 능동 서비스 관리자가 그 시간을 사용자에게 환기시키게 되는 것이며, 또한 이 서비스를 모닝콜 기능으로 사용할 수도 있는 것이다.

6. 결 론

본 논문에서는 사용자 친화적 지능형 공간 제어 시스템

을 제안하였다. 이 시스템은 인간 중심적인 음성 인터페이스를 제공하며, 또한 사용자의 행동 패턴을 분석하고 이를 기반으로 지능적인 서비스를 제공한다. 또한 웹 표현(web presence)을 제공하여 브라우징이 가능한 단말기를 휴대한 사용자는 언제 어디서나 시스템에 접근할 수 있다. 현재 이 시스템은 내장형 보드와 임베디드 리눅스를 기반으로 구현되었다.

제안된 시스템은 최근 주목받고 있는 유비쿼터스 컴퓨팅이 요구하는 속성들을 만족할 수 있도록 설계되었다. 특히 자원 제약이 있는 저가형 PDA급 하드웨어 환경에서 소프트웨어적인 음성 인식이 가능하다는 것을 보였으며, 능동 서비스 관리자는 사용자의 행위로부터 규칙성을 파악하고 이를 기반으로 미래를 예측하여 지능화된 문맥 인식 서비스(context aware service)를 제공할 수 있음을 보였다. 이 내용들은 본 논문에서 제안한 시스템과 기존의 다른 시스템 간에 차별되는 특징이며, 향후 스마트 홈, 지능형 사무실, 생각하는 사물(Things That Think) 등의 연구에 효과적으로 이용될 수 있을 것으로 기대한다.

현재 시스템은 기본적인 동작을 검증할 수 있는 프로토타입이 구현된 상태이며, 향후 우리는 다음의 방향으로 본 연구를 확장할 것이다. 첫째, 실험 결과 질에서 이미 언급했듯이 능동 서비스 관리자는 잘못된 예측에 대한 보완을 할 수 있어야 하며, 이를 위한 사용자 위치나 존재 여부를 파악하는 방법에 대해 연구할 것이다. 또한, 현재 구현된 능동 서비스 관리자는 사람 행동에 대한 매우 간단한 휴리스틱(Heuristic)을 사용하였는데, 이것도 다양한 인공 지능 기법과 확률 알고리즘을 이용하여 좀 더 인간에게 편리한 서비스를 제공할 수 있도록 발전시킬 것이다. 둘째, 가전 제품을 단순히 제어하는 것뿐만 아니라 제품의 상태 파악 및 관리를 할 수 있는 네트워크 정보 가전(network information appliances) 관리 시스템으로 확대시킬 것이다. 셋째, 현재 연구에서는 음성을 그대로 사용자 인터페이스로 사용했는데, 이 경우 사용자의 의도하지 않은 음성에 대해 시스템이 반응하여 오동작을 할 수도 있다. 즉 음성 인터페이스의 정형화된 구조에 대한 연구도 필요하다.

참 고 문 헌

- [1] M. Weiser, "Some Computer Science Problems in Ubiquitous Computing," *Communication of the ACM*, pp.75-84, July, 1993.
- [2] M. Weiser, <http://www.ubiq.com/hypertext/weiser/>.
- [3] L. Kleinrock, "Nomadic computing : anytime, anywhere in a disconnected world," *Mobile Networks and Applications*, Vol.1, No.4, 1997.
- [4] M. Satyanarayanan, "Pervasive Computing Vision and Challenges," *IEEE Personal Comm.* Vol.6, No.8, pp.10-17, August, 2001.
- [5] The Pervasive Computing, <http://www.ibm.com/>.

- [6] The Disappearing Computer, <http://www.disappearing-computer.net/>.
- [7] 노무라총합연구소 저, "유비쿼터스 네트워크와 신사회 시스템", 전자신문사, 2003.
- [8] 이은경, 하원규, "유비쿼터스 컴퓨팅 비전과 주요국의 연구 동향", 전자통신동향분석, 제17권 제6호, pp.1-10, Dec., 2002.
- [9] MIT Oxygen Project, <http://oxygen.lcs.mit.edu/>.
- [10] D. Garlan, D. P. Siewiorek, A. Smailagic and P. Steenkiste, "Project Aura : Toward Distraction-Free Pervasive Computing," IEEE Pervasive Computing, April-June, 2002.
- [11] Aura Project at CMU, <http://www-2.cs.cmu.edu/~aura/>.
- [12] CoolTown at HP, <http://www.cooltown.com/>.
- [13] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic, "People, Places, Things : Web Presence for the Real World," HP Technical Report, HPL-2001-279, 2001.
- [14] Endeavour Project at Berkeley, <http://endeavour.cs.berkeley.edu/>.
- [15] Aware Home at Georgia Tech., <http://www.awarehome.gatech.edu/>.
- [16] Infosphere Project at Georgia Tech., <http://www.cc.gatech.edu/projects/infosphere/>.
- [17] Portolano Project at UW, <http://portolano.cs.washington.edu/>.
- [18] EasyLiving at MS, <http://research.microsoft.com/easy-living/>.
- [19] GoAhead web server, <http://www.goahead.com/>.
- [20] HTK Projects, <http://htk.eng.cam.ac.uk/>.
- [21] Hybus, <http://hybus.co.kr/>.
- [22] 음성인식, http://inc2.ucsd.edu/~owkwon/srhome/lecture/speech_recog_system.html.
- [23] Daniel P. Bovet and Marco Cesati, "Understanding the LINUX KERNEL," O'REILLY, 2002.



최 종 무

e-mail : choijm@dankook.ac.kr
 1993년 서울대학교 해양학과(이학사)
 1995년 서울대학교 대학원 컴퓨터공학과 (공학석사)
 2001년 서울대학교 대학원 컴퓨터공학과 (공학박사)

2001년~2003년 유비쿼스(주) 기술연구소 책임연구원
 2003년~현재 한국정보과학회 논문지 편집 위원
 2003년~현재 서울대학교 컴퓨터신기술연구소 연구원
 2003년~현재 단국대학교 정보컴퓨터학부 컴퓨터과학전공 전임 강사
 관심분야 : 시스템 소프트웨어, 내장형 시스템, 파일 시스템, 유비쿼터스 컴퓨팅 등



백 창 우

e-mail : bckddn@daum.net
 1997년 동국대학교 정보산업학부 입학
 2001년 삼성 테크윈 리눅스 프로그래밍 과정 수료
 2002년 리눅스원 리눅스 SE 전문가 과정 수료

2003년 동국대학교 소프트웨어 콘테스트 대상
 2003년~현재 동국대학교 컴퓨터 공학과 재학
 관심분야 : 운영체제 커널, 시스템 소프트웨어, 컴퓨터 보안, 내장형 시스템



구 자 경

e-mail : kooz@ssm.samsung.co.kr
 2000년~2002년 KATUSA 군복무
 2002년~2003년 박영만 전산학원 SE과정 118기 수료
 2003년 제1회 임베디드 소프트웨어 공모대진(정통부 주최) 장려상

2003년~현재 삼성전자 소프트웨어 멤버십 13기 정회원
 1998년~현재 단국대학교 경영정보, 전산학과 복수 전공
 관심분야 : 임베디드 시스템, 유비쿼터스 컴퓨팅, 커널, 휴대폰 운영체제, WIPI



최 용 석

e-mail : hicys@integra.co.kr
 1999년~2003년 인테그라 정보통신(주) 기업부설연구소 주임연구원(군무중)
 2003년~현재 단국대학교 정보컴퓨터학부 컴퓨터과학전공 재학 중
 관심분야 : 내장형시스템, HDTV시스템, 유비쿼터스 컴퓨팅 등.



조 성 제

e-mail : sjcho@dankook.ac.kr
 1989년 서울대학교 컴퓨터공학과(학사)
 1991년 서울대학교 대학원 컴퓨터공학과 (공학석사)
 1996년 서울대학교 대학원 컴퓨터공학과 (공학박사)

1996년~1997년 서울대학교 컴퓨터신기술연구소 연구원
 2001년 미국 University of California, Irvine 객원 연구원
 1997년~현재 단국대학교 정보컴퓨터학부 컴퓨터과학전공 조교수
 관심분야 : 컴퓨터 보안, 시스템 소프트웨어, 실시간 시스템, 분산 시스템 등