웹기반 원격제어시스템을 위한 이동통신문자서비스

○
이현수*, 최진석*, 박은영*, 박근효*, 김수정*,
김용대*, 박남섭*, 윤종준*, 이영란*, 김삼룡**,이정배*
*부산외국어대학교 컴퓨터공학과
**경남정보대학 컴퓨터정보과

A text delivery service of mobile communication for web based remote control system

○
Hyun-Soo Lee*, Jin-Suk Choi*, Eun-Young Park*,
Keun-Hyo Park*, Soo-Jung Kim*, Yong-Dae Kim*, Nam-Sup Park*,
Jong-Joon Yun*, Young-Ran Lee*, Sam-Ryong Kim**, Jeong-Bae Lee*
*Dept. of Computer Enginneering, Pusan University of Foreign Studies
**Kyungnam College of Information Technology

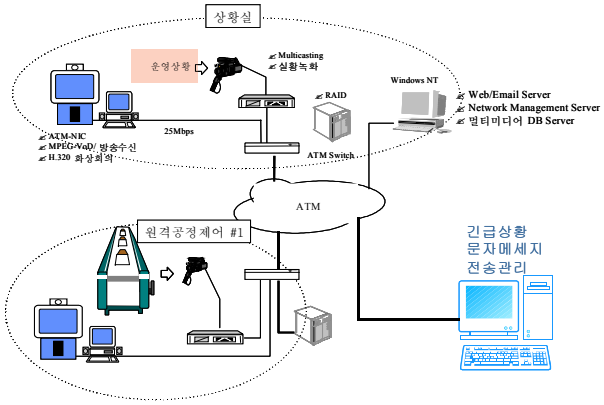
요 약

본 논문에서는 클라이언트/서버 형태로 긴급 이벤트 자동 호출 기능을 처리할 수 있는 원격제어 시스템을 설계하고 구현하였다. 웹 애플릿을 기반으로 구현된 클라이언트 시스템은 상황실에 위치하여 직접적으로 제어를 담당하는 서버에게 원격 제어 명령을 내린다. 제어 명령은 서버시스템으로 전달되며, 서버시스템에 존재하는 구동기가 실질적으로 컨베이어 시스템을 제어하는 기능을 가진다. 여기서 컨베이어 시스템은 승용차 조립라인을 시뮬레이션 한 것이다. 이러한 원격제어 시스템에 이동통신의 문자서비스를 이용하여 긴급한 상황을 자동으로 원격지에 있는 관리자에게 알려주는 기능까지 확대시켰다. 관리자는 원격 시스템을 항상 감시할 필요가 없이 어느 장소에서나 상황보고를 받을 수 있게 된다.

1.서 론

현재는 정보교환시대라고 일컬어도 과언이 아니다. 이에 이동통신과 웹이 지원해 주는 역할이 더 부각되고 있는 실정이다. 웹의 여러 기능중에서 우리가 쉽게 이용할 수 있는 서비스 중에서 이동통신 서비스 지원이 각 업체별로 활발히 진행되고 있고 이를 이용하여 웹상에서 간단한 문자메시지를 주고 받을 수 있게 되었다. 이러한 이동통신의 기능을 다른 분야에 접목시켜 봄으로써 응용시스템의 개발이 더 용이하게 될 것이다. 본 연구에서는 그림 1의 원격제어 시스템의 구성도에서 보는 바와 같이 Client/Server

환경에서 초고속정보통신망과 분산 멀티미디어를 이용하여 컨베이어 시스템을 원격으로 제어하는 시스템의 개발을 목표로 하고 있다. 더불어 공정 상황에서 발생하는 긴급 이벤트의 효과적인 처리 방안으로서 기존의 이동통신문자 서비스 인터페이스를 이용하여

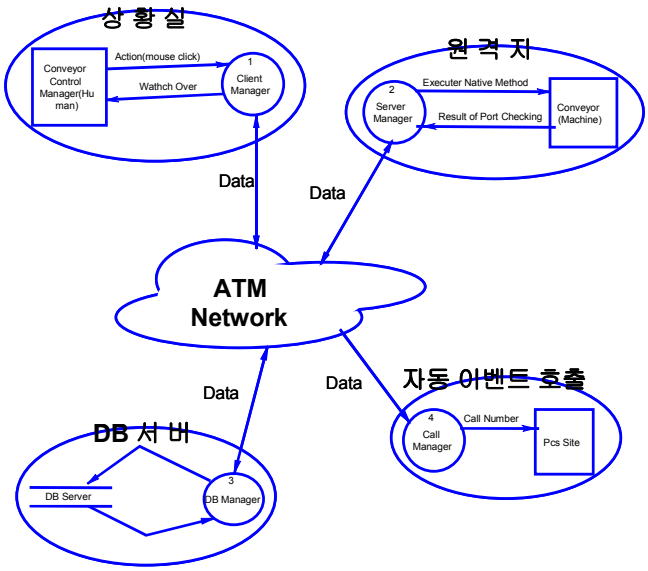


(그림 1) 원격제어시스템의 구성도

관리자에게 긴급 상황을 문자 전송함으로써 관리자가 좀 더 빠른 시간내에 이벤트를 처리할 수 있는 효과를 얻는 것을 목표로 하고 있다. 여기서 원격제어는 Inter/Intra-net의 웹을 통하여 가능하도록 한다.

2. 이동통신 문자 서비스 기반 원격제어 전체 구성

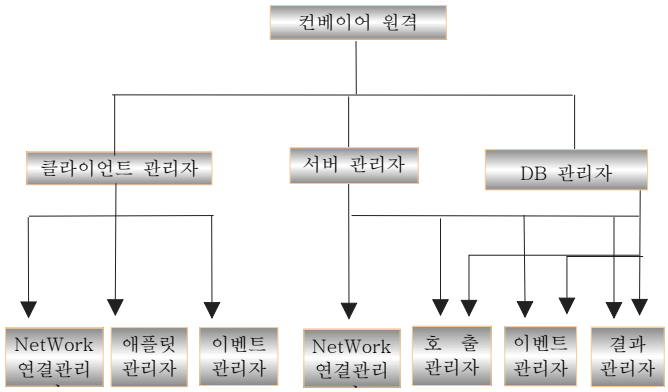
모델전체구성도는 그림 2 에서 보는 바와 같이 클라이언트 측 애플릿 프로그램과 서버 측 제어프로그램과 데이터베이스 시스템으로 구성된다. 본 논문에서는 클라이언트측과 제어프로그램이 내장된 서버측



(그림 2) 긴급상황 이동통신 문자 서비스 구성도

과의 원격제어 기능을 설명하고 긴급상황시에 관리자 휴대폰으로 문자를 전송하는 자동 이벤트 호출을 한다.

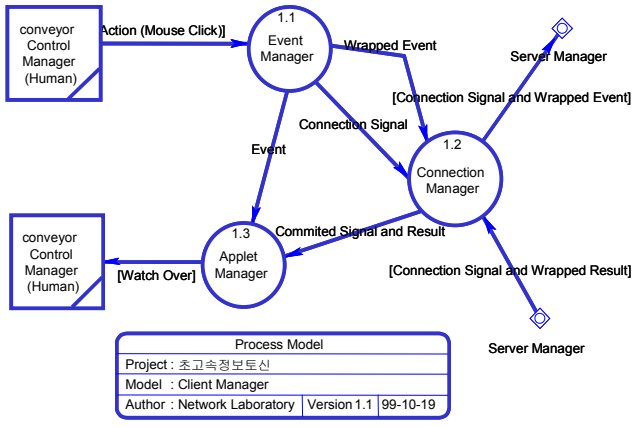
각 프로세스의 소개에 앞서 원격 제어시스템의 전체 프로세스 계층도를 살펴보면 그림 3 에서 보는 바와 같다. 전체 시스템은 클라이언트와 서버 그리고 DB 관리자로 분류된다. 컨베이어 구동기는 서버관리자 하부의 이벤트 관리자에 의해 구동이 된다.



(그림 3) 컨베이어 원격 제어 시스템 프로세스

3. 클라이언트 시스템

클라이언트 시스템은 그림 4에서 보는 바와 같이 원격지 상황실에 존재하며 크게 네트워크 연결관리자, 애플릿 관리자, 이벤트 관리자의 3개의 프로세스로 구성되어 있다.



(그림 4) 클라이언트 관리자 구성도

상황실 관리자에 의해 이벤트가 발생하면 네트워크 연결관리자를 통해 현지 공장에 있는 서버 프로세스와 연결하고 주어진 이벤트는 이벤트 관리자에서 포장이 되어져 다시 네트워크 연결관리자에 의해 서버 프로세스로 전송된다. 서버 프로세스에서 처리된 결과는 네트워크 연결관리자를 통해 애플릿 관리자로

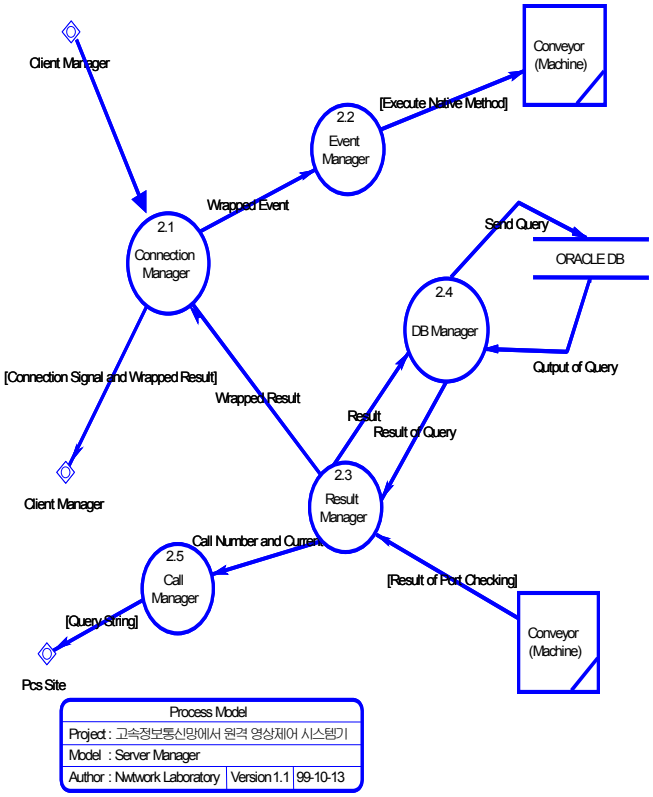
전송되어 상황실 관리자에서 상황을 보여준다.

4. 이동통신 문자전송 서비스 기반의 서버 시스템

서버 시스템은 원격지 현장에 위치하며 그림 5에서 보는 바와 같이 구성되며, 네트워크 연결관리자, DB 관리자, 이벤트관리자, 결과관리자, 호출관리자의 5개의 프로세스로 구성되어 있다.

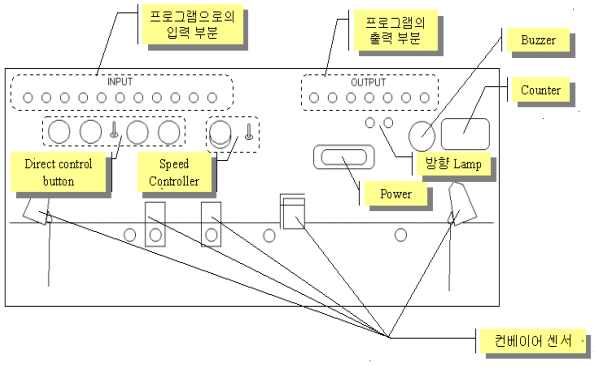
서버 시스템은 클라이언트 시스템으로부터의 이벤트를 분석하여 컨베이어 시스템에 적용시키기도 하며 DB 서버에 질의를 보내기도한다. 또, 컨베이어 시스템을 체크하여 이벤트 발생시에 DB 서버에 기록하기도 하며 다시 컨베이어 시스템과 클라이언트 시스템, 원격지에 있는 책임자에게 문자 메세지를 보내어 현재의 상황을 전송하는 일을 한다. 여기서 문자메시지의 전송은 기존의 이동통신 문자전송 인터페이스를 이용한다.

여기서 서버 시스템의 컨베이어 시스템 루틴을 좀더 자세히 살펴 보자.



(그림 5) 서버관리자 구성도

본 논문에서 사용되는 컨베이어 시스템은 자동자 조립공정을 모델링 한 것으로 실제 모양은 그림 6에서 보는 바와 같다.



(그림 6) 컨베이어 시스템의 구성도

- 프로그램으로의 입력부분 : 컨베이어의 출력으로서 프로그램에 입력되어 현재 컨베이어의 상태를 파악하게 해준다.
- 프로그램의 출력부분 : 프로그램에서 컨베이어 구동기에 의해 구동되는 부분이다.
- Direct control button : 중앙에 있는 auto/man 스위치를 조작하여 man으로 세트된 경우에는 컨베이어 시스템을 현장에서 시스템 관리자가 직접 제어할 수 있도록 해 주며, auto로 세트되는 경우에는 원격으로 컨베이어 시스템을 제어할 수 있는 환경을 제공해 준다.
- Speed Controller : man으로 세트된 경우, 현장에서 컨베이어의 속도를 직접 제어할 수 있도록 한다.
- Power : 컨베이어의 전원 스위치이다.
- 방향 Lamp : 정·역 방향 표시등이다.
- Buzzer : 컨베이어 자체에서 소리를 내는 부분이다.
- Counter : 생산된 승용차의 수를 카운터하는 부분이다.
- 컨베이어 센서 : 각각의 센서는 승용차 생산라인의 한 공정을 의미한다.

이상의 여러 기능들은 컨베이어 시스템 자체뿐만 아니라 상황실에서 원격 제어 할 수 있는 기능들이다.

컨베이어 원격 제어용 구동기는 서버 시스템에 설치되어 있으며 실질적으로 컨베이어 시스템을 구동하는 함수들을 가지고 있다. 구동기는 Java의 네이티브 메서드로 구현되어 있다.

컨베이어 시스템의 제어는 그림 7에서 서술한 기능에서 보는 봐와 같이 클라이언트 시스템으로부터

발생되는 이벤트를 체크하는 메인 쓰레드와 그림 8에서 서술된 기능에서 보는 바와 같이 컨베이어 시스템의 상황을 주기적으로 체크하는 서브 쓰레드로 구성된다.

```
public static void main(String args[]) {
{       . . . . .
        while(true) {
        클라이언트시스템으로부터 시그널을 기다림
        시그널이 오면 이벤트 처리자를 생성
        }
}
```

(그림 7) 서버의 메인 쓰레드

서버 시스템의 메인 쓰레드는 예약된 소켓 포트로 클라이언트 시스템으로부터 접속을 기다리며 접속이 되면 이벤트 관리자를 생성하여 시그널(이벤트)를 처리하게 한다.

```
public void run()
{
        while(true) {
        이전의 컨베이어 시스템의 상황을 저장
        쓰레드를 0.05 초간 쉼
        현재의 컨베이어 시스템의 상황을 단계별 체트
            긴급상황이면
                메시지 관리자를 호출
        이전의 상황과 현재의 상황이 다를때 적절한 이벤트
                처리자를 실행
        }
}
```

(그림 8) 서버의 서브 쓰레드

서브 쓰레드는 주기적으로 이전의 컨베이어 시스템의 상황과 현재의 컨베이어 시스템의 상황을 비교하여 차이나는 부분에 해당하는 이벤트 관리자를 구동시켜 상활실 관리자 및 현재의 컨베이어 시스템에 적용시키는 일을 수행한다.

컨베이어 시스템은 컨베이어 서버 시스템에 장착된 마이크로 칩을 이용하여 3e8H(A Port), 3e9H(B Port), 3eaH(C Port), 3ebH(Init Port) 의 4개의 I/O address를 사용하여 컨베이어 시스템과 정보를 교환한다. 각각의 I/O address 별로 간단히 설명하면, 3e8H(A Port) address는 컨베이어 시스템의 현재 상황을 서버 시스템으로 전송하는데 사용된다. 3e9H(B Port) address 와 3eaH(C Port) address는 서버의 시그널과 컨베이어 시스템의 속도를 전송하는데 사용된다. 그리고, 3ebH(Init Port) address는 컨베이어 서버 시스템에 정착된 마이크로 칩을 초기화 하는데 사용된다. A,B,C 포트의 구성은 그림9에서 보는 바와 같다.

<A Port>

- 서버로 시그널 전송에 사용 (I/O Address 3E8H)

8번 bit	7번 bit	6번 bit	5번 bit	4번 bit	3번 bit	2번 bit	1번 bit
PROX1	LS2	LS1	REV	FOR	AUTO	STOP	START

<B Port>

- 컨베이어 시스템으로 시그널 전송에 사용 (I/O Address 3E9H)

not used	not used	COUNT	BUZ ZER	REV_L	FOR_L	REV	FOR

<C Port>

- 서버로 시그널 전송과 컨베이어 시스템 속도 설정에 사용 (I/O Address 3EAH)

50%속도	22.5% 속도	13.5% 속도	6% 속도	not used	not used	PHS	PROX2

(그림 9) A, B, C Port 의 구성도

먼저 컨베이어 시스템의 초기화 과정은 그림 10과 같다. 우선 Init Port에 91H 값을 전송하여 마이크로 칩을 초기화한다. 다음으로 관리자의 입력으로 받아들여진 컨베이어 시스템의 초기 속도값을 셋팅한다. 그리고 초기에 정해진 방향으로 컨베이어 시스템의 방향을 설정하고 서버로부터 시그널 전송이 이루어질 B Port를 초기화한다.

```
void Server__init(struct HServer *this, long Speed)
{
Init Port에 91H 값을 전송하여 마이크로 칩 초기화;
C Port에 입력된 Speed 값을 전송하여 컨베이어 시스템
   속도 초기화;
A Port에 움직이는 방향 설정;
B Port에 00H 값을 입력하여 초기화;
}
```

(그림 10) 컨베이어 시스템 초기화 루틴

컨베이어 시스템의 구동 그림 11은 관리자의 입력으로 받아들여진 컨베이어 시스템의 구동방향을 점검하여 B Port에 그 방향을 전송한다.

```
void Server__act_start(struct HServer *this, long direction)
{
현재 컨베이어 시스템에 설정된 방향이 정방향이면,
        B Port에 정방향 시그널을 전송하여 구동시킨다.
현재 컨베이어 시스템에 설정된 방향이 역방향이면,
        B Port에 역방향 시그널을 전송하여 구동시킨다.
}
```

(그림 11) 컨베이어 시스템의 초기 구동 루틴

이 컨베이어 시스템의 초기 구동 루틴과 아울러 진행 중 속도 증가나 감소시에도 방향의 전향이 있을 경우에는 그림 12와 같이 그림 11과 유사한 루틴을 사용한다. 예를 들면, 컨베이어 시스템의 정방향 진행 시 최소 속도에서 속도감소 시그널을 만나면 현재의 방향을 체크하여 그와는 반대 방향으로 속도를 증가시켜 준다.

```
void  Server__act_start_reverse(struct HServer *this)
{
B Port의 상태를 읽는다.
B Port 상태가 정방향이면
        B Port에 역방향 시그널을 전송한다.
B Port 상태가 역방향이면
        B Port에 정방향 시그널을 전송한다.
}
```

(그림 12) 컨베이어 시스템의 방향 전환 루틴

컨베이어 시스템의 정지 루틴 그림 13은 현재의 상태를 체크하여 구동시에만 B Port에 정지 시그널을 전송하여 컨베이어 시스템을 정지 시킨다.

```
void Server__act_stop(struct HServer *this)
{
        현재의 컨베이어 시스템의 상황을 체크한다.
        B Port에 정지 시그널을 전송한다.
}
```

(그림 13) 컨베이어 시스템 정지 루틴

그리고, 각포트의 체크 루틴은 그림 14에서 보는 바와 같이 상황을 읽어 그 값을 리턴한다.

또, 컨베이어 시스템의 속도 제어 루틴은 컨베이어 시스템을 체크하여 총 32 단계로 속도를 제어할 수 있다.

```
long Server__check각 포트(struct HServer *this)
{
        A, B, C Port의 상황을 체크한다.
        A, B, C Port의 현재 값을 리턴한다.
}
```

(그림 14) A, B, C Port의 체크 루틴

속도 증가 루틴 그림 15는 현재의 컨베이어 시스템의 방향이 정방향일때는 최고 속도가 아니라면 한 단계 높은 속도의 시그널을 보낸다. 역방향일때는 정방향 진행이 속도의 증가이므로 최저속도가 아니라면 한 단계 낮은 속도를 적용시킴으로써 속도의 향상을 시킨다. 그리고 역방향 최저속도일때는 방향을 바꾸고 한 단계 높은 속도를 적용시킨다.

```
void Server__SpeedUp(struct HServer *this)
{
현재의 컨베이어 상황을 체크한다.
컨베이어 시스템의 방향이 정방향 일때
        최고속도가 아니면
                한단계 높은 속도를 적용한다.
컨베이어 시스템의 방향이 역방향 일 때
        최저속도이면
                방향을 바꾸고 속도를 증가시킨다.
        최저속도가 아니면
                한단계 낮은 속도를 적용한다.
}
```

(그림 15) 컨베이어 시스템의 속도 증가 루틴

속도 감소 루틴은 그림 16에서 보는 바와 같다. 컨베이어 시스템의 방향이 정방향 일 때 최저속도가 아니면 한 단계 낮은 속도를 적용시킨다. 최저속도라면 방향을 바꾸고 속도를 증가시키며 역방향 일때, 최대속도가 아니면 한 단계 높은 속도를 적용시키고 최대속도일때는 그대로 둔다.

그 외에 컨베이어 시스템에 부착된 부저를 울리게 하는 루틴은 그림 17에서 보는 바와 같고, 카운트를 증가시키는 루틴은 그림 18에서 보는 바와 같다. 부저를 울리게 하는 루틴은 일정시간동안 B Port에 부저 시그널을 보내어 부저를 울리게 한다. 카운트 제

```
void Server__SpeedDown(struct HServer *this)
void cgi__translate ( long  emergency_rate )
{
{   현재의 컨베이어 상황을 체크한다.
  관리자의 호출빈도에 따라
    컨베이어 시스템의 방향이 정방향 일때
    011 이면
        011에 맞는 cgi
        최저속도이면
    017 이면     방향을 바꾸고 속도를  증가시킨다.
        017에 맞는 cgi
        최저속도가 아니면
    016 이면
        016에 맞는 cgi 낮은 속도를 적용한다.
        한단계
    019 이면
    컨베이어 시스템의 방향이 역방향 일 때
        019에 맞는 cgi
        최고속도가 아니면
}
                한단계 높은 속도를 적용한다.
}
```

(그림 20) Query_String 생성루틴

(그림 16) 컨베이어 시스템 속도 감소 루틴

```
void Server__Buzzer(struct HServer *this)
{
현재의 컨베이어 시스템을 체크한다.
현재의 컨베이어 시스템의 상황에 부저 시그널을 OR
    시켜 적용한다.
잠시 딜레이를 준다.
현재의 컨베이어 시스템을 체크한다.
현재의 컨베이어 시스템의 상황에 부저 시그널을 빼고
    적용한다.
}
```

(그림 17) 컨베이어 시스템의 부저 제어루틴

```
void Server__Counter(struct HServer *this)
{
현재의 컨베이어 시스템을 체크한다.
현재의 컨베이어 시스템의 상황에 카운트 시그널을 OR
        시켜 적용한다.
잠시 딜레이를 준다.
현재의 컨베이어 시스템을 체크한다.
현재의 컨베이어 시스템의 상황에 카운트 시그널을 빼
        고 적용한다.
}
```

(그림 18) 컨베이어 시스템의 카운트 제어 루틴

어 루틴은 부저 제어 루틴과 마찬가지로 B Port에 카운트 시그널을 보내어 카운드를 증가 시켜 주는 역할을 한다.

A,B,C port의 체크 루틴에서 리턴된 값을 판단하여 원격지의 책임자에게 메시지를 보낸다.

이동통신 문자 서비스 인터페이스를 통하여 긴급상황 메시지를 전송은 그림 8 에서 긴급상황이 발생하였을 때 이루어진다. 절차는 그림 19에서 보는 바와 같다.

그림 20에서 보는 바와 같이 호출 관리자는 관리자의 인적상황을 통해 해당되는 이동통신 문자전송 서비스를 지원하는 사이트를 이용하기 때문에 그에 맞는 cgi 생성하여야 한다.

cgi에서 생성된 query_string은 그림 21에서 보는 바와 같다.

이상에서 살펴본 컨베이어 시스템의 제어 루틴은 최하위 루틴들로써 컨베이어 시스템을 물리적으로 제어한다. 이를 제어하는 루틴은 자바 애플릿 응용기술

```
void message__call ( long  emergency_rate )
{
  현재의 상황을 판단
      컨베이어 시스템 정지 일 때
      불량품의 과다 생산 일 때
      생산속도의 급격한 저하 일 때
        긴급상황에 맞는 메시지를 선택한다.
        호출 타이머를 3으로 셋한다.
      관리자에 의한 조건
        관리자에 의해 정해진 메시지를 선택한다.
        관리자에 의해 정해진 호출 타이머가 셋된다.
   while( emergency_flag ) {
      관리자에게 긴급상황을 알린다.
                        }
}
```

(그림 19) 컨베이어 시스템의 메시지 호출 루틴

을 이용함으로써 우수한 제어 사용자 인터페이스를 제공한다.

5. 결　　론

본 연구에서는 클라이언트/서버 형태로 Java 네이티브 메서드를 이용하여 컨베이어 시스템을 원격으로 제어하는 방법을 제시하고 구현하였다. 이것은 이동통신의 문자전송 시비스를 기반으로 한다. 이러한 시

(그림 21) 017cgi에서 생성된 query_string

스템의 구성은 근거리통신망으로 원격지에 연결된 컨베이어 시스템을 상황실에 위치한 클라이언트 시스템에서 웹 애플릿을 사용자 인터페이스로 사용하여 원격 제어하는 형태를 가진다. 또, 서버 시스템에서는 컨베이어 시스템을 통과하는 생산품 내역에 대한 데이터베이스를 저장, 유지, 관리한다. 또한 긴급 이벤트 발생시 효과적인 처리 방안으로서 기존의 이동통신을 이용하여 관리자에게 긴급상황을 문자전송함으로써 관리자가 효과적으로 이벤트를 처리하게 하였다. 본 연구에서 사용된 시스템은 승용차 조립라인을 모델링한 시스템으로 구성되어 있다. 이동통신 문자전송서비스 기반의 원격제어 모델은 승용차 조립라인의 컨베이어 시스템에 적용되었다.

이러한 이동통신 문자전송 서비스 기반의 원격제어가 성공적으로 이루어지므로써 초고속정보통신망에서 분산 멀티미디어 사용 기술의 확산은 더욱 더 활발해 질 것이다. 좀 더 나아가 원격지 공정과 상황실 시스템간의 편리한 대화를 위해서는 텍스트, 그래픽, 이미지, 사운드, 비디오 등 멀티 미디어에 의한 대화가 가능해야 한다. 이러한 시스템의 개발로 말미암아 다양한 미디어에 의한 인터페이스 개발의 가속화를 기대할 수 있게 된다. 하이퍼미디어에 바탕을 둔 사용자 인터페이스에 관한 연구는 객체 지향 기술, 시각 프로그래밍, 멀티미디어 기술 등 여러 분야에 파급되어 편리한 사용자 환경 구축을 통해 전문가가 아닌 초심자들도 쉽게 조작할 수 있을 것이다. 공정 제어 분야에서는 공장 운영에 상당한 비용 절감을 기대해 볼 수 있고, 생산성 향상을 꾀할 수 있을 것으로 기대된다. 이로써 초고속정보통신망 하에서 원격 감시 및 제어 시스템의 시장성은 대단하다고 볼 수 있고, 전체 산업 발전에 끼치는 영향은 크다고 할 것이다.

[참 고 문 헌]

[1] Craig M. Wittenbrick, Eric C. Rosen, Darrell D. E. Long,"Real-time System for Managing Environmental Data." Proceeding of Conference on Software Engineering and Knowledge Engineering, June 1996

[2] Theodore R. Haining, Darrell D.E. Long, Patric E. Mantey, Craig M. Wittenbrink, "The Real-Time Environmental Information Network and Analysis System (REINAS),"Proceeding of COMPCON, March 1995

[3] 이 정배, 김 인홍,“원격 영상 감시 및 제어 자동화,” 정보처리학회지, 1997. 7.