NexWave 소프트웨어 인프라스트럭처 개요

임베디드 소프트웨어 개발툴 'NSI'

넥스웨이브솔루션에 의해 개발된 임베디드 소프트웨어 개발툴인 'NSI(NexWave Software Infrastructure)'는 개발비용 부담을 줄이고, 생산성을 향상시킬 수 있는 혁신적인 솔루션이다. 지금부터 NSI의 소개와 함께 그 특징들을 살펴보자.

자료제공: 넥스웨이브솔루션/www.nexwave-solutions.com

컴포넌트화 된 소프트웨어 인프라스트럭처는 생산성을 크게 증대시키는 시도로서 널리 인식되고 있다. 즉, 제품 출시에 소요되는 시간을 줄이고(타임-투-마켓), 품질의 향상과 유지보수가 쉽다. 왜냐하면 이미 개발된 소프트웨어를 재사용할 수 있고, 유연하게 재배치가 가능한 컴포넌트 기반이기 때문이다.

현재 비즈니스, 데스크톱 그리고 서버 등의 소프트웨어들은 CORBA와 같은 비동시성의 메시지 전달 컴포넌트 모델을 사용하여 애플리케이션 레벨의 컴포넌트화와 런-타임 업그레이드를 시도하고 있지만, 이러한 작업은 앞으로도 많은 노력을 필요로 한다.

일부 가전업체들(CE Manufacturers)은 임베디드 애플리케이션의 컴포넌트화를 목적으로 내부 소스코드에 기초를 둔 'Build-Time' 컴포넌트 모델의 개발을 시도해 왔지만, 임베디드 시장을 위한 상업적 컴포넌트 시스템은 업계에 거의 전무한 상황이다.

계속되는 소프트웨어 개발비용의 증가와 이에 따른 부담에 대응하기 위해 NexWave는 모든 OS에 이식이 가능한 효율적인 컴포넌트 인프라스트럭처를 개발하는데 주력해 왔다.

NexWave 소프트웨어 인프라스트럭처(NSI)는 호스트 OS와는 독립적으로 컴포넌트화를 해주는 모델로서 모델 컴포넌트를 정의하고, 서로 상호 작용하는 컴포넌트들로 이루어진 완벽한 도구이다.

NSI 컴포넌트 바이너리 파일들은 다양한 임베디드 OS에 호환될 수 있도록 만들어지며, 모든 플랫폼에서 설치되어 작동한다. NSI는 다른 비즈니스 컴포넌트 인프라스트럭처와는 달리 동시적인 운영을 통해 컴포넌트들끼리 서로 의존해 동작하게 하는 수동적 모델이다.

이러한 방법은 프로세서나 메모리 자원의 사용에 있어 경제적일 뿐만 아니라, 컴포넌트 인프라스트럭처의 리얼-타임 오퍼레이션의 완벽성을 보장한다. 또한 임베디드 소프트웨어 개발자들에게 익숙한 C나 C++ 언어의 사용을 가능하게 한다.

NSI는 리얼-타임 성능을 저하시키지 않는 매우 효율적이고 혁신적인 프로그래밍 모델이다. Live 업그레이드가 필요치 않은 이상, 개발 시스템 상에 있는 NSI 컴포넌트들을 통해 전체 소프트웨어 시스템을 구성하는 것이 가능하다. 이를 위한 어떤 런-타임 소프트웨어도 필요 없다.

NSI는 컴포넌트 기반 소프트웨어 개발의 일반적 장점과 더불어 다음과 같은 강점을 가지고 있다:

- 컴포넌트들의 재사용을 통한 소프트웨어 개발의 생산성과 품질 향상

- 생산과정의 자연스런 통합, 쉬워진 시제품 개발, 발 빠른

시장대응, 기존 컴포넌트들의 유연한 조합.

- 지속적으로 증가하는 소프트웨어 개발비용을 컴포넌트 바이너리 표준 파일로 생산자간 교환 및 재사용으로 IPR과 라이선스를 간단히 함으로써 투자비용 회수.

- NexWave의 런-타임 dependency 운영과 소프트웨어 인프라스트럭처를 사용함으로써 소프트웨어 업데이트와 확장.

- 가전, 통신, 네트워크 등 광범한 분야에서 소프트웨어를 목적에 맞게 재구성 할 수 있음.

- 소프트웨어 운영과 배치 솔루션으로서 컴포넌트 모델, 바이너리 포맷, 컴포넌트 생성, 어셈블리와 컨피규레이션 운영 도구를 포함하고 있으며, 런-타임 인프라스트럭처, 소프트웨어 업데이트와 확장을 위한 도구가 옵션으로 제공된다.

NSI

NSI는 효과적인 컴포넌트 구조에 기초를 둔 완벽한 소프트웨어 관리 및 개발 솔루션이다. NSI를 사용하게 되면 임베디드 소프트웨어 개발시 독립된 컴포넌트들로 개발이 가능하고, 생산 라인에 맞춰 다양한 구성을 가질 수 있다. 그에 따라 재사용과 독립적인 업데이트가 가능하다.

NSI의 특징 중 하나는 소프트웨어를 바이너리 레벨의 컴포넌트들로 분리한다는 점이며(단순한 소스 코드 레벨이 아닌), 컴포넌트들로 하여금 시스템 내에서 오프라인(statically) 또는 임베디드 타깃에서 온라인상(dynamically)으로 개발 및 조합이 가능하다.

컴포넌트 구조

컴포넌트 구조는 소프트웨어 컴포넌트들을 분석 및 구체화하고, 다른 컴포넌트들에게 제공되는 기능을 어떻게 전달할지, 그리고 다른 컴포넌트로부터 필요로 하는 기능들을 어떻게 전달할지 등을 지정해준다.

아울러 어떻게 컴포넌트들이 바이너리 형태로 저장이 되고, 어떻게 특정 시스템에 대하여 바이너리 컴포넌트들이 조합되는지, 그리고 어떻게 컴포넌트들이 물리적으로 서로 묶여지는지에 대해 설명한다.

컴포넌트 구조는 다음과 같이 정의된다:

- '.dep' 언어: 컴포넌트와 인터페이스를 모델화 하는 데에 쓰인다.

- 컴포넌트 인터페이스: 의존적인 운용과 컴포넌트 어셈블리 업무를 위해 필요한 컴포넌트의 설명.

- Standard data type: 컴포넌트 인터페이스를 정의하기 위해 쓰이는 일반적 데이터 타입들

- XCOM 바이너리 포맷: 코드, 데이터, 그리고 바이너리 된 컴포넌트의 인터페이스 저장 방법, 컴포넌트간의 통신을 위한 인터페이스 정의

Building

NSI는 소스 내에 있는 컴포넌트들을 독립적인 바이너리로 변경하는 커맨드 라인 도구를 제공한다. 그리고 이를 통해 만들어진 바이너리 컴포넌트들은 제공된 구성에 따라서 소프트웨어 시스템으로 조합된다.

- NexDev: NSI 컴포넌트들을 컴파일 하는 Makefiles의 표준 모음으로 상용 컴파일러들처럼 무료로 공급된다. 개발자의 내부 개발 툴을 지원할 수 있도록 쉽게 확장이 가능하다.

- Depc: '.dep' 인터페이스 컴파일러이다.

- GNU gcc: NSI에 의해 지원되는 모든 프로세서 아키텍처를 위한 디폴트 컴파일러로서, 아키텍처 기반으로 상업적 컴파일러나 내부 컴파일러로 대체가 가능하다.

- GNU binutils와 ld: XCOM 포맷에서 바이너리 컴포넌

트들을 만드는데 쓰이는 링커(linker)이다.

- xcomstrip/xcompostld: XCOM 포맷에서 바이너리 컴포넌트에 대한 'production code optimizer' 이다.

- Mkimage: 독립적인 컴포넌트들을 실행 가능한 이미지로 해주는 패키지. 컴포넌트들을 NSI 런-타임 시 독립적인 바이너리 형태들로 남아있고, 타깃 디바이스에 독립적으로 다른 컴포넌트와 교체되거나 침투될 수 있다.

- Mksimplefs: 파일 시스템 이미지 생성 도구이다.

Framework

NSI는 모든 임베디드 애플리케이션과 RTOS abstraction layer에 의하여 요구되는 기본적인 서비스들을 제공하는 컴포넌트들로 이루어져 있다. 이러한 RTOS abstraction layer는 개발자로 하여금 RTOS에 독립적인 컴포넌트를 만들게 해준다.

- NexGUI: NexGUI는 가전기기와 임베디드 애플리케이션을 위한 GUI(graphical user interface) framework을 제공한다. NexGUI의 적용 범위는 프레임 버퍼를 액세스하는 소프트웨어부터 데스크톱 레벨의 애플리케이션 영역까지 해당한다.
  NexGUI는 핵심 컴포넌트들의 집합으로 이루어져 있으며(이미지 디코더, 폰트 디코더, clipper/drawer, widget engine 등을 포함), 높은 호환성과 유연성을 제공한다. 특히 뛰어난 동작성과 작은 풋프린트를 가지며, 프로그래머들이 자유자재로 그 핵심 컴포넌트에 다른 컴포넌트들을 추가하거나 변형시킬 수 있도록 구성되어 있다.
  또한 NSI 런-타임과 NexGUI 컴포넌트들이 개발자의 PC에서 사용 가능하므로, 타깃 하드웨어를 기다릴 필요 없이 애플리케이션을 PC상에서 개발할 수 있다.

- LibTRON는 μTRON 3.0과 4.0 legacy code를 어떤 NSI 환경(리눅스, 윈도우 등)에서도 작업할 수 있도록 도와주는 프로그램이다. LibTRON은 컨피규레이션 도구와 NSI 컴포넌트들로 구성되어 있다.
  LibTRON 컴포넌트는 어떤 NSI 환경에서도 주어진 RTOS 추상화(abstraction) 위에서 실행된다. NSI는 리얼타임 서비스를 제공하는 컴포넌트들을 생성할 수 있게 하고, OS에 상관없이 사용될 수 있다는 것을 보여준다.

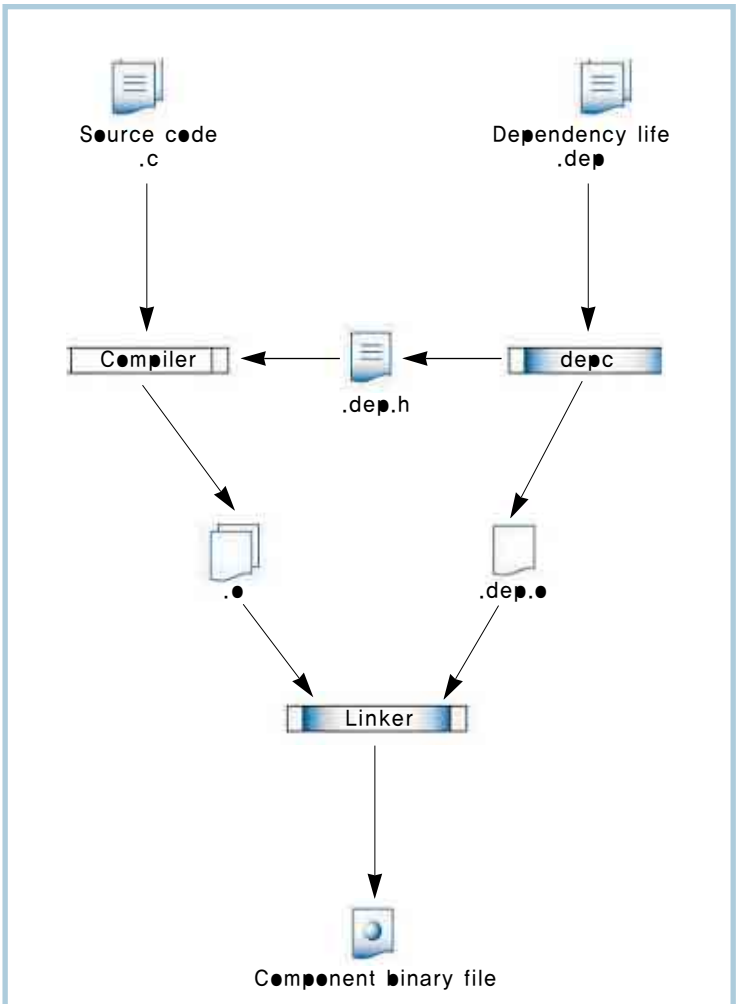

그림 1

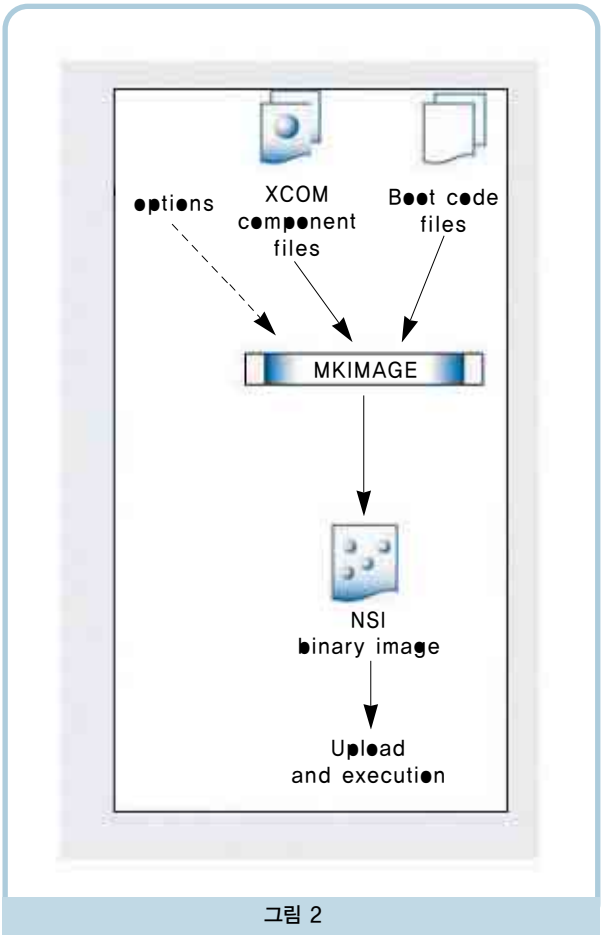

그림 2

배치 툴(Deployment Tool)

command-line 툴은 개발자와 설계자들이 소프트웨어 시스템 내에서 컴포넌트를 조합하고 타깃 디바이스에서 수정할 수 있는 그래픽 개발 환경에서 동작한다.

- NexBuilder: NexBuilder는 build-time(off-line) 환경이다. 컴포넌트들을 목록화 하고 브라우징 하며, 시스템 컨피규레이션 관리(templates, selection of policies and implementations), 컴포넌트 조합, dependency checking이나 소프트웨어 시스템의 생성 등의 일을 맡는다.

- NexGUIBuilder: NexGUIBuilder는 NexGUI framework을 사용하여 쉽게 컴포넌트를 제작할 수 있도록 해주는 그래픽 툴이다. 내부적으로 상용화가 가능한 NSI 환경에서 동작할 수 있는 유일한 기능 덕분에, 실제의 WYSIWYG(what you see is what you get) 상황을 제공한다. 또한 소프트웨어 아키텍처에 맞도록 컴포넌트화가 가능하다.

런-타임 인프라스트럭처

NSI의 옵션인 런-타임 인프라스트럭처는 'run-time dependency management' 기능을 제공하며, 타깃 디바이스에서 소프트웨어를 위해 swapping(updates)과 probing, 그리고 extending을 하게 한다.

NSI는 개발자에게 소프트웨어 컴포넌트를 임베디드 상품을 위한 완전한 소프트웨어로 만들 수 있도록 하게 하는 일련의 규격과 도구를 제공한다. 전체 생성 방법은 두 단계로 요약된다. 먼저 싱글 컴포넌트를 개발한 후, 그 컴포넌트를 구성하고 재배치해 시스템을 만든다.

이 진행 과정에서 actual 코드와 인터페이스 definition이 생성된다. 코드와 인터페이스 모두 바이너리 포맷으로 표현된다. 최종 결과물은 하나의 바이너리 컴포넌트가 된다.

이 단계에서 컴포넌트 간의 dependency가 정의되고 운영되며, 모든 선택된 컴포넌트들이 타깃의 환경에 따라 조합된다.

On-line dependency는 마지막 이미지에 포함되어 있는 NSI 런-타임 컴포넌트들에 의해 다뤄진다. 이 컴포넌트들은 dependency를 동적으로 그리고 최종 bootable 이미지 내에 있는 컴포넌트의 교환 및 삽입을 주관한다.

NSI의 특징

Total life cycle 소프트웨어 관리

OOP(Object Oriented Programming)와 같은 개발환경 툴이나 생산성 향상 툴과는 달리 NSI는 임베디드 소프트웨어 컴포넌트의 완벽한 life cycle을 관리한다. 기존의 툴은 컴파일과 디버깅 단계 후에 종료된다. 이에 반해 NSI는 유연한 설정에 따라 컴파일 된 바이너리 컴포넌트가 나중에 조합되는 것을 가능하게 한다. 또한 원격 진단과 원격 지원과

같은 새로운 기능들이 개별적인 컴포넌트를 업데이트하는 생산자에게 연결시켜주는 'resulting software system'을 제공한다.

NSI는 컴파일부터 large 시스템에 조합되고 디버깅, 배치, 원격 진단, 새로운 버전에 의해 교체 될 때까지 따라 다닌다. 소프트웨어의 개발 및 배치, 품질보증, 공장시험에 대한 완벽한 툴 세트를 제공한다.

Self-contained binary components

NSI는 독립적으로 컴파일 된 바이너리 컴포넌트로부터 완벽한 소프트웨어 시스템을 조합한다. 이러한 형상은 소프트웨어 개발자가 재설정이 필요한 소프트웨어 시스템에 대하여 소스코드를 공급해야 하는 필요성을 없도록 해준다.

컴포넌트들은 각각의 프로세서 아키텍처에 대해 컴파일 되고 가상 머신 없이 실행된다. NSI 컴포넌트 관리는 다른 단계와 해야 할 일들을 인식한다. 컴파일 프로세스는 self-contained 바이너리 컴포넌트를 만든다. 이 단계에서 컴포넌트들은 각자의 타깃 환경으로부터 완전히 독립적이다. 적절한 타깃 환경에 대한 준비는 세부적인 조합 절차를 거쳐 일어난다.

Complete component assembly와 configuration support

컴포넌트들에게 각자의 독립성을 표현하기 위하여 NSI는 동작하고 있는 소프트웨어 시스템에 다양하게 공급되는 유기적인 컴포넌트들의 조합을 허용한다. NSI는 개발자가 self-contained, 그들의 소프트웨어 시스템으로부터 또는 외부로부터 바이너리 컴포넌트를 갖는 것을 허용한다.

Online management of 'live' components

NSI는 런-타임으로 'live' 타깃에 컴포넌트들을 관리한다. NSI는 실질적인 자체진단을 하는 툴로서 QA팀과 FT(Factory Tester)에게 제공하며 타깃 상에서 돌아가는 소프트웨어의 단위 시험으로서 제공한다.

NSI는 생산자 스스로가 타깃을 식별할 수 있도록 자체 진단을 제공하며, 원격 지원으로 개별적인 소프트웨어 컴포넌트들을 확실하게 다운로드 받을 수 있다. 동시에 생산자에 의해 암호화된 서명을 받게 한다. NexWave는 새로운 컴포넌트들과 dependency들을 가져오는 것을 다루는 것과 런-타임 컴포넌트들에 대해서 동적으로 액세스하는 것을 다루는 자체의 컴포넌트 세트를 개발해 왔다. 이러한 'run-time support components'는 NSI 패키지의 일반적인 부분이다.

NSI의 장점

NSI는 일반적으로 컴포넌트 기반의 소프트웨어 개발이 가질 수 있는 장점 이외에도 호환성 및 효율성, 그리고 타임-투-마켓 등 이점을 제공한다.

NSI는 호스트 OS와 독립적인 동기방식의 모델(synchronous model)을 사용하는 특성 때문에 매우 효율적이다. 이 모델은 주어진 소프트웨어 환경 내에서 NSI를 채택함으로써 전형적인 프로그램 방식으로 빠져들게 한다.

- Legacy system은 '컴포넌트화'와 새로운 컴포넌트들로 조합 되어질 수 있다.
- 성능은 컴포넌트화 되지 않은 코드와 동등하다.
- 풋프린트는 관리될 수 있으며, 빌드-타임 어셈블리의 경우 오버헤드가 전혀 없다.
- 호스트 OS로부터의 독립성은 시스템 개발자로 하여금 기존 환경을 보존하고 특정한 플랫폼이나 벤더에 종속되는 것을 막아준다.

또한 바이너리 포맷의 장점은 지적 자원의 보호와 컴포넌트 공급자에 대한 신뢰성 부담을 줄여주는데, 서로 다른 팀 간에 컴포넌트를 교환하는 것을 가능하게 하는 것이 핵심 기술이다(비즈니스 유닛과 생산자간 바이너리를 교환함으로써 소프트웨어에 대한 투자를 점점 더 많이 회수하게 한다). NSI의 바이너리 포맷과 효율성은 하나의 운영체계 API를

위해 상품을 정형화 하는 것 보다 훨씬 큰 경제적 이익을 제공한다.

Swapping and probing

런-타임 dependency 관리 능력을 갖추고 있는 NSI는 동시에 독립적인 컴포넌트들의 동적 로딩 및 언로딩을 할 수 있다. 개발환경에서 시스템 내부에 형성된 컴포넌트들은 수정되거나 재 컴파일하지 않고 런-타임 시에 관리되어 질 수 있다.

네트워크 능력을 갖춘 디바이스에 배포될 때에, 런-타임 인프라스트럭처에 의하여 관리되는 컴포넌트들은 개별적으로 식별이 가능하고 디바이스가 동작될 때 교체가 가능하다. 이러한 능력은 제품 개발 및 관리에 대하여 광범위한 영역에서 이익을 창출한다.

조기 시제품 제작(assembled from configurable, 바이너리 컴포넌트)

NSI는 생산자로 하여금 준비 가능한 바이너리를 유연하게 조합되도록 배타적으로 설계할 수 있다. 재사용과 구성 능력은 시제품에 대하여 자연적으로 따라오는 이점으로 NSI는 기존 컴포넌트를 이용해 조기에 시제품을 제작할 수 있다.

결 론

NSI는 시작부터 임베디드 소프트웨어 시장을 겨냥해 설계되었으며, μTRON, 리눅스 등 다양한 플랫폼에서 사용할 수 있다.

특정 환경에서 NSI의 사용을 제한하는 어떠한 기술적 한계는 없으나, NSI의 형상은 특정한 시장에 더욱 자연스럽다. 바이너리 포맷은 실리콘 제품과 함께 소프트웨어 패키지가 늘어나는 반도체 시장에서 특히 더 유용하다.

특히 NSI의 풍부한 컴포넌트 구성과 어셈블리는 적은 마진과 잦은 상품 개발이 이뤄지는 가전시장에서 효율적이다. 가전업체들은 생산 과정을 체계화하기를 원해 왔으며, 가전제품 내에 있는 소프트웨어의 크기와 복잡성이 점점 커지는 것을 고려할 때 정형화된 생산 방식이 소프트웨어에도 적용되길 요구하고 있다. NSI는 소프트웨어 개발 및 관리의 최적화를 통해 완벽하게 지원한다.

요 약

- 동작: 기존의 컴포넌트들을 이용하여 신속하게 훌륭히 동작하는 프로그램을 만든다.

- 유지: 컴포넌트화 된 솔루션의 모듈 구조는 개별 컴포넌트들의 교환을 매우 쉽게 한다.

- 효율성: 동작 장애들은 쉽게 발견될 수 있고, 그러한 장애들을 위한 조정은 몇몇 동작에 필요한 컴포넌트들만 찾아냄으로써 쉽게 해낼 수 있다.
  컴포넌트들은 더 나은 동작을 위해 그들의 고유한 성격을 변화시키지 않고 재구성 할 수 있다. 더 나은 동작을 위해 플랫폼 간에 옮겨 다닐 수 있고, 컴포넌트들은 애플리케이션의 동작이나 사용에 전혀 영향을 미치지 않고도 재구성 될 수 있다.

- 안정성: 정형화되고 완벽한 컴포넌트의 정의가 이루어지게 된다면, 컴포넌트의 안정성은 간단한 질문에 달려있다. 컴포넌트가 정의된 업무를 정확하게 처리하는가? 전체적인 애플리케이션은 다른 주제이지만, 컴포넌트가 안정되어 있다면 그 컴포넌트들이 모여 만들어진 애플리케이션의 안정성은 크게 증가될 것이다.

- 유동성: 컴포넌트의 특정화는 플랫폼의 영향을 받지 않는다. 그러므로 새로운 플랫폼을 위해 새로운 형태의 컴포넌트가 재생산될 수 있고, 변형된 컴포넌트는 다른 컴포넌트에 전혀 영향을 미치지 않는다.

NSI에 대한 더 자세한 정보를 원하시면, 웹사이트 www.nexwave-solutions.com나 한국 판매 대리점 GQTECH (www.gqtech.co.kr)에 문의하면 된다. RTime