工學碩士 學位論文

LabVIEW를 사용한 AMS 및 고장진단 시스템 개발

Development of the AMS and Failure Diagnosis System by Using LabVIEW

指導敎授 趙 權 回

2006 年 2月

韓國海洋大學校 大學院

機關시스템工學科

張 台 璘

목 차

List of figures

List of tables

Development of the AMS and Failure Diagnosis System by Using LabVIEW

Tae - Lin JANG

Department of Marine Engineering, Graduate School,

Korea Maritime National University,

Busan, Korea

(Supervisor : Prof. Kwon - Hae CHO)

Abstract

An engine room in a ship is a space of highly intricate system consisted of not only the main engine but the auxiliary machinery. On the ground of this, the occurrence of a problem in a machine could generate problems of other machines in succession, on account of close connection between them.

For 4 academic years, to embark in a ship as a marine engineer, students learn and practice a lesson such as a basic and technical knowledge and theory for operating and managing a ship system. However, after graduation, when they are faced with practical ship operation and systems as a new engineer, they would tend not to find out the reason for trouble because of lack of knowledge about relationship between machines.

Therefore, it is necessary to have an education through practical use of

system which has systematic and composite technique for analysis. Using this, students could acquire how to easily cope with an emergency such as a breakdown of system or unusual working.

In this context, it is discussed the construction for alarm monitoring and failure diagnosis system on this thesis, and used software is G programming language, LabVIEW. But, only partial construction of alarm monitoring and diagnosis system was achieved because of deficient in given condition. And program was constructed to make up for it of available.

제 1 장 서 론

선박의 기관실은 주기관(main engine) 하나만으로 구성되어 있는 것이 아니라 주기관을 구동하기 위한 보기(auxiliary machinery)와 결합되어 있는 매우 복잡한 시스템이다. 각 기기들이 서로 밀접한 관계를 가지고 전체 시스템을 이루기 때문에 한 기기에서 발생한 문제가 다른 기기의 작동에 문제를 발생시키기도 한다. 따라서 기관사는 각 기기에 대한 지식뿐만 아니라 기기들 간의 관계를 잘 파악하여 고장의 원인을 기기 내부 및 외부에서 쉽게 찾아낼 수 있는 능력을 가져야 한다.

선박 기관사로 승선을 하기 위해 학생들은 학부과정 4년 동안 선박 시스템을 관리·운용하기 위하여 기관실을 구성하고 있는 각종 기기에 대한 기초지식부터 전문지식까지 다양한 이론을 배우고 실습하는 과정을 거치게 된다. 하지만 졸업 후에 초임 사관으로 승선하여 실제 선박 시스템을 접하게 되면 각 기기 사이의 상호관련성에 관한 지식이 부족하여 고장의 원인을 쉽게 찾아내지 못하는 경우가 종종 발생한다.

따라서 학생들이 시스템 고장 및 비정상적인 운전과 같은 위급한 상황에 쉽게 대처할 수 있는 능력을 배양하는 것이 요구되며, 이를 위해서는 선박 시스템에 대한 지능적인 진단기능을 가진 AMS(alarm monitoring system) 및 고장진단 시스템의 활용을 통한 교육이 필요하다.

고장진단 시스템(failure diagnosis system)이란 발생 가능성이 있는 이상상태, 계측 항목간의 인과관계 및 관찰된 시스템의 동작에 대한 정보로부터 이상의 원인을 찾아내는 시스템을 말한다. 고장진단 시스템은 시스템의 운용에 있어서 고도의 안전성과 고장에 대한 빠른 조치를 요구하는 곳인 원자력 또는 화력 발전소의 온라인 고장진단 시스템, 통신망 고장진단 시스템, 전력계통 고장진단 시스템 등에 주로 적용되고 있다.

이상의 배경으로부터 본 논문에서는 감시 및 고장진단 시스템 구축에 관하여 연구하고자 하며, 기본적인 감시 및 고장진단 시스템의 기본적인 틀만을 구현하고 향후 프로그램을 보완할 수 있도록 구성한다.

진단 모듈을 구축하기 위해서는 먼저 진단대상에 대한 지식베이스와 합당한 추론기구 구축이 필요하며, 전문가시스템 개발도구를 이용하는 경우에는 지식베이스만 있으면 된다. 본 논문에서는 전문가시스템 개발도구를 사용하지 않기 때문에 지식베이스와 추론기구를 모두 구축하도록 한다.

진단 모듈 구축을 위해서는 우선 대상 분야에 적합한 지식표현법을 선정하여 지식베이스를 구성해야 하는데, 이 경우 일반적으로 전문가와 협의를 통해서 진단지식을 추출한다. 다음으로 추출한 진단지식은 인과관계를 정의하여 컴퓨터가 인식할 수 있는 진단지식으로 변환시키고 최종적으로 지식베이스가 구축된다.

본 논문의 고장진단 시스템 구현에도 위와 같은 절차를 따르며 G 프로그래밍 언어인 LabVIEW를 사용하도록 한다. LabVIEW는 프로그래밍 언어를 처음 접하는 사람도 쉽게 배울 수 있어서, 단시간에 복잡한 시스템을 꾸밀 수 있는 장점 때문에 최근에 많이 사용하고 있는 프로그래밍 언어이다. 본 논문에서는 LabVIEW를 사용하여 아래와 같은 내용으로 시스템을 구현하도록 한다.

실시간으로 받아들인 센서의 값을 화면에 나타냄과 동시에 진단이 이루어지도록 하며, 이벤트 구동방식을 사용하여 프로그램의 CPU와 메모리의 사용량을 줄인다. 추후 수정과 확장을 위해 하부 VI를 사용하여 프로그램의 각부를 모듈화하고 고장진단을 위한 진단규칙을 사용자가 프로그램 상에서 직접 추가, 삭제, 수정할 수 있도록 구성한다.

또한 실시간으로 받아들인 데이터 중 아날로그 센서의 데이터, 모든 센서에서 발생한 경보 내역, 사용자가 구성한 진단규칙을 프로그램 실행 중 일정 시간마다 저장을 하도록 구성한다. 이를 위해서는 데이터베이스 툴킷(database

toolkit)을 사용하여 효율적으로 데이터베이스를 구축할 필요가 있으나 본 논문에서는 비용상의 문제로 LabVIEW 자체의 파일 입출력 기능을 사용하여 데이터베이스를 구축하도록 한다. 사용자가 요구할 경우에는 데이터베이스에 저장된 데이터를 프로그램 실행 중에 화면상에서 보고 수정할 수 있고 프로그램 종료 후에도 LabVIEW가 아닌 윈도우즈 상의 다른 프로그램을 통해서 볼 수 있도록 구성한다.

화면은 선박에서 흔히 볼 수 있는 AMS 화면과 비슷하게 구성한다. 한국해양대학교 신조 실습선인 한바다호의 파이프라인을 참고하여 화면을 구성하며, 중요하다고 생각되는 계통만을 구현하도록 한다. 각 계통에는 그 계통화면에 속하는 각종 센서를 표시하고 경보가 발생했을 경우 화면상에 표시되도록 구현한다.

본 논문의 구성은 다음과 같다. 2장에서는 시스템을 구현하기 위해 사용된 LabVIEW라는 프로그램과 시스템의 전반적인 구성, 3장에서는 시스템의 화면 구성과 블록 다이어그램 구성에 대하여 설명하고, 4장에서는 구현한 시스템의 실행결과를 통해 본 논문에서 개발한 진단시스템의 타당성을 보여준다. 마지막으로 5장에서는 결론 및 향후 과제를 제시한다.

제 2 장 LabVIEW를 사용한 시스템 개발

2.1 시스템 전체 구성

시스템 전체 구성은 아래 **Fig. 2.1**과 같다. 계측부로부터 들어오는 데이터를 프로그램에서 받은 후, 경보 발생 유무를 판단하고 발생한 경보에 대한 진단을 내린 후 결과를 화면상에 나타내도록 구성한다. 또한 데이터베이스를 구축하여 실시간으로 들어오는 아날로그 센서의 값을 저장하고 발생하거나 소거된 모든 센서의 경보 내역을 저장한다.

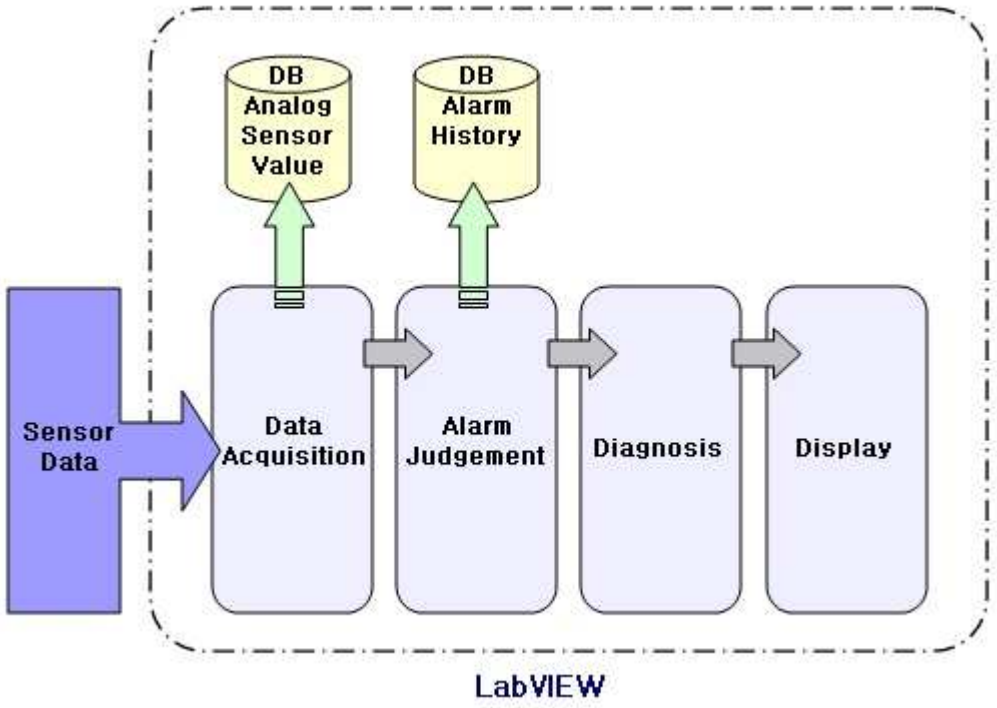


Fig. 2.1 Diagram of organization

본 논문에서는 시스템 구성을 위하여 LabVIEW라는 프로그래밍 툴을 사용한다.

2.2 LabVIEW의 특성

LabVIEW[i]는 NI(National Instruments)사에서 개발한 강력하고 유연한 계측 및 자동화 소프트웨어 개발 어플리케이션이다. LabVIEW로 만들어진 프로그램은 가상기계(virtual instrument), 즉 VI라고 불리며 vi라는 확장자를 가진 파일 형태로 저장된다.

LabVIEW는 텍스트 기반의 프로그래밍 언어인 포트란이나 C 언어와는 달리 G 프로그래밍 언어로 알려진 그래피컬한 프로그래밍 언어이다. 블록 다이어그램이라 불리는 흐름도와 유사한 가상기계 프로그램을 생성하고, 아이콘으로 만들어진 그래픽 기호를 사용하여 하나의 컴포넌트를 형성하게 된다. G 프로그램을 만들기 위해 LabVIEW에서 사용되는 그래피컬한 아이콘은 그 형태만으로도 기능을 파악할 수 있다.

Fig. 2.2, **Fig. 2.3**과 같이 LabVIEW는 블록 다이어그램(block diagram)과 프런트 패널(front panel)이라 불리는 두 개의 윈도우로 구성되어 있다.

Fig. 2.2는 프로그램을 실행했을 때 사용자가 특정 값을 입력하거나 시스템이 사용자에게 결과를 보여주기 위해 구성된 프런트 패널 윈도우이다. 프런트 패널은 다양한 유형의 컨트롤과 인디케이터를 포함하는 VI 코드 인터페이스이다. 그래프, 버튼, 텍스트, 불린(boolean) 등과 같은 컨트롤과 인디케이터가 프런트 패널에 위치하게 된다. 사용자는 프런트 패널에 위치한 컨트롤과 인디케이터를 통해 프로그램과 상호 작용을 한다.

Fig. 2.3은 프로그램이 실행될 때 사용자에게 보이지는 않지만 사용자가 지정한 알고리즘을 수행하는 역할을 하는 블록 다이어그램이다. 함수, 구조, 데이터를 하나의 객체에서 다른 객체로 옮기는 와이어뿐만 아니라, 프런트 패널 상의 컨트롤과 인디케이터에 대응하는 터미널과 그래픽 형태의 프로그램 코드를 포함한다. 블록 다이어그램의 아이콘들은 하위 레벨의 VI, 내장 함수와 프로그램 제어 구조를 표현한다.

---

i) LabVIEW : Laboratory Virtual Instrument Engineering Workbench

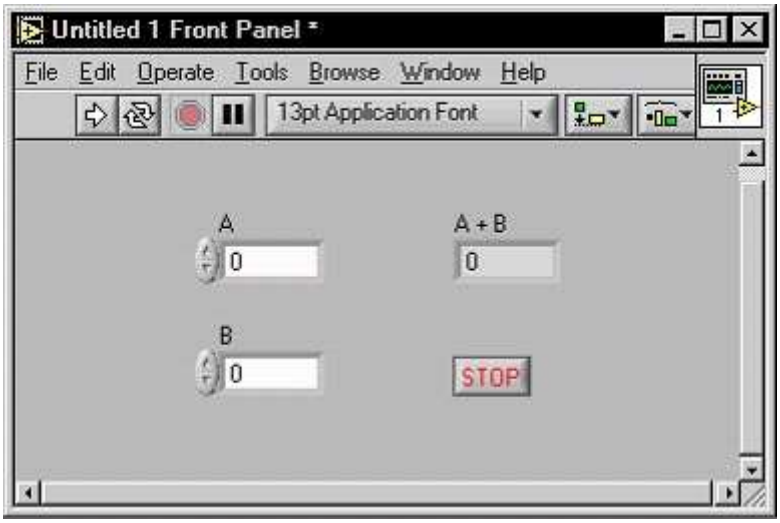


Fig. 2.2 Front Panel

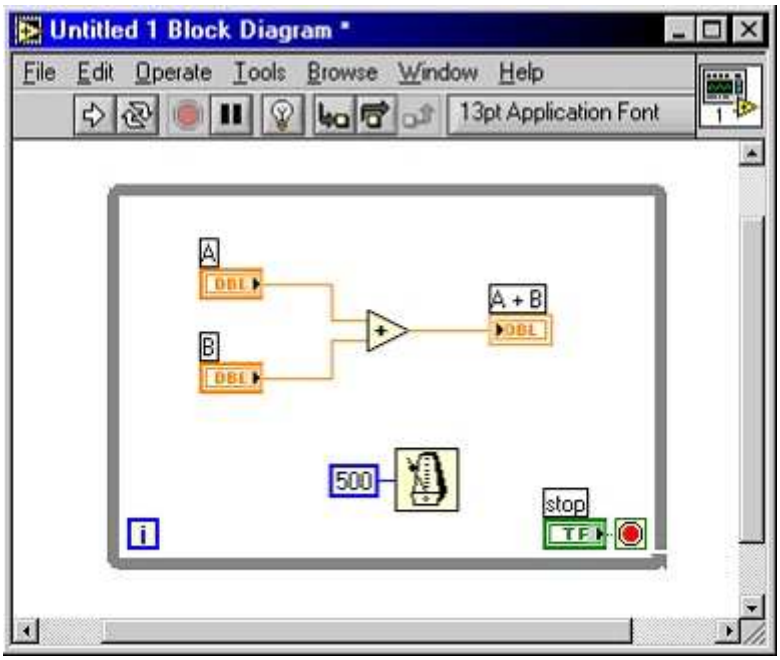


Fig. 2.3 Block diagram

아래는 LabVIEW의 대표적인 특징을 정리한 것이다.

1) 텍스트 기반의 프로그래밍 언어(포트란[ii], C 언어[iii])와는 달리 G 프로그래밍 언어로 알려진 그래피컬한 프로그래밍 언어를 사용한다. 따라서 프로그래밍 경험이 없어도 쉽게 프로그램을 짤 수 있어서 프로그램 개발 시간을 단축시킬 수 있다.
2) 순차적으로 진행되는 대부분의 프로그래밍 언어와는 달리 와이어를 통해 데이터가 흐르는 순서에 따라 진행되는 데이터 흐름(data flow) 방식이다.
3) 프로그램을 모듈화할 수 있다. 주어진 태스크(task)를 몇 개의 더 간단한 하부 태스크(subtask)로 나눈 후에(G 프로그래밍에서 이러한 하부 태스크들은 하부 VI(sub VI)라고 불려지고 서브루틴과 유사함), 각 하부 태스크를 수행하는 가상기계를 설계할 수 있다. 완전한 프로그램 형태인 최상위 레벨의 블록다이어그램에서 조합된다. 모듈성은 각 하부 VI를 독립적으로 실행할 수 있는 특성이다. 이러한 모듈성으로 인하여 하부 VI의 디버깅과 검증이 쉬워진다. 더욱이, 하부 VI가 일반 목적을 위하여 구성되었다면 다른 프로그램에서도 그 하부 VI를 사용할 수 있다.
4) LabVIEW에서 제공하는 기능 중 하나인 「Application builder」 기능을 사용하여 프로그램을 하나의 실행 파일로 만들 수 있다. 코딩한 프로그램의 지적 재산권을 보호할 수 있으며 다른 컴퓨터에 쉽게 설치가 가능하다. 실행 파일을 만들면 소스 코드인 블록 다이어그램을 볼 수 없으며, LabVIEW가

---

ii) 포트란 (FORTRAN) : Formular translator의 약자. 알골과 함께 과학 계산용으로 주로 사용되는 언어이며, ANSI에서 수정하고 능력을 확장시켜 포트란을 완성하였다. 산술 기호를 그대로 사용할 수 있으며 기초적인 수학 함수들을 그대로 불러내어 쓸 수 있으나 현재는 거의 쓰이지 않는 언어.

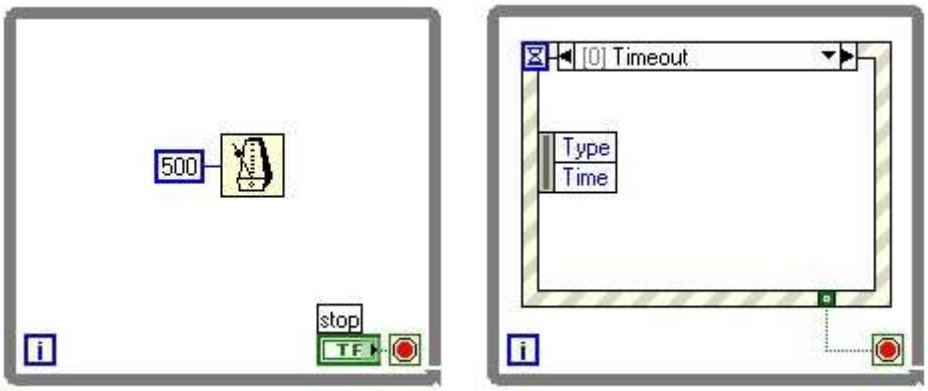
iii) C 언어 (C Language) : 벨 연구소에서 1971년경부터 리치(D.M.Ritchie) 등에 의해서 개발된 시스템 기술용의 프로그래밍 언어. UNIX 오퍼레이팅 기술에 사용할 것을 목적으로 설계한 언어로 UNIX OS의 대부분이 이 언어로 개발되었다. 컴퓨터의 구조에 밀착한 기초 기술이 가능한 것과 간결한 표기가 될 수 있는 것 등을 특징으로 하고 있다. 프로그램 오류를 쉽게 발견하기 위한 기능은 부족하지만, 고수준 언어에서 자주 볼 수 있는 기술상의 제약이 적기 때문에 오히려 프로그래밍하기 쉬운 편리한 언어로 평가된다.

설치되어 있지 않은 컴퓨터에서 실행파일만으로 프로그램을 실행할 수 있다.

위에서 기술한 LabVIEW의 특징 때문에 본 논문에서는 LabVIEW를 사용하여 시스템을 개발한다.

2.3 경보의 발생과 진단 메커니즘

감시 및 진단 시스템을 구성하기 위해서는 프로그램의 화면상에 나타낼 데이터를 실시간으로 갱신하는 것이 필요하며, 이를 위해 **Fig. 2.4**와 같은 반복 루프(while loop)를 이용한 폴링(polling)방식 혹은 이벤트 구조(event structure)를 사용한 이벤트 구동 방식을 사용할 수 있다.



(a) Polling (b) Event-driven

Fig. 2.4 Comparison between polling and event-driven method

Fig. 2.4 (a)와 같이, 폴링 방식에서 블록 다이어그램은 프런트 패널의 상태가 바뀌었는지를 확인하기 위하여 사용자가 지정한 시간마다 변화를 확인하는 루프를 계속적으로 돌아야 한다. 확인루프는 CPU 타임을 많이 사용하게 되고, 루프의 주기보다 빨리 일어나는 변화는 감지하지 못할 수도 있다.

Fig. 2.4 (b)의 이벤트 구동 방식 프로그램은 일반적으로 이벤트가 발생하기를 기다리는 루프를 가지고 있다. 이벤트에 반응하여 실행을 하고, 다음 이벤트가 발생할 때까지 대기한다. 프로그램이 어떻게 각 이벤트에 반응할 것인지는 이벤트에 대한 코드 작성에 의존하며, 프로그램의 실행순서는 어느 이벤트가 어떤 순서로 발생했는지에 따라서 결정된다.

이벤트를 사용하면 프로그램 상에서 지정해둔 이벤트가 발생할 때마다 블록 다이어그램에 정확하게 통보해주어, 사용자의 특정 조작에 바로 반응하기 때문에 확인루프를 사용하지 않아도 된다. 이벤트 구동 방식의 사용은 프로그램의 CPU 사용량을 줄이고, 블록 다이어그램 코드를 단순화시키며, 사용자의 상호작용에 블록 다이어그램이 정확히 반응하는 것을 보장한다.

본 논문에서는 실시간으로 들어오는 센서 데이터에서 발생하는 이벤트를 감지하여 경보 리스트를 갱신하고 진단이 이루어지도록 시스템을 구성한다.

(1) 디지털 센서에서의 경보 발생 원리

디지털 센서의 데이터는 0과 1의 상태로 들어오며, **Fig. 2.5**의 흐름도와 같이 프로그램에서 데이터의 변화를 감지하여 알람의 발생 및 소거를 판단한다. 경보가 발생하거나 소거되는 것과 같은 이벤트가 발생할 경우에만 경보 리스트를 갱신하고 진단이 이루어지도록 구성한다.

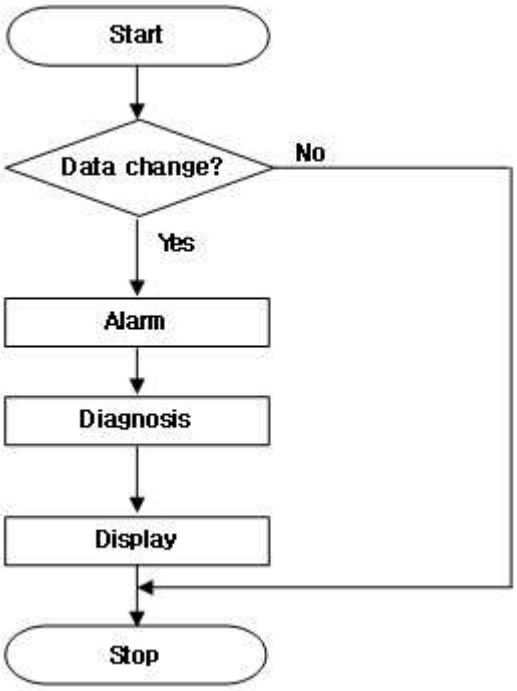


Fig. 2.5 Flowchart of digital sensor alarm

(2) 아날로그 센서에서의 경보 발생 원리

아날로그 센서의 경우, **Fig. 2.6**의 흐름도에서와 같이 들어온 데이터에 변화가 있는지 확인하고 변화가 있을 경우에만 경보 레벨과 비교하도록 구성한다. 데이터가 L 레벨과 H 레벨 사이에 있으면 정상값이라 판단하고 화면상에 정상적으로 표시하며, 사이값을 벗어날 경우 경보를 발생시킨다. L 레벨만 있을 경우 H 레벨만을 판단값으로 사용하며 H 레벨만 있을 경우 L 레벨만을 판단값으로 사용한다. 경보 레벨이 지정되어 있지 않은 경우 경보발생을 위한 비교판단 절차를 무시하고 아날로그 센서의 값만을 화면상에 보여준다.

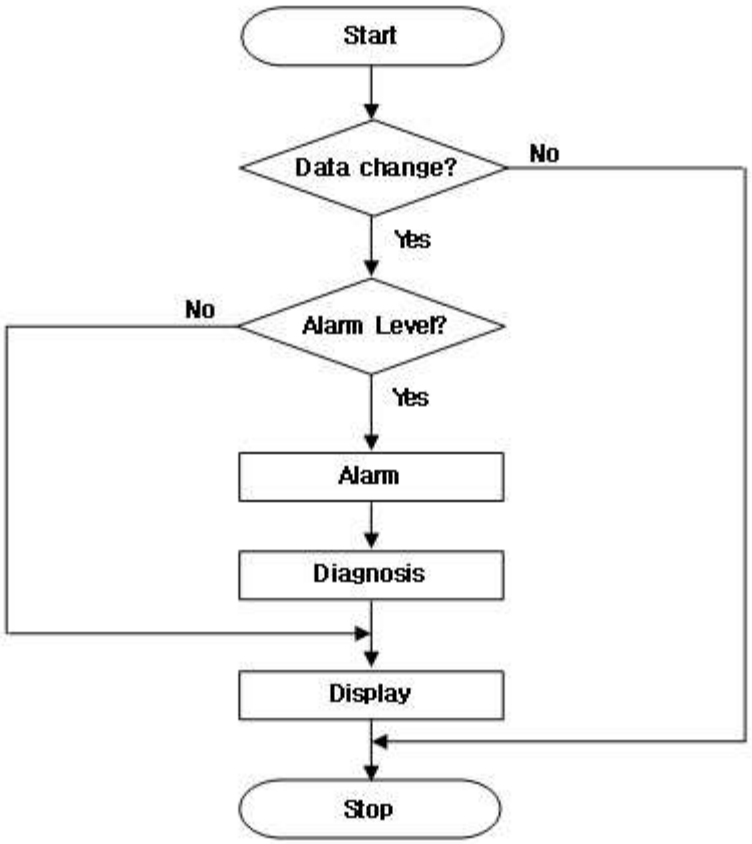


Fig. 2.6 Flowchart of analog sensor alarm

(3) 진단 메커니즘

전문가의 조언을 얻어 지식베이스를 구축하고 그에 따라 진단규칙을 작성하더라도 사용자가 원하는 바와 다른 경우가 있다. 따라서 본 논문에서는 사용자가 이미 구성된 진단 규칙을 수정하고 삭제하거나 원하는 진단 규칙을 추가할 수 있도록 구성한다.

사용자가 입력한 진단규칙을 프로그램에 바로 적용하고 새로운 경보가 발생하거나 소거되는 것을 이벤트로 지정하고, 이벤트가 발생했을 경우에만 구성된 진단규칙과 비교분석하여 일치하는 진단과 자세한 설명을 찾아내도록 구성한다.

2.4 데이터 저장

LabVIEW에서 파일 입·출력은 ASCII[iv] 형식, 이진(binary) 형태 및 앞의 두 가지 방식에 대한 디스크 스트리밍 네 가지 방식이 있다. 디스크 스트리밍이란 데이터를 HDD(hard disk driver)와 같은 저장매체에 연속적으로 저장하는 것을 의미하며, **Table. 2.1**에서 ASCII와 이진형태의 파일 입출력을 비교하였다.

Table 2.1 Comparison of binary and ASCII

	Binary	ASCII
저장 속도	ASCII보다 빠름	형변환 과정이 CPU와 메모리의 리소스를 많이 사용하여 느림
파일 크기	헤더(header) 정보가 필요하고 데이터 정보를 100% 유지하기 때문에 파일 크기가 크다.	형변환 과정에서 파일 크기를 줄일 수 있다.
형변환	없음	문자열로 형변환
파일 읽기	LabVIEW 이외의 파일에서는 읽을 수 없다.	Notepad, 한글, MS Word 등의 다른 프로그램에서 접근이 가능하며, 스프레드시트 파일은 엑셀에서 바로 읽어서 사용할 수 있다.

프로그램 실행 중에는 프로그램 상에서 데이터를 확인하고 프로그램 종료 후에도 LabVIEW 이외의 다른 응용 프로그램을 통하여 저장된 데이터를 읽고 수정이 가능하며 파일 크기가 작은 ASCII 파일 형태로 아날로그 센서의 데이터와 경보 발생 및 소거 기록을 저장한다.

---

iv) ASCII (American Standard Code for Information Interchange) : 미국에서 표준화가 추진된 정보교환용 7비트 부호. 256가지의 영역마다 어떤 원칙에 의해 표현 가능한 모든 숫자, 문자, 특수문자를 하나씩 정해 놓은 것이다.

(1) 경보 내역 저장

디지털 센서, 아날로그 센서에서 경보가 발생하거나 소거되는 것과 같은 이벤트가 발생했을 때만 내역이 실시간으로 저장하도록 구성한다.

이벤트가 발생하게 되면 경보명과 경보 레벨(아날로그인 경우 LL, L, H, HH, 디지털인 경우 on, off)과 경보 발생 혹은 소거 시각을 저장한다.

경보 내역은 프로그램이 실행 중일 때는 기능 버튼 중 경보 히스토리에서 볼 수 있게 구성한다. CSV[v] 파일이 아닌 탭으로 분리되어 저장하여 종료되었을 때는 엑셀뿐만 아니라 노트패드 등의 프로그램을 통해서도 그 내역을 편리하게 볼 수 있도록 구성한다.

(2) 아날로그 데이터 저장

디지털 센서의 경우 on/off만으로 표시되기 때문에 경보 히스토리에서 데이터를 확인할 수 있으며 진단에 사용하는 경우에는 그 상태만을 사용하면 된다. 하지만 아날로그 센서의 경우 그 값이 계속 변하기 때문에 경보 상태만이 아니라 데이터의 경향까지 파악해야만 더 정밀한 진단을 할 수 있다.

따라서 모든 아날로그 센서의 값을 1초 간격으로 저장하여 그 경향을 파악하고 사용자의 요구가 있을 때는 그 내역을 볼 수 있도록 구성한다. 경보 내역과 마찬가지로 프로그램이 실행 중일 때는 기능 버튼 중 히스토리를 눌러서 볼 수 있으며 프로그램이 종료되었을 때는 엑셀이나 메모장 등의 프로그램을 통해서도 그 내역을 볼 수 있도록 구성한다.

---

v) CSV file (comma-separated values file) : 각 항목의 값들이 쉼표(comma)에 의해 분리되는 일련의 아스키 텍스트 라인들로 구성되며, 레코드간의 구분은 새 줄(new line)로 구분된다.

제 3 장 시스템 구현

3.1 시스템 구현 방법

추후 프로그램이 수정될 경우를 고려하여 수정이 간편하고 디버깅이 쉽도록 시스템을 모듈화하였다. 또한 용량이 커지더라도 되도록 신속하고 정확한 반응을 보장하도록 이벤트 구동 방식으로 블록 다이어그램을 구성했다.

화면 구성을 위해 한국해양대학교 신조 실습선인 한바다호의 파이프라인을 참고했다.

3.1.1에서는 구성된 화면의 종류, 3.1.2에서는 블록 다이어그램을 구성하기 위해 사용된 방법에 관해 설명한다.

3.1.1 화면구성

Table 3.1과 같이 중요하다고 생각되는 8개 계통과 사용자의 편의를 위한 2개의 트렌드(trend) 화면으로 구성했다. 대부분의 사용자에게 익숙한 2차원 화면으로 구성했으며, 주요 기기는 3차원으로 구성했다. 화면은 파이프라인과 가능한 비슷하게 구현하였으며, 파이프는 선박에서 주로 사용하는 색상으로 표시하고, 화살표로 파이프 내 유체의 흐름 방향을 나타내었다. 발생한 경보는 해당 계통 화면의 센서가 위치한 곳에 적색으로 표시되며, 화면 좌측의 OCP 상단의 경보 목록창에도 표시된다.

Table 3.1 List of mimic and trend diagram

Mimic	Exhaust gas system
	Main & thrust bearing system
	High temperature cooling fresh water system
	Low temperature cooling fresh water system
	Sea water system
	M/E fuel oil service system
	M/E lubricating oil service system
	Boiler feed water service system
Trend	History trend
	Alarm history trend

3.1.2 블록 다이어그램 구성

중복되는 부분이나 독립적인 기능을 가진 부분은 디버깅과 수정이 쉬운 하부 VI로 구성하여 시스템을 모듈화하였다. 구성된 하부 VI를 기능이 중복된 여러 곳에서 사용하거나 사용자가 원할 시에만 실행되도록 하여 프로그램을 효율적으로 구성하였다.

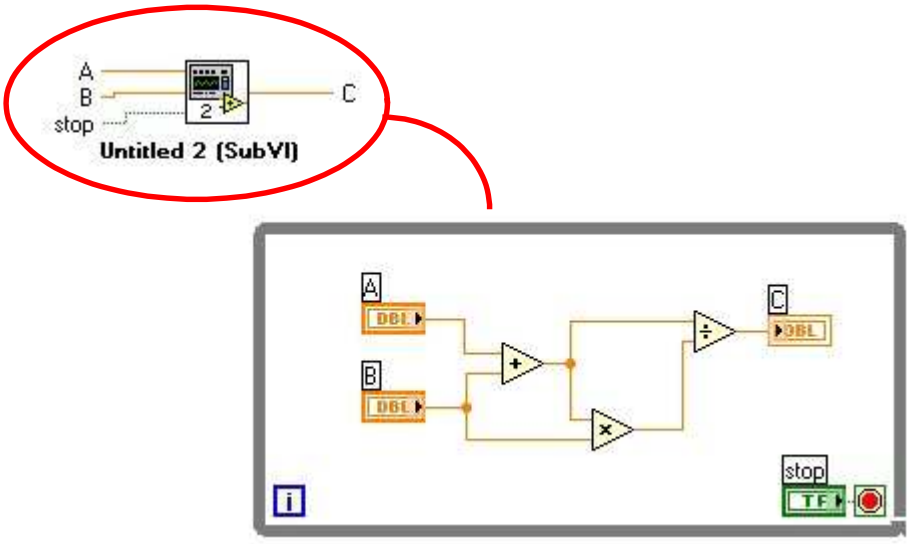


Fig. 3.1 Example of sub VI

하나의 하부 VI에는 외부로부터 데이터를 받아들이는 입력 노드와 하부 VI 내부 코드를 거쳐 외부로 데이터를 내보내는 출력 노드가 있다. **Fig. 3.1**에서 보이는 것처럼 사용자가 지정한 부분을 입·출력 노드를 가진 아이콘 형태의 하부 VI로 구성한다. 구성된 하부 VI는 외부에서 데이터를 입력받고 하부 VI 내부 프로그램을 거친 후 출력노드를 통해 데이터를 내보낸다.

Fig. 3.2는 LabVIEW가 자체적으로 지원하는 하부 VI가 아닌 사용자가 구성한 하부 VI의 목록 중 일부를 보여준다. 주실행프로그램을 실행시키면 Main.vi에 속해있던 하부 VI들이 실행되며, Change.vi, Global - Alarm Name.vi, Print.vi, Sum.vi는 사용자의 요구가 있는 경우에만 실행되기 때문에 CPU와 메모리의 사용량을 줄일 수 있다.

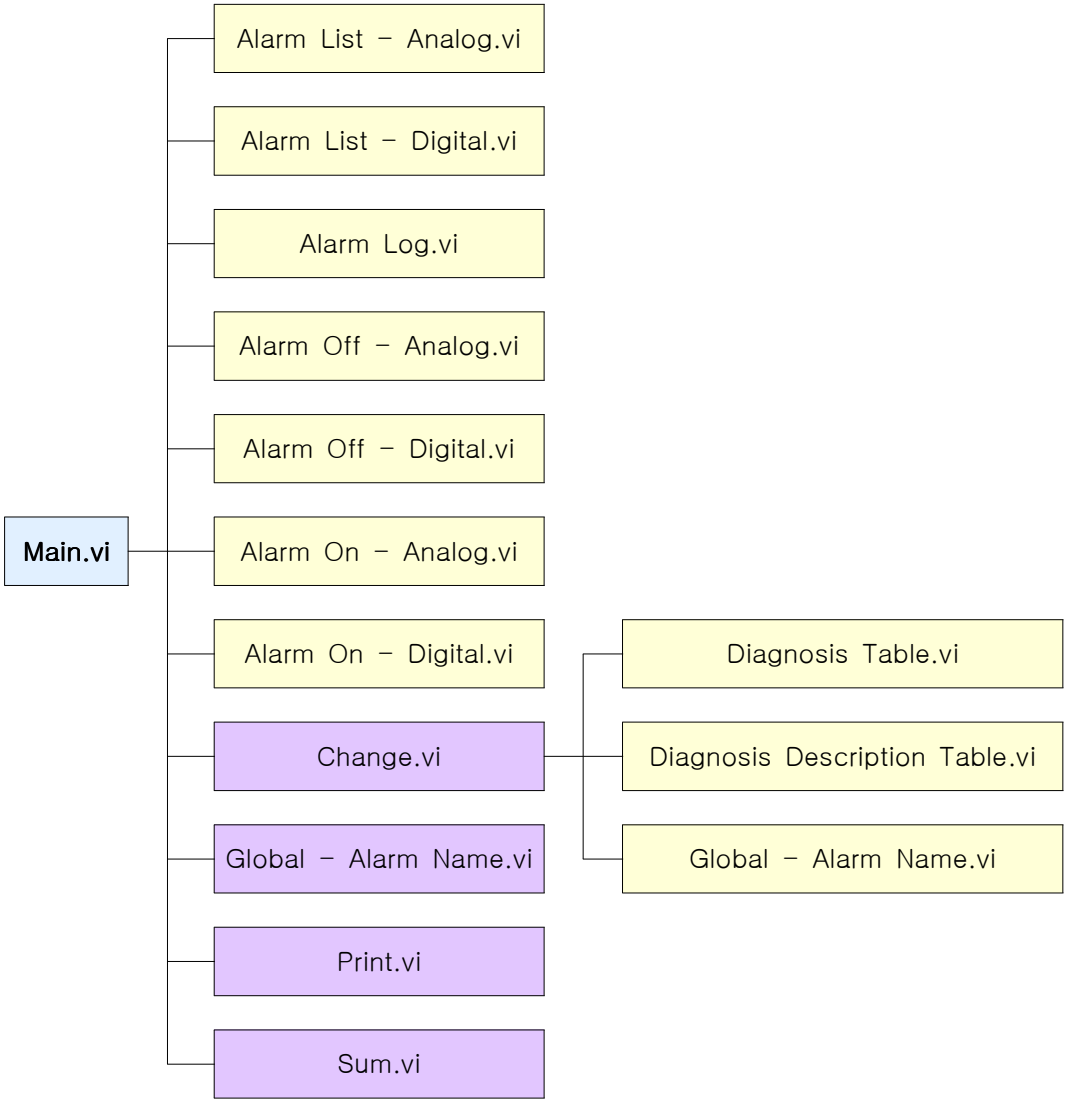


Fig. 3.2 List of used sub VI

사용자가 구성한 하부 VI뿐 아니라 LabVIEW에서 지원하는 하부 VI 모두를 계통도(hierarchy) 화면에서 한 눈에 볼 수 있다. **Fig. 3.3**과 같이 최상위에 주 실행프로그램인 Main.vi가 표시되며, 그 아래에 각 VI에서 사용된 하부 VI들의 계층별 구조를 볼 수 있다.

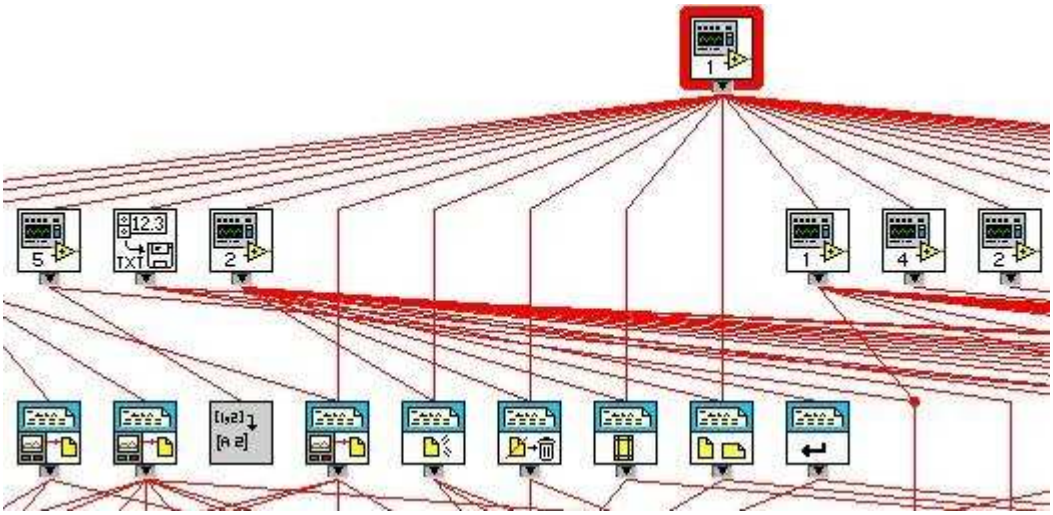


Fig. 3.3 Hierarchy of sub VI

3.2 계통별 화면구성

(1) 배기 시스템(exhaust gas system)

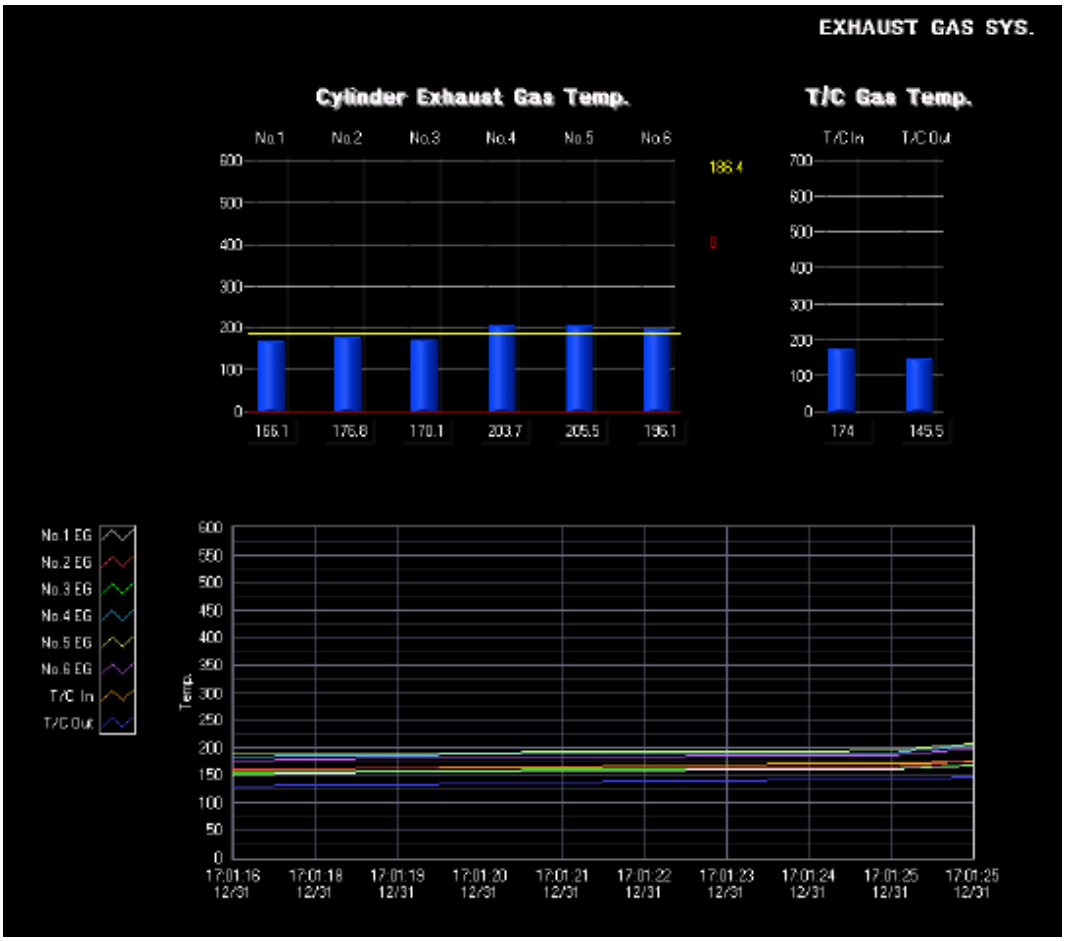


Fig. 3.4 Exhaust gas system

총 6개의 실린더에서 배출되는 각각의 배기 온도와 T/C(turbo charger) 입·출구 온도를 한 화면에서 관찰할 수 있으며, 전체 배기의 경보 레벨 및 현재 평균값을 막대그래프와 텍스트를 통해 실시간으로 확인할 수 있도록 구성했다. 또한 입력되는 아날로그 데이터의 실시간 확인 및 관찰이 가능하도록 실시간 트렌드(realtime trend)를 구성했다.

(2) 메인 베어링과 추력 베어링 시스템(main & thrust bearing system)

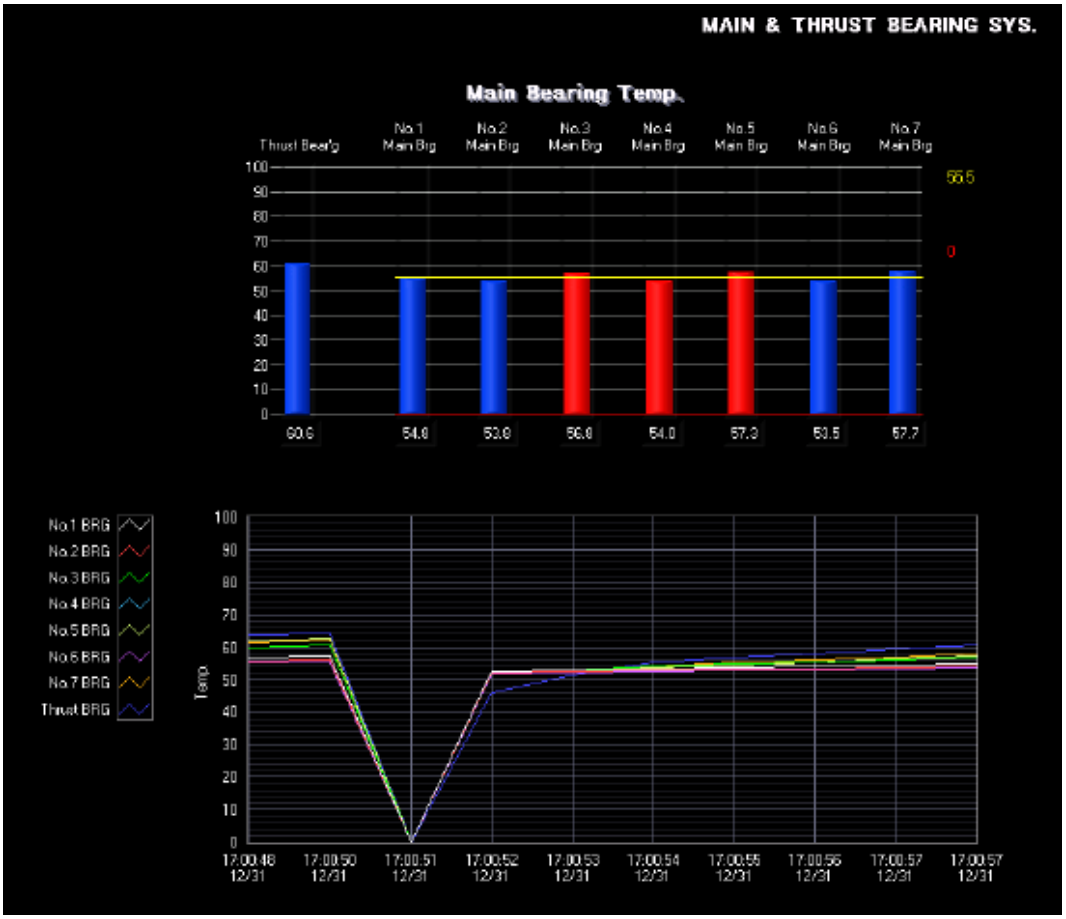


Fig. 3.5 Main & thrust bearing system

메인 베어링(main bearing)과 추력 베어링(thrust bearing) 온도를 한 화면에서 관찰할 수 있으며 전체 베어링의 경보 레벨 및 현재 평균값을 막대그래프와 텍스트를 통해 실시간으로 확인할 수 있도록 구성했다. 또한 입력되는 아날로그 데이터의 실시간 확인 및 관찰이 가능하도록 실시간 트렌드를 구성했다.

(3) 고온 냉각 청수 시스템(high temperature cooling fresh water system)

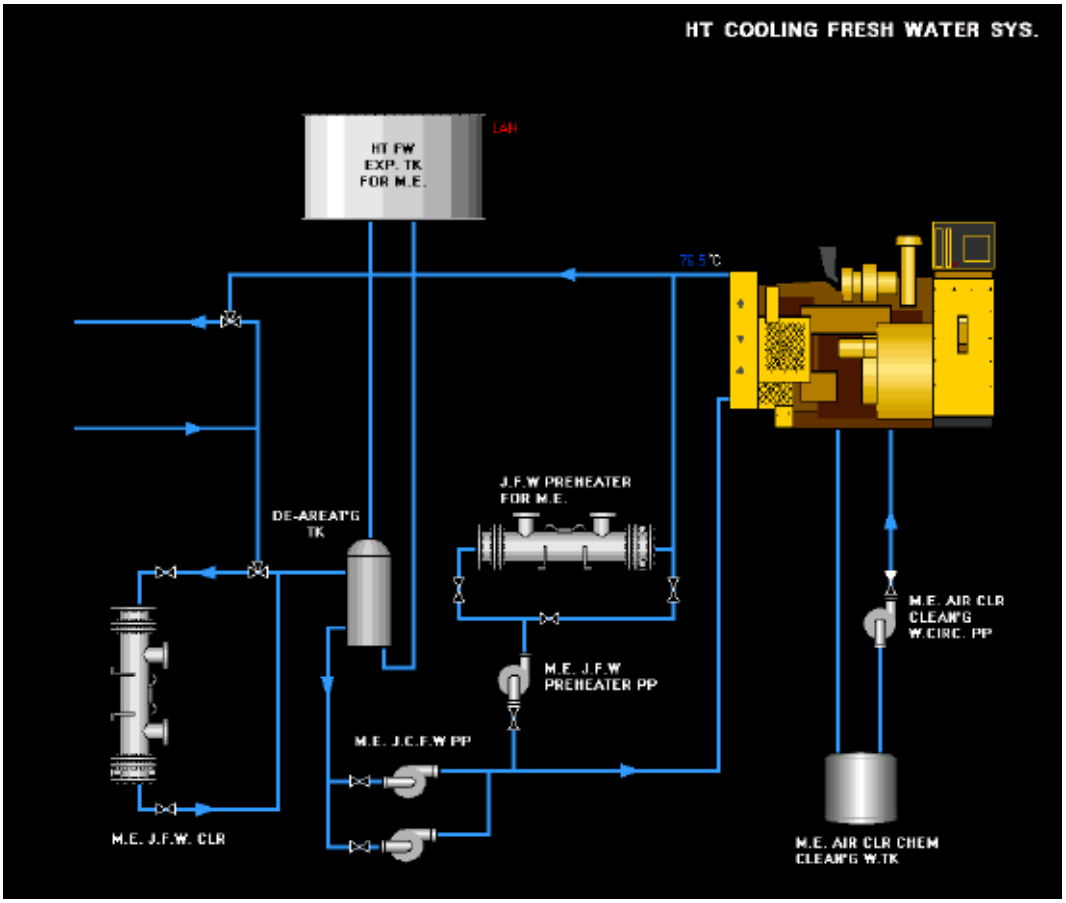


Fig. 3.6 High temperature cooling fresh water system

고온 냉각 청수 계통의 설계도면을 바탕으로 각 탱크, 펌프, 쿨러 및 파이프만을 간결하게 표현하며, 출구 냉각수 온도의 경우는 실시간 확인 및 관찰을 위해 아날로그 텍스트 형태로 표시했다.

(4) 저온 냉각 청수 시스템(low temperature cooling fresh water system)

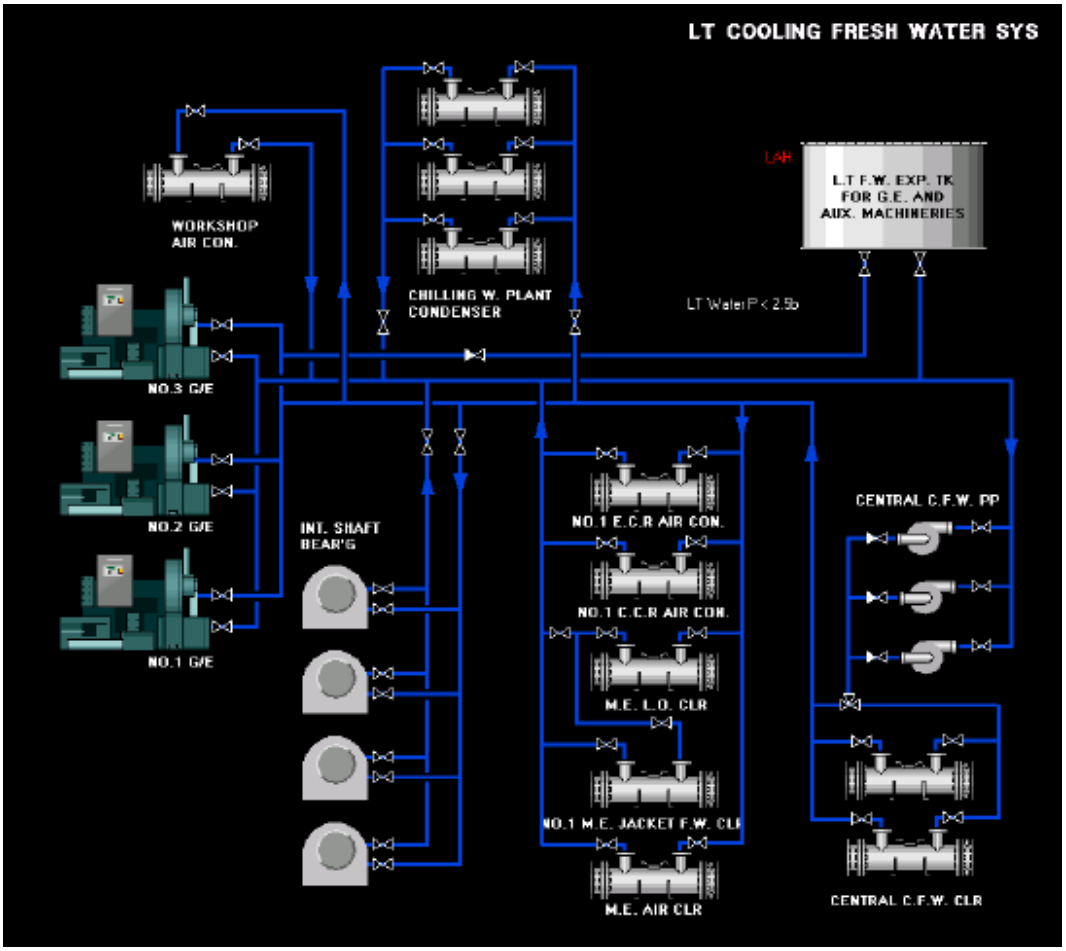


Fig. 3.7 Low temperature cooling fresh water system

저온 냉각 청수 계통의 설계도면을 바탕으로 각 탱크, 펌프, 쿨러, 복수기(condenser) 및 파이프만을 간결하게 표현했다.

(5) 냉각 해수 시스템(cooling sea water service system)

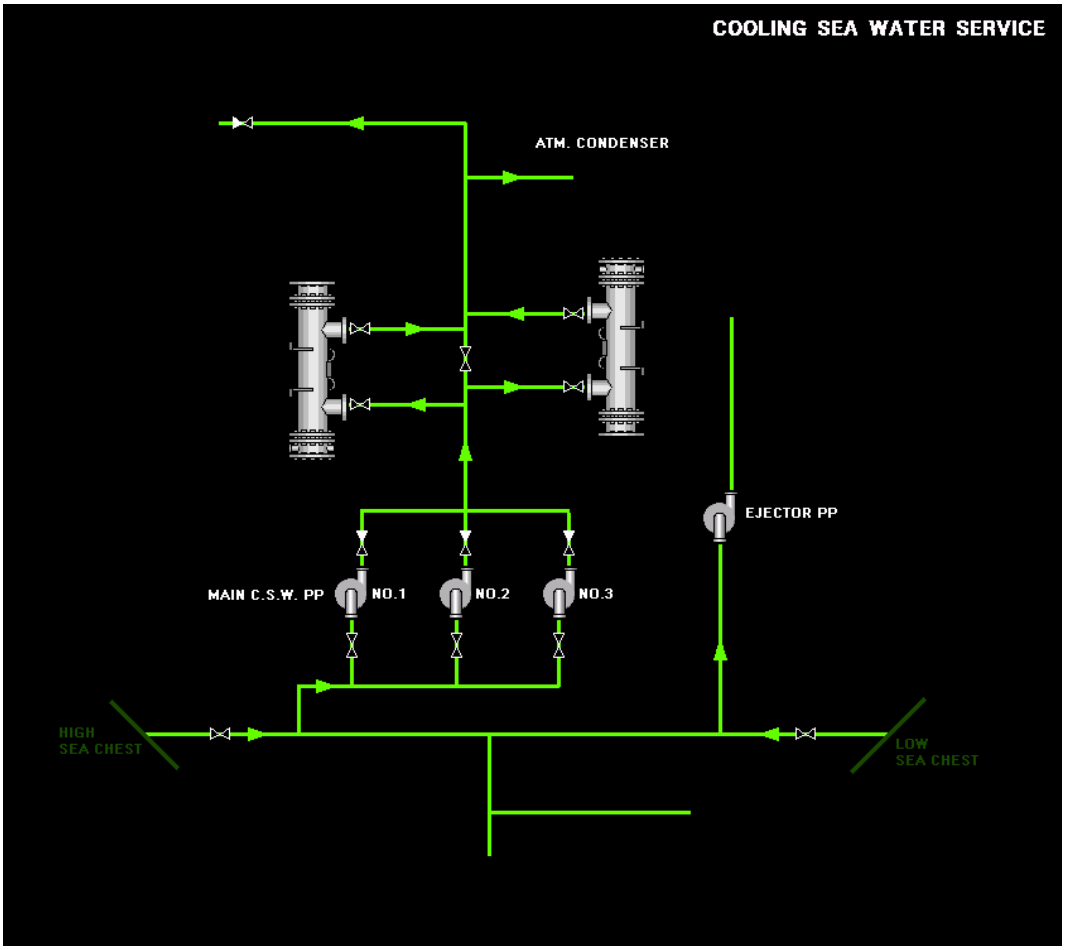


Fig. 3.8 Cooling sea water service system

냉각 해수 계통의 설계도면을 바탕으로 각 탱크, 펌프, 쿨러 및 파이프만을 간결하게 표현했다.

(6) 주기관 연료유 공급 시스템(M/E fuel oil service system)

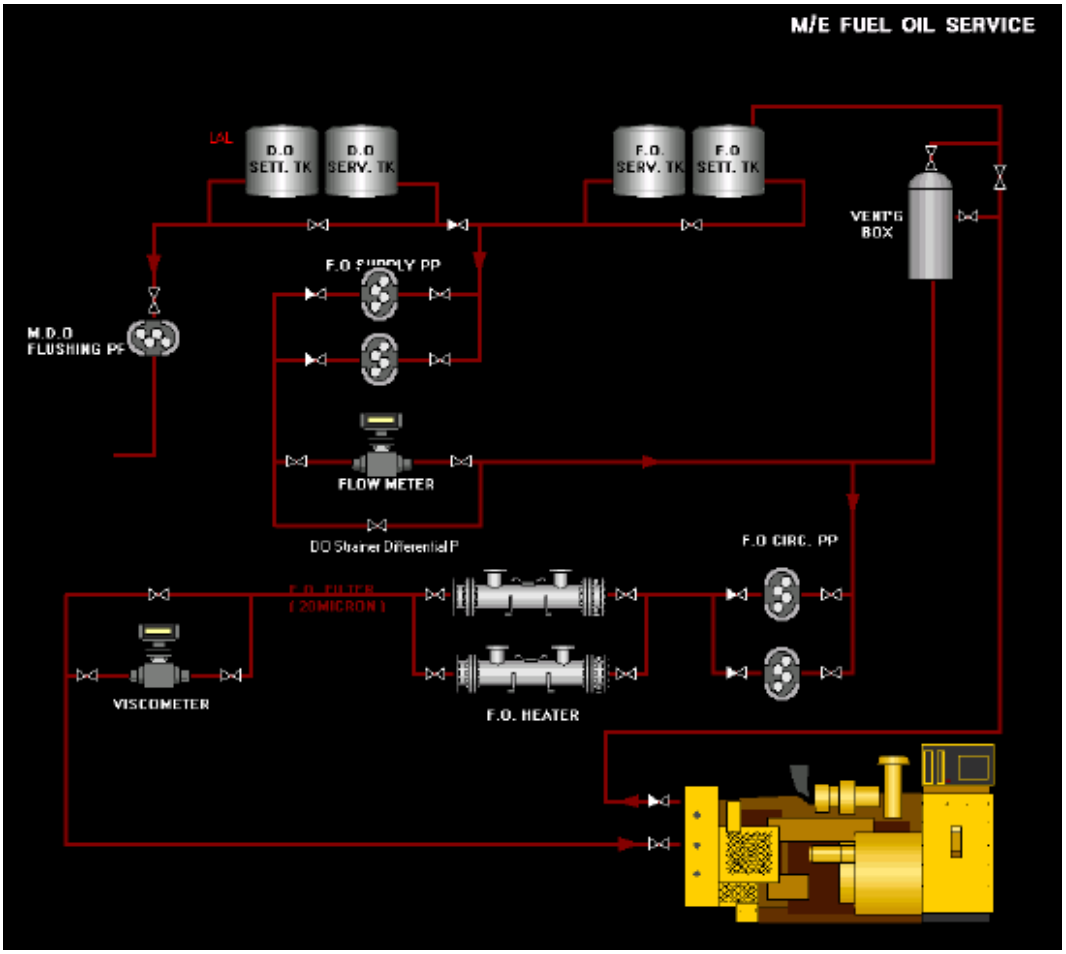


Fig. 3.9 M/E fuel oil service system

주기관 연료유 공급 계통의 설계도면을 바탕으로 각 탱크, 펌프, 히터 및 파이프만을 간결하게 표현했다.

(7) 주기관 윤활유 공급 시스템(M/E lubricating oil service system)

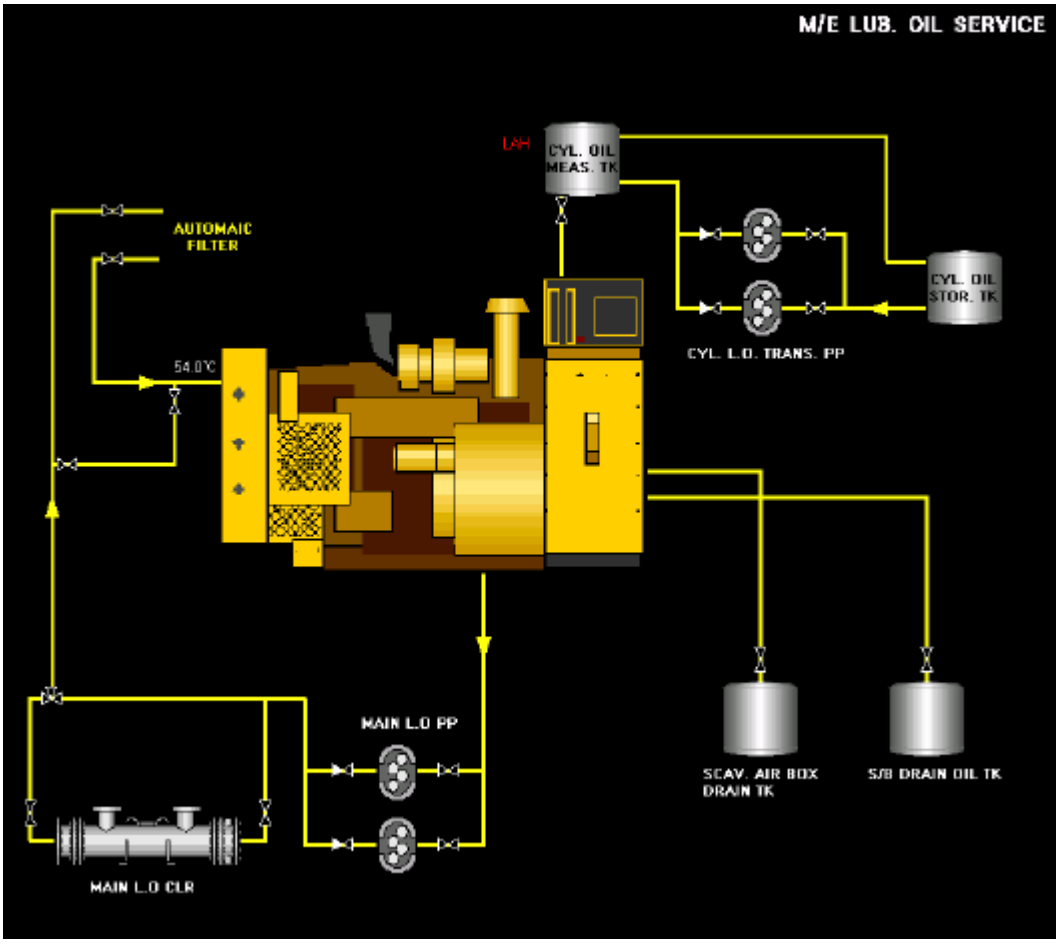


Fig. 3.10 M/E lubricating oil service system

주기관 윤활유 공급 계통의 설계도면을 바탕으로 각 탱크, 펌프, 쿨러 및 파이프만을 간결하게 표현했다. 엔진 입구 윤활유 온도의 경우 실시간 확인 및 관찰을 위해 아날로그 형태로 표시했다.

(8) 보일러 급수 시스템(boiler feed water service system)

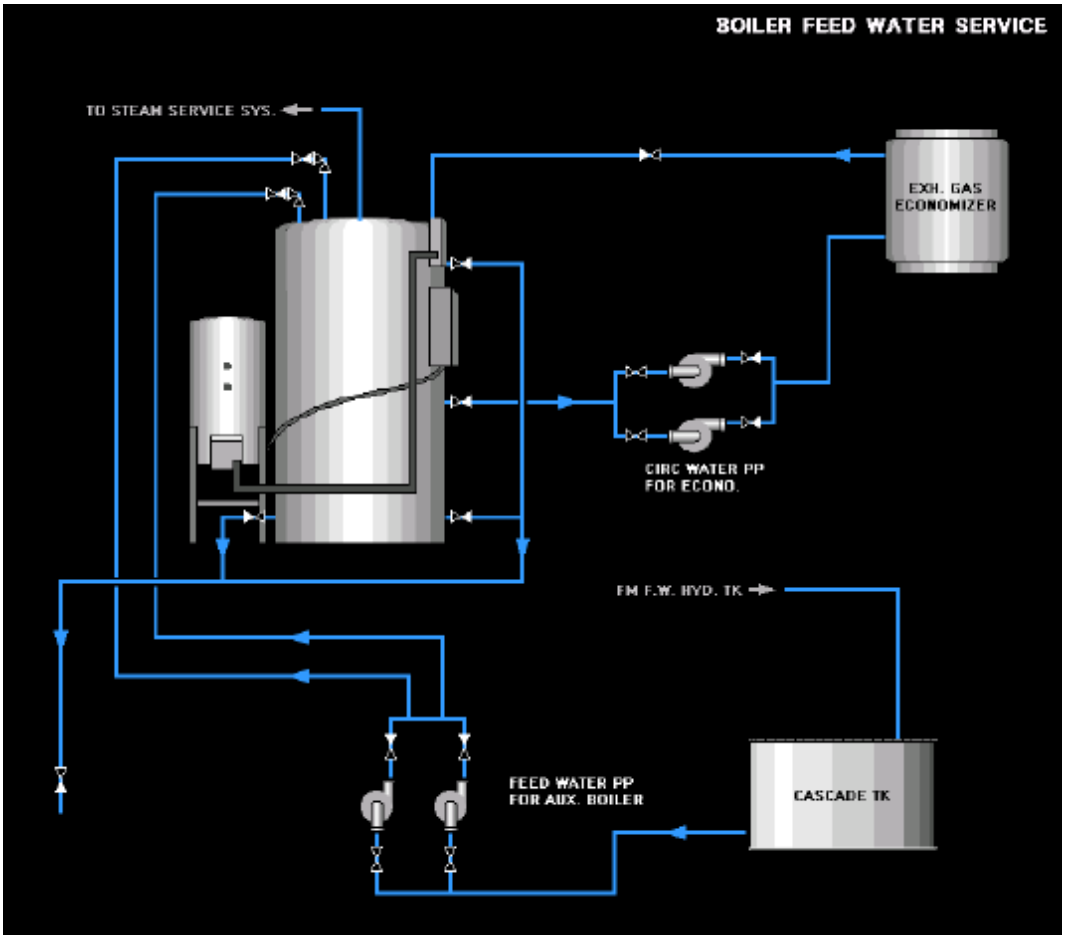


Fig. 3.11 Boiler feed water service system

보일러 급수 계통의 설계도면을 바탕으로 각 탱크, 펌프, 절탄기(economizer) 및 파이프만을 간결하게 표현했다.

3.3 트렌드

(1) History trend

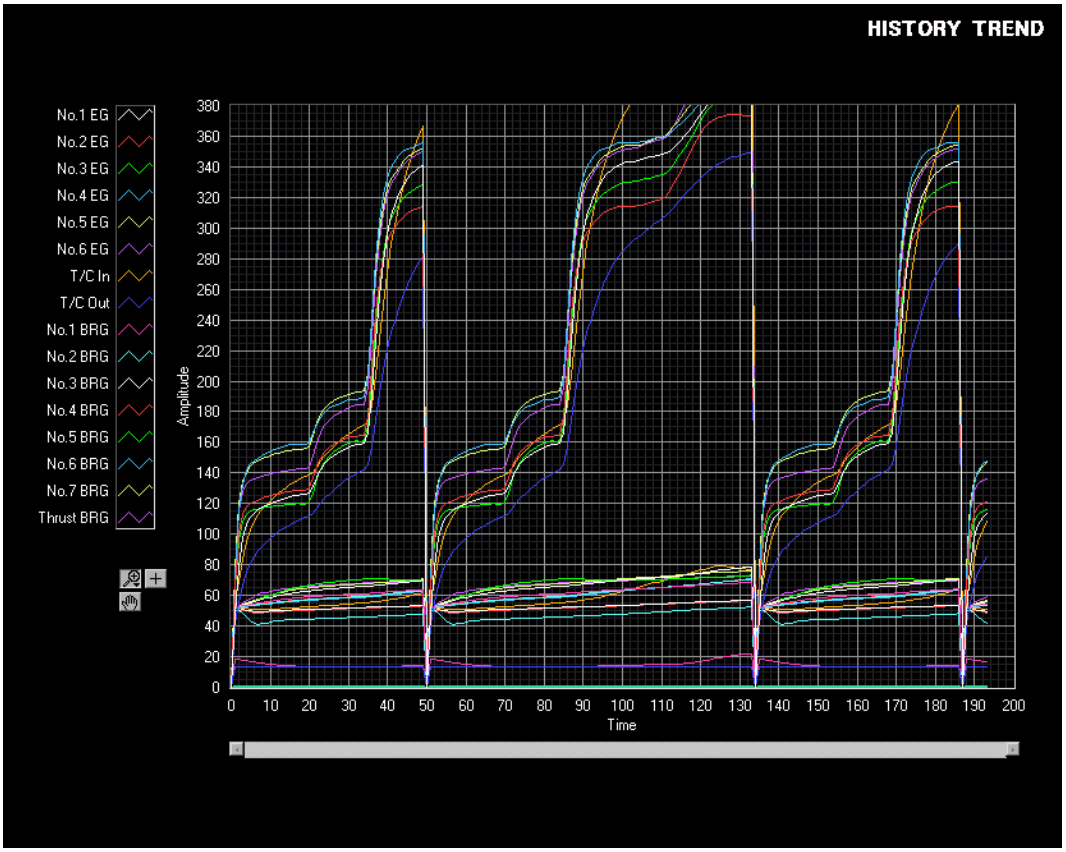


Fig. 3.12 History trend

모든 아날로그센서의 과거값을 볼 수 있도록 구성했다. 정해진 시간마다 아날로그 센서값이 저장되며, 사용자가 요구할 시 저장된 모든 데이터를 보여준다. 하단의 스크롤바를 움직여 좌우로 그래프를 이동할 수 있으며 좌측 도구를 사용하여 확대나 축소가 가능하다.

(2) Alarm history

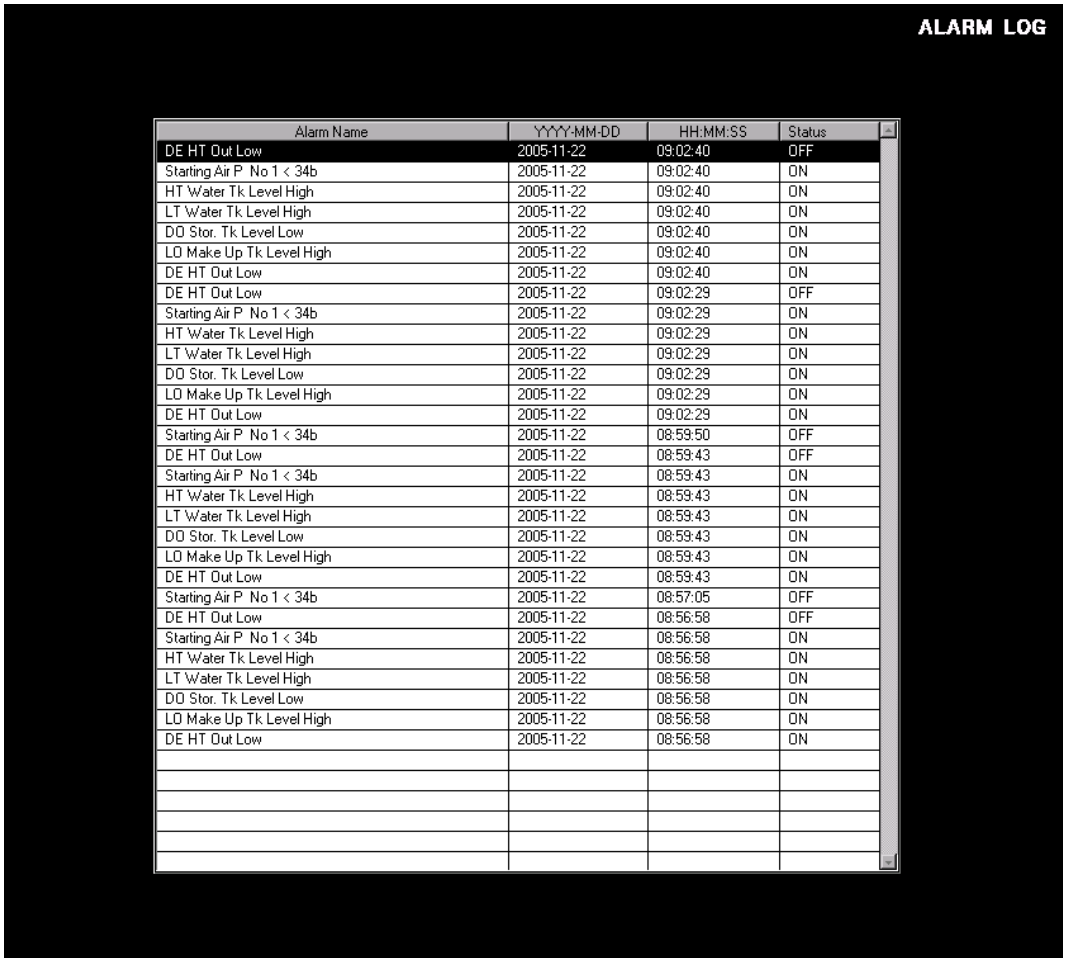


Fig. 3.13 Alarm history

모든 센서의 경보 발생 및 소거 내역이 경보명, 발생시각, 상태와 함께 저장된다. 디지털 센서인 경우 경보명만, 아날로그 센서인 경우 경보명과 경보 발생 레벨이 함께 저장된다.

3.4 OCP(Operating console panel)

Fig. 3.14에서 보는 것과 같이, 메인 화면의 좌측에는 사용자의 편의를 위하여 OCP가 구성되어 있다. 현재 발생한 경보의 목록을 보여주는 경보 리스트, 발생한 경보들을 이미 짜여진 진단규칙과 실시간으로 비교하여 진단 결과를 나타내는 진단창, 진단에 관한 상세한 설명을 나타내는 진단 설명창, 총 8개로 구성된 각 계통의 화면으로 이동하기 위한 MIMIC 버튼과 사용자의 편의를 위해 구성된 기능 버튼(function button)들, 프로그램 종료를 위한 종료(exit) 버튼으로 구성되어 있다.

각각의 기능에 대해 자세하게 설명한다.

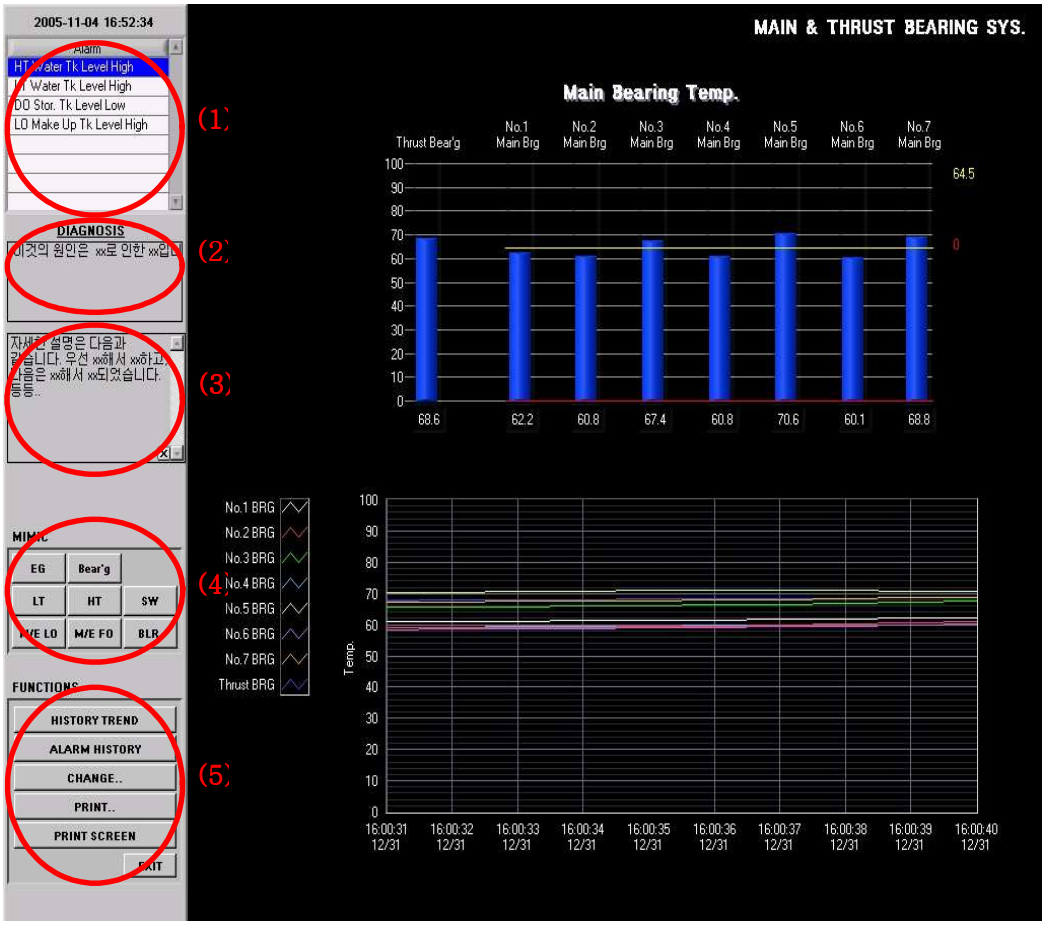


Fig 3.14 Operating console panel

(1) 경보 리스트 창

Fig. 3.14의 (1)에서 보이는 것과 같이, 발생한 경보의 이름이 실시간으로 경보 리스트 창에 나타난다. 최근에 발생한 경보가 가장 위에 위치하며, 경보가 소거되었을 때는 경보 리스트에서도 사라진다. 하지만 경보 히스토리에는 발생하거나 소거된 내용이 기록된다. 경보 리스트의 상단에는 시스템으로 읽어온 현재 날짜와 시각이 표시된다. 아날로그 경보는 경보명과 LL(Low Low), L(Low), H(High), HH(High High)의 레벨이 함께 나타난다.

(2) 진단 창

Fig. 3.14의 (2)는 발생한 경보에 대한 진단 결과를 실시간으로 나타내주는 진단창이다. 경보가 발생하거나 소거되는 것과 같은 이벤트가 발생할 때마다 새로운 진단을 하고 진단리스트를 갱신한다. 진단결과에 대한 자세한 설명이 필요할 경우에는 진단결과 중 하나를 더블클릭하면 **Fig. 3.14**의 (3)에서 보이는 것과 같이 진단창 아래에 해당 진단에 관한 상세한 진단 설명창이 나타난다. 진단 설명창을 닫고 싶을 때는 ☒ 를 클릭하면 된다. 진단에 관한 설명을 바꾸고 싶을 때는 **CHANGE..** 메뉴로 팝업창을 띄워 바꿀 수 있다.

(3) MIMIC

Fig. 3.14의 (4)에서 보이는 것과 같이, 앞서 설명한 8계통의 화면으로 이동하기 위한 버튼이 구성되어 있다. 각 버튼을 누르면 해당 화면으로 이동한다. CPU 타임을 절약하기 위하여 버튼을 마우스로 클릭하는 것을 이벤트로 설정하고 선택 상태를 유지하기 위하여 쉬프트 레지스터(shift register)를 사용했다. **Fig. 3.15**는 **Table 3.1**의 8계통의 MIMIC과 기능버

튼을 누르는 것을 이벤트로 지정하여 블록다이어그램으로 구성한 것을 보여주는 그림이다.

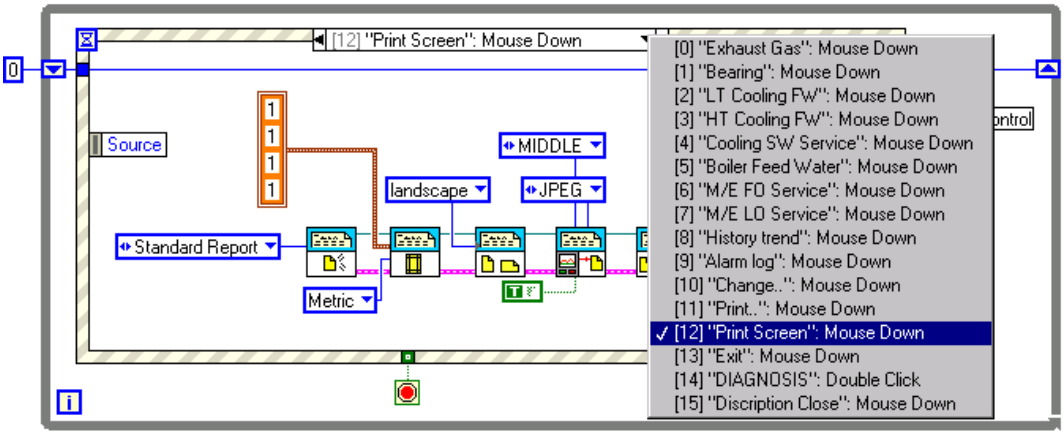


Fig. 3.15 Menu block diagram

(4) 기능 버튼

Fig. 3.14의 (5)에서 보이는 것과 같이, **HISTORY TREND**, **ALARM HISTORY**, **CHANGE..**, **PRINT**, **PRINT SCREEN**, **EXIT** 등 사용자의 편의를 위한 버튼으로 구성되어 있다. 메뉴 버튼과 마찬가지로 CPU 타임을 절약하기 위하여 버튼을 마우스로 누르는 것을 이벤트로 설정하고 선택 상태를 유지하기 위하여 쉬프트 레지스터를 사용했다.

① HISTORY TREND

모든 아날로그 센서의 과거값을 볼 수 있다. 트렌드의 좌측에는 색으로 구분된 그래프 범례가 있고 그 아래에는 시간 조정, 부분 확대 및 축소가 가능한 도구가 있다.

② ALARM HISTORY

발생하거나 소거된 경보의 모든 내역을 볼 수 있다. 가장 최근에 발

생하거나 소거된 경보명이 가장 위에 나타난다. **Fig. 3.16**은 하부 VI로 구성된 경보내역 트렌드 블록 다이어그램을 보여주는 그림이다. 그림의 가장 좌측의 하부 VI에서 특정 장소의 파일을 읽고 파일을 테이블 구조로 변환하도록 구성되어 있다. 파일의 마지막 라인까지 읽은 후 변환을 위한 반복루프를 종료시켜 테이블을 사용자에게 보여준다.

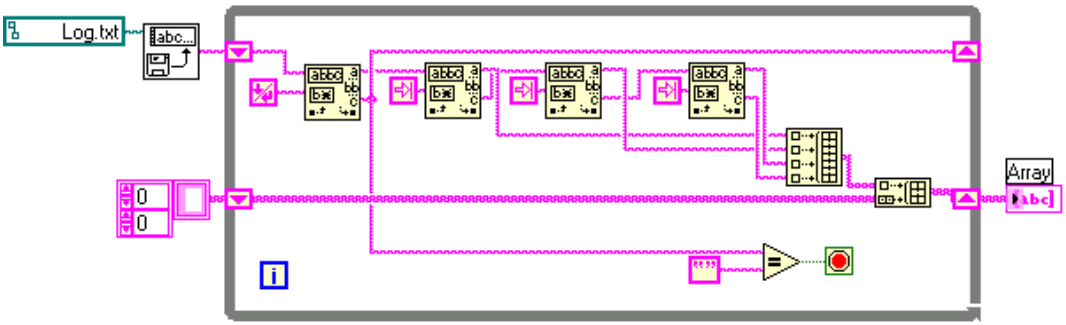


Fig. 3.16 Alarm history trend block diagram

③ CHANGE..

아날로그 경보 레벨과 진단 설명을 프로그램 상에서 쉽게 수정하거나 삭제할 수 있고 사용자가 원하는 진단규칙을 추가할 수 있도록 구성하였다. **Fig. 3.17**과 같이 사용자가 변경을 원할 경우 **CHANGE..** 버튼을 누르면 하부 VI로 구성된 change.vi가 실행되어 팝업창이 나타나고 아날로그 경보 레벨 수정과 진단 수정 및 추가를 선택할 수 있는 탭이 창의 상단에 보인다.

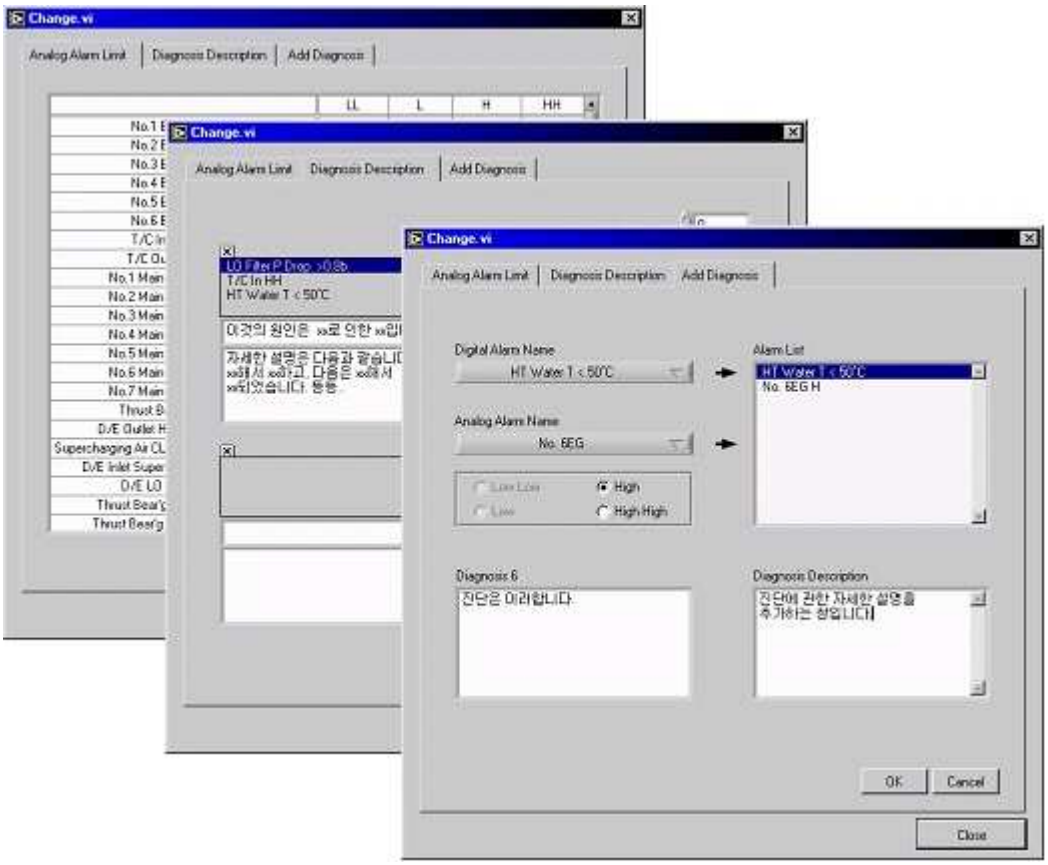


Fig. 3.17 Modification, addition and deletion of data file

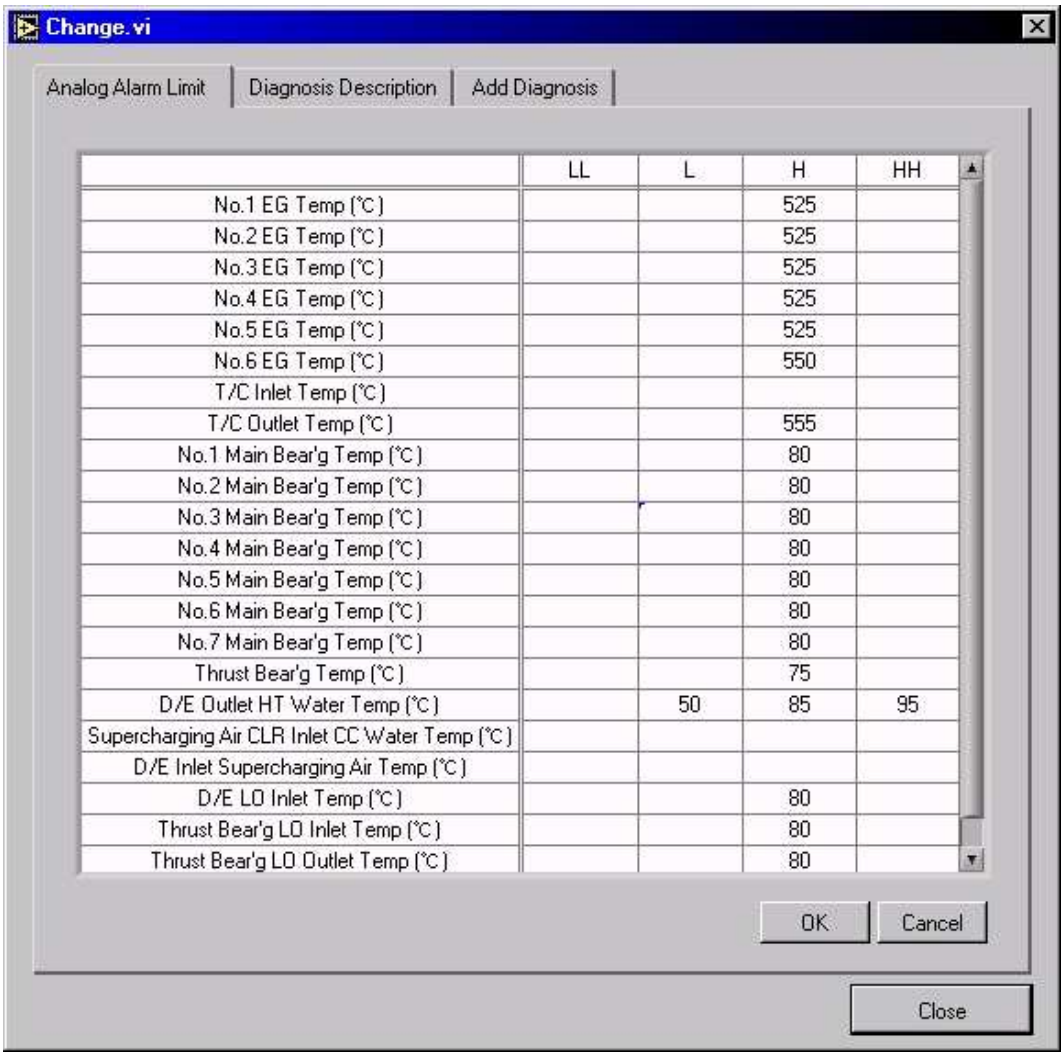


Fig. 3.18 Modification of analog alarm limit

Fig. 3.18과 같이 **Analog alarm limit** 탭에서는 아날로그 센서명과 각 센서의 경보 발생값을 보여준다. 각 센서마다 LL, L, H, HH의 네 가지 경보 레벨이 있으며 경보 발생값을 없애고 싶을 때는 공란으로 만들면 된다. 모든 레벨이 공란으로 지정되면 실시간으로 값만 보여진다.

OK 버튼을 누르면 변경된 내용이 저장이 되고, 취소를 하고 싶을 때 **Cancel**을 누르면 가장 마지막에 저장된 상태로 돌아간다.

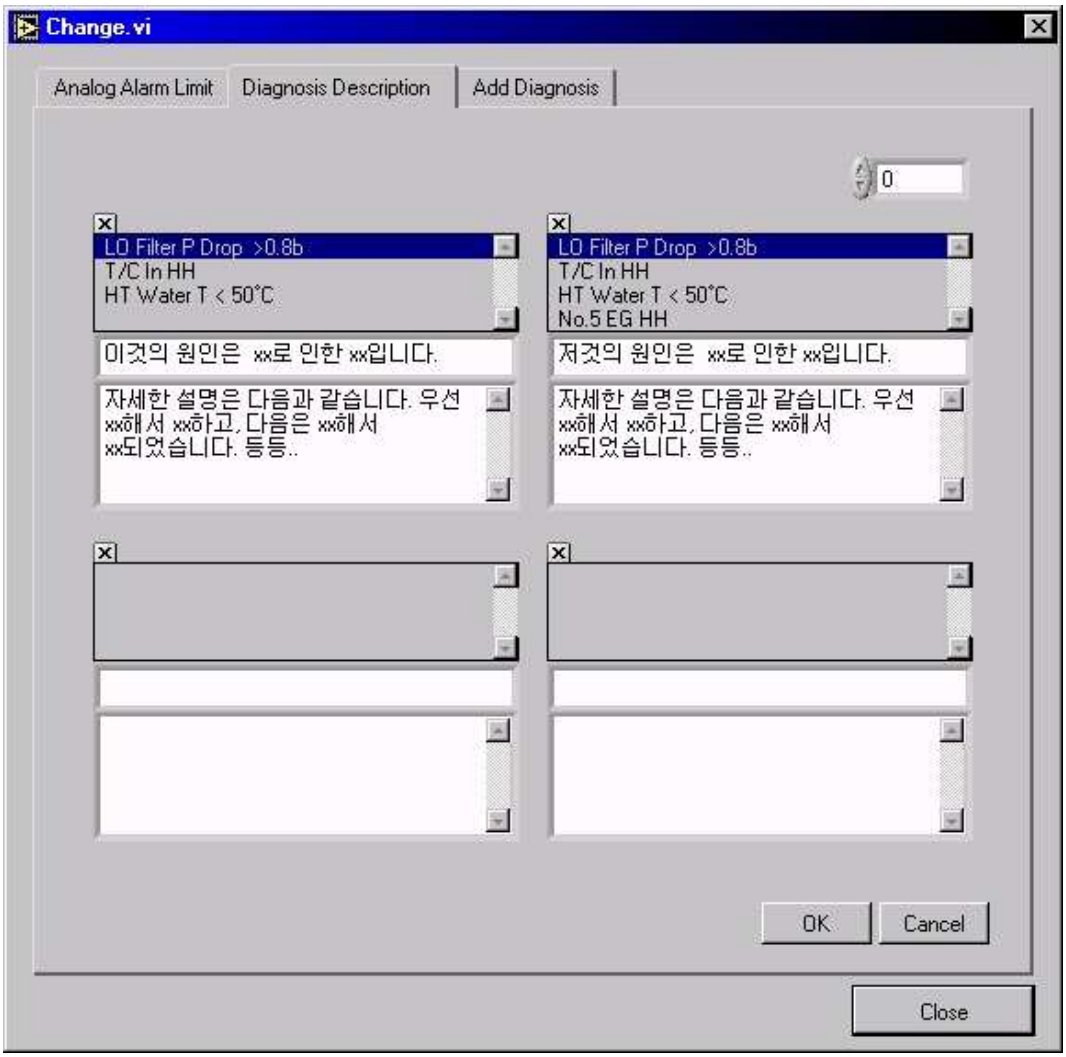


Fig. 3.19 Modification of diagnosis description

Fig. 3.19와 같이 **Diagnosis description** 탭에서는 발생한 경보에 대하여 내리게 될 진단과 진단에 관한 자세한 설명을 보여준다. 미리 구성된 설명에 내용을 부가, 삭제할 수 있다. 좌측 상단의 인덱스를 사용하여 더 많은 진단 설명을 볼 수 있다. **Analog alarm limit** 탭에서와 마찬가지로 변경 후 **OK** 버튼을 누르면 변경내용이 저장이 되고, **Cancel** 버튼을 누르면 가장 마지막에 저장된 상태로 돌아간다.

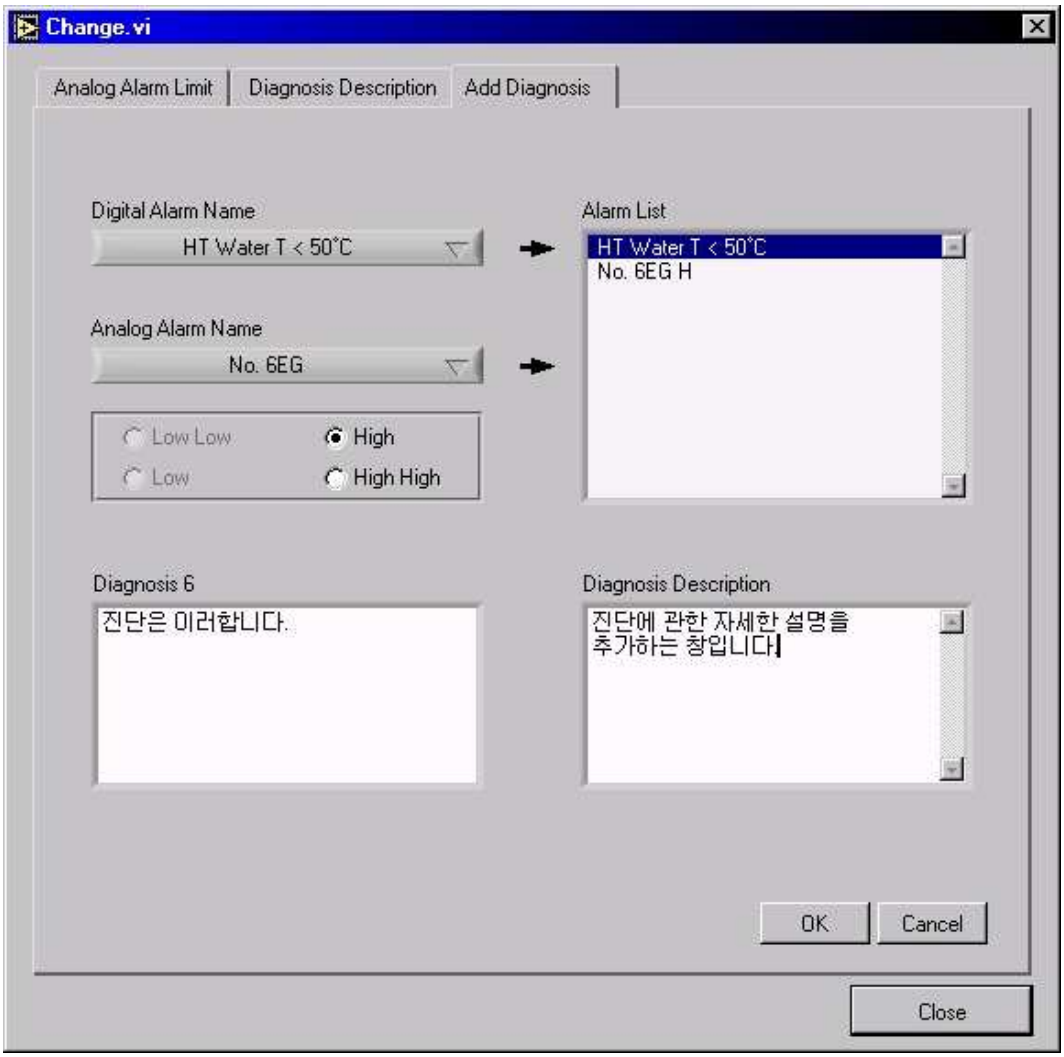


Fig. 3.20 Addition diagnosis

Fig. 3.20에서 보는 것과 같이 **Add diagnosis** 탭에서는 진단규칙을 추가할 수 있다. 좌측의 디지털 경보명, 아날로그 경보명과 레벨을 선택한 후 화살표를 눌러서 만들고자 하는 진단규칙에 사용될 경보 리스트를 작성한다. 경보 리스트에서 삭제하고자 하는 경보는 경보명을 더블클릭하면 삭제된다. 작성된 리스트에 관한 진단과 진단 설명을 넣은 후 저장을 하면 진단규칙이 추가된다.

④ PRINT

Fig. 3.21은 하부 VI 중 하나인 print.vi를 실행시켰을 때 나타나는 프린트 패널이다. 아날로그 센서의 경보 레벨과 경보 로그 등을 인쇄할 수 있다. 인쇄를 원하는 항목에 체크를 한 후 프린트 버튼을 누르면 체크된 항목이 순서대로 인쇄된다. 프린트 매수와 용지의 가로, 세로 설정 등을 할 수 있다. 프린트를 취소하고자 할 때는 취소 버튼을 누르면 된다.

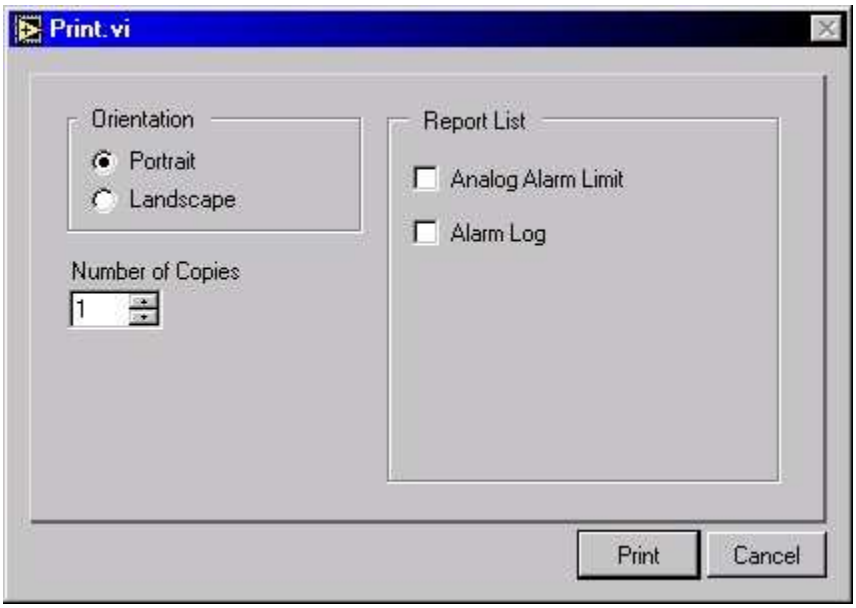


Fig. 3.21 Print

Fig. 3.22는 print.vi의 블록 다이어그램 중 일부이다. 글자체, 글자 굵기, 크기 등은 사용자가 구성을 할 수 없도록 상수로 구성되어 있으며, 용지 방향과 인쇄 매수만을 사용자가 선택할 수 있다.

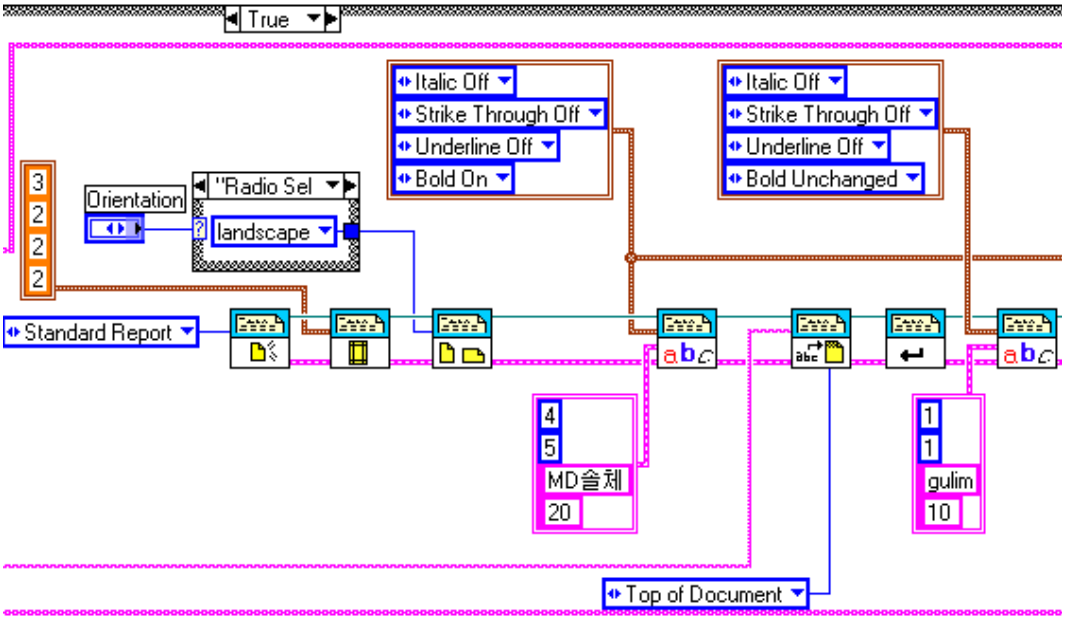


Fig. 3.22 Print block diagram

⑤ PRINT SCREEN

Fig. 3.23은 **PRINT SCREEN**을 실행시키기 위해 구성된 블록 다이어그램으로서 현재 프런트 패널 상에 보이는 화면을 그대로 인쇄한다.

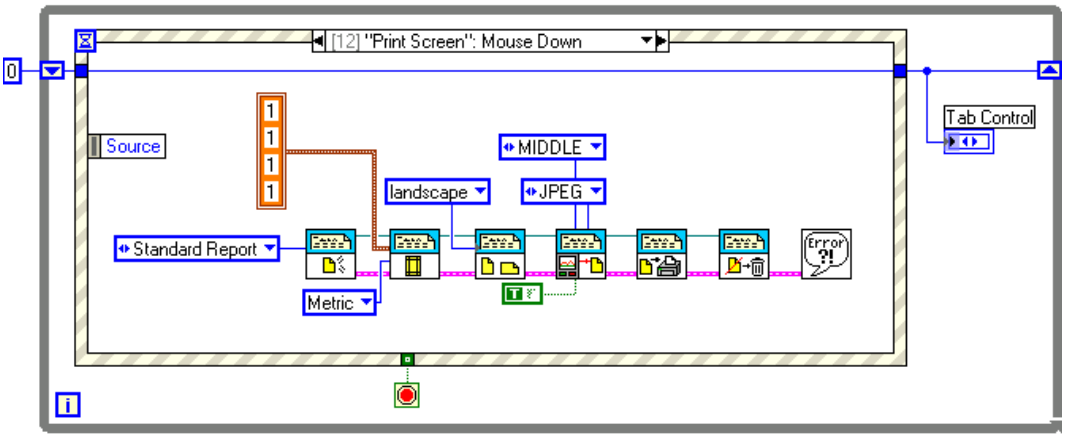


Fig. 3.23 Print screen block diagram

⑥ EXIT

프로그램을 종료한다.

제 4 장 프로그램 실행 및 검토

프로그램 시뮬레이션을 위해 한국해양대학교 신조 실습선인 한바다호의 센서값을 사용하여야 하나, 본 논문에서는 원자력발전소의 비상디젤발전기 시운전 결과로 MySQL 데이터베이스에 저장된 데이터의 일부를 엑셀 파일로 변환한 후 데이터를 한 줄씩 받아들여 프로그램의 시뮬레이션 데이터로 활용하였다.

Fig. 4.1 Performance flow

Fig. 4.1과 같은 실행 흐름도에 따라 프로그램이 작동하는지의 여부와 각 실행 단계에서 구성된 알고리즘에 따라 정상적으로 작동하는지를 확인하였다.

구성한 프로그램을 실행하고 아래 사항을 확인하였다.

1) 아날로그 데이터를 사용자가 지정한 경보 레벨과 비교하여 경보 레벨을 초과할 시 경보가 발생함.
2) 경보가 발생하거나 소거되었을 때 현재 경보리스트가 정상적으로 갱신됨.
3) 경보에 대한 현재 센서의 상태값과 구축된 진단규칙을 바탕으로 추론을 시작하여 진단결과를 감시부에 전송하고 감시부에서는 이 결과를 고장진단 결과창에 표시함.
4) 각 버튼에 지정된 이벤트가 정상적으로 작동함.
5) 사용자가 요구할 시 진단에 관한 상세한 설명이 진단창 아래에 위치한 진단설명창에 나타남.
6) 각 계통 화면에서 디지털 센서에 알람이 발생하였을 때 발생한 알람이 표시됨.
7) 아날로그 센서값이 화면상에 정상적으로 표시가 되며 알람이 발생했을 때 센서값이 붉은색으로 변함.
8) 사용자가 아날로그 센서의 경보 발생 레벨을 수정하거나 진단을 추가, 삭제, 수정할 경우 실시간으로 프로그램 상에 반영됨.

확인 결과 **Fig. 4.2**와 같이 정상적으로 작동함을 확인할 수 있었다.

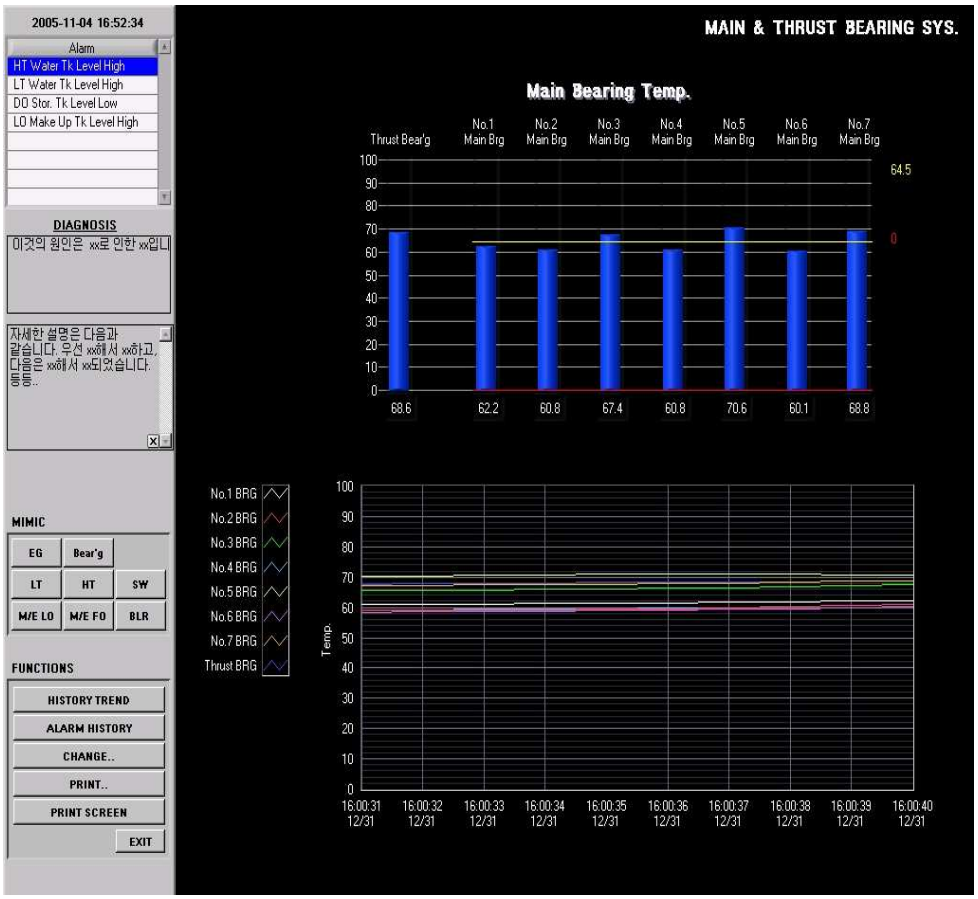


Fig. 4.2 Operation of system

제 5 장 결 론

본 논문에서는 LabVIEW를 사용하여 산업 현장에서 사용되고 있는 감시 시스템과 고장진단 시스템을 교육용 감시 진단 시스템으로 구축하는 문제를 다루었다. 복잡한 기관 시스템을 숙지하지 못하고 있는 선박 기관사가 감시 및 진단 시스템을 통해 기기간의 상호관련성을 바탕으로 전체 시스템을 이해하는데 유용하게 사용할 수 있다.

프로그램 구현 결과는 아래와 같다.

(1) 사용자가 프로그램 상에서 이미 구성되어 있던 진단규칙을 수정하거나 삭제하고 원하는 진단규칙을 추가할 수 있도록 구성하여 진단에 탄력성을 부여하였다. 또한 진단에 관한 상세한 설명을 수정, 삭제, 추가할 수 있게 하여 사용자의 수준에 맞도록 직접 구성이 가능하다.

(2) 고장진단은 계측된 데이터가 정상치에서 벗어날 경우에만 실행되며 지식베이스에서 인과관계를 통한 추론으로 이루어진다. 이러한 데이터 분석과 진단을 위하여 폴링 방식을 사용하여 계속적으로 감시하지 않고 이벤트 구동방식의 구조로 블록 다이어그램을 구성하여 CPU와 메모리의 사용량을 줄였다.

(3) 계측된 데이터를 화면상에 실시간으로 표시하며, 정상 데이터는 그대로 표현하지만 이상 데이터는 경보와 함께 고장진단을 실시한 결과를 동시에 표시하도록 구성하였다. 경보 발생 시 해당 센서가 위치한 계통 화면상에 깜빡이는 적색의 글자로 표시된다.

(4) 추후 프로그램의 확장, 수정을 용이하게 하기 위해 각부를 모듈화하였다. 특정 기능을 수행하는 부분을 하부 VI로 구성하여 수정 및 디버깅이 쉽도록 구성하였다. 또한 기능이 중복되는 부분을 하부 VI로 구성하고 여러 곳에서 사용하여 프로그램의 용량을 줄였다.

본 프로그램을 교육용 감시 및 진단시스템으로 상용화시키기 위해서는 아래와 같은 수정 및 보완이 필요하며, 이를 향후 연구 과제로 남겨둔다.

(1) 편의상 중요하다고 판단되는 8개의 계통만을 구현하였으며 센서의 상태값만을 진단에 활용하였다. 정밀한 진단을 위해서는 센서의 상태뿐 아니라 경향을 자세하게 분석하여 증상까지도 추론하여 정밀한 진단에 사용하는 것이 필요하다.

(2) 비용 문제로 LabVIEW 자체 기능을 사용하여 경보 내역과 아날로그 센서값에 대한 데이터베이스를 구축하였으나, 데이터베이스 툴킷 등을 사용하여 프로그램을 효율적으로 구현할 필요가 있다.

(3) 각 기기 및 센서간의 관계를 규정하고, 그에 관한 설명을 추가, 수정, 삭제할 수 있도록 하여 사용자가 복잡한 선박 시스템을 체계적으로 이해할 수 있도록 구성하는 것이 필요하다.

참 고 문 헌

[1] 이재규 외 5명 공저, ≪전문가시스템 원리와 개발≫, 법영사, 1996

[2] 모경주 외 3명, 「화학 공정의 이상진단을 위한 조업지원 시스템의 개발」, 전문가시스템학회지, 제 2권, 제 1호, pp.11~26, 1996

[3] 정학영, 박현신, 「경보처리 기반 진단시스템 개발, 전문가시스템학회지」, 제 4권, 제 1호, pp.103~113, 1998

[4] R. Khosla and T. Dillon, Enabling Technology for Diagnostic Applications, (Industrial and Engineering Application of Artificial Intelligent and Expert Systems) : Proceedings of the Eight International conference on held in Melbourne, Australia, pp.263~272, 1995

[5] 박종일, ≪비상발전기용 디젤엔진을 위한 진단기술에 관한 연구≫, 한국해양대학교 대학원 석사학위논문, 2005

[6] 곽두영, ≪LabVIEW Express(고급)≫, Ohm사, pp.3~6, 2003

[7] The measurement and automation catalog 2004, National Instruments, p.64, 2004

[8] Piping system diagram in E/R, STX B5001, 2004

[9] 박홍복, ≪LabVIEW 7.0 입문≫, 정익사, pp.126~127, 2004

[10] http://www.ni.com/korea, Support > KnowledgeBase, Support > Troubleshooting

[11] http://www.mylv.net, LabVIEW 아카데미

[12] STX, “건조사양서 B5001”, 2002

[14] National Instruments Korea, ≪Data Acquisition & Signal Conditioning≫, 2002

[15] 조권회, ≪비상디젤발전기계통 성능 및 고장 분석기술 개발≫, 한국원자력안전기술원, 2004