웨어러블 운영체제의 동향 및 전망

황재민
충남대학교 컴퓨터공학과

남병규
충남대학교 컴퓨터공학과

Ⅰ. 서 론

최근 컴퓨팅의 패러다임이 모바일에서 웨어러블로 급속히 넘어가는 상황에서 웨어러블 컴퓨터는 더 이상 상상 속의 개념이 아닌 현실이 되고 있다. 2012년 Google Glass 〈그림 1〉의 발표 이후, 우리에게 친숙해진 웨어러블 컴퓨터는 인포테인먼트, 피트니스를 비롯하여 산업 전반에 있어 활용도를 높여 가고 있으며 산업의 패러다임을 바꾸어 놓고 있는 중이다. 이러한 웨어러블 컴퓨터는 사용자의 몸에 항시 착용되는 특성으로 인하여 제한된 무게와 크기를 가지게 되며 배터리 용량 또한 한정적일 수밖에 없다. 따라서 제한된 성능, 배터리, 그리고 불편한 UI는 웨어러블 컴퓨터가 시장에서 수용되기 위하여 반드시 해결되어야 할 과제이다. 그러므로 웨어러블 운영체제는 〈그림 2〉와 같이 사용자의 자연스러운 행위를 UI로 활용해야 하며 부족한 성능을 해결하기 위해 클라우드 서버와 같은 외부의 자원을 적극 활용해야 한다. 이와 더불어 기존의 모바일 환경보다 한 층 강화된 저전력 기술을 구현하여 초저전력, 초경량화된 환경을 실현해야 하는 과제를 안고 있다.

현재 웨어러블 컴퓨터를 위한 운영체제들은 기존의 스마트 기기를 위한 운영체제 또는 실시간 운영체제를 기반으로 하여 웨어러블 환경의 특성에 맞추어 최적화된 형태로 구현되고 있다.

현재 웨어러블 컴퓨터를 위한 운영체제들은 기존의 스마트 기기를 위한 운영체제 또는 실시간 운영체제를 기반으로 하여 웨어러블 환경의 특성에 맞추어 최적화된 형태로 구현되고 있다. Android, Tizen, iOS와 같은 기존의 스마트 기기용 운영체제를 기반으로 하는 Android Wear, Tizen 웨어러블 프로파일, Watch OS의 경우에는 웨어러블 기

〈그림 1〉 Google Glass

〈그림 2〉 웨어러블 플랫폼 주요 이슈

기에 알맞게 불필요한 부분을 제거하고 경량화 된 형태로 사용되고 있다. 또한 소형 임베디드 시스템에 탑재되어 온 실시간 운영체제들은 웨어러블 기기마다의 특성을 살려 각 기기에 특화된 형태로 사용되고 있다.

본 고에서는 이러한 웨어러블 기기용 운영체제들의 특징을 살펴보고 웨어러블 플랫폼이 해결해야 할 이슈들에 대한 운영체제 측면에서의 기법들에 대하여 알아본다. 먼저 2장에서 웨어러블 기기의 응용분야별 주요 운영체제에 대하여 살펴보고 3장에서 최신 웨어러블 운영체제의 기법들을 바탕으로 향후 웨어러블 운영체제의 전망을 살펴본다. 그리고 마지막으로 4장에서 결론을 맺는다.

웨어러블 기기용 운영체제들의 특징을 살펴보고 웨어러블 플랫폼이 해결해야 할 이슈들에 대한 운영체제 측면에서의 기법들에 대하여 알아본다.

Ⅱ. 웨어러블 응용에 따른 운영체제 동향

웨어러블 컴퓨터는 반도체 기술의 소형 · 고성능화에 힘입어 스마트 워치 및 스마트 글래스와 같은 인포테인먼트 분야까지 그 영역을 확장해 가고 있다. 이러한 웨어러블 컴퓨터는 응용분야에 따라 〈그림 3〉과 같이 크게 4가지의 분야로 분류되며 각 분야에 따른 제한조건 및 목적이 상이한 관계로 탑재되는 운영체제도 서로 다르게 된다.[1]

첫째, 인포테인먼트 분야의 웨어러블 컴퓨터는 스마트폰과 연동하여 사용자로 하여금 스마트폰의 다양한 기능을 활용할 수 있도록 하는데 목적이 있다. 단순 정보제공뿐만 아니라 스마트폰의 기능을 일부 수행하는 수준까지 발전한 단계로 이를 충분히 운용할 수 있는 운영체제가 탑재되어 있다. 현재 데스크톱, 모바일 기기에서 사용되는 Linux, iOS 기반의 운영체제가 주를 이룬다.

둘째, 피트니스 분야의 웨어러블 컴퓨터는 항시 사용자의 몸에 착용되어 있는 관계로 매우 작은 크기를 요구한다. 메모리나 배터리도 매우 제한적일 수밖에 없으며 이를 운용하기 위한 운영체제 또한 경량 및 저전력에 초점을 맞춘 실시간 운영체제가 선호된다.

셋째, 헬스케어 및 메디컬 분야의 웨어러블 컴퓨터는 피트니스 분야와 마찬가지로 사용자가 항시 착용해야 하는 환경과 그로 인한 크기의 제한에 따라 경량의 운영체제가 요구된다. 특히 높은 실시간성을 필요로 하므로 실시간 운영체제가 탑재된다.

넷째, 산업 · 군사 분야의 웨어러블 컴퓨터는 앞서 살펴보았던 분야의 웨어러블 컴퓨터에 비해 보다 가혹한 환경에서 운용되는 것이 특징이며, 고도의 실시간성이 요구되는 관계로 높은 안정성을 지닌 실시간 운영체제가 사용된다.

운영체제 측면에서 살펴보면, 스마트 웨어러블 기기를 위한 Linux, iOS 기반의 고성능 운영체제와 경량 및 실시간 웨어러블 기기를 위한 초경량 실시간 운영체제로 이분화 됨을 볼 수 있다. 본 장에서는 이러한 분류에 따라 웨어러블 컴퓨터에서 주로 사용되고 있는 고성능 운영체

〈그림 3〉 웨어러블 컴퓨터의 분류

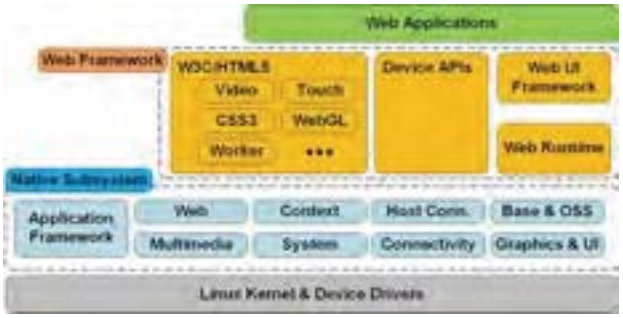
제 및 초경량 운영체제의 구조와 특성에 대하여 각각 살펴본다.

1. 웨어러블 스마트 기기를 위한 운영체제

삼성전자, 구글, 애플 등 모바일 스마트 기기를 선도해 오던 기업들이 사업을 확장하여 스마트 워치, 스마트 글래스 등의 웨어러블 스마트 기기들을 잇달아 출시함에 따라, 이들 웨어러블 기기를 운용하기 위한 운영체제 또한 Android, iOS 등과 같이 기존 스마트기기용 운영체제에 바탕을 둔 제품들이 각광을 받고 있다. 이러한 인포테인먼트용 웨어러블 운영체제로는 주로 Linux 기반의 Tizen, Android Wear와 애플 제품을 위한 Watch OS가 사용되고 있다. 이들 모두 SDK (Software Development Kit)가 제공되어 손쉬운 응용프로그램 개발이 가능하다는 장점을 가진다.

1) Tizen

삼성 기어2 등에 탑재되고 있는 Tizen은 삼성, Intel 및 개인 개발자들이 다수 참여하여 개발한 리눅스 기반 오픈 소스 플랫폼이다.[2] Tizen은 스마트폰, 태블릿뿐만 아니라 TV, 카메라 등의 가전제품과 나아가 스마트 워치 등의 인포테인먼트 웨어러블 기기에까지 사용되고 있다. Tizen 웨어러블 프로파일 구조는 〈그림 4〉에 보인 것과 같이 리눅스 커널과 디바이스 드라이버 위로 운영체제

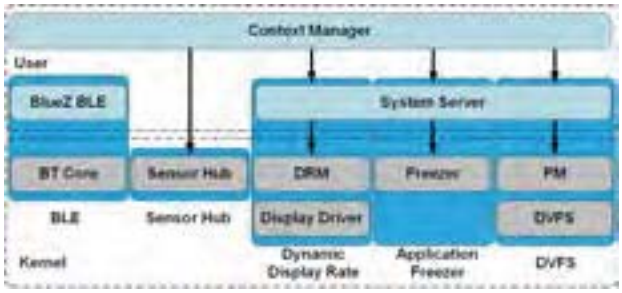


〈그림 4〉 Tizen 웨어러블 프로파일 구조

의 기본 기능을 담당하는 네이티브 서브시스템이 탑재된다.[3] 웹 기능을 지원하기 위한 웹 프레임워크가 추가로 탑재된 후 웹 애플리케이션이 최상위에서 동작하게 된다. 웹 애플리케이션은 HTML5 기반으로 개발되어 타 플랫폼에 쉽게 이식이 가능하다는 장점이 있다.

Tizen 웨어러블 프로파일은 성능 면에서의 최적화를 도모하기 위해 Web UI 프레임워크의 일부를 변경하였다. 기존에는 HTML5로 구성되어 있던 Web widget을 순수한 Javascript로 구성된 TAU (Tizen Advanced Web UI Framework)로 변경하여 웹 애플리케이션이 시작되는 시간을 단축시키고 있다. 또한 경량화를 위하여 웨어러블 기기에서는 불필요한 Webkit의 기능을 축소하였다. 나아가 웹 및 2D UI 툴킷으로서 EFL (Enlightenment Foundation Libraries)을 사용하여 메모리 사용량을 줄이고 있다. EFL은 Qt나 Gtk+에 비하여 빠른 소프트웨어 렌더링을 지원하고 메모리 사용량이 작은 것이 특징이다.

Tizen은 소비 전력을 줄이기 위해 〈그림 5〉와 같이 5가지 특징을 가지고 있다. 먼저 스마트폰과의 연동을 위한 블루투스를 저전력으로 동작시키기 위하여 BLE (Bluetooth Low Energy)를 채택하고 있다. BLE는 기존 블루투스 대비 적은 프로토콜 스택을 지니며 데이터 패킷을 단순화시켜 전력 소비를 절감시키는 기술이다.[4] 그리고 메인 보드와 분리된 Sensor Hub가 센서 값을 모니터링 하여 필요할 때만 메인보드로 센서 값을 전달함으로써 전력 소비를 줄인다. 또한 디스플레이와 관련해서는 애플리케이션의 컨텍스트에 따라 디스플레이 갱신 주기를 동적으로 변경해주는 Dynamic Display Rate 기법을 통하여 저전력을 구현하고 있다. 애플리케이션 관리 측면에



〈그림 5〉 Tizen의 저전력 기법

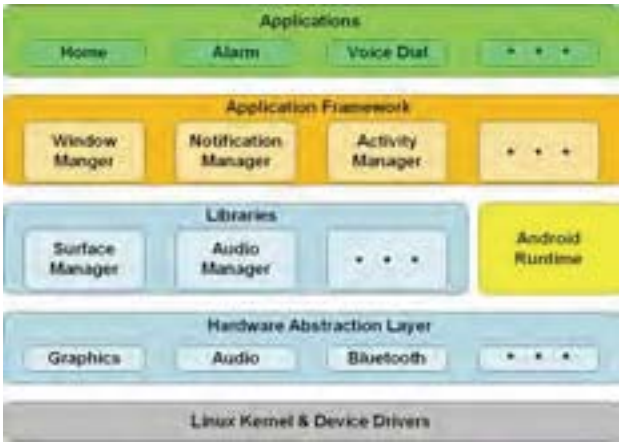
서는 백그라운드로 진입한 애플리케이션은 Freeze 시킴으로써 불필요한 CPU 사용을 막아준다. 마지막으로 프로세서의 전압 및 주파수를 작업량에 따라 동적으로 변경하여 전력소모를 줄이는 DVFS (Dynamic Voltage and Frequency Scaling) 기법을 웨어러블 애플리케이션에 최적화하여 구현하고 있다.[5]

2) Android Wear

스마트 워치에 주로 탑재되는 Android Wear는 구글이 개발한 웨어러블 기기용 운영체제이며 현재 일부 소스 코드만이 공개된 상태이다. Android 4.3 이상의 스마트폰과 연동하여 동작하며 구글 Now를 통하여 음성으로 스마트 워치를 조작할 수 있다. Android Wear는 모바일 운영체제인 Android와 크게 다르지 않을 것으로 보이는데, Android의 구조는 〈그림 6〉에 나타나 있다. 이는 리눅스 커널을 기반으로 하며 하드웨어 추상화 계층 위로 네이티브 언어로 기술되는 라이브러리 및 Java로 기술되어 애플리케이션을 관리하는 프레임워크가 올라간다. 최상위에서 Java 애플리케이션이 동작함으로써 Android가 구동된다.

구글은 최근에 Android 5.0을 공개하였으며 이 버전에서는 기존 Dalvik 가상머신이 ART (Android RunTime) 라이브러리로 교체되었다.[6] ART는 Java 애플리케이션이 인스톨될 때 AOT (Ahead-of-Time) 컴파일을 통하여 디바이스의 바이너리로 변환한 후 애플리케이션을 동작시키므로, 기존 Dalvik 가상머신의 JIT (Just-in-Time) 컴파일 기법에 비하여 성능을 향상시켰으며 CPU 사용량을 줄여 배터리에 친화적으로 설계되었다.

더욱이 Android 5.0에서는 Project Volta를 통하여 전



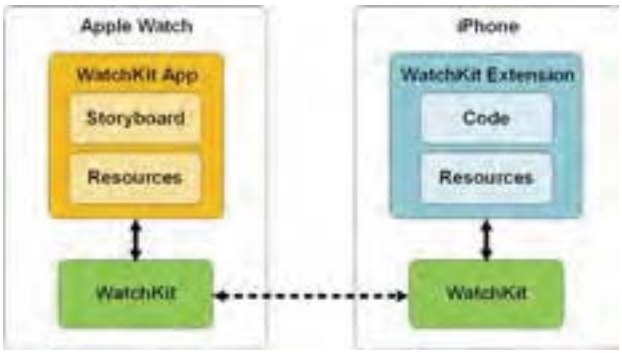
〈그림 6〉 Android 구조

력 소모량을 줄이고 있는데, Project Volta는 Android 5.0 버전에서 새롭게 도입된 JobScheduler, Battery Historian을 포함한 배터리 친화적 기술을 통칭하는 것이다.[7] JobScheduler는 개발자가 사용할 수 있는 API이며 애플리케이션이 특정 상황에서만 동작하도록 정의한다. 이를 활용하면 긴급하지 않은 애플리케이션은 충전 중과 같은 배터리에 부담이 없는 상황에 동작시켜 웨어러블 기기의 가동 시간을 연장시킬 수 있다. Battery Historian은 애플리케이션마다 배터리 사용 내역을 시각화하여주는 툴로써 개발자로 하여금 효율적인 애플리케이션 개발을 가능케 해준다.

3) Watch OS

아이폰으로 스마트폰 시장을 양분하였던 애플이 최근 애플워치와 함께 웨어러블 플랫폼을 위한 운영체제인 Watch OS를 출시하였다. 애플워치는 iOS 8.2 이상을 탑재하고 있는 아이폰과 연동하여 동작하며 아이폰과 마찬가지로 Siri를 통하여 음성 명령을 내릴 수 있다. Watch OS의 구조는 〈그림 7〉과 같은 iOS 8의 구조를 기반으로 한다.[8] 이 구조는 커널의 역할을 수행하는 Core OS 계층 위로 네트워크, 데이터베이스 등의 서비스를 제공하는 Core Services 계층을 둔다. 그 위로는 그래픽스, 오디오 기능 등을 제공하는 Media 계층이 위치하며 최상위에서 애플리케이션을 직접 지원하기 위한 프레임워크를 제공하는 Cocoa Touch 계층이 있다.

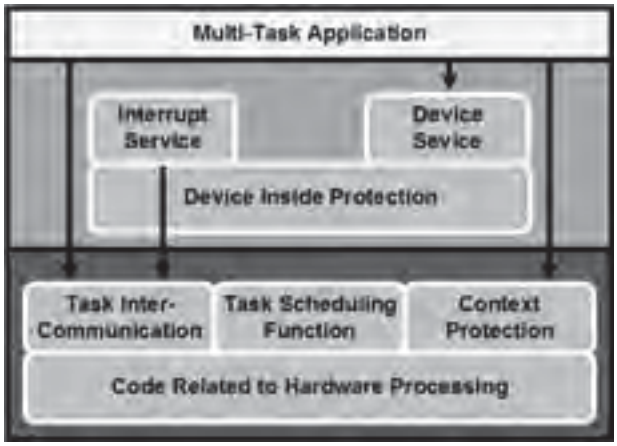
〈그림 7〉 iOS 구조



〈그림 8〉 WatchKit 동작 구조

모바일 환경을 고려하여 iOS 8은 전력 소모를 줄이기 위한 몇 가지 특징을 가진다. 먼저 홈 화면으로 돌아갈 경우 실행 중인 애플리케이션 중 백 그라운드에서 수행할 일이 없는 애플리케이션을 freeze-dried 상태로 두어 CPU를 점유하지 않도록 한다. 또한 VoIP (Voice over Internet Protocol)를 사용하는 애플리케이션이 푸시 알림이 있을 경우에만 네트워크 연결을 유지하게 하여 전력 소비를 줄인다. 추가로 저전력 블루투스인 BLE를 지원한다.

이밖에 애플워치는 아이폰의 대체가 아닌 확장을 목적으로 하므로, 서드 파티 애플리케이션을 동작시키기 위하여 〈그림 8〉과 같은 구조를 가진다. 그림에서 보이는 바와 같이 애플워치는 애플리케이션의 인터페이스에 해당하는 Storyboard와 관련 리소스만을 가지며 실제 애플리케이션은 아이폰에 위치한다. 즉, 애플워치는 인터페이스에 대한 그래픽 처리를 담당하며 아이폰은 애플리케이션을 제어하게 된다. 이는 Tizen, Android Wear와 달리 Watch OS는 아이폰을 보조하며 인터페이스를 확장하는 역할임을 알 수 있다.



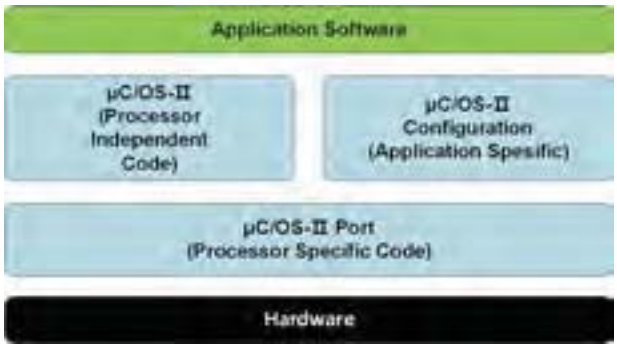
〈그림 9〉 FreeRTOS 구조

2. 초경량 웨어러블 기기를 위한 운영체제

피트니스 목적의 웨어러블 컴퓨터는 사용자의 신체에 오랫동안 접촉해 있으면서 움직임에 방해를 주지 않아야 하는 제약 조건으로 인해 매우 제한적인 부피와 무게를 지닐 수밖에 없다. 이러한 환경에서는 CPU, 메모리, 배터리 등의 하드웨어 자원이 극히 제한되므로 운영체제 또한 CPU를 오래 점유해서는 안 되며 매우 경량으로 구현되어야 한다. 이러한 이유로 피트니스 목적의 웨어러블 컴퓨터에서는 초경량 실시간 운영체제를 주로 사용한다. 실시간 운영체제는 기존 임베디드 환경에서 주로 사용되어 왔으며, 차후 IoT에 적합한 운영체제라 평가받는 만큼 모바일 및 데스크톱 운영체제에 비하여 매우 가볍고 운영체제에 필수적인 기능만을 담고 있다. 현재 웨어러블 기기에서 주로 사용되는 운영체제로는 Real Time Engineer의 FreeRTOS와 Micrium의 µC/OS, 그리고 SEGGER의 embOS 등을 들 수 있다.

1) FreeRTOS

Real Time Engineers에서 개발한 실시간 운영체제인 FreeRTOS는 매우 작은 메모리 용량만을 차지하며 오픈소스 운영체제 중 Android 다음으로 점유율이 높은 것으로 알려져 있다.[9] 그 구조는 〈그림 9〉에 나타난 바와 같이 인터럽트 서비스, 태스크 간 통신 및 동기화 기능, 스케줄러 등으로 구성되어 있다.[10] FreeRTOS는 실시간성을 위하여 우선순위와 라운드-로빈 스케줄링 방식이 사용되고 있다. 이 외에도 생성되는 태스크 및 우선순위 개



〈그림 10〉 µC/OS-Ⅱ 구조

수에 제한이 없다는 특징이 있다.

메모리 사용량을 줄이면서 멀티태스킹의 효과를 내기 위하여 코루틴 사용도 가능하다. 코루틴은 태스크와 비슷하지만 하나의 애플리케이션을 구성하는 코루틴끼리는 스택을 공유하도록 하여 메모리 사용을 절약하는 특징을 가진다. 하지만 코루틴 간에는 비선점 스케줄링만이 가능하므로 실시간성을 필요로 하지 않는 응용에서만 사용할 수 있다는 제약을 가진다.

전력 소비를 줄이기 위하여 tick 인터럽트를 일정 시간 동안 차단하는 Tickless Idle Mode도 지원된다. 이는 실행 가능한 태스크가 없을 때는 Idle 태스크가 수행되어 시스템이 저전력 모드로 진입하지만 tick 인터럽트가 매번 발생하여 저전력 모드에서 벗어나는 문제를 해결해 준다.

2) µC/OS-Ⅱ의 구조 및 특징

Micrium의 µC/OS-Ⅱ는 교육 목적으로 자유로운 사용이 가능한 오픈소스 실시간 운영체제이다. 이는 〈그림 10〉에 보이는 바와 같이 프로세서 의존적인 모듈과 독립적인 모듈, 애플리케이션을 위한 설정 모듈로 구성되어 있으며, 세부적으로는 임계 영역 관리, 태스크 간 통신, 타이머 관리, 메모리 관리 등의 기능을 가진다.[11] 최대 64개의 태스크를 수행할 수 있으며 사용자의 요구에 따라 필요 기능만 포함하여 컴파일 되므로 메모리 요구량을 조절할 수 있다는 특징을 지닌다. 이 외에도 신뢰성 및 안정성이 뛰어나 의료 장비 등에 사용되기에 적합한 것으로 알려져 있다.

µC/OS-Ⅱ는 태스크 간 동일 우선순위를 허용하지 않으며 우선순위에 의한 선점형 스케줄링을 수행한다. 또한 개발자는 임계 영역에 접근할 때 인터럽트를 막지 않는 대신 스케줄러를 lock할 수 있다. 이는 인터럽트에 대한 빠른 대처를 가능하도록 해준다.

3) embOS의 구조 및 특징

embOS는 SEGGER에서 개발한 상용 실시간 운영체제로서 embOS/IP, emUSB, emWin 등의 미들웨어와 함께 제공된다.[12] 더욱이 embOS 자체도 모듈화 되어 있어 필요한 기능만을 포함하여 더욱 경량화 할 수 있도록 하고 있다. 이 외에도 높은 우선순위의 인터럽트는 다른 태스크에 의해 방해받지 않도록 하여 인터럽트 지연을 최소화 하는 특징을 가지고 있다.

실시간 운영체제이지만 실시간성에 대한 필요가 없이 작고 단순한 시스템을 위한 superloop 설계도 허용한다. 이는 루프를 돌며 하나의 태스크만 실행하며 인터럽트 서비스로 약간의 실시간성만을 제공하는 것을 뜻한다. 하나의 스택만 존재하면 되므로 메모리 제약이 큰 시스템에 적합하다.

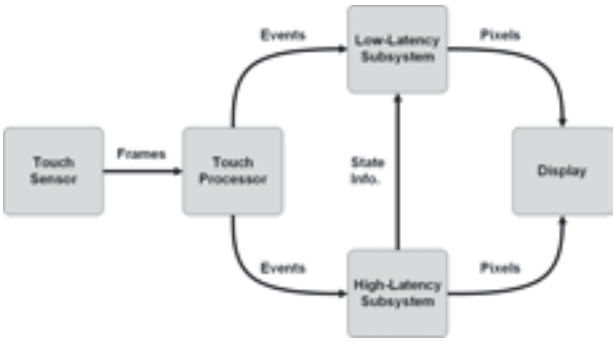
embOS는 전력 소비를 줄이기 위한 방법으로 FreeRTOS와 마찬가지로 Tickless 모드를 지원한다. 외부에서 또는 Tickless 주기마다 발생하는 인터럽트가 감지되거나 어떤 태스크가 실행 가능해 질 때까지 시스템을 저전력 상태로 둠으로써 불필요한 전력 소비를 막는다.

Ⅲ. 웨어러블 운영체제를 위한 커널 요소

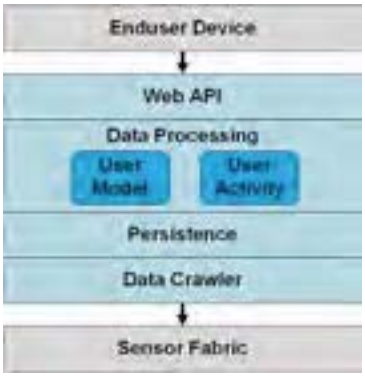
1. NUI (Natural User Interface)

스마트 워치, 스마트 글래스 등의 웨어러블 기기는 제한된 인터페이스로 인해 사용자의 음성이나 신체 움직임을 활용하는 NUI (Natural User Interface)를 적극 필요로 한다. 이러한 기기의 발전에 맞물려 운영체제 측면에서는 기존의 터치 제스처 뿐만이 아니라 음성 및 모션 인식과 같은 새로운 입력 방식을 지원하기 위한 연구를 활발히 진행 중이다.

터치 제스처의 경우 스마트폰, 스마트 워치 등 화면에 직접 터치가 가능한 환경에 맞추어 반응성을 중요한 요소로 여긴다. 이는 마우스와 같은 입력 장치와는 달리 출



〈그림 11〉 터치 제스처 반응성 향상 기법



〈그림 12〉 스마트 의복을 위한 NUI 구조

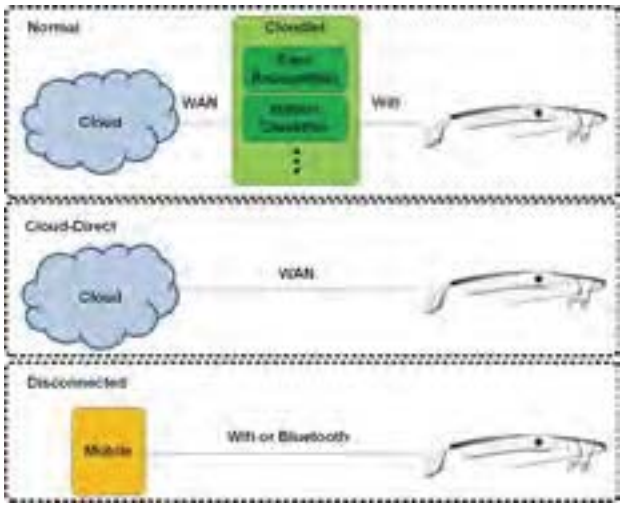
력 장치인 디스플레이에 직접 입력을 가하는 형태로 입·출력의 차이가 명확하게 드러나기 때문이다. 마이크로소프트는 하드웨어 가속 및 터치 이벤트의 처리 구조를 변경하여 반응성을 개선한 연구를 수행하였는데, 이 연구는 〈그림 11〉에 보이는 바와 같이 터치 이벤트를 처리하여 픽셀을 계산하는 부분을 기존의 속도가 느린 저속처리부(High-Latency Subsystem)와 더불어 하드웨어 가속을 받는 고속처리부 (Low-Latency Subsystem)를 혼합하여 사용함으로써 반응성을 개선하고 있다.[13] 구체적으로는 터치 이벤트에 반응하여 고속처리부가 터치가 발생한 부분만 하드웨어 가속을 통해 빠르게 픽셀 연산을 수행하고 디스플레이로 출력한다. 동시에 저속처리부는 전체 픽셀에 대한 계산을 수행하여 전체 화면을 갱신한다. 전체 화면의 갱신 시간은 개선되지 않지만 터치가 일어난 부분의 화면은 빠르게 갱신되므로 반응성의 향상을 가져온다.

웨어러블 기기에서의 새로운 입력 방식을 효과적으로 활용하기 위한 소프트웨어 구조도 최근 독일의 슈투트가르트 대학의 연구팀에 의하여 제안되었다.[14] 이는 애플리케이션 개발을 용이하게 하는 툴킷으로서 심박 센서, 가속도 센서와 같이 사용자의 신체 정보와 움직임을 측정하는 센서 기반의 스마트 의류를 대상으로 한다. 이 툴킷은 〈그림 12〉와 같이 최하단의 Data Crawler 계층에서 센서로부터의 입력을 취합하며, 그 위의 Persistence 계층이 애플리케이션 개발 및 분석을 위해 센서 값을 DB에 저장한다. Data Processing 계층은 디지털 필터 및 NUI 인식을 위한 알고리즘들을 포함하며 이벤트를 관리한다. 그리고 사용자의 포즈, 심박수와 같은 생체 정보를 저장하는 User Model과 뛰거나 걷는 등의 행위를 저장하는 User Activity 모듈을 지닌다. 이 계층은 애플리케이션이 직접 센서 값에 접근하여 신호 처리를 할 필요성을 없애준다. 마지막으로 다양한 기기에서 손쉽게 스마트 의복과 통신하기 위한 Web API를 애플리케이션에 제공한다. 이러한 구조를 통하여 스마트 의복을 비롯한 웨어러블 기기에서의 NUI를 효과적으로 제공하고 있다.

2. 클라우드 인터페이스

앞서 서론에서 살펴본 바와 같이, 웨어러블 기기는 사용자의 몸에 항시 착용되어야 하는 관계로 작고 가벼우며 매우 한정적인 자원만을 가지게 된다. 이러한 제약사항으로 말미암아 웨어러블 기기에서는 외부 기기의 도움을 활용하여 복잡한 연산을 처리해 내는 오프로딩(Offloading) 기법이 필수적이다. 현재 오프로딩 서비스의 대표적인 예로는 구글의 음식인식 서비스를 들 수 있는데, 이러한 오프로딩 기법에서는 클라우드 서버를 사용하여 단말기에서 수행하기 어려운 복잡한 연산들을 처리하고 있다.

오프로딩 기법은 두 가지 측면에서의 이슈가 발생하는데, 첫째는 디바이스와 클라우드 서버간의 통신 시간인 RTT (Round-Trip Time)가 크다는 것이며, 둘째는 회선 등의 문제로 네트워크 연결이 불안정할 수 있다는 것이다. 이러한 문제를 해결하고자 여러 연구가 진행되었으며, 최근 카네기 멜론 대학의 연구팀에서는 〈그림 13〉과 같이 Cloudlet의 사용 및 오프로딩 실패에 대한 정책을 고안하여 오프로딩의 문제에 대한 해결책을 제시하였다.[15] 여기서 Cloudlet이란 클라우드 서버와 웨어러블 기



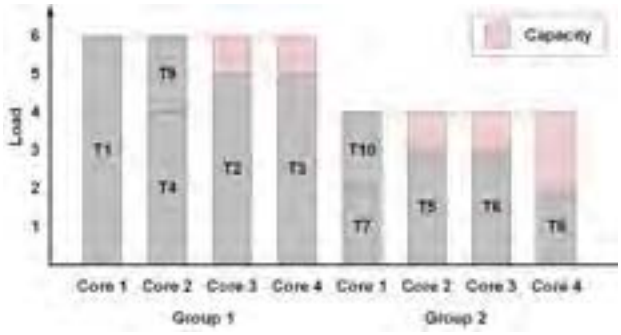
〈그림 13〉 오프로딩 정책

기 사이의 중간 매개체이자 디바이스로부터의 요청을 처리하여 주는 머신으로서 Wifi를 통해 디바이스와 통신한다. 운영체제는 Cloudlet, 클라우드 서버, 휴대용 단말기 기 순으로 오프로딩을 시도하여 네트워크 연결 실패에 대비하며, 가장 높은 품질의 서비스를 제공하면서도 짧은 RTT를 보장받도록 정책을 수립한다.

현재 웨어러블 플랫폼에서는 이러한 클라우드 서비스를 활용하기 위한 수단으로 웹 표준 언어인 HTML5를 주목하고 있다. HTML5는 웹 API를 제공하여 웹 브라우저를 통한 웹 애플리케이션을 구동해 줄 수 있으며 TV, 자동차 등 타 플랫폼으로의 이식성도 뛰어난 장점을 가지고 있다.[16] 또한 웹 애플리케이션은 네이티브 애플리케이션과 달리 업데이트가 필요 없으며 서버로부터 항상 최신의 애플리케이션 데이터를 제공받는다. 더욱이 제공되는 API 중 애플리케이션 캐시는 서버로부터의 데이터를 캐싱하여 서버로의 로드 요청 횟수를 줄이고 속도를 향상시키며 네트워크 연결이 끊어진 경우에도 웹 애플리케이션이 동작할 수 있도록 하여 웨어러블 환경에 적합한 인터페이스로 평가된다.

3. 저전력 스케줄러

태스크 스케줄링은 운영체제의 가장 핵심 요소이며 수많은 연구가 진행되고 있는 분야이다. 초기 연구는 단일 프로세서 코어에 대한 것이었지만 멀티 코어가 대중화됨



〈그림 14〉 VILCF 스케줄링 알고리즘 예시

에 따라 멀티 코어에서의 스케줄링 기법에 대한 연구가 다수 진행되었다. 뿐만 아니라 최근에는 모바일 기기의 저전력화 요구에 맞추어 전력 소모를 줄이는 스케줄링 기법에 대한 연구도 활발히 진행되고 있다.

프로세서의 전력 소비를 줄이는 가장 보편화된 기법으로는 DVFS를 들 수 있다.[5] DVFS는 앞서 언급한 바와 같이 프로세서의 부하량에 따라 전압 및 주파수를 동적으로 변경하여 전력소모를 줄이는 방법이다. 저전력 스케줄링에 대한 연구들은 이러한 DVFS 기법을 태스크 스케줄러와 연동하여 다룸으로써 DVFS 정책에 의하여 변경된 주파수가 데드라인을 놓치지 않고 나아가 에너지 효율적인 스케줄링을 달성하도록 하고 있다.

이러한 동향에 따라 최근 미시간 대학의 연구팀은 멀티 코어에서의 에너지 효율적인 스케줄링을 위하여 VILCF (Voltage Island Largest Capacity First) 스케줄링 알고리즘을 제안하였다.[17] 이 연구는 대부분의 멀티코어 프로세서가 설계의 복잡성 문제로 개별 코어에 개별 전압을 공급하지 않고 그룹화 된 코어들에 하나의 전압만을 공급한다는 점에 착안하였으며, 이 때 데드라인을 놓치지 않도록 하기 위해서는 그룹화 된 코어들 중 가장 부하가 심한 코어에 맞추어 DVFS 정책을 이행해야 하는 한계를 지닌다. 따라서 하나의 그룹 안에서 높은 전압 및 주파수에도 불구하고 부하가 적은 태스크를 수행함으로 인해 가용한 처리성능을 가진 코어가 발생하게 되고 이 가용 자원을 Capacity라 부르고 있다. 모든 코어의 Capacity가 0이 되도록 스케줄링을 하는 것이 최적인데, 이는 NP-hard 문제에 해당한다. 즉, 최적의 스케줄링은 계산시간이 오래 걸리며 근사화된 방법으로 가용한 처리성능이 가

장 큰 코어에 태스크를 할당하는 방식이 VILCF스케줄링 알고리즘이다.

구체적으로 VILCF 알고리즘은 모든 태스크에 대해 먼저 Capacity가 가장 큰 코어를 찾는다. 태스크의 부하가 이 코어의 Capacity보다 작다면 해당 코어에 태스크를 할당하지만, 그렇지 않은 경우, 부하량이 가장 작은 코어에 태스크를 할당한다. 코어 후보가 여럿 있다면 인덱스가 가장 낮은 코어를 선택하여 할당한다. 이와 같은 한 번의 스케줄링이 이루어질 때마다 각 코어가 처리할 부하량에 따라 코어들을 내림차순으로 정렬하여 다음 태스크에 대한 검색을 준비한다. 이러한 알고리즘의 시간 복잡도는 에 해당하며 여기서 는 태스크 개수를 의미한다.

VILCF 스케줄링 알고리즘의 동작 과정을 예를 들어 살펴보면 다음과 같다. 두 개의 그룹이 있고 각 그룹 내에 네 개의 코어가 있으며 태스크 집합 T = {6, 5, 5, 4, 3, 3, 2, 2, 2, 2}가 있다고 가정할 때, 태스크 1은 부하량이 6이며 모든 코어의 Capacity가 0이므로 그룹 1의 코어 1에 할당된다. 이에 따라 그룹 1의 코어 1의 Capacity는 0이 되고 나머지 코어들의 Capacity는 6이 된다. 태스크 2, 3, 4는 그룹 1의 코어 2, 3, 4가 가진 Capacity보다 부하량이 작으므로 각각 그룹 1의 코어 2, 3, 4에 할당된다. 그리고 이 코어들의 Capacity는 1, 1, 2로 갱신된다. 태스크 5는 그룹 1에서 적합한 Capacity를 지닌 코어를 찾지 못하여 그룹 2의 코어 1에 할당된다. 마찬가지로 그룹 2 코어 1의 Capacity는 0이 되며 그룹 2의 나머지 코어들의 Capacity는 3이 된다. 이와 같은 과정을 반복적으로 수행하면 〈그림 14〉에 나타난 결과를 얻을 수 있다.

이러한 일련의 스케줄링을 통하여 코어마다의 부하량이 결정되고, 데드라인 넘지 않기 위해 각 그룹별로 가장 높은 부하량을 가진 첫 번째 코어에 맞추어 해당 그룹의 전압 및 주파수를 결정하므로 결과적으로 Capacity가 최소화되어 모든 코어를 에너지 효율적으로 사용하게 된다.

Ⅳ. 결론

스마트 폰 이후 IT 업계의 새로운 화두로 떠오르고 있는 웨어러블 컴퓨터는 사용자의 몸에 항시 착용되는 특성으로 인하여 매우 제한적인 자원을 가진다. 본 고에서는 이러한 웨어러블 컴퓨터의 제한조건을 극복하기 위해 주요 웨어러블 운영체제들이 가지는 구조 및 특성에 대해 살펴보았다. 또한 다른 컴퓨팅 모델과 달리 웨어러블 컴퓨터에서 발생하는 주요 이슈인 NUI, 클라우드 인터페이스, 초저전력 기법에 대하여 효과적으로 접근하고 있는 최신 연구들에 대해 살펴보았다. 결론적으로 웨어러블 운영체제는 다양한 입력 수단을 지원하기 위해 통일된 규격의 UI 구조를 가지게 되며, 타 플랫폼과의 호환성을 위하여 웹 기반 구조를 채택하고, 클라우드와의 연계를 통하여 웨어러블 단말의 부족한 자원을 극복해 나아갈 것으로 전망된다.

웨어러블 운영체제는 다양한 입력 수단을 지원하기 위해 통일된 규격의 UI 구조를 가지게 되며, 타 플랫폼과의 호환성을 위하여 웹 기반 구조를 채택하고, 클라우드와의 연계를 통하여 웨어러블 단말의 부족한 자원을 극복해 나아갈 것으로 전망

감사의 글

본 연구는 산업통상자원부 및 한국산업기술평가관리원의 산업핵심기술개발사업의 일환으로 수행하였음. [10049270, 웨어러블 스마트 기기를 위한 컴퓨터비전 기반 UI/UX용 SoC 및 SW 플랫폼 연구]

참 고 문 헌

[1] 손용기, "웨어러블 컴퓨터 기술 현황과 전망," 한국산업기술진흥협회 기술과경영, Oct. 2013

[2] Olga Gadyatskaya, et al., "Security in the Firefox OS and Tizen Mobile Platforms," Computer, Vol. 47, no 6, pp. 57–63, 2014

[3] Sachin Dev Sharma and Taesoo Jun, All Connected with Tizen, Connect yours!, Samsung Developer Conference, 12 Nov. 2014, available on: http://samsungdevcon.com

[4] Elke Mackensen, et al., "Performance analysis of an Bluetooth Low Energy sensor system," IEEE 1st International Symp. on

Wireless Systems (IDAACS-SWS), pp. 62-66, Sept. 2012

[5] Kevin J. Nowka, et al., "A 32-bit PowerPC system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling," IEEE J. of Solid-State Circuits, Vol. 37, no 11, pp. 1441-1447, Nov. 2002

[6] Xueqiang Wang, et al., "DeepDroid: Dynamically Enforcing Enterprise Policy on Android Devices," Proc. of 18th Annual Network and Distributed System Security Symposium (NDSS), Feb. 2015

[7] Meghan Desai and Matthew Williams, "Introduction to Project Volta," Google I/O 2014, 25 June 2014, available on: http://www.google.com/events/io

[8] iOS Technology Overview, 17 Sept 2014, available on: http://developer.apple.com

[9] Rick Merritt, "Slideshow: 10 Embedded Design Trends," EE Times, 21 Apr. 2014

[10] Shancao Niu, et al., "Analysis and Implementation of Migrating Real-Time Embedded Operating System FreeRTOS Kernel Based on S3C44B0 Processor," IEEE International Symp. on Information Science and Engineering (ISISE), pp. 430-433, Dec. 2012

[11] Jean, J. Labrosse, MicroC/OS-II the real-time kernel, CMP Books, 2003

[12] embOS Real-Time Operating System CPU-independent User & Reference Guide, Segger Microcontroller GmbH & Co. KG, Mar. 2015, available on: http://www.segger.com

[13] Albert Ng, et al., "Designing for low-latency direct-touch input," Proc. of the 25th annual ACM symp. on User interface software and technology, pp. 453-464, Oct. 2012

[14] Stefan Schneegass, et al., "Towards a garment OS: supporting application development for smart garments," Proc. of the 2014 ACM International Symp. on Wearable Computers: Adjunct Program, pp. 261-266, Sept. 2014

[15] Kiryong Ha, et al., "Towards wearable cognitive assistance," Proc. of the 12th annual international conf. on Mobile systems, applications, and services, pp. 68-81, June. 2014

[16] 채원석 외 5명, "모바일 웹 앱을 위한 HTML5 및 프레임워크 동향," 전자통신동향분석, pp. 92-100, June 2012

[17] Jun Liu and Jinhua Guo, "Voltage Island Aware Energy Efficient Scheduling of Real-Time Tasks on Multi-core Processors," IEEE International Conf. on High Performance Computing and Communications, IEEE 6th International Symp. on Cyberspace Safety and Security, IEEE 11th International Conf. on Embedded Software and System (HPCC,CSS,ICESS), pp. 645-652, Aug. 2014

황 재 민

- 2014년 충남대학교 메카트로닉스공학/컴퓨터공학 복수전공 학사 졸업
- 2014년 ~ 현재 충남대학교 컴퓨터공학 석사과정

〈관심분야〉
웨어러블 플랫폼, 모바일 GPU, SW-SoC 융합

남 병 규

- 1999년 경북대학교 컴퓨터공학 학사 졸업
- 2001년 KAIST EECS 석사 졸업
- 2007년 KAIST EECS 박사 졸업
- 2001년 ~ 2002년 ETRI 컴퓨터시스템 연구부 연구원
- 2007년 ~ 2010년 삼성전자 SystemLSI 사업부 책임연구원
- 2010년 ~ 현재 충남대학교 컴퓨터공학과 조교수

〈관심분야〉
모바일 GPU, 임베디드 CPU, 저전력 SoC 설계, 임베디드 SW 플랫폼