PC기반의 생산시스템을 위한 운용소프트웨어 구조

An Operating Software Architecture for PC-based Manufacturing Systems

박 남 준, 김 홍 석, 박 종 구
(Nam-Joon Park, Hong-Seok Kim, and Jong-Koo Park)

Abstract : In this paper, a new architecture of operating software associated with the component-based method is proposed. The proposed architecture comprises an execution module and a decision-making module. In order to make effective development and maintenance, the execution module is divided into three components. The components are referred to as Symbol, Gateway, and Control, respectively: The symbol component is for the GUI environments and the standard interfaces; the gateway component is for the network communication and the structure of asynchronous processes; the control component is for the asynchronous processing and machine setting or operations. In order to verify the proposed architecture, an off-line version of operating software is made, and its steps are as follows: i) Make virtual execution modules for the manufacturing devices such as dual-arm robot, handling robot, CNC, and sensor; ii) Make decision-making module; iii) Integrate the modules and GUI using a well-known development tools such as Microsoft's Visual Basic; iv) Execute the overall operating software to validate the proposed architecture. The proposed software architecture in this paper has the advantages such as independent development of each module, easy development of network communication, and distributed processing of resources, and so on.

Keywords : component, interface, activeX, MTS, UML, PC-based manufacturing system

I. 서론

미래의 경쟁사회에서는 소비자의 다양한 요구를 충족시킬 수 있는 제품을 생산하는 기업만이 살아남을 수 있다. 즉, 차별화된 디자인과 모델의 제품을 적절한 시기에 만들어 내야 한다. 이를 위해서 생산시스템은 새로운 제품생산을 위한 빠른 시스템의 변경과 새로운 기능을 쉽게 추가할 수 있는 유연한 시스템이어야 한다[5].

최근에는 이러한 유연한 생산시스템의 구축을 위하여 PC 기반의 생산시스템에 대한 연구가 활발히 이루어지고 있다[9]. 이러한 시스템은 모든 작업과 추가되는 기능을 소프트웨어로 처리할 수 있기 때문에 유연한 생산시스템을 구축할 수 있으며, 작업기기의 하드웨어 구성을 범용 PC로 쉽게 연결할 수 있는 장점이 있다. 그러나 PC기반의 생산시스템은 운용소프트웨어의 개발과 유지보수에 많은 비용이 들어가는 문제점이 있다. 이는 기존의 소프트웨어 구조와 개발방법론의 문제로부터 기인한다. 따라서 본 논문에서는 기존의 소프트웨어의 문제점을 분석하여 해결방안을 찾으며 운용소프트웨어의 새로운 구조를 제안한다.

기존의 생산시스템을 위한 운용소프트웨어의 개발에 주로 사용되었던 절차지향방법은 필요한 기능을 개발자가 직접 함수로 만들어 순차적으로 호출하면서 문제를 해결하는 것으로 개발이 쉽고 구조가 단순하다. 하지만 이 방법은 개발의 비용과 시간이 많이 들며, 개발한 함수의 비표준화로 인하여 재사용이 어렵고, 자료의 관리와 보호가 어렵다.

이러한 문제를 해결하고자 객체지향방법을 적용하여 운용 소프트웨어를 개발하고 있다. 객체지향방법은 현실세계의 시스템을 사용자의 관점에서 추상화하여 모델링하는 것이다. 이 방법론의 장점은 친숙한 사용자 인터페이스(interface)를 제공한다는 것과 소프트웨어 개발을 위한 모델링과 방법론을 제공한다는 점이다. 그러나 객체지향방법도 참조의 문제, 소스차원의 재사용, 객체지향의 분석 및 설계의 어려움 등의 문제점이 있다. 따라서 객체지향방법을 사용한 운용소프트웨어도 시스템 개발과 수정에 상당한 시간과 노력이 필요하다.

이에 대한 한 가지 해결책으로서 최근에는 컴포넌트(component) 기반의 운용소프트웨어가 개발되고 있다. 컴포넌트란 하나의 완성된 응용 소프트웨어를 만들기 위해서 개발된 단위기능의 소프트웨어를 말한다[7]. 컴포넌트를 이용하면 생산성과 품질을 높일 수 있고, 객체지향방법의 한계점을 극복할 수 있다. 컴포넌트의 장점은 다음과 같다. 첫째, 컴포넌트는 객체간의 통신을 하기 위해 인터페이스를 이용하므로, 객체지향방법에서의 객체간 참조 문제가 없다. 둘째, 컴포넌트는 미리 검증된 작은 기능을 바이너리 형태로 제공하므로 소스코드 수준의 재사용을 위해 소스코드를 이해해야 하는 어려움이나 소스공개의 문제가 없다. 셋째, 전체 시스템을 분석하고 설계하는 어려움 없이 작은 기능의 부분만 설계하여 컴포넌트로 개발하므로 시스템 설계가 용이하고, 상용의 컴포넌트를 이용할 수 있어 쉽고 빠르게 응용시스템의 개발이 가능하다[7]. 컴포넌트를 개발하기 위해서는 컴포넌트의 개발을 위한 표준규약이 필요하며, 본 논

접수일자 : 2000. 1. 29., 수정완료 : 2000. 8. 29.
박남준, 박종구 : 성균관대학교 전기전자 및 컴퓨터 공학부
김홍석 : 한국생산기술연구원

문에서는 마이크로소프트에서 제안한 컴포넌트 표준규약인 COM(component object model)을 사용한다. COM을 사용한 이유는 PC 기반의 시스템에서 사용하기 쉽고, 마이크로소프트의 운영체계에서 잘 동작하기 때문이다[6].

위에서 설명한 바와 같이 운용소프트웨어 개발을 위해서 최근에는 소프트웨어의 기술과 PC의 우수한 성능이 연결된 생산시스템이 개발되고 있다[8]. 이러한 기술개발의 흐름에서 본 논문에서는 다품종 소량생산 체계를 위한 PC기반의 생산시스템에 사용되는 운용소프트웨어를 컴포넌트 구성된 효율적인 운용소프트웨어의 새로운 구조를 제안한다. 기존의 연구들은 개념적인 구조와 전략만을 제시하였다[4][5]. 또한 물리적인 환경은 고려하지 않고 단지 운용소프트웨어 자체의 효율성만을 강조하여 운용소프트웨어의 개발과 수정이 전체 운용소프트웨어에 영향을 주는 비효율적인 구조도 많았다[3].

따라서 본 논문에서는 이러한 문제점의 대안을 찾고 컴포넌트 소프트웨어의 기술과 PC의 장점을 통합하여 새로운 운용소프트웨어를 개발하고자 한다. 이를 위해 논문에서는 먼저 생산시스템의 물리적인 구조를 바탕으로 운용소프트웨어를 분석 및 설계를 하며 전체 구조를 설계한다. 또한 개발과 유지보수 비용을 줄일 수 있도록 각 시스템을 최대한 모듈화하며 모듈간의 통신을 단순화하고 물리적인 작업기기의 변동에도 유연하게 대처할 수 있는 구조가 될 수 있도록 한다. 마지막으로 본 논문에서 제안한 구조를 쉽게 작성하여 하나의 운용소프트웨어로 통합하는 방법에 대해 설명한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 본 논문에서 제안한 운용소프트웨어의 구조에 대해서 기술하고, 제 3장에서는 제안된 운용소프트웨어의 구조를 작업셀 시스템의 운용소프트웨어에 적용 해 본다. 마지막으로 제 4장에서는 결론을 내리고 향후과제를 제시한다.

II. 제안된 운용소프트웨어의 구조

본 장에서는 PC 기반의 생산시스템을 위한 운용소프트웨어의 새로운 구조를 제시하기 위해서 PC 기반의 생산시스템을 정의하고 분석하여 이를 모듈별로 개발 및 유지보수를 할 수 있는 운용소프트웨어의 구조와 개발방법에 대해서 논한다.

1. PC 기반의 생산시스템 구조

일반적인 PC 기반의 생산시스템은 그림 1에서와 같이 전체시스템을 관리하는 운용 PC와 작업기기, 센서의 처리를 위한 PC가 표준 네트워크인 TCP/IP로 연결되어 있다. 각 PC에서는 작업기기와 센서의 정보 및 명령전달을 위하여 안정화된 인터페이스 카드를 사용한다. 따라서 PC에서 사용하는 일반적인 방법으로 쉽게 시스템을 구성할 수 있으며, 또한 확장성이 높은 시스템 구현이 가능하다. 하지만 PC 기반의 시스템은 실시간 OS(Operating System)를 사용하지 않아 완벽하게 실시간성을 보장할 수 없으며, 또한 TCP/IP를 통한 데이터 전송도 실시간성을 완전히 보장하지 못한다. 하지만 앞으로는 실시간 OS를 기반으로 한 PC가 나오고 네트워크의 속도향상이 기대되므로 향후에는 실시간성이 보장되는 PC 기반의 생산시스템도 가능할 것이다.

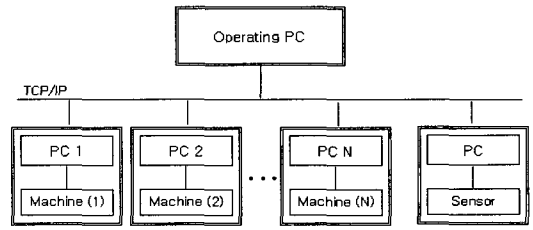


그림 1. 일반적인 PC기반의 생산시스템.

Fig 1. The general PC-based manufacturing systems.

2. 운용소프트웨어의 구조

본 논문에서는 일반적인 PC 기반의 생산시스템의 구성을 물리적인 관점에서 분석 및 설계하여 이를 새로운 운용소프트웨어 구조로 개발하는 과정을 논한다.

이를 위하여 먼저 새로운 운용소프트웨어에 필요한 요구사항을 정의한다.

1) 운용소프트웨어는 각 기능을 모듈로 구성하여 개발 및 향후 유지보수를 쉽게 할 수 있어야 한다.

2) 새로운 작업기기의 추가나 변동과 같은 물리적인 변화에도 운용소프트웨어가 유연하게 대처할 수 있어야 한다.

3) 작업일정의 변경이나 통신의 설정이 간단해야 한다.

4) 작업을 분산시켜 전체 시스템의 자원을 효율적으로 사용하여야 한다.

본 논문에서는 위의 요구사항을 만족하기 위하여 운용소프트웨어를 모듈화하고 변화에 유연하며 효율적인 구조가 되도록 다음과 같이 나눈다.

다음의 표 1은 운용소프트웨어의 구성을 보인다.

표 1. 운용소프트웨어의 구성.

Table 1. The configuration of operating software.

Module		Mission	Component
Decision-making		Schedule Control	Noting
Execution	Machine	Operating Machine Status Information	Symbol Gateway Control
	Sensor	Sensor Information Transfer	Symbol Gateway

운용소프트웨어는 각 작업기기를 제어하는 실행모듈과 이 실행모듈을 전체적으로 관리하는 의사결정모듈로 구성된다. 실행모듈은 생산시스템의 작업기기나 센서들을 네트워크를 통해서 원격으로 제어 및 감시하는 부분이며, 실행모듈을 작업기기와 센서로 나눈 이유는 센서는 컨트롤 기능은 없고 단지 정보만을 전달하므로 작업기기와 다른 구조를 갖기 때문이다. 의사결정모듈은 작

업일정에 맞추어 실행모듈에게 작업지시를 내리는 부분이다. 위의 운용소프트웨어의 모듈은 실제 PC상의 물리적인 환경에서 PC 단위로 나뉜다. 이는 각 작업기기의 추가 및 제거가 다른 작업기기에 영향을 주지 않기 위함이 있다. 또한 센서만을 전문적으로 처리하는 PC를 두었으며 이는 제어가 필요하지 않아 컨트롤 컴포넌트가 필요하지 않은 구조로 구성된다.

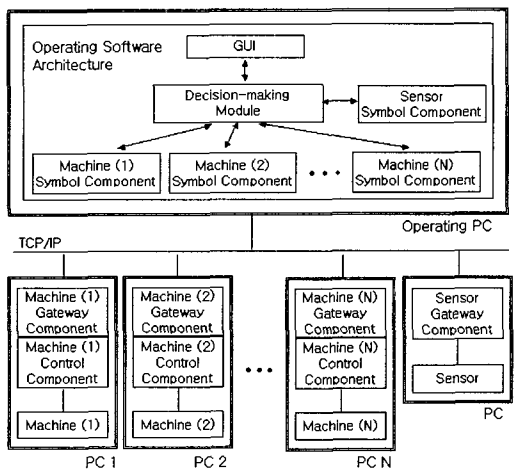


그림 2. 운용소프트웨어의 구조.

Fig 2. The structure of operating software architecture.

그림 2는 운용소프트웨어의 물리적인 환경에서의 모듈의 위치와 정보전달체계를 보인다. 전체적인 작업일정을 관리하는 운용 PC에 속해 있는 모듈의 역할은 주로 작업일정과 현재상태표시 및 작업기기와 센서들과의 통신이며, 각 작업기기와 센서들을 관리하는 PC에서는 운용 PC와의 통신부분 및 명령을 작업기기에게 전달하고 현재 상태정보를 얻는 역할을 한다.

3. 모듈의 구조와 역할

본 장에서는 운용소프트웨어의 의사결정모듈과 실행모듈의 구조와 역할에 대해서 논한다. 실행모듈은 센서에 컨트롤하는 기능이 없기 때문에 작업기기와 센서 실행모듈로 나누어 설계한다.

3.1 작업기기 실행모듈

작업기기 실행모듈은 작업기기의 실행에 관련된 사용자 인터페이스와 네트워크 통신 및 작업기기 제어의 역할을 하는 핵심 부분이면서, 한편으로는 운용소프트웨어의 구조를 복잡하게 하는 부분이기도 한다. 따라서 본 논문에서는 작업기기 실행모듈의 주요 기능을 컴포넌트로 설계한다. 작업기기 실행모듈을 컴포넌트로 설계하면 다음과 같은 장점이 있다. 첫째, 다른 운용소프트웨어를 개발할 때 빠른 개발과 통합을 위한 노력을 줄일 수 있다. 둘째, 실행모듈을 컴포넌트로 개발하면 구조를 단순화하고 작업기기의 추가나 변경에도 유연한 구조를 유지 할 수 있다. 이러한 이유로 본 논문에서는 작업기기 실행모듈을 컴포넌트로 개발하며 또한, 작업기기 실행모듈을 세 부분의 컴포넌트로 구성하여 효율적인 구조를 제시한다. 실행모듈을 세 부분의 컴포넌트로 나눈 이유는 다음과 같다. 첫째, 실행모듈의 주요 세 가지 기능인 사용자 인터페이스와 네트워크 설정 및 작업기기 제어가 서로 독립적인 기능이므로 이를 컴포넌트로 구성하면 독립적인 개발 및 유지보수를 동적으로 수행할 수 있다. 둘째, 각 컴포넌트간의 통신을 COM에서 제공하는 위치투명성과 MTS(Microsoft Transaction Server)를 사용할 수 있어 네트워크의 세부사항을 모르고도 네트워크 통신이 가능하다. 셋째, 각 컴포넌트들은 자신의 고유 스레드로 작업을 수행하므로 자원의 효율적인 분산처리가 가능하다.

작업기기 실행모듈은 그림 3에서와 같이 전체적으로 네트워크에 연결된 클라이언트-서버의 구조이다. 서버의 위치에는 게이트웨이 컴포넌트와 컨트롤 컴포넌트가 있다. 게이트웨이 컴포넌트는 MTS의 내부로 들어가 심볼 컴포넌트에게 서비스를 제공하며 MTS가 대신 네트워크의 세부사항을 처리하며 서버의 역할을 한다. 컨트롤 컴포넌트는 게이트웨이 컴포넌트에게 참조되며 이 때문에 두 컴포넌트 사이에 정보가 교환될 수 있다. 심볼 컴포넌트는 네트워크로 연결된 클라이언트의 위치에 해당되며 MTS 서버와 정보를 교환한다.

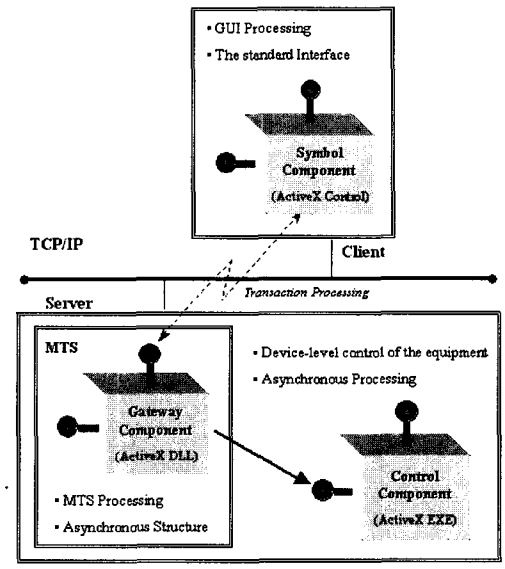


그림 3. 작업기기 실행모듈의 구조.

Fig. 3. The structure of machine execution module.

다음은 실행모듈을 구성하는 컴포넌트들의 정의와 역할을 설명한다.

3.1.1 심볼 컴포넌트

심볼 컴포넌트의 역할은 작업기기의 현재상태를 2차원의 그래픽 요소로 표현하는 것과 의사결정모듈에 표준 인터페이스를 제공하는 것이다. 심볼 컴포넌트에 제공하는 표준 인터페이스는 다음과 같다. 이 표준인터페이스는 다른 인터페이스와는 달리 최종 사용자에게 제공하여 생산시스템의 고유한 생산일정을 작성하는데 이용되는 것으로 본 논문에서는 일반적이며 간결한 세 종

류의 인터페이스로 시스템을 운용할 수 있도록 설계한다.

• Operation 인터페이스

이 인터페이스의 역할은 작업기기에 초기화, 일시정지, 종료 및 고유 작업지시를 전달하는 것이다. 다음은 비주얼베이직 언어로 구현된 Operation 인터페이스를 보인다.

```
Public Sub Operation(ByVal JobN)
  ......
  Call Screen(JobN)          ' 2차원 화면구성안 호출
  Machine1.SelectJob (JobN) ' 작업기기에 명령전달
  ......
End Sub
```

• Complete 인터페이스

이 인터페이스의 역할은 작업기기가 작업수행여부를 알려주는 것으로 이를 이용하여 새로운 작업지시를 내릴지 여부를 판별한다.

```
Public Property Get Complete() As Boolean
Complete = Machine1.EventFlag      '작업종료 체크
End Property
```

• Status 인터페이스

이 인터페이스의 역할은 작업기기가 최종적으로 수행한 작업식별자의 값을 얻는 것이며, 얻어낸 작업식별자를 이용하여 의사결정모듈에서 다음 작업지시를 판별하는 요소로 사용하게 한다. 또한, 현재 작업상태를 알기 위해서 각 작업기기에 해당하는 센서의 정보를 알 수 있도록 한다.

```
Public Property Get Status() As Integer
Status = Machine1.lastJob   '작업기기의 작업식별자
End Property
```

3.1.2 게이트웨이 컴포넌트

게이트웨이 컴포넌트는 일종의 중개자로서, 심볼 컴포넌트로부터 작업지시를 받아서 컨트롤 컴포넌트에게 전달하는 역할과 현재 작업기기 정보를 컨트롤 컴포넌트로부터 심볼 컴포넌트에게 전달하는 두 가지 역할을 한다.

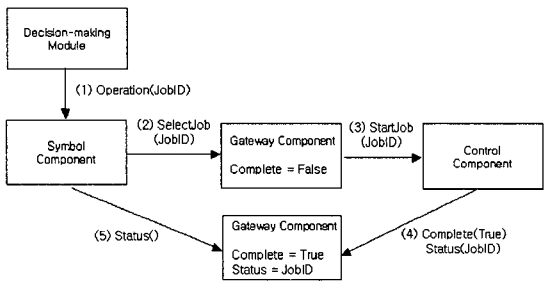


그림 4. 작업기기 실행모듈의 동작원리.

Fig. 4. The operating principle of machine execution module.

3.1.3 컨트롤 컴포넌트

컨트롤 컴포넌트는 게이트웨어 컴포넌트로부터 작업지시를 받으면 독립적인 실행 스레드를 생성하여 작업을 수행한 뒤 결과를 게이트웨어 컴포넌트에게 전달하는 역할을 한다.

그림 4는 위에서 설명한 작업기기 실행모듈의 동작원리를 보인다.

다음은 그림 4의 작업기기 실행모듈의 동작원리를 순서대로 설명한 것이다.

1) 의사결정모듈에서 심볼 컴포넌트에 작업지시를 한다.
   - 해당 인터페이스 : Operation(JobID)
2) 심볼 컴포넌트에서 게이트웨이 컴포넌트에게 작업식별자를 전달한다.
   - 해당 인터페이스 : SelectJob(JobID)
3) 게이트웨이 컴포넌트는 컨트롤 컴포넌트에 비동기식 처리 방법으로 작업을 지시한다.
   - 해당 인터페이스 : StartJob(JobID)
4) 작업이 끝나면 컨트롤 컴포넌트가 이벤트를 발생시켜 작업의 상태를 게이트웨이 컴포넌트에 전달한다.
   - 해당 인터페이스 : Complete()
5) 현재 작업의 상태를 알기 위해서 작업의 상태나 센서의 정보를 읽어 온다.
   - 해당 인터페이스 : Status()

제안된 컴포넌트 기반의 실행모듈 구조의 장점은 다음과 같다. 첫째는 실행모듈의 모듈화이다. 실행모듈은 세 부분의 독립적인 컴포넌트로 구성되므로 각 컴포넌트들의 수정을 별도로 할 수 있다. 예를 들면 작업기기의 컨트롤 컴포넌트를 수정하여도 네트워크 설정이나 사용자 인터페이스에는 그대로 사용할 수 있다. 둘째는 실행모듈의 표준화이다. 제안된 실행모듈은 PC에 연결된 어떠한 종류의 작업기기에도 적용이 가능하다. 특히 작업 중 작업기기의 고장으로 다른 회사의 제품을 사용해야 될 경우라면 컨트롤 컴포넌트의 수정만으로 작업을 재개할 수 있다. 셋째는 실행모듈의 재사용성 이다. 한번 개발된 작업기기 실행모듈은 바이너리 형태의 컴포넌트이다. 따라서 개발된 실행모듈을 수정 없이 다른 컴포넌트 기반의 운용소프트웨어에 추가할 수 있다. 넷째 실행모듈의 쉬운 사용법이다. 실행모듈은 Operation, Status, Complete 인터페이스만으로 작업기기의 제어 및 감시를 할 수 있다.

3.2 센서 실행모듈

센서 실행모듈의 역할은 작업기기의 현재상태를 전달하는 것이다. 이 모듈과 작업기기 실행모듈과 차이점은 첫째, 동기식 처리 방식이다. 이것은 센서의 정보를 얻는데 시간이 많이 걸리지 않기 때문이다. 둘째, 컨트롤 컴포넌트가 필요 없다. 이것은 제어부분이 없기 때문이

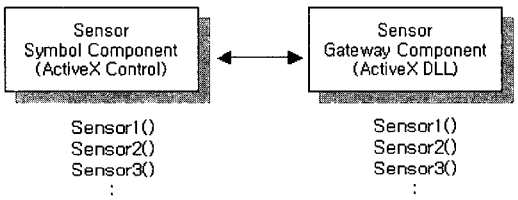


그림 5. 센서 실행모듈의 구조.

Fig. 5. The structure of sensor execution module.

다. 셋째는 표준 인터페이스가 없다는 점이다. 이는 센서의 공통된 특징을 찾기가 어려워 표준 인터페이스를 정의 하기가 힘들며, 향후 센서의 추가에 유연하기 위함이다. 따라서 센서 실행모듈의 인터페이스는 각 센서의 특성을 살린 명칭으로 정의한다.

그림 5는 센서 실행모듈이 심볼 컴포넌트와 게이트웨이 컴포넌트로 구성되어 있는 것을 보인다.

3.3 의사결정모듈

의사결정모듈은 GUI에서 작업지시를 받으면 작업일정에 따라 실행모듈에게 해당작업을 지시하며, 또한 현재 시스템의 작업상태를 파악하여 새로운 작업을 판별하는 역할을 한다. 실행모듈에게 작업지시를 내리기 위해서 본 논문에서 제안된 표준 인터페이스를 사용하며, 의사결정모듈에서 새로운 작업을 판별하기 위한 알고리즘은 다음과 같다. 먼저 작업기기가 수행 중인지를 체크하고 수행중이 아니면 관련센서와 작업기기의 상태 및 플래그의 정보를 이용하여 작업일정에 의거하여 새로운 작업을 결정한다. 작업이 결정되면 작업식별자와 표준 인터페이스를 이용하여 작업을 수행한다. 이 과정은 작업기기별로 순차적으로 수행하며 작업 목표량이 끝날 때까지 주기적으로 반복한다. 그림 6은 의사결정모듈의 알고리즘을 순서도로 표현한 것이다.

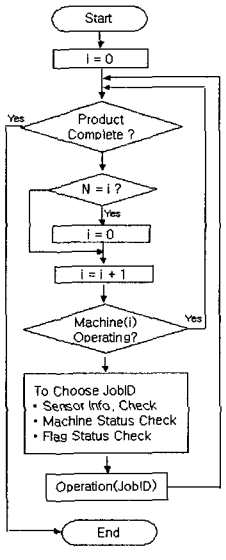


그림 6. 의사결정모듈의 순서도.

Fig 6. The flowchart of decision-making module.

다음은 의사결정모듈이 각 실행모듈이 제공하는 인터페이스를 이용하여 순서도를 바탕으로 구현된 것이다.

```
'일정간격으로 실행되는 함수
Private Sub Forever_Timer()
'작업기기 상태(작업의 유무) 체크
If Machine1.Complete = True Then
Call Work1    '작업이 없는 상태이면 해당작업 실행
End If
If Machine2.Complete = True Then
  Call Work2
End If
 ...
If MachineN.Complete = True Then
  Call WorkN
End If
End Sub
' 각 해당작업에 관한 함수
Private Sub Work1()
'센서와 플래그 상태체크
If Flag = True And Sensor.Sensor1 = True Then
  Machine1.Operation(JobID)  '해당작업시작
End if
End Sub
...
Private Sub WorkN()
...
End Sub
```

4. 모듈의 통합과정

본 논문에서 제안한 작업기기 실행모듈과 센서 실행모듈 및 의사결정모듈을 통합하기 위해서 표준 소프트웨어 개발도구를 사용한다. 표준개발도구로는 비주얼 C++, 비주얼 베이직, 델파이 등이 있다. 모듈들의 통합과정은 다음과 같다. 첫째, 각 실행모듈의 심볼 컴포넌트를 운영체계에 등록한다. 심볼 컴포넌트를 등록시키면 개발도구환경의 도구상자에 심볼 컴포넌트가 생기게 된다. 둘째, 이 심볼 컴포넌트를 개발도구의 폼에 올려놓는다. 셋째, 심볼 컴포넌트와 개발도구의 다른 컴포넌트를 이용하여 운용 소프트웨어의 GUI를 설계한다. 넷째, 심볼 컴포넌트의 인터페이스를 이용하여 의사결정모듈을 작성한다. 그림 7은 각 모듈을 통합시켜 운용소프트웨어를 개발하는 순서를 보인다.

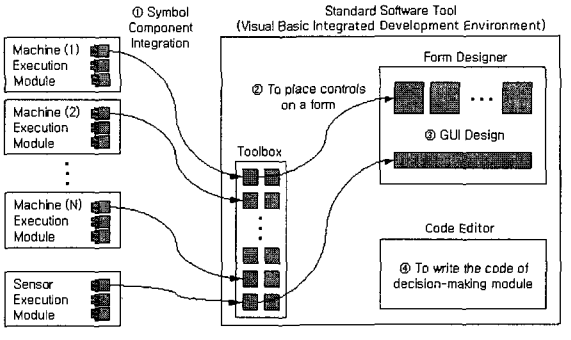


그림 7. 운용소프트웨어 개발순서.

Fig. 7. The development procedure of operating software.

III. 적용사례

본 장에서는 제안한 운용소프트웨어의 구조와 개발방법론을 시험해보기 위하여 한국생산기술연구원의 작업셀시스템을 대상으로 운용소프트웨어를 작성한다.

1. 작업셀시스템의 구성

작업셀시스템은 PC 기반의 생산시스템 개념으로 구성된 기초 모델이다. 작업셀의 역할은 Peg과 Hole을 가공

하고 가공된 부품을 이동하여 조립하는 것이다. 작업기기의 종류는 가공작업을 위한 CNC(Computer Numerical Control)와 Peg-in-Hole 조립작업을 위한 듀얼암 로봇(Dual-armed Robot)과 갠트리(Gantry) 그리고 부품이동을 위한 핸들링 로봇(Handling Robot)이 있다. 그리고 작업기기의 센서와 컨베이어(Conveyor), 스택커(Stacker)의 센서정보를 실시간으로 처리하기 위해서 CAN (Controller Area Network)을 이용한다. 그림 8은 위에서 설명한 작업기기들이 네트워크 연결된 PC 기반의 작업셀시스템의 구조도를 보인다.

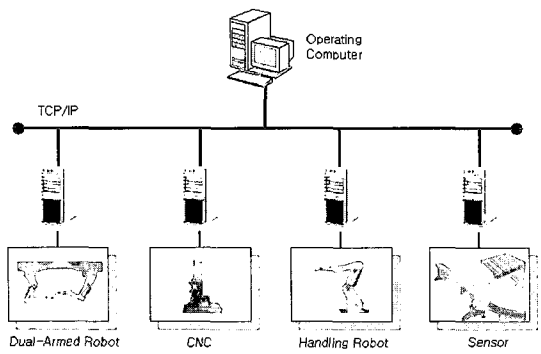


그림 8. 작업셀시스템의 구조도.

Fig 8. The diagram of workcell system.

2. 실행모듈 작성

2.1 작업기기 실행모듈 작성

• 핸들링로봇 실행모듈

핸들링로봇의 실행모듈은 작업셀시스템의 가공 및 조립작업을 위해서 Peg과 Hole이 필요한 작업기기들에게 전달하는 것이 주요역할이다. 그림 9는 핸들링로봇의 실행모듈을 구성하는 UML(Unified Modeling Language)을 이용한 컴포넌트 구조도를 보인다. 본 논문에서 사용한 UML은 소프트웨어의 설계된 내용을 표현하기 위해서 OMG(Object Management Group)에서 정한 국제표준 비주얼 모델링 언어이다[1]. UML은 소프트웨어의 청사진을 작성하기 위한 표준 언어이며 전체적인 시스템 구성에 대한 시각화, 규정화, 구성화 및 문서화로 표현하는데 사용된다. UML을 이용하면 사용자와 개발자간이나 개발자간의 공식적인 의견교환을 쉽게 할 수 있다. UML의 구성은 비주얼 모델들과 모델을 이용한 다양한 관점의 시각적인 구조도로 이루어져 있다. 본 논문에서는 UML의 여러 모델 중 컴포넌트의 대한 내용만을 사용한다. 그림 9의 컴포넌트 구조도는 컴포넌트와 인터페이스 및 컴포넌트를 구성하는 클래스 구조도로 구성된다. 컴포넌트는 직사각형모양에 탭(tab)이 붙은 모양이다. 컴포넌트의 위에 있는 글자는 컴포넌트의 이름을 가리키며 아래에 있는 글자는 컴포넌트를 구성하는 클래스를 나타낸다. 컴포넌트 밑에 있는 직사각형은 컴포넌트의 인터페이스를 정의한 것이다. 또한 인터페이스 밑에 있는 것은 컴포넌트 안에 있는 클래스를 정의한 것이며 굵은 테두리의 액티브(active) 클래스는 비동기식 처리방식으로 발생된 신호(signal)들을 처리하기 위한 메소드를 표현할 수 있는 UML에서 정의한 클래스의 한 종류이다. 그리고 컴포넌트간의 점선은 참조(reference)를 나타내는 것이며, 컴포넌트와 인터페이스의 점선은 연관(association)을 나타낸 것이다.

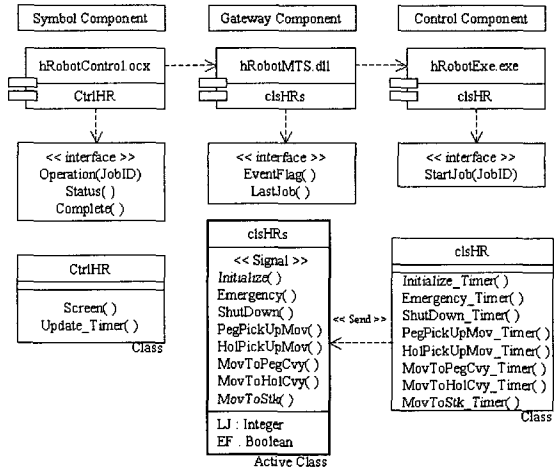


그림 9. 핸들링로봇의 실행모듈.

Fig 9. The execution module of handling robot.

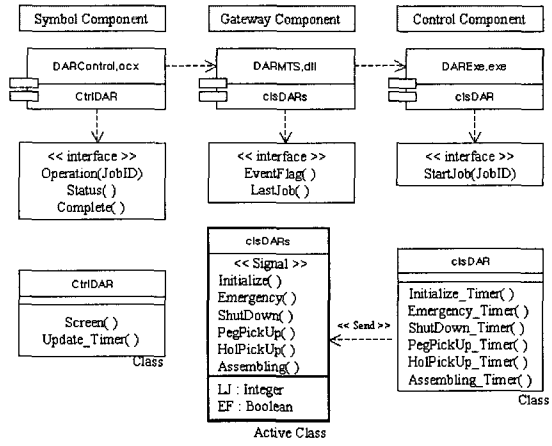


그림 10. 듀얼암 로봇의 실행모듈.

Fig. 10. The execution module of dual-armed robot.

그림 9에서 심볼 컴포넌트의 클래스는 표준 인터페이스의 Operation, Status, Complete의 구현 메서드와 화면처리를 위한 메서드로 구성된다. 게이트웨이 컴포넌트의 클래스는 비동기식 처리를 한 후에 발생하는 이벤트 처리 메서드와 컨트롤 컴포넌트의 상태정보를 심볼 컴포넌트로 전달하기 위한 인터페이스 구현 메서드로 구성된다. 컨트롤 컴포넌트의 클래스는 작업을 수행하기 위한 메서드와 작업이 끝난 후 이벤트를 발생시키는 메서드로 구성된다.

• 듀얼암로봇 실행모듈

듀얼암로봇 실행모듈의 주요역할은 다음과 같다. 첫째는 Peg과 Hole 장착이고, 둘째는 Peg-in-Hole 조립작업한 메서드와 작업이 끝난 후 이벤트를 발생시키는 메서

드로 구성된다.

• 듀얼암로봇 실행모듈

듀얼암로봇 실행모듈의 주요역할은 다음과 같다. 첫째는 Peg과 Hole 장착이고, 둘째는 Peg-in-Hole 조립작업이며, 마지막으로 조립완성품을 완성품적재함에 적재하

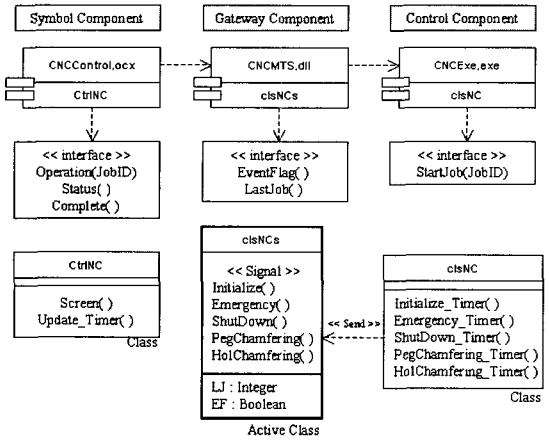


그림 11. CNC의 실행모듈.

Fig. 11. The execution module of CNC.

는 것이다. 그림 10은 듀얼암로봇 실행모듈의 작업과 인터페이스를 정의한 컴포넌트 구조도를 보인다.

• CNC 실행모듈

CNC의 실행모듈은 작업 부품인 Peg과 Hole을 가공하는 것이 주요역할이다. 그림 11은 CNC 실행모듈의 컴포넌트 구조도를 보인다.

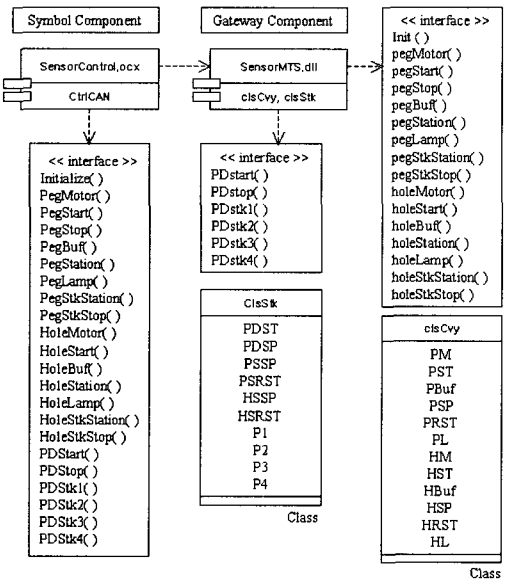


그림 12. 센서의 실행모듈.

Fig. 12. The execution module of sensor.

2.2 센서 실행모듈

센서 실행모듈은 센서들의 정보를 운용소프트웨어에 제공하는 역할을 한다. 센서들은 크게 부품의 이동을 위한 컨베이어의 센서와 부품의 공급과 적재를 위한 스택커의 센서로 나뉜다. 그림 12는 센서 실행모듈의 컴포넌트 구조도를 보인다. 센서 실행모듈의 인터페이스는 본 논문에서 제안한 대로 각 센서의 특성을 살린 고유한 인터페이스로 정의하였다.

3. 작업 시나리오 및 의사결정모듈

다음은 작업셀시스템의 데모를 위하여 설정된 작업일정을 시간의 순서대로 작성한 것이다.

① 사용자가 작업기기들과 센서들을 초기화 한다.

② 초기화 완료 후 사용자는 작업시작 명령을 내린다.

③ 핸들링로봇이 Peg을 집어 CNC에 장착한다.

④ CNC는 Peg 가공작업을 한다.

⑤ 핸들링로봇이 가공완성품을 Peg컨베이어에 적재한다.

⑥ Peg컨베이어는 Peg을 듀얼암로봇의 공급위치까지 이송한다.

⑦ 핸들링로봇이 Hole을 집어 CNC에 장착한다.

⑧ CNC는 Hole 가공작업을 한다.

⑨ 핸들링로봇은 가공완성품을 Hole컨베이어에 적재한다.

⑩ Peg컨베이어는 Hole을 듀얼암로봇의 공급위치까지 이송한다.

⑪ 듀얼암로봇이 양팔에 각각 Peg과 Hole을 장착한다.

⑫ 듀얼암로봇이 Peg-in-Hole 조립작업을 수행한다.

⑬ 듀얼암로봇은 완성품을 완성품컨베이어에 놓는다.

⑭ 완성품컨베이어는 조립완성품을 이송한다.

⑮ 핸들링로봇은 완성품을 완성품스택커에 적재한다.

적용사례의 작업일정은 기본적인 순서는 있지만 작업은 비동기식 처리방식을 이용하므로 순서에 상관없이 실행 가능한 작업을 먼저 수행할 수 있으며, 또한 동시에 여러 작업이 가능하다. 본 논문에서는 작업일정을 분리하여 의사결정모듈 메서드를 Handling()과 Chamfering() 및 PegInHole()로 나누었다. 그림 13은 의사결정모

```
Private Sub DecisionMakingModule()
  If DA.Complete = True Then
    Call PegInHole  'Dual-armed Robot dosen't working
  End If
End Sub

Private Sub PegInHole()
  If HR.Status = 6 And Sensor.PegStation = True Then
    DA.Operation (11)  'To pick up a peg
    PegPick = True
  ElseIf HR.Status = 7 And Sensor.HoleStation = True Then
    DA.Operation (12)  'To pick up a hole
    HolePick = True
  ElseIf PegPick = True And HolePick = True Then
    DA.Operation (13)  'Peg-in-hole assembly work
  End If
End Sub
```

그림 13. 의사결정모듈의 예.

Fig. 13. The example of decision-making module.

듈 중에서 듀얼암로봇의 조립작업과 연결된 PegIn Hole 의사결정모듈의 예를 보인다.

위 그림에서 DecisionMakingModule() 메서드는 듀얼암로봇의 작업 유무를 주기적으로 확인하며, 작업이 없을 경우에는 조립작업에 해당하는 PegInHole() 메서드를 호출한다. 호출된 메서드에서는 센서 정보와 작업일정에 의해 결정된 작업지시를 듀얼암로봇에 전달한다.

4. 운용소프트웨어 작성

지금까지 개발한 작업기기 모듈들과 센서 실행모듈 및 의사결정모듈을 가지고 표준개발도구를 사용하여 운용소프트웨어를 작성하였다. 그림 14는 작성된 운용소프트웨어가 가상의 작업을 수행하고 있는 것을 보인다.

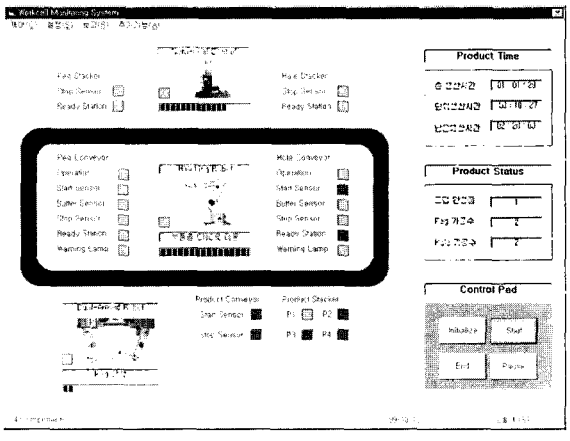


그림 14. 운용소프트웨어의 GUI.

Fig. 14. The GUI of operating software.

본 논문에서 개발된 운용소프트웨어는 실행모듈이 실제 작업기기와 연결되지 않은 상태에서 개발된 오프라인(off-line) 운용소프트웨어이며, 이를 위해서 각 작업들은 가상으로 일정한 시간동안 동작하도록 설정하였다. 운용소프트웨어의 작동요령은 다음과 같다. 첫째, GUI의 아래쪽 초기화 버튼을 이용하여 작업기기들과 센서를 초기화한다. 둘째, 초기화가 완료되면 작업시작 버튼을 누른다. 셋째, 작업이 시작된 후 긴급상황이 발생하면 일시정지 버튼을 누른다. 넷째, 모든 작업이 끝나면 종료버튼을 누른다.

작업이 시작되면 각 작업기기들은 심볼 컴포넌트에서 제공하는 작업의 진행상황과 내용을 사용자가 볼 수 있도록 화면에 표시한다. 그리고 운용소프트웨어의 GUI이 오른쪽 부분에 작업셀시스템의 작업의 총 생산시간과 단위시간 및 남은 예상시간과 생산되는 부품의 수량을 표시하여 작업의 종료시점을 예측할 수 있게 하였다.

V. 결론

향후의 생산자동화시스템은 다품종 소량 생산에 적합한 PC 기반의 생산자동화시스템으로 변하게 될 것으로 예측된다. 이는 PC 기반의 생산자동화시스템은 다양한 S/W와 H/W를 사용할 수 있으며 네트워크를 통한 원격제어 등 많은 장점을 가지고 있기 때문이다. 하지만 이러한 장점에도 불구하고 PC 기반의 생산자동화시스템은 시스템의 안정성 문제나 기존의 운용 소프트웨어의 복잡성으로 인한 개발 및 유지보수에 많은 비용이 드는 문제가 있다. 안전성의 문제는 향후 PC의 발전속도를 볼 때 충분히 해결될 것이라 예측된다. 운용소프트웨어의 문제는 기존의 운용소프트웨어의 구조와 개발 방법론과 일반적인 소프트웨어와는 달리 물리적인 구조의 복잡성 때문에 생기는 문제라 생각한다.

본 논문에서는 이러한 문제점 중 운용소프트웨어 복잡성을 해결하고자 새로운 구조를 제안하였다. 제안된 구조는 운용소프트웨어를 PC 기반의 생산시스템의 물리적인 구조와 운용소프트웨어의 확장성 및 유연성을 위해 각 기능과 통신 및 유지 보수의 최소단위가 될 수 있도록 모듈로 나누어 설계한 구조로 구성된다. 구체적으로 모듈은 실행모듈과 의사결정모듈로 구성되며 모듈의 기능은 시스템의 작업기기의 추가와 같은 물리적 변화와 네트워크를 통한 데이터 전송 및 설정, 그리고 복잡한 GUI환경 개발을 쉽게 한다.

따라서 본 논문에서 제안한 모듈 구조를 사용하면 다음과 같은 장점이 있다.

첫째, 각 모듈별로 독립적인 개발 및 수정이 가능하여 각 작업기기만의 고유작업을 여러 사람이 동시에 독립적으로 개발할 수 있다.

둘째, 한번 개발된 실행모듈을 다른 운용소프트웨어에 재사용 할 수 있고 쉽게 작업일정 프로그램을 작성할 수 있다.

셋째, 운용소프트웨어에 새로운 작업기기를 추가하는 경우에 작업기기의 실행모듈만 개발하면, 다른 작업기기와는 별도로 설치를 할 수 있으므로 이미 개발된 운용소프트웨어의 변경을 최소화하여 빠른 시간 내에 생산시스템을 재구성 할 수 있다.

넷째, 각 모듈들이 각 기능을 분담하므로 많은 자원을 필요로 하는 운용소프트웨어의 작업을 분산처리 할 수 있다.

다섯째, 일반적인 표준개발도구를 사용하여 개발되어 누구나 쉽게 접근하여 각 모듈을 개발할 수 있고 또한 각기 개발된 모듈의 통합도 용이하다.

따라서 본 논문의 의의는 미래의 생산자동화시스템인 PC 기반의 생산시스템의 운용소프트웨어 구조를 제시하였고, 또한 기존의 복잡한 운용소프트웨어 개발의 문제점을 분석하고 이에 대한 해결안으로 생산시스템의 실제 구축된 환경을 바탕으로 운용소프트웨어 구조를 설계하여 각 기능의 모듈성, 유지보수를 위한 유연성 및 자원의 분배와 통신설정의 용이성에 적합한 효율적인 시스템의 구조를 제시하였다는 점이다.

또한, 본 논문에서는 제안된 운용소프트웨어 구조의 설계와 통합을 검증하기 위하여 적용사례로 작업셀시스템을 대상으로 운용소프트웨어를 작성했다. 이를 위해 먼저 각 작업기기의 실행모듈과 센서 실행모듈을 및 작업일정에 의해 동작하는 의사결정모듈을 작성하였다. 그리고 이러한 여러 모듈을 통합하여 하나의 오프라인으

로 동작하는 운용소프트웨어를 작성하였다.

향후 계획은 본 논문에서 검증된 운용소프트웨어의 구조로 작성된 오프라인 운용소프트웨어를 사용하여 실제 작업기기들과 연결하여 작업셀시스템을 운용해 보는 것이고 좀 더 나아가 실시간 시스템에 대한 연구를 하여 PC 기반의 실시간 생산자동화시스템과 이를 운용소프트웨어에 대해서 논하는 것이다.

참고문헌

[1] G. Booch, J. Rumbaugh, and I. Jacobson, *The unified modeling language user guide,* Addison-Wesley, 1998.

[2] G.-T Kim, S. D. Cha, and D. H. Bae, "Task object model based approach for robot workcell application programming," *Computer Software and Applications Conf.*, pp. 109-114, 1997.

[3] J. H. Park, J. W. Kim, and W. H. Kwon, "Specification of a software architecture and protocols for automated VLSI manufacturing system operation," 제어·자동화·시스템공학 논문지, vol. 3, no. 1, pp. 94-100, 1997.

[4] J. M. Reagin, J. E. Beck, T. E. Sweeny, R. L. Anderson, and T. D. Garner, "A component-based software architecture for robotic workcell applications," *IEEE Trans. on Electronics Packaging Manufacturing*, vol. 22, no. 1, pp. 85-94, 1999.

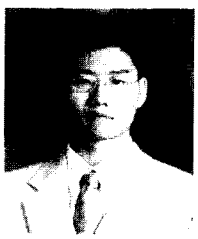
[5] J. S. Smith, "Design and implementation of FMS control software," *Proc. of the 1996 FAIM Conf.*, 1996.

[6] *COM Specification*, Microsoft Corporation, 1995.

[7] 고석범, 노대식, 민미경, 김일곤, "컴포넌트 객체 모델을 기반으로 한 서비스 아키텍쳐의 확장과 추론엔진의 컴포넌트화," 정보과학회 논문지(C), vol. 3, no. 4, pp. 419-427, 1997.

[8] 박재현, "개방형 생산 자동화 시스템 기술 동향," 전자공학회지, vol. 26, no. 3, pp. 57-61, 1999.

[9] 백준걸, 오훈언, 신현준, 김성식, 이홍철, "객체지향 제조관리 시스템 평가를 위한 시뮬레이터 개발," 한국시뮬레이션학회 논문지, vol. 8, no. 1, pp. 1-18, 1999.

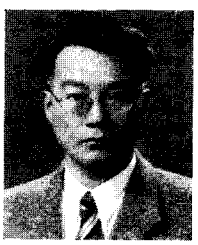


박 남 준

1998년 성균관대 제어계측공학과 졸업. 2000년 성균관대학교 석사과정 졸업. 2000년~현재 (주)아트닉스 연구원. 관심분야는 모델링. 공장자동화, 소프트웨어 공학, 디지털 제어시스템.

박 종 구

1987년 서울대학교 제어계측공학과 졸업. 동 대학원 석사(1989), 동 대학원 박사(1993). 1995 ~ 현재 성균관대학교 전기전자 및 컴퓨터공학부 부교수. 관심분야는 제어이론 및 응용, 컴퓨터 응용 제어 시스템, 추정이론, 가상현실 시스템 및 응용.



김 홍 석

1980년 서울대 전기과 졸업. 서울대 제어계측공학과 석사(1983), 1983 ~ 1987년 한국과학기술연구원 계측소자 연구실, 1990 ~ 1991, 한국과학기술연구원 광전기술센타, 동대학 박사(1990). 1990년~현재 한국생산기술연구원 수석연구원. 관심분야는 최적제어.