碩 士 學 位 論 文

하이브리드 DVR을 위한 소프트웨어 플랫폼

-DVR 시스템에 적용할 통합 소프트웨어 플랫폼-

2011年

漢 城 大 學 校 大 學 院

工 學 科

工 學 專 攻

趙 儁 來

碩 士 學 位 論 文
指導敎授 李珉秭

하이브리드 DVR을 위한 소프트웨어 플랫폼

-DVR 시스템에 적용할 통합 소프트웨어 플랫폼-

Software Platform for Hybrid DVR

-Integrate Software Platform Apply to the DVR System-

2010年 12月 日

漢城大學校 大學院

工 學 科

工學專攻

趙 儁 來

碩 士 學 位 論 文
指導敎授 李珉秭

하이브리드 DVR을 위한 소프트웨어 플랫폼

-DVR 시스템에 적용할 통합 소프트웨어 플랫폼-

Software Platform for Hybrid DVR

-Integrate Software Platform Apply to the DVR System-

위 論文을 컴퓨터 工學 碩士學位 論文으로 提出함

2010年 12月 日

漢城大學校 大學院

工 學 科

工學專攻

趙 儁 來

趙僑來의 工學 碩士學位論文을 認准함

2010年 12月 日

審査委員長 _____ 印

審 査 委 員 _____印

審 査 委 員 _____印

목 차

【 표 목 차 】

【 그 림 목 차 】

제 1 장 서 론

세계적으로 보안 영상 시장이 기존 아날로그 카메라와 직접 연결하는 방식의 DVR 중심 시장에서 IP 카메라와 NVR(Networked Video Recorder), CMS(Centralized Monitoring Station 또는 Cental Monitoring System) 등으로 구성된 IP 보안 감시 시장으로 바뀌어 가고 있다. 이 전이 과정에서 16/32 채널의 HD급 또는 D1급 카메라와 IP 카메라, 다른 DVR 등 네트워크에 연결된 다른 장치를 동시에 접속할 수 있는 고기능 하이브리드 방식 DVR의 수요가 급증하고 있다. 하이브리드 방식 DVR은 그 기능면에서 매우 다양한 장치, 영상 압축 기술, 접속 방법을 수용해야 하기 때문에, 플랫폼 기반의 소프트웨어 설계가 필수적이다.

DVR의 네트워크화를 위해서 기존 독립형 DVR이 갖지 못했던 고해상도 GUI 환경에서 동작하는 CMS 기능들과 다양한 소프트웨어, 하드웨어 모듈들을 유연하게 지원해야 하는 소프트웨어 플랫폼을 개발하는 것이다.

본 논문은 2장에서 DVR의 연구 배경을 설명하고, 3장에서 하이브리드 DVR의 소프트웨어 플랫폼 구조를 기술하고, 어떻게 다양한 하드웨어 및 소프트웨어 플랫폼을 높은 생산성으로 수용할 수 있는지, 또 사용자 인터페이스를 어떻게 자동으로 생성할 수 있는지를 설명한다. 4장에서는 설계한 소프트웨어 플랫폼을 바탕으로 구현한 시스템의 구조와 GUI, 사용된 라이브러리, 개발환경을 설명한다. 5장에서는 구현된 시스템의 결과에 대해설명하고 6장에서는 결론과 향후 연구에 대해 설명한다.

제 2 장 연구 배경

DVR (Digital Video Recorder)

DVR은 카메라에 잡히는 영상을 비디오테이프 없이 디지털화시켜 하드디스크(HDD)에 압축, 저장하는 차세대 영상 저장장치다. 아날로그 시스템에서는 별도로 있어야 하는 화면을 바꿔주는 스위치, 화면 분할기, VCR(비디오 재생기), 센서 및 알람 제어기 등을 하나로 통합한 시스템이다. DVR은 영상압축 알고리즘인 H.263, MPEG, MJPEG 등을 기반으로 한 기술로 동화상을 압축하고 저장한다.

DVR은 디지털 이미지로 변환되어 녹화된 영상을 반영구적으로 HDD에 저장하는 기능과 사용자가 녹화된 데이터를 순간검색 할 수 있는 기능, 여러 대의 카메라 영상을 1대의 모니터에서 분할하여 감시할 수 있는 멀티플렉서기능(Multiplexer), 먼 거리의 원격지에서도 모뎀이나 LAN, xDSL 등에서 녹화검색 및 실시간 화면을 감시할 수 있는 화상전송 기능이 내장되어 있다.

동영상을 디지털 신호로 바꿔 저장하고 전송하는 DVR는 90년대 중반부터 기존 CCTV의 단점을 개선한 틈새 상품으로 출발했다.

한국에서 지난 1997년 DVR 상업화에 성공했으며 한국 제조업체들이 세계 시장의 80% 이상을 차지하고 있다.

현재 보안장비 시장에서 각광받고 있는 DVR은 영상회의 디지털 방송 차세대 VCR인 개인 비디오 녹화기(PVR) IMT-2000(차세대 이동통신)의 영상 커뮤니케이션 등에도 다양하게 활용될 수 있다.

아날로그 영상 감시 장치인 CCTV를 대체하는 디지털 방식의 영상 감시 장비, CCTV에 비해 화질이 뛰어난 점외에도 컴퓨터의 하드디스크를 저장 매체로 사용하기 때문에 녹화테이프를 교체할 필요가 없고 인터넷을 통한 실시간 영상 전송 및 원격지 감시 기능이 있어 네트워크로 통합화하고 있는 정부 및 기관, 기업체들에게 가장 적절한 영상 감시 시스템으로 평가 받고 있다. DVR는 전 세계적으로 연간 3조 5000억원 규모에 달하는 CCTV 시장을 향후 5년 내에 대체할 것으로 전망되고 있으며 장기적으로 홈 시큐리티 등에 응용돼 개인시장으로도 확대될 가능성이 높아 성장 잠재력이 매우 높은 분야다.

NVR (network video recorder)

NVR이란 DVR처럼 전용 녹화기를 의미하며 최근에 IP 카메라의 수요가 증가하면서 NVR(녹화장치)에 대한 관심도 높아졌다. NVR은 PC처럼 자체적으로 IP를 설정할 수 있으므로 공유기와 같은 라우터에 연결한 후 자체 주소를 생성하고 추후에 NVR에 접속하려면 인터넷 주소창에 생성한 주소를 입력하기만 하면 된다. 주소를 입력하면 LIVE VIEW(라이브 뷰)와 녹화된 파일 보기(PLAY BACK)으로 들어갈 수 있다.

네트워크상에 설치된 카메라나 비디오 서버의 영상녹화, 모니터링, 이벤트 관리, 재생 등을 위한 전용 PC 서버, IP 카메라를 통해 디지털 영상을 전송받아 압축 저장하는 기능으로, IP 전용 저장 장치이기 때문에 아날로그를 디지털로 변환하는 장치가 필요 없고 기존 디지털 비디오 녹화기(DVR)를 대체하는 장치이다.

네트워크 카메라는 기본적으로 웹서버를 내장하고 있기 때문에 프로그램을 별도로 설치할 필요 없이 웹브라우저 즉 인터넷 익스플로러에 카메라 주소를 입력하기만 하면 볼 수 있다. 그리고 녹화를 하려면 녹화 소프트웨어를 PC 또는 서버에 설치하면 그 설치한 PC 또는 서버가 CCTV의 DVR처럼 녹화기의 역할을 하게 된다. 24시간 녹화를 위한 전용 녹화 장치가 필요한 경우 PC보다는 서버를 선택해야 하는데 서버 + 녹화 소프트웨어의 가격이 비싸기 때문에 일반 [1]SOHO 유저의 경우 어려움을 겪고 있었다. 그러나 NVR 제품을 사용하면 이러한 문제를 해결할 수 있다.

IP Camera

유무선 인터넷에 연결하여 사용하는 카메라, 카메라 모듈, 디코더, 영상 압축 칩, CPU, 네트워크 전송 칩 등으로 구성된다. 카메라 모듈로부터 받은 아날로그 신호는 디코더를 통해 디지털로 바뀌고, 압축 칩에서 압축을 거쳐 전송되는 흐름이다. IP 카메라는 외부에서도 집 안 상황을 휴대폰으로 점검할 수 있으며, 사용자 제작 콘텐

---

1) 작은 사무실 혹은 가정을 사무실로 이용한다는 개념. 본래 SOHO는 마케팅 용어로, 급격하게 발전하는 정보통신과 사무용품 시장에서 특정 소비자층을 지칭하는 의미로 사용하기 시작, 보다 새로운 미래의 사업 형태를 지칭하는 용어로 의미가 확대됐다. SOHO가 갑작스럽게 관심의 대상이 된 이유는 인터넷의 폭발적인 인기 때문이다. 1990년대 초반 웹이 등장한 이래 현실 세계에 있던 수많은 분야들이 인터넷이라는 가상공간에 자리를 잡아, 안방 컴퓨터 앞에 앉아서도 전세계를 방문하는 것은 물론 별다른 자본 없이도 원하는 사업을 할 수 있다는 가능성을 보여주었다. 또한 기업의 감원은 결과적으로 많은 소규모 사업자와 재택 근무자를 낳게 되고 기업으로서도 직원에게 들어가는 각종 연금과 복리 비용을 절감할 수 있어, SOHO의 수요는 갈수록 증가하고 있다

츠(UCC)와 홈 네트워크를 위한 도구로 사용된다.

제 3 장 소프트웨어 플랫폼 설계

본 논문에서는 하이브리드 DVR을 위한 소프트웨어 플랫폼의 구성을 위하여, 시장에 출시된 DVR, CMS, VNR, 네트워크 카메라, DVR용 반도체 등의 기능, 제품의 특징을 분석하고, 기능적으로는 라이브 뷰, 레코딩, 검색, 재생, 설정, 이벤트 처리, 네트워크 접속이라는 큰 그룹으로 나누고 각 그룹에서 세부적으로 어떤 기능을 제공하는지 조사하였다. 조사한 여러 제품들을 비교하여 많은 제품에서 동일하게 나타나는 일반적인 기능들과 몇 종류의 제품들에서 나타나는 특수한 기능으로 분류하여 정리하여 소프트웨어 플랫폼을 설계하는 기반으로 사용하였다. 또한 이 기능 분석 결과는 기본적으로 사용자의 편의성을 극대화 시킬 수 있는 GUI를 설계하는데 이용될 수 있다.

제 1 절 프레임워크 컨셉

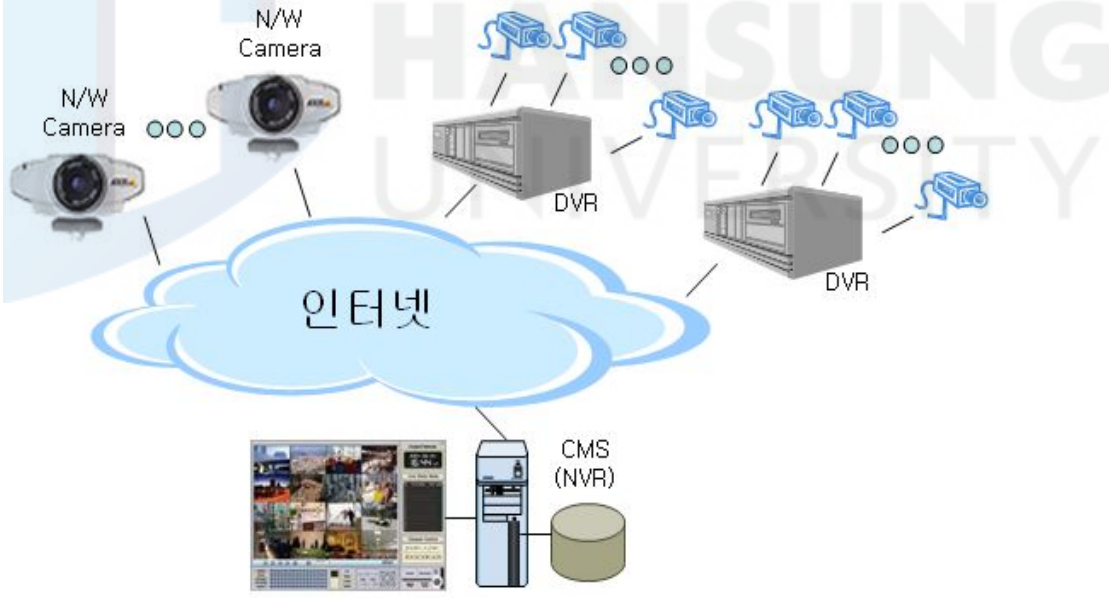

그림 3-1 프레임 워크 컨셉

그림 3-1과 같이 개발 대상 프레임 워크는 CMS(NVR)상에서 실행되는 소프트웨어를 의미하고 로컬카메라, 네트워크 카메라, 원격 DVR들을 통합 관리하며, 모든 시스템으로부터 영상, 오디오, 센서, 접근 기록 등 보안 데이터를 실시간 확인, 저장, 검색할 수 있는 것을 컨셉을 한다.

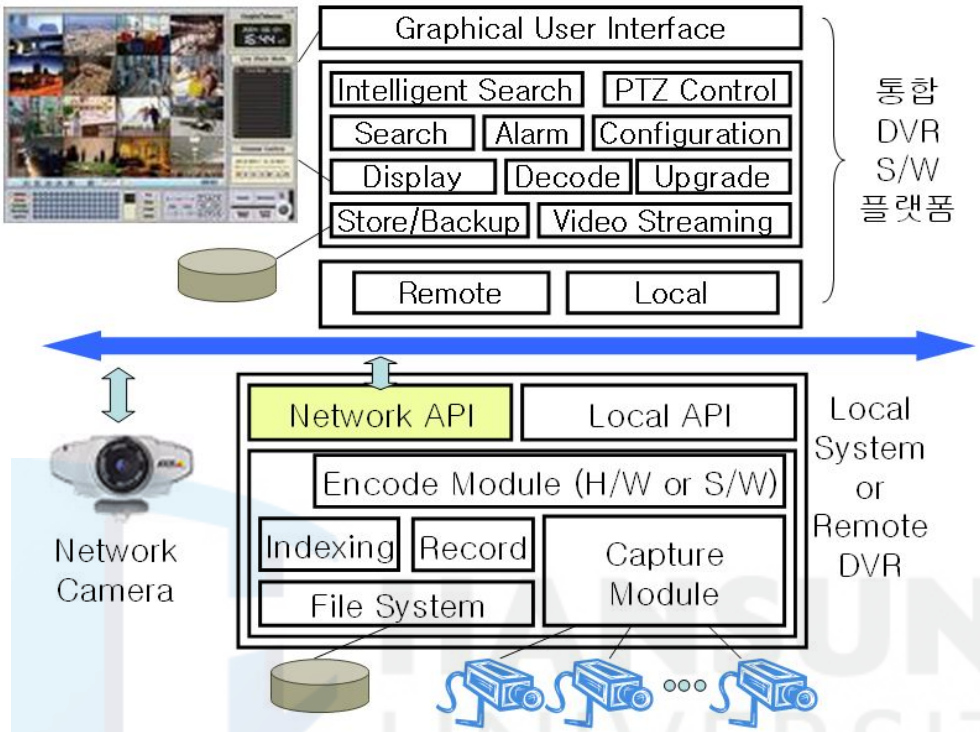

그림 3-2 네트워크 기반 DVR 통합제어 구조

그림 3-2와 같이 네트워크 기반 DVR 통합제어 소프트웨어 플랫폼을 개발하는 것으로서, 기본적으로 DVR의 기능을 수행하고, 네트워크 카메라, 원격지 DVR을 관리하는 기능을 가지는 소프트웨어 플랫폼이다. 플랫폼으로서의 개발 목표는 각 소프트웨어 모듈의 API를 공고히 하여, 다양하게 변하는 시스템 환경, 새로운 기술의 도입과 전체 시스템의 확장성에 유연하게 대처하는 소프트웨어 구조를 만드는 것이다.

제 2 절 하이브리드 DVR 소프트웨어 플랫폼

개발 대상 DVR 소프트웨어 플랫폼은 앞서 기술한 하이브리드 DVR이나 CMS 상에서 실행되는 소프트웨어를 의미한다. 본 논문에서는 그림 3-1과 같이 소프트웨어 플랫폼이 로컬 카메라, 네트워크 카메라, 원격 DVR을 통합 관리, 제어하며, 다양한 영상/오디오 코덱, 스트리밍, 파일 시스템, 센서 및 접근 기록 등 보안 데이터, 비디오 분석을 통한 지능형 검색, DB 기반 검색, 네트워크 운용, 자동 업그레이드 등 모든 필요한 기능을 정현한 계층 구조와 API의 정의를 통해 연결될 수 있도록 설계하였다.

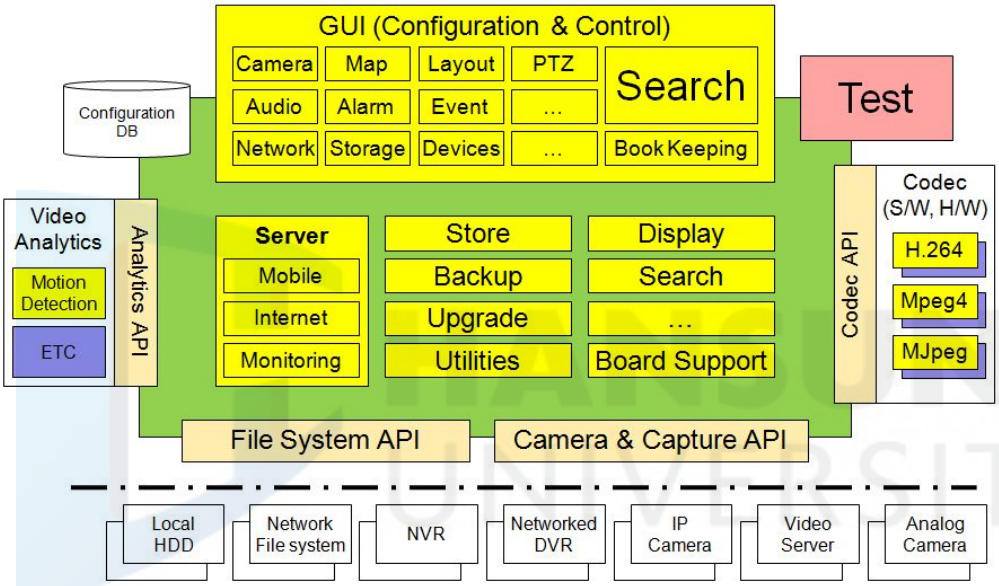

그림 3-3 하이브리드 DVR의 소프트웨어

그림 3-3에서 카메라, 캡춰, API 부분만을 구체적으로 보면 그림 3-4와 같다. 그림에서는 캡춰보드를 이용하는 아날로그 카메라, 코덱 H/W, IP 카메라, 비디오 서버, 네트워크에 연결된 DVR까지 모두 하나의 API를 통해 접근이 가능한 것을 볼 수 있다.

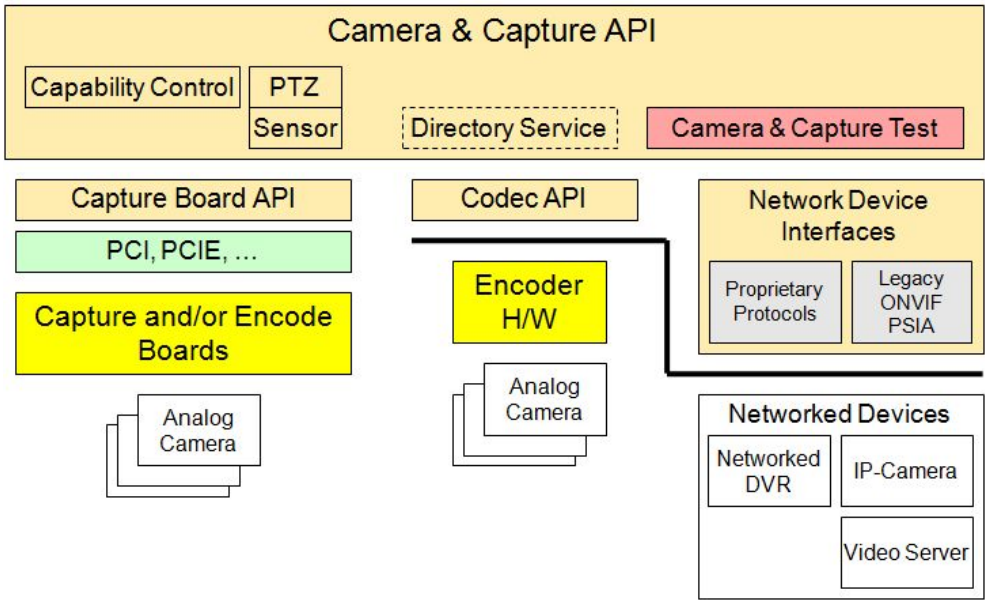

그림 3-4 카메라 및 캡춰 API

설계에 적용되는 모든 하드웨어, 소프트웨어 기능 모듈은 그림 3-5에서 처럼 Device Definition Package로 기능을 정의한 Capability와 표준 API를 따르는 Code, 설정과 제어에 추가적으로 필요한 GUI로 구성되어 있다.

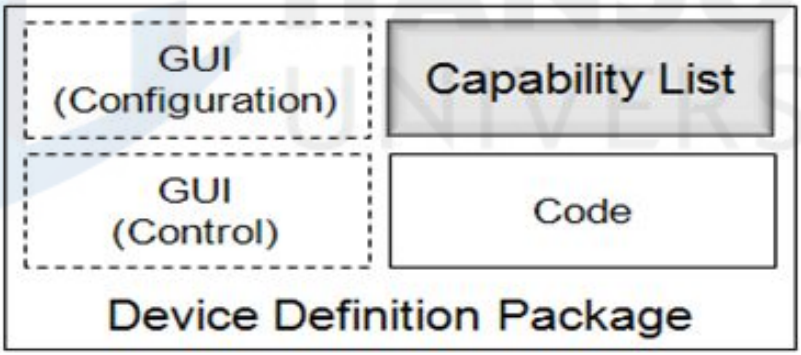

그림 3-5 컴포넌트 구조

또한 원격에 있는 카메라 또는 DVR과 같은 복합 기능 장치들과 같은 컴포넌트를 SOAP/RTSP 프로토콜을 거쳐서 같은 방식으로 이용할 수 있기 때문에, 같은 응용 프로그램으로 로컬 및 네트워크 운용을 동시에 수행할

수 있다.

제 3 절 GUI 자동 생성기

본 논문에서는 다양한 사용자의 편의성 요구와 새로운 프로토콜, 하드웨어 등을 추가적인 코딩 없이 기능 기술만으로 설정하고, 제어하기 위한 GUI 자동 생성기를 같이 설계하였다.

최근들어 ONVIF나 PSIA 등과 같은 IP 카메라 표준이 만들어지고 있으나, CMS 소프트웨어의 가장 중대한 어려움은 수 많은 DVR, IP 카메라, 비디오 서버들이 각자 나름대로의 기능과 프로토콜, API를 사용한다는 것이다. 이 과정은 소프트웨어적인 인터페이스 뿐만 아니라, 기능의 설정, 제어를 위한 GUI 구현에도 상당히 많은 비용이 들어간다는 것을 의미한다. 따라서 이 논문에서는 그림3-6과 같이 각 기능 모듈의 시본 사용자 인터페이스에서 벗어나는 부분의 GUI를 컴포넌트 Capability로부터 자동으로 생성하여 코딩 없이 지원할 수 있는 방안을 제공하고 있다.

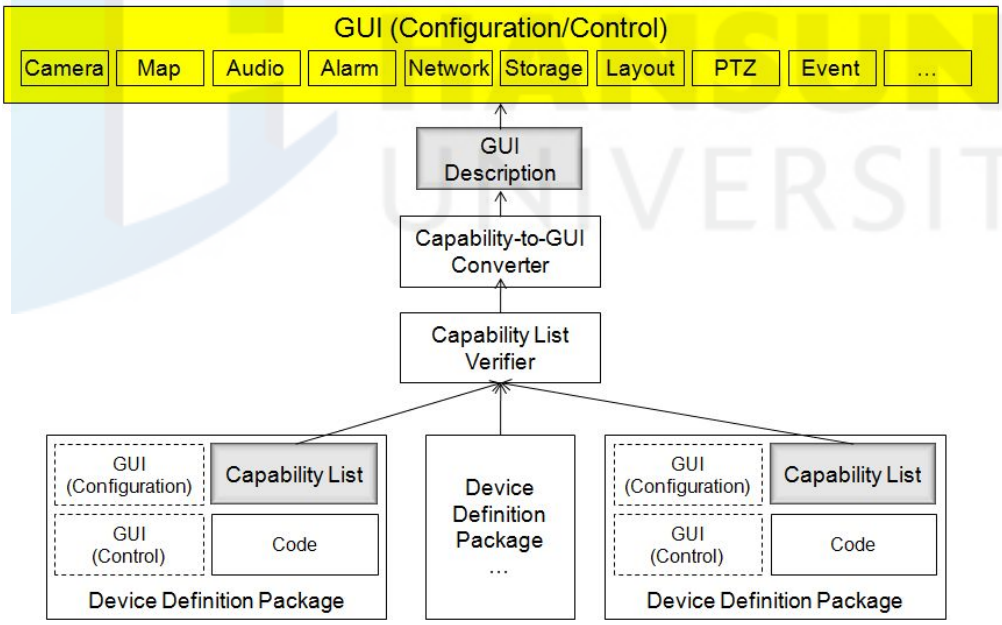

그림 3-6 GUI 자동생성기 구조

제 4 장 구 현

제 1 절 개발 환경

소프트웨어를 개발한 실행환경은 표 4-1과 같다.

	개발 환경	실행 환경
하드웨어	인텔 코어 2 듀오	인텔 코어 2 듀오
운영체제	페도라 8	페도라 8
에디터	Eclipse CDT	
컴파일러	gcc	
디버거	gdb	
개발언어	C++	

표 4-1 개발 및 실행 환경

개발 및 실행환경은 동일한 환경으로 구성되었고 운영체제는 페도라 8을 사용하였고 그 위에서 Eclipse CDT(Ver3.5) 에디터를 이용하였으며 개발언어는 C++을 사용하였고 컴파일 및 디버깅을 위해 GNU tool인 gcc와 gdb(페도라 8 제공)를 Eclipse를 이용하여 사용하였다.

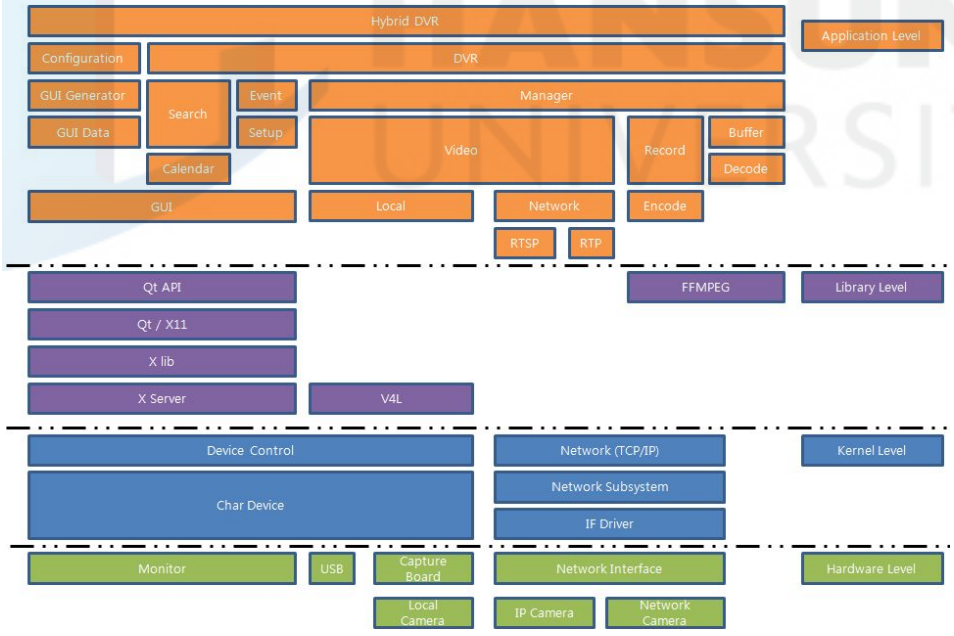

그림 4-1 전체 소프트웨어 구조

그,림4-1은 본 논문에서 구현한 소프트웨어의 전체적인 구조이다. V4L 라이브러리는 페도라 8에서 제공되고 Qt는 Version4.6.0를 설치하고 이클립스와 연동하기 위해 qt-eclipse-integration-linuxx86.1.6.0를 설치한다. ffmpeg은 yum을 이용하여 최신 버전을 설치한다.

제 2 절 라이브러리

소프트웨어 개발을 위해서 필요한 라이브러리는 Qt, V4L, FFmpeg 등이 있다.

1. Qt

Qt는 GUI 프로그램 개발을 위한 크로스 플랫폼 위젯 툴킷이다. 주로 C++를 사용하며, 파이썬, 루비, C, 펄, 파스칼의 바인딩을 제공한다. 많은 플랫폼에서 동작하며, 상당히 좋은 국제화를 지원한다. SQL 데이터베이스 접근, XML 처리, 스레드 관리, 단일 크로스 플랫폼 파일 관리 API를 제공한다.

Qt는 다음과 같은 플랫폼으로 개발된다. Linux/X11 - X윈도 시스템(유닉스/리눅스), Mac OS X - 맥 오에스 텐, Windows - 마이크로소프트 윈도, Embedded Linux - PDA, 스마트폰 등 임베디드 플랫폼, Windows CE - 윈도 CE, Symbian - 신비안, Maemo - maemo의 플랫폼으로 개발할 수 있다.

여기서는 Qt4.0을 사용하여 Linux/X11 플랫폼에서 개발하였다. Qt 4.0에서 제공하는 클래스는 다음과 같다. QtCore - 핵심 클래스, QtGui - 그래픽 사용자 인터페이스 구성요소, QtNetwork - 네트워크 구성요소, QtOpenGL - OpenGL 구성요소, QtSql - SQL 데이터베이스 구성요서, QtSvg - SVG 그림 구성요소, QtXml - XML 파서 구성요소, QtDesigner - Qt 디자이너를 위한 클래스, QtUiTools - Qt 디자이너 폼을 다루는 클래스, QtAssistant - 온라인 도움말을 위한 클래스, Qt3Support - Qt 3 호환성 클래스, QtTest - 프로그램 테스트를 위한 클래스이다. 또한 Qt 라이브러리를 사용하는데 도움을 주는 도구는 다음과 같다. Qt-Designer - Qt 프로그램의 폼을 해준다. Qt-Assistant - 온라인 문서를 제공한다. Qt-Linguist, update and release - 프로그램의 지역화를 도와준다. qmake - Qt에서 제공하는 라

이브러리와 개발한 언어의 컴파일을 쉽게 해준다. moc - Qt 메타 정보 컴파일러, uic - Qt 폼 컴파일러(*.ui -> *.h), rcc - Qt 리소스 컴파일러가 있다.

Qt는 GUI의 완전한 추상화가 가능하다. 자체 페인팅 엔진과 컨트롤을 사용하며, 실행되는 플랫폼의 모습을 최대한 따라한다. Qt는 플랫폼에 의존적인 코드를 거의 사용하지 않았으므로 서로 다른 플랫폼으로 이식하는 것이 쉽다. 하지만 이 방법은 서로 다른 플랫폼의 모습을 정교하게 따라하도록 해야 하는 것이다. 서로 다른 플랫폼의 자체 API를 사용해서 Qt 컨트롤을 제어한다.

약칭 “moc”로 알려져 있는 이 프로그램은 Qt에서 사용되고 있는 비표준적인 메타 데이터들을 처리하기 위한 도구이다. 예로 Qt에서 사용하고 있는 이벤트 처리방식인 시그널-슬롯 방식은 표준 C++ 처리방식이 아니다.

우리는 Qt 라이브러리를 이용해 플랫폼에 무관한 소프트웨어의 구현을 목표로 하였고, Qt 라이브러리에서 제공하는 GUI를 생성하고 제어하는 클래스를 이용하여 프로그램을 작성하였다.

Qt 라이브러리를 사용하기 위해서 페도라에 라이브러리를 다음과 같이 설치하고 Eclipse를 이용하기 위한 패키지도 설치한다.

Qt 라이브러리를 제공하는 www.trolltech.com에서 라이브러리를 다운받아 설치한다. 설치 방법은 다운받은 파일의 압축을 풀고 configure한 뒤 make하여 설치한다. Eclipse에서 Qt를 사용하기 위한 패키지인 intergration을 다운받아 압축을 풀어 Eclipse에 덮어 쓰고 Eclipse를 업데이트하면 Eclipse에서 Qt 라이브러리 패스를 설정하여 Qt 라이브러리를 사용할 수 있다.

2. V4L

V4L은 Video4Linux의 약자로 리눅스에서 비디오 디바이스를 제어하고 사용하기 위한 API이다. 비디오 디바이스는 튜너를 포함한 비디오 카드나 웹캠과 같이 영상 정보를 입력할 수 있는 장치를 뜻하며, 튜너를 포함한 디바이스의 경우 텔레비전이나 AM/FM 라디오, 텔레텍스트(Teletext)등의 방송을 시청/청취할 수 있다. V4L은 리눅스 커널에 기본적으로 포함되어 있으며, 커널 컴파일을 통해 커널 모듈, 혹은 정적으로 컴파일하여 사용이 가능하다. 각 디바이스를 위해 표4-2와 같은 파일을 제공하고 있으며, 이를 이용하기 위해서는 이 디바이스 파일을 열고 검색하여 지원하

는 기능을 찾아서 API를 이용할 수 있다.

V4L은 페도라에서 기본적으로 제공해준다. 프로그램을 작성할 때 필요한 헤더를 추가하여 API를 사용하면 된다.

디바이스 기능	디바이스 이름	마이너 범위
비디오 캡쳐 인터페이스	/dev/video	0 - 63
AM/FM 라디오 디바이스	/dev/radio	64 - 127
Teletest 인터페이스 칩	/dev/vtx	192 - 223
Raw VBI 데이터 (Intercast/teletext)	/dev/vbi	224 - 239

표 4-2 디바이스 기능 및 위치

3. FFmpeg

FFmpeg은 디지털 음성스트림과 영상스트림에 대해서 다양한 종류의 형태로 기록하고 변환하는 컴퓨터 프로그램이다. 명령어를 직접 입력하는 방식으로 동작하며 여러 가지 자유 소프트웨어와 오픈 소스 라이브러리로 구성되어 있다. 라이브러리 중에는 libavcodec도 들어있는데, 이 라이브러리는 음성/영상 코덱 라이브러리로 여러 프로젝트에서 쓰인다. 또 libavformat이라는 음성/영상 다중화, 역다중화 라이브러리도 있다. FFmpeg은 리눅스 기반 하에 개발 되었지만 Mac OS X, Microsoft Windows, AmigaOS 등 대부분의 운영체제에서 컴파일이 가능하다.

FFmpeg를 사용하기 위해서는 ffmpeg를 다운받아 configure한 뒤 make하여 설치하여 사용할 수 있다.

제 3 절 소프트웨어 구조

개발 환경에 필요한 라이브러리를 설치하고 다음과 같이 소프트웨어를 구성하였다.

그림 4-1은 2장에서 구성한 플랫폼에 해당하는 클래스를 보여준다. GUI 부분은 Reader, Parser, Description, MappingCData를 기본으로 사용하거나 virtual 클래스인 경우 상속을 통해서 구현하였다. Codec API 부분은 Encode, Decode를 virtual 클래스로 구현하였고 이를 상속받아 ffmpeg 라이브러리를 사용하는 클래스로 구현하였다. File System API 부분은 Input, Output을 virtual 클래스로 구현하였

고 이를 상속받아 로컬 드라이브에서 동작하는 클래스를 구현하였다. Camera & Capture API 부분은 Video를 virtual 클래스로 구현하였고 이를 상속받아 V4L 라이브러리를 사용하는 클래스를 구현하였다. 기본적으로 사용되는 Display, Record, Search, Calendar, Queue를 기본으로 사용하거나 virtual 클래스인 경우 상속을 통해서 구현하였다.

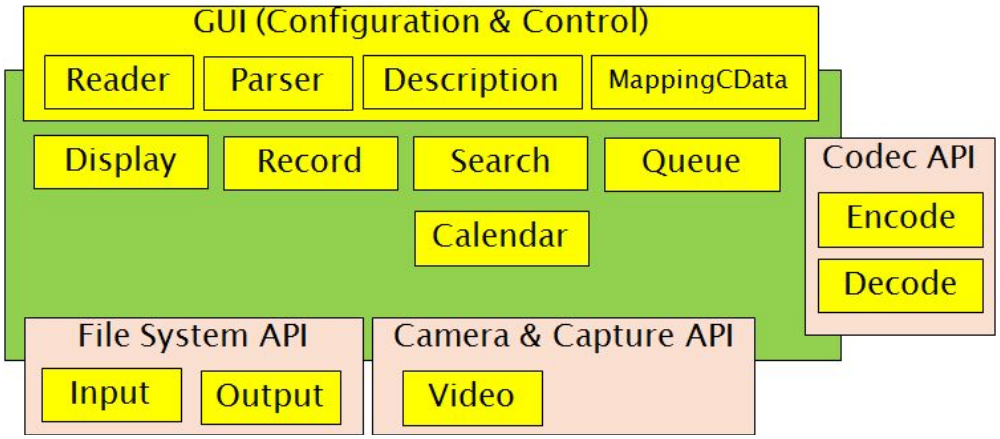

그림 4-2 각 플랫폼에 해당하는 클래스

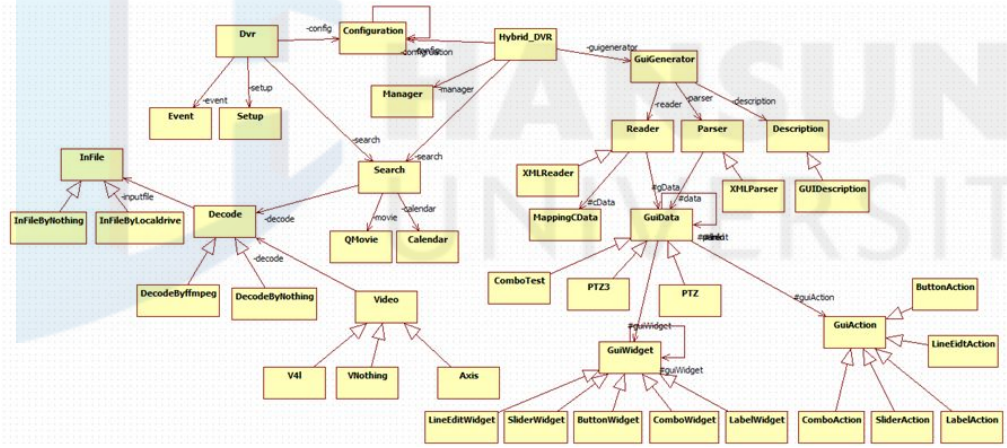

그림 4-3 구현된 전체 클래스 구조

그림 4-2은 구현된 전체 클래스의 모습으로 밑부분의 세세한 부분을 제외한 전체적인 구조를 볼 수 있는 클래스 다이어그램이다.

1. Codec API

Codec의 Encode, Decode는 소프트웨어 플랫폼의 Codec API 부분으로 설정할 때 CameraData 클래스에 저장된 정보를 이용한다.

그림 4-3는 영상을 Encode, Decode 하는 부분으로 ffmpeg 라이브러리를 이용하여 구현하였다. 실시간 영상을 Encode하여 저장 파일로 저장하는 기능과 저장된 파일을 Decode하여 저장된 영상을 출력하는 기능을 가진다. 다른 라이브러리를 사용하려면 virtual 클래스인 Encode, Decode를 상속받아 구현하면 된다.

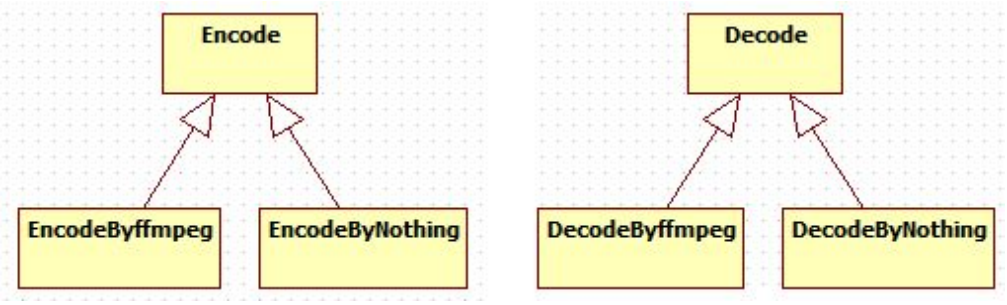

그림 4-4 Codec API 상속관계

Encode는 영상을 압축하여 영상을 저장 할 수 있게 하는 기능을 가진 클래스이다. 여러 가지 Encode 기법을 Encode 클래스를 상속받아 계속적으로 추가 할 수 있다.

virtual int init(int width, int height)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, encode의 구조를 설정하고 그 결과를 반환한다.
virtual int set(unsigned char *buf)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, encode의 buffer를 설정하고 그 결과를 반환한다.
virtual int close()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, encode를 종료하고 그 결과를 반환한다.

표 4-3 Encode Class

EncodeByffmpeg는 Encode를 상속받고 ffmpeg 라이브러리를 이용하여 인코딩하는 기능을 가진 클래스이다.

AVCodecContext *codec_context
ffmpeg의 구조체로 코덱 구조를 가지는 변수이다.
AVFrame *picture
ffmpeg의 구조체로 frame 정보를 가지는 변수이다.
uint8_t *outbuf
buffer의 정보를 가지는 변수이다.
int OUTBUF_SIZE
buffer의 크기 정보를 가지는 변수이다.
int width
encode의 width의 정보를 가지는 변수이다.
int height
encode의 width의 정보를 가지는 변수이다.
QMutex mutex
encode를 위한 mutex 변수이다.
OutFile *outFile
output을 위한 outFile 변수이다.
void setArea(int width, int height)
encode의 크기를 설정하는 함수이다.

표 4-4 EncodeByffmpeg Class

EncodeByNothing는 Encode를 상속받은 Encodebynoting은 아무 일도 하지 않는 클래스이다.

Decode는 영상의 압축을 풀어 영상을 재생 할 수 있게 하는 기능을 가진 클래스이다. 여러 가지 Decode 기법을 Decode 클래스를 상속받아 계속적으로 추가 할 수 있다.

virtual int init(char *)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, decode의 구조를 설정하고 그 결과를 반환한다.
virtual int init()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, live stream를 처리하기 위한 decode의 구조를 설정하고 그 결과를 반환한다.

virtual unsigned char* decoding(unsigned char*, int)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, live stream을 디코딩해주는 함수로 프레임 단위로 디코딩되며 디코딩된 영상을 반환한다.
virtual int updataDecodePos()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, decode의 위치를 설정하고 그 결과를 반환한다.
virtual void setStartTime()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 시작 시간을 설정하는 함수이다.

표 4-5 Decode Class

DecodeByffmpeg는 Decode 클래스를 상속받고 ffmpeg 라이브러리를 이용하여 디코딩하는 기능을 담당하는 클래스이다.

AVFormatContext *pFormatCtx
ffmpeg의 구조체로 format의 구조를 가지는 변수이다.
int vidoeStream
video stream의 번호를 가지는 변수이다.
AVCodecContext *pCodecCtx
ffmpeg의 구조체로 코덱 구조를 가지는 변수이다.
AVCodec *pCodec
ffmpeg의 구조체로 코덱 정보를 가지는 변수이다.
AVFrame *pFrame
ffmpeg의 구조체로 frame 정보를 가지는 변수이다.
AVPacket packet
ffmpeg의 구조체로 packet 정보를 가지는 변수이다.
int frameFinished
frame의 마지막 위치 정보를 가지는 변수이다.
float aspect_ratio
상의 비율 정보를 가지는 변수이다.
int frame_count
frame의 개수 정보를 가지는 변수이다.
QMutex mutex
decode를 사용하기 위한 mutex 변수이다.
PLAY_STATE state
ffmpeg의 Play 상태를 가지는 변수이다.

unsigned char *buffer
buffer의 정보를 가지는 변수이다.
InFile *inputfile
입력 파일의 정보를 가지는 변수이다.
void setImage(unsigned char *buf, int widte, int height)
디코딩된 데이터를 보내주는 함수이다.
unsigned char* picture2yuv(AVFrame *picture)
2yuv 형식으로 변환해 반환하는 함수이다.
void avReadFrameFlush()
frame을 읽어서 flush 하는 함수이다.
void flushPacketQueue()
Queue에 packet을 flush 하는 함수이다.
int decode()
디코딩을 시작하는 함수이다.
int decodeGet(unsigned char **data)
버퍼에 있는 데이터로 디코딩하는 함수이다.
void updateRangePos()
Range 위치를 업데이트 하는 함수이다.

표 4-6 DecodeByffmpeg Class

DeocdeByNothing은 Decode 클래스를 상속받은 아무 일도 하지 않는 클래스이다.

2. Camera & Capture API

소프트웨어 플랫폼의 Camera & Capture API 부분으로 설정할 때 camera 클래스에 저장된 정보를 이용하여 설정을 한다.

연결된 장치(local camera, usb camera, ip camera 등)을 제어하는 부분으로 V4L 라이브러리를 사용하여 구현되었고 영상을 캡쳐하여 실시간 영상으로 출력하는데 사용된다. 다른 라이브러리(ip camera, onvif camera 등)를 사용하려면 그림 4-4과 같이 virtual 클래스인 Video를 상속받아 구현하면 된다.

Video는 Qt 라이브러리의 QThread를 상속받고 Decode 클래스를 포함하고 영상을 제어하는 기능을 담당하는 클래스이다. 여러 가지 카메라나 다양한 압축 방식을 적용시킬 수 있도록 이 클래스를 상속받아 추가적으로 Video 기능을 담당하는 클래스를 생성할 수 있다.

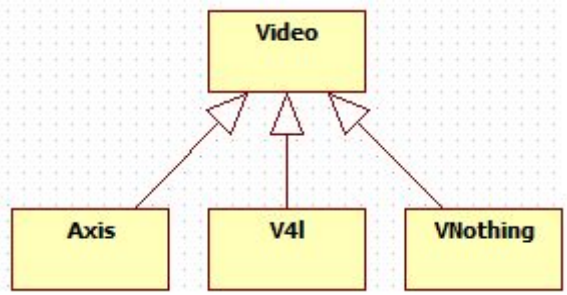

그림 4-5 Camera & Capture API 상속관계

virtual int init()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, video의 설정을 초기화 하는 함수이다.
virtual int capture()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, video의 이미지를 가져오는 함수이다.
virtual int setProperties()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, video 속성을 설정한다.
virtual unsigned char* getPicData()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, video의 YUV 이미지를 가져오는 함수이다.
virtual int getWidth()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, video 이미지의 가로 길이를 가져오는 함수이다.
virtual int getHeighet()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, video 이미지의 세로 길이를 가져오는 함수이다.

표 4-7 Vidleo Class

V4L은 Video를 상속받고 V4l 라이브러리를 이용하여 로컬 카메라를 제어하는 기능을 담당하는 클래스이다.

QString *device
장치에 대한 정보를 가지는 변수이다.
unsigned char *pic_data
pic 데이터의 정보를 가지는 변수이다.
unsigned char *rgb_data
rgb 데어터의 정보를 가지는 변수이다.
int depth
depth의 정보를 가지는 변수이다.
void *map
map의 정보를 가지는 변수이다.
char *image_buffer
image buffer의 정보를 가지는 변수이다.
int maxwidth, minwdith, maxheight, minheight
가능한 크기정보를 가지는 변수이다.
int width, height
실제 display 할 크기를 가지는 변수이다.
int norm
어떠한 형식인지 저장하는 변수이다.
int frame
frame의 크기를 저장하는 변수이다.
unsigned char *buf
buffer를 가지는 변수이다.
int video
video 번호를 가지는 변수이다.
struct video_capability cap
V4L의 struct 구조이다.
struct video_channel chan
V4L의 struct 구조이다.
struct video_picture pic
V4L의 struct 구조이다.
struct video_mbuf m_buf
V4L의 struct 구조이다.
struct video_mmap v_map
V4L의 struct 구조이다.
struct video_window win
V4L의 struct 구조이다.
int brightness
video의 밝기 정보를 가지는 변수이다.
int hue

video의 색조 정보를 가지는 변수이다.
int contrast
video의 대비 정보를 가지는 변수이다.
int fno
fno 정보를 가지는 변수이다.
int flag
flag 정보를 가지는 변수이다.
void outVideo(unsigned char *picData, int width, int height)
queue에 데이터를 보내는 함수이다.
int setVideoPicture()
video picture를 설정하는 함수이다.
int setVideoWindow()
video window를 설정하는 함수이다.
int setVideoChannel()
video channel을 설정하는 함수이다.
int dumpVideoCapability()
video capability를 dump하는 함수이다.
int dumpInformationOfChannels()
video channel 정보를 dump하는 함수이다.
int dumpVideoWindow()
video window를 dump하는 함수이다.
int dumpVideoPicture()
video picture를 dump하는 함수이다.

표 4-8 V4L Class

Axis는 Video를 상속받고 네트워크로 연결된 카메라를 제어하는 기능을 담당하는 클래스이다.

NetworkCamera *nCamera
networkCamera의 정보를 가지는 클래스로 AXIS Network Camera와 연결하기 위한 정보를 가지는 변수이다.
FrameData *framedata
네트워크 카메라의 한 프레임을 저장하기 위한 변수이다.
Decode *decode
networkCamera에 연결될 decode 변수이다.
int makeRtsp()

Rtsp를 연결하기 위해 Rtsp 명령어를 만드는 함수이다.
void rtspAddr(char *addr, int num, char *buffer)
Rtsp 연결을 위해 연결할 주소를 만드는 함수이다.
void rtspResolution(char *buf, char *res)
명령어에 해상도를 추가하는 함수이다.
void rtspFps(char *buf, int fps)
명령어에 프레임 수를 추가하는 함수이다.
void rtspBRate(char *buf, int brate)
명령어에 비디오 비트 비율을 추가하는 함수이다.
void rtspGSize(char *buf, int size)
명령어에 프레임 인터벌 크기를 추가하는 함수이다.
int connectRtspServer()
네트워크 카메라의 Rtsp를 연결하는 함수이다.
int creatRtpSocket()
네트워크 카메라의 Rtp를 생성하는 함수이다.
void EncodeBuffer(char *pInputBuffer, unsigend int pInputBufferLength, char *pOutputBufferString)
명령어를 암호화를 위한 버퍼링하는 부분이다.
void EncodeByteTriple(char *pInputBuffer, unsigend int InputCharacters, char *pOutputBufferString)
명령어를 암호화하는 함수이다.
unsigned int CalculateRecquiredEncodeOutputBufferSize (unsigned int pInputByteCount)
암호화된 명령어의 크기를 계산하는 함수이다.
int setDescribe()
Rtsp 명령어를 생성하여 네트워크 카메라의 정보를 설정하고 그 결과를 받는 함수이다.
int setOption()
Rtsp 명령어를 생성하여 네트워크 카메라에 옵션을 설정하고 그 결과를 받는 함수이다.
int setSetup()
Rtsp 명령어를 생성하여 네트워크 카메라의 셋업을 설정하고 그 결과를 받는 함수이다.
int setPlay()
Rtsp 명령어를 생성하여 네트워크 카메라에서 Rtp로 스트리밍을 보내라고 요청

하고 그 결과를 받는 함수이다.
int streamingTeardown()
Rtsp 명령어를 생성하여 네트워크 카메라와 연결을 해제하고 그 결과를 받는 함수이다.
int sendGetParam()
Rtsp 명령어를 생성하여 네트워크 카메라와 연결을 유지하고 그 결과를 받는 함수이다.
FrameData* capture()
Rtp 스트리밍으로 전송되는 영상스트리밍을 한 프레임씩 저장하는 함수이다.

표 4-9 Axis Class

VNothing은 Video를 상속받고 아무 일도 하지 않는 클래스이다.

3. File System API

소프트웨어 플랫폼의 FileSystem API 부분으로 설정할 때 camera 클래스에 정보가 저장되어 있다.

저장된 파일에 관한 정보를 가지는 InFile은 저장된 파일을 읽어서 영상을 출력할 수 있게 Decode와 연결이 되고 출력할 파일에 대한 정보를 가지는 OutFile은 실시간 영상이 Encode를 통해 압축된 데이터를 파일로 저장한다. 로컬 드리이브가 아닌 다른 형식의 저장 방법은 그림 4-5과 같이virtual 클래스인 InFile, OutFile를 상속받아 구현하면 된다.

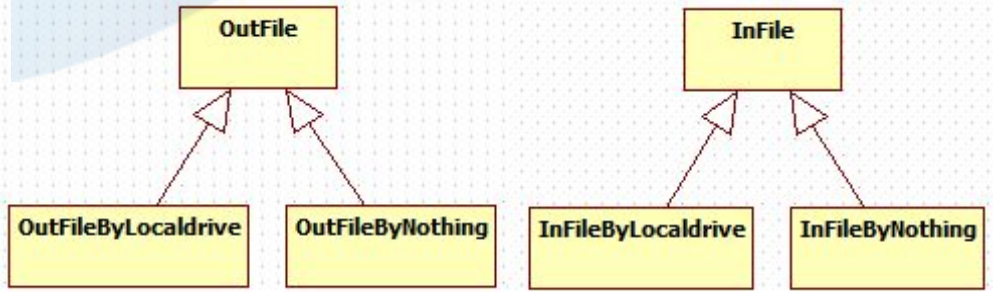

그림 4-6 File System API 상속관계

InFile은 프로그램에서 Input으로 읽어올 파일에 대한 정보를 가진다. Input File에

대한 위치 정보, 연결 정보 등에 대한 정보를 포함하는 클래스이다.

virtual int init()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, InFiel의 초기 설정을 하는 함수이다.
virtual int setFileInfor(char *chanPath)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 파일의 정보를 설정하는 함수이다.
virtual char* getFilename()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 파일의 이름을 반환하는 함수이다.
virtual int setStartTime(int hour, int minute, int sec)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 시작 시간을 설정하는 함수이다.
virtual int getPos()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 위치를 반환하는 함수이다.
virtual void setPos()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 위치를 설정하는 함수이다.
virtual int updateDecodePos()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 디코드 위치를 업데이트 하는 함수이다.
virtual int setNextPos(int *pos)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 다음 위치를 설정하는 함수이다.
virtual int setStartPos()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 시작 위치를 설정하는 함수이다.

표 4-10 InFile Class

Infilebylocaldrive은 InFile를 상속받고 로컬 드라이브에 있는 파일을 입력으로 받을 수 있는 정보가 있는 클래스이다.

File *f
파일을 저장할 수 있는 변수이다.
char *filename
파일 이름을 저장할 변수이다.
int hour
시를 저장할 변수이다.
int minute
분을 저장할 변수이다.
int sec
초를 저장할 변수이다.
int pos
위치 값을 저장할 변수이다.
QMutex mutex
파일을 읽어 올 때 중복되지 않도록 하는 mutex 변수이다.
int moveNextPos(int *pos)
다음 이동 위치를 반환하는 함수이다.
int moveEndPos(int *pos)
이동의 끝 위치를 반환하는 함수이다.
int decode()
디코딩 하는 함수이다.
void updateRangePos()
Range 위치를 업데이트 하는 함수이다.
int decodeGet(unsigned **data)
디코드의 정보를 가져오는 함수이다.
int moveStartPos()
이동이 시작하는 위치를 반환하는 함수이다.
int setFileName(QString &fileName)
파일 이름을 설정하는 함수이다.

표 4-11 Infilebylocaldrive Class

Infilebynothing은 InFile를 상속받은 Infilebynoting은 아무 일도 하지 않는 클래스이다.

OutFile은 프로그램에서 Output으로 출력할 파일에 대한 정보를 가진다. Output File에 대한 위치 정보, 연결 정보 등에 대한 정보를 포함하는 클래스이다.

virtual int init()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, OutFile의 초기 설정하는 함수이다.
virtual int close()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 클래스를 닫는 함수이다.
virtual void write(unsigned char *outbuf, int out_size)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, output으로 내보낼 정보를 쓰는 함수이다.

표 4-12 OutFile Class

Outfilebylocaldrive OutFile를 상속받고 로컬 드라이브에 있는 파일을 출력으로 내보낼 수 있는 정보가 있는 클래스이다.

char indexFile[1024]
파일에 쓸 때 사용하는 버퍼 변수이다.
int cameraindex
저장할 이미지를 보내주는 카메라의 번호 정보를 갖는 변수이다.
FILE *f
저장할 파일의 변수이다.
FILE *f_index
저장할 파일의 정보를 가지는 파일의 변수이다.
int frame_count
프레임 수를 가지는 변수이다.
int pos
위치 값을 가지는 변수이다.
int hour
시간을 가지는 변수이다.
char* getIndexFile()
버퍼의 위치를 받는 함수이다.
FILE *getFile()
저장할 파일을 반환하는 함수이다.
FILE *getFileIndex()
파일정보를 가지는 파일을 반환하는 함수이다.
void setPos(int pos)

위치를 설정하는 함수이다.
int getPos()
위치를 반환하는 함수이다.
int existDirectory(char *path)
존재하는 디렉토리인지 확인하는 함수이다.
int makeDirectory(char *basic, int year, int mon, int day)
존재하지 않을 때 디렉토리를 생성하는 함수이다.
int makeIndexfile(char *indexFile)
저장할 파일의 정보를 가지는 파일을 생성하는 함수이다.
int alloc(char *fileName, char *indexFile)
파일을 열어 이어 쓰는 함수이다.
char* getFileName(char *indexFile)
파일의 이름을 반환하는 함수이다.
void timeconfirmation()
시간 정보의 싱크를 맞추는 함수이다.

표 4-13 Outfilebylocaldrive Class

Outfilebynothing은 OutFile를 상속받은 Outfilebynoting은 아무 일도 하지 않는 클래스이다.

4. 기본 구조

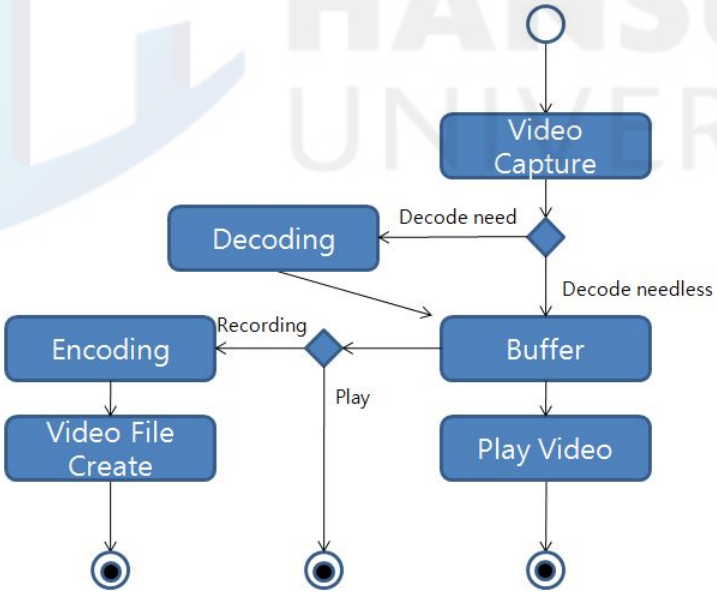

그림 4-7 Play / Record Activity Diagram

그림 4-7와 그림 4-8은 소프트웨어 기본 구조로 프로그램이 가져야 할 기본적인 기능들을 가지고 있다. 그림 9에서 Camera Device에서 캡쳐된 영상을 Buffer 클래스에 보내면 Buffer 클래스에서 저장하고 Display 클래스와 Record 클래스에게 보내주면서 저장된 파일을 삭제 한다. Display 클래스는 Buffer 클래스에서 실시간 영상데이터를 받아서 실시간 영상을 출력한다. Record 클래스는 Buffer 클래스에서 실시간 영상데이터를 받아서 Encode와 Outfile을 이용하여 실시간 영상을 압축하여 파일로 저장한다.

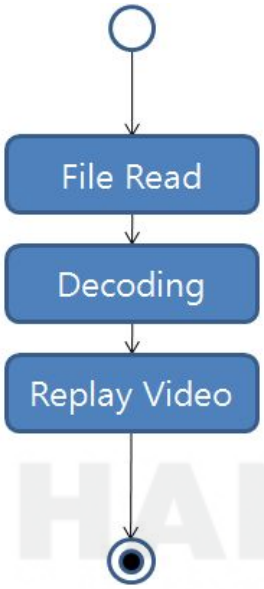

그림 4-8 Replay Activity Diagram

Search 클래스는 저장된 영상을 보여주는 부분으로 Calendar 클래스로 저장된 Video File을 찾아서 Infile과 Decode를 이용하여 저장된 파일의 영상 압축을 풀어 영상을 출력한다.

Hybrid_DVR은 Qt 라이브러리의 QMainWindow 클래스를 상속 받은 클래스로 구현된 전체 클래스의 최상위에 위치하고 있다. 이 클래스는 각 필요한 부분의 클래스 GUIGenerator, LiveView, Search, Event, Setup의 레퍼런스를 가지고 있고 표와 같다.

Ui::HybridDVR *ui
hybriddvr.ui로 QMainWindow 클래스를 기반으로 구성되어 있고 XML형식으로 표현되어있다. 이 클래스 기본 프레임워크로 각종 Widget을 포함하고 있다.
Configuration *configuration
Camera 클래스를 관리하는 클래스의 변수이다.
GuiGenerator *guigenerator
동적으로 GUI를 적용시키기 위한 클래스로 Capability list(XML)를 입력으로 받아 동적으로 GUI를 적용시키는 클래스의 변수이다.
LiveView *liveview
실시간 View에 대한 정보와 제어를 하는 클래스의 변수이다.
Search *search
저장된 영상을 재생하기 위한 클래스의 변수이다.
Setup *setup
각종 설정을 할 수 있는 클래스의 변수이다. (미구현)
Event *event
각종 이벤트를 관리하는 클래스의 변수이다. (미구현)
void on_FileOpen_clicked()
저장된 Capability list(XML) 파일을 읽어 list를 생성한다. 생성된 list의 첫 번째 정보를 기반으로 GUI 및 카메라 정보를 적용시키는 함수이다.
void on_FileList_currentIndexChanged(int index)
생성된 list의 선택이 바뀔 때 마다 바뀐 정보를 기반으로 GUI 및 카메라 정보를 적용시키는 함수이다.
void on_Live_clicked()
적용된 정보를 기반으로 실시간 View를 적용 시켜 Widget에 출력하는 함수이다.
void on_Record_clicked()
실시간 View되고 있는 영상을 녹화시키는 함수이다.
void on_Search_clicked()
저장된 영상을 제어하고 출력 할수 있는 다이얼로그 창을 불러 오는 함수이다.
void on_PowerOFF_clicked()
실행되고 있는 프로그램을 종료하는 함수이다.

표 4-14 Hybird_DVR Class

MDisplay는 Qt 라이브러리의 QWidget를 상속 받고 영상을 보여주는 기능을 한다. QueueBuffer로부터 데이터를 받아 화면에 출력한다.

virtual void setSize(int width, int height)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, display의 크기를 설정하는 함수이다.
virtual void setDisplayNumber(int camNumber)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, display의 번호를 설정하는 함수이다.
Virtual void displayVideo(unsigned char*, int, int)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, display될 데이터를 play하는 함수이다.

표 4-15 MDisplay Class

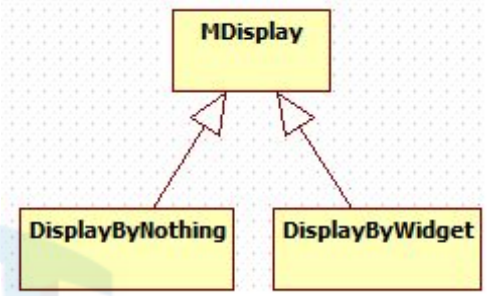

그림 4-9 MDisplay 상속관계

DisplayByWidget은 그림4-9과 같이 MDisplay를 상속받은 Displaybywidget은 QueueBuffer에서 받은 데이터를 widget에 출력하는 기능을 하는 클래스이다.

QImage *image
이미지를 출력하기 위한 변수이다.
unsigned int ng_yuv_gray[]
yuv 데이터 타입 중 gray를 저장하기 위한 변수이다.
unsigned int ng_yuv_red[]
yuv 데이터 타입 중 red를 저장하기 위한 변수이다.
unsigned int ng_yuv_blue[]
yuv 데이터 타입 중 blue를 저장하기 위한 변수이다.
unsigned int ng_yuv_g1[]
yuv 데이터 타입 중 g1를 저장하기 위한 변수이다.
unsigned int ng_yuv_g2[]

yuv 데이터 타입 중 g2를 저장하기 위한 변수이다.
unsigned int ng_clip
yuv 데이터 타입 중 clip를 저장하기 위한 변수이다.
unsigned char* image_buffer
image를 위해 사용되는 buffer를 저장하기 위한 변수이다.
QBrush background
image를 출력하기 위한 변수이다.
int width
출력될 display의 width 크기를 가지는 변수이다.
int height
출력될 display의 height 크기를 가지는 변수이다.
int camNumber
display의 번호를 가지는 변수이다.
bool view
view에 대한 설정 값을 가지는 변수이다.
bool text
text에 대한 설정 값을 가지는 변수이다.
bool imgflag
img에 대한 설정 값을 가지는 변수이다.
bool fullScreenflag
full Screen에 대한 설정 값을 가지는 변수이다.
QString *strText
출력될 text를 가지는 변수이다.
void setText(bool b)
text를 출력에 대한 설정을 하는 함수이다.
void paint(QPainter *painter, QPaintEvent *event)
각 설정에 맞도록 출력하는 함수이다.
void yuv2rgb_init()
yuv2의 rgb초기 설정을 하는 함수이다.
void yuv2rgb(unsigned char* in_addr, unsigned char* out_addr, int originalWidth, int orginalHeight)
rgb를 yuv2의 변환하는 함수이다.
unsigned int getGray(unsigned char val)
gray 값을 반환하는 함수이다.
unsigned int getRed(int gray, unsigned char red)
red 값을 반환하는 함수이다.
unsigned int getGreen(int gray, unsigned char red, unsigned blue)
green 값을 반환하는 함수이다.

unsigned int getBlue(int gray, unsigned char blue)
blue 값을 반환하는 함수이다.
void paintEvent(QPaintEvent *event)
paint에 대한 이벤트를 주는 함수이다.

표 16 DisplayByWidget Class

DisplayByNothing는 그림 4-9과 같이 MDisplay를 상속받은 Displaybynothging는 아무 일도 하지 않는 클래스이다.

Record는 실시간 재생되고 있는 영상을 녹화하는 기능이다. 이 클래스는 재생되고 있는 영상을 저장하는 기능을 하고 Qt 라이브러리의 QThread를 상속받고 있으며 영상 압축에 필요한 Encode 클래스를 포함한다. 영상을 저장하게 되면 QueueBuffer에서 데이터를 받아 압축하여 저장한다. Recode 클래스는 스레드로 실행되면서 선택한 Encode를 이용하여 영상을 압축하여 저장하는 기능을 가진 클래스이다.

QMutex mutex
저장을 위한 mutex 클래스 변수이다.
Encode *encode
encodefmf 저장하기 위한 변수이다.
int camNumber
녹화할 카메라 번호를 저장할 변수이다.
void stop()
녹화를 멈추는 함수이다.
void encodeClose()
encode를 종료하는 함수이다.
void setRecordFloag(bool recordFlag)
녹화에 대한 설정을 하는 함수이다.
bool getRecordFlag()
녹화에 대한 설정 값을 반환하는 함수이다.
void getPacket(unsigned char*, int, int)
QueueBuffer로부터 데이터를 받아오는 함수이다.

표 4-17 Record Class

Search는 저장되어 있는 영상을 재생하기 위한 기능을 가진다. 이 클래스는 저장된 영상을 찾아서 재생해주는 기능을 가진 클래스로 저장된 영상을 찾는 기능을 하는

Calendar 클래스와 LiveView에서 사용되었던 Play 클래스가 포함된다.

Ui::SearchClass ui
searchclass.ui로 QDialog 클래스를 기반으로 구성되어 있고 XML형식으로 표현되어 있다. 이 클래스에 각종 Widget을 포함시킬 수 있다.
QMovie *movie
저장된 영상을 play하기 위한 변수이다.
Calendar *calendar
저장된 영상을 찾기 위한 변수이다.
QString currentMovieDirectory
저장된 영상의 path를 저장하기 위한 변수이다.
MDisplay *mDisplay
영상을 display하기 위한 변수이다.
Decode *decode
저장된 영상을 decode하기 위한 변수이다.
void openFile(QString &fileName)
지정된 파일을 open하는 함수이다.
void on_quitButton_clicked()
다이얼 로그를 종료하는 함수이다.
void on_searchButton_clicked()
저장된 영상을 찾기 위한 다이얼로그를 불러오는 함수이다.
void on_playButton_clicked()
지정된 영상을 play하는 함수이다.
void on_pauseButton_clicked()
play되는 영상을 잠시 멈추는 함수이다.
void on_stoptButton_clicked()
play되는 영상을 정지시키는 함수이다.
void on_leftEndButton_clicked()
play되는 영상을 많이 되감기하는 함수이다.
void on_leftSmallButton_clicked()
play되는 영상을 조금 되감기하는 함수이다.
void on_rightEndButton_clicked()
play되는 영상을 많이 빨리 감기하는 함수이다.
void on_rightSmallButton_clicked()
play되는 영상을 조금 빨리 감기하는 함수이다.
void updateButton()
button을 업데이트하는 함수이다.
void putPath(char *chanPath, QString *decodeType, QString *inputFile,

QTime *time)
저장된 영상의 위치와 디코드 정보, input, 시간을 주는 함수이다.
void goToFrame(int frame)
해당 프레임으로 넘어가는 함수이다.
void fitToWindow()
fit를 확인하는 함수이다.
void updateFrameSlider()
프레임 슬라이더를 업데이트하는 함수이다.

표 4-18 Search Class

Calendar는 저장되어 있는 영상을 찾기 위한 기능을 가진다. 이 클래스는 저장된 영상을 찾는 기능을 담당하는 클래스이다.

Ui::CalendarClass ui
calendarclass.ui로 QDialog 클래스를 기반으로 구성되어 있고 XML형식으로 표현되어 있다. 이 클래스에 각종 Widget을 포함시킬 수 있다.
int year
년도 정보를 가지는 변수이다.
int mon
월 정보를 가지는 변수이다.
int day
날 정보를 가지는 변수이다.
int start_time
시작 시간을 가지는 변수이다.
void putPath(char *chanPath, QString *decodeType, QString *inputFile, QTime *time)
저장된 영상의 위치와 디코드 정보, input, 시간을 보내는 함수이다.
void selectedDateChanged()
선택된 데이터가 바뀌는 함수이다.
void on_selectButton_clicked()
선택을 하는 버튼 함수이다.
void on_okButton_clicked()
확인 버튼 함수이다.
void on_cancelButton_clicked()
취소 버튼 함수이다.

표 4-19 Calendar Class

Buffer는 영상을 버퍼링 하는 기능을 가진다. 이 클래스는 Video 클래스에서 재생 가능하게 제어된 데이터를 임시로 쓰는 버퍼로 Qt 라이브러리의 QThread를 상속받으며 Packet 클래스를 포함한다. Buffer은 스레드로 실행되면서 Video 클래스에서 받은 데이터를 Mdispaly 클래스나 Recode 클래스에게 전달하는 기능을 한다.

QMutex mutex
Queue 클래스를 위한 mutex 변수이다.
Static const int PACKET_COUNT_MAX
packet 크기를 설정하는 변수, 100으로 설정되었다.
QWaitCondition bufferNotEmpty
buffer가 비어있는지 확인하는 변수이다.
QWaitcondition bufferNotFull
buffer가 차 있는 지 확인하는 변수이다.
QQueue<Packet*> queue
Queue 변수이다.
int count
저장된 Packet의 수를 저장하는 변수이다.
bool record
record에 대한 정보를 저장하는 변수이다.
void recordEnable(bool b)
record를 설정하는 함수이다.
bool isRecord()
녹화가 진행되는 지 확인하는 함수이다.
void outVideo(unsigned char*, int, int)
실시간 재생을 할 때 Display에 보내는 packet을 보내는 함수이다.
void outRecord(unsigned char*, int, int)
녹화를 할 때 Record에 보내는 packet을 보내는 함수이다.
void setText(bool text)
녹화할 때 text를 설정하는 함수이다.
void enQueue(unsigned char*, int, int)
queue에 packet을 넣는 함수이다.
void deQueue()
queue에서 packet을 빼는 함수이다.

표 4-20 Buffer Class

Packet은 Buffer에서 사용하는 데이터이다. 이 클래스는 Buffer에서 이용하는 packet 클래스로 데이터의 단위나 형식을 정의하고 있는 클래스이다.

unsigned char* picData
전송할 데이터를 저장하는 변수이다.
int width
전송할 데이터의 width를 저장하는 변수이다.
int height
전송할 데이터의 height를 저장하는 변수이다.
void setData(unsigned char *picData, int width, int height)
데이터를 설정하는 함수이다.
unsigned char* getPicdata()
데이터를 반환하는 함수이다.
int getWidth()
width를 반환하는 함수이다.
int getHeight()
height를 반환하는 함수이다.

표 4-21 Packet Class

Manager는 ViewInfo 클래스를 가지는 클래스로 실시간 재생과 저장 기능을 담당하는 클래스이다.

QWidget *parent
재생될 영상의 담을 widget를 저장하는 변수이다.
QHash<int, ViewInfo*> hash
ViewInfo 클래스를 저장하는 hash 변수이다.
QMutex mutex
hash에 접근할 이용할 mutex 변수이다.
int insertCamera(int cameraIndex)
카메라를 추가하여 실시간 재생 또는 녹화가 가능하게 하는 함수이다.

표 4-22 Manager Class

ViewInfo는 Record와 Play 클래스를 가지고 있고 실시간 재생이 가능한 연결된 camera의 정보를 관리하는 기능을 담당하는 클래스이다.

Play *play

play 클래스를 저장하는 변수이다.
Record *record
record 클래스를 저장하는 변수이다.
void setRecord(Record *record)
record를 설정하는 함수이다.
void setPlay(Play *play)
play를 설정하는 함수이다.
Reocrd* getRecord()
record를 반환하는 함수이다.
Play* getPlay()
play를 반환하는 함수이다.

표 4-23 ViewInfo Class

Play는 Vidoe, QueueBuffer, MDisplay 클래스를 가지며 Play 클래스에서는 재생이 가능하도록 제어하고 디코딩하여 디스플레이하는 기능을 담당하는 클래스이다.

Video *video
Video 클래스를 저장하는 변수이다.
MDisplay *display
MDisplay 클래스를 저장하는 변수이다.
QueueBuffer *buffer
QueueBuffer 클래스를 저장하는 변수이다.
void setVideo(Video *video)
video를 설정하는 함수이다.
void setDisplay(MDisplay *display)
display를 설정하는 함수이다.
void setQueueBuffer(QueueBuffer *buffer)
buffer을 설정하는 함수이다.
Video* getVideo()
video를 반환하는 함수이다.
MDisplay* getDisplay()
display를 반환하는 함수이다.
QueueBuffer* getBuffer()
buffer을 반환하는 함수이다.

표 4-24 Play Class

제 4 절 GUI

소프트웨어 플랫폼의 GUI(Configuration&Control) 부분으로 GUI Configuration 부분과 기본적인 GUI, 동적인 GUI를 구성하기 위한 클래스로 구성되어 있다.

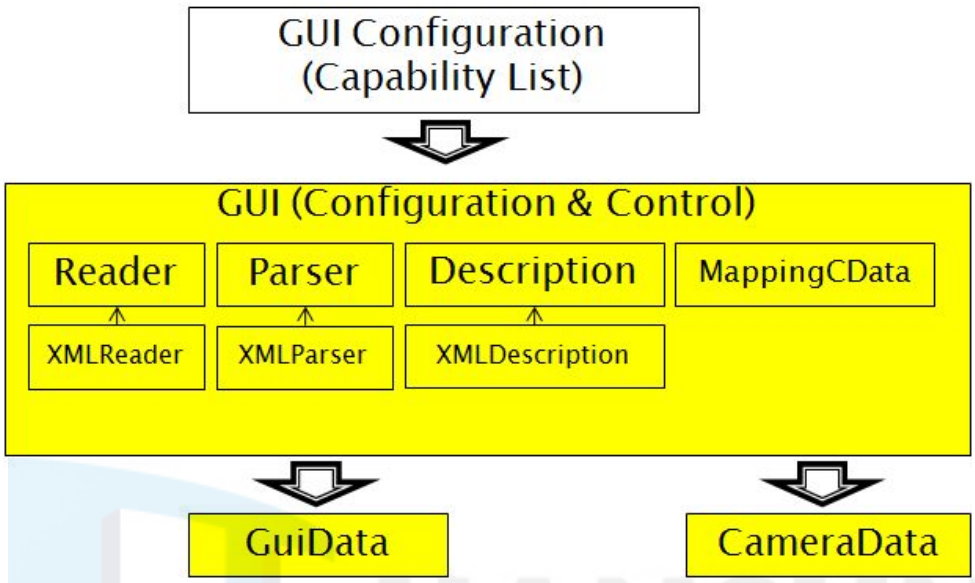

그림 4-10 GUI 클래스 구조

Capability List를 읽어서 Reader(XMLReader), Parser(XMLParser), Description (XMLDescription), MappingCData로 처리하여 GUI의 정보를 가지는 GuiData와 카메라 정보를 가지는 CameraData에 정보를 저장하고 동적인 GUI를 생성한다.

1. Gui Configuration (Capability List)

Capability List는 동적으로 적용될 GUI의 type, 이름, 위치, 크기, 액션 정보와 연결될 카메라의 이름, 번호, camera type, display type, encode type, decode type, input type, output type, control type, 장치 위치, brightness, saturation, hue, contrast 정보를 Camera-1의 구성요소로 Camera-1, Camera-2 등으로 카메라, GUI set의 정보를 XML 형식으로 표현하는 데이터이다. 이 데이터는 XML 파일로 프로그램이 실행되면 읽어서 동적으로 GUI와 camera 정보를 적용시킨다.

```
<?xml version="1.0" encoding="euc-kr" ?>
<Onvif>
    <Camera-1>
        <Camera value="Camera" >
            <cameraIndex value="1" />
            <cameraType value="localCamera" />
            <camdisplayType value="qwidget" />
            <encodeType value="ffmpeg" />
            <decodeType value="ffmpeg" />
            <inputFileSystemType value="localdrive" />
            <outputFileSystemType value="localdrive" />
            <videoControlType value="videoforlinux" />
            <device value="/dev/video0" />
            <Struct value="cameraVideoQuality" >
                <brightness value="32768" />
                <saturation value="32768" />
                <hue value="32768" />
                <contrast value="32768" />
            </Struct>
        </Camera>
        <CapabilityList>
            <PTZ type = "Control" widgettest = "TestPTZ">
                <PTZUP parent = "PTZ" widget = "Button" sizew = "50"
                        sizeh = "30" locx = "10" locy = "10"
                        Text = "UP" action = "1"></PTZUP>
                <PTZDOWN parent = "PTZ" widget = "Button" sizew = "50"
                        sizeh = "30" locx = "70" locy = "10"
                        Text = "DOWN" action = "2"></PTZDOWN>
                <PTZLEFT parent = "PTZ" widget = "Button" sizew = "50"
                        sizeh = "30" locx = "130" locy = "10"
```

```
                Text = "LEFT" action = "3"></PTZLEFT>
        <PTZRIGHT parent = "PTZ" widget = "Button" sizew = "50"
                sizeh = "30" locx = "190" locy = "10"
                Text = "RIGHT" action = "4"></PTZRIGHT>
    </PTZ>
    <PTZ2 type = "Control" widgettest = "TestPTZ2">
        <PTZUP parent = "PTZ" widget = "Button" sizew = "50"
                sizeh = "30" locx = "10" locy = "10"
                Text = "UP" action = "1"></PTZUP>
        <PTZDOWN parent = "PTZ" widget = "Button" sizew = "50"
                sizeh = "30" locx = "70" locy = "10"
                Text = "DOWN" action = "2"></PTZDOWN>
        <PTZLEFT parent = "PTZ" widget = "Button" sizew = "50"
                sizeh = "30" locx = "130" locy = "10"
                Text = "LEFT" action = "3"></PTZLEFT>
        <PTZRIGHT parent = "PTZ" widget = "Button" sizew = "50"
                sizeh = "30" locx = "190" locy = "10"
                Text = "RIGHT" action = "4"></PTZRIGHT>
    </PTZ2>
    <TestConfigure type = "Configuration" widgettest = "TestConf">
        <Name parent = "TestConfigure" widget = "Button"
          sizew = "100" sizeh = "30" locx = "10" locy = "50"
          Text = "TestConfigure" action = "1"></Name>
        <TestCombo parent = "TestConfigure" widget = "ComboBox"
          sizew = "100" sizeh = "30" locx = "120" locy = "50"
          action = "1">
           <ComboString parent = "TestCombo"
             widget = "ComboString" Text = "test1"></ComboString>
           <ComboString parent = "TestCombo"
             widget = "ComboString" Text = "test2"></ComboString>
           <ComboString parent = "TestCombo"
```

```
                    widget = "ComboString" Text = "test3"></ComboString>
                </TestCombo>
                <TestButton parent = "TestConfigure"
                 widget = "Button" sizew = "100" sizeh = "30"
                 locx = "230" locy = "50" Text = "TestButton" action = "1">
                </TestButton>
            </TestConfigure>
            <TestConfigure2 type = "Configuration" widgettest = "TestConf2">
                <Name parent = "TestConfigure"
                 widget = "Button" sizew = "100" sizeh = "30" locx = "10"
                 locy = "50" Text = "TestConfigure" action = "1"></Name>
                <TestCombo parent = "TestConfigure" widget = "ComboBox"
                    sizew = "100" sizeh = "30" locx = "120"
                    locy = "50"  action = "1">
                    <ComboString parent = "TestCombo"
                      widget = "ComboString" Text = "test1"></ComboString>
                    <ComboString parent = "TestCombo"
                      widget = "ComboString" Text = "test2"></ComboString>
                    <ComboString parent = "TestCombo"
                      widget = "ComboString" Text = "test3"></ComboString>
                </TestCombo>
                <TestButton parent = "TestConfigure"
                 widget = "Button" sizew = "100" sizeh = "30"
                 locx = "230" locy = "50" Text = "TestButton" action = "1">
                </TestButton>
            </TestConfigure2>
        </CapabilityList>
    </Camera-1>
    <Camera-2>
        <Camera value="Camera" >
            <cameraIndex value="1" />
```

```
                .
                .
                .
        </CapabilityList>
    </Camera-2>
</Onvif>
```

표 4-25 Gui Configuration XML

2. 기본 GUI 구조

이미 구성된 GUI는 프로그램이 기본적으로 가지는 기본 GUI로 Qt 라이브러리에서 제공하는 GUI 구성 방식으로 XML형식을 취하고 있고 구성된 XML 형식의 파일을 읽어 헤더 파일로 제공되는 형식이다. 프로그램에서 사용될 기본적인 GUI에 대해서만 구성되었다.

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
    <class>HybridDVR</class>
    <widget class="QMainWindow" name="HybridDVR">
        <property name="enabled">
            <bool>true</bool>
        </property>
        <property name="geometry">
            <rect>
                <x>0</x>
                <y>0</y>
                <width>1300</width>
                <height>850</height>
            </rect>
        </property>
        <property name="windowTitle">
            <string>Hybrid DVR</string>
        </property>
        <widget class="QWidget" name="centralwidget">
```

```
        <widget class="QWidget" name="ViewWidget" native="true">
            <property name="geometry">
                <rect>
                    <x>10</x>
                    <y>10</y>
                    <width>960</width>
                    <height>730</height>
                </rect>
            </property>
        </widget>
        <widget class="QPushButton" name="FileOpen">
            <property name="geometry">
                <rect>
                    <x>10</x>
                    <y>760</y>
                    <width>90</width>
                    <height>27</height>
                </rect>
            </property>
            <property name="text">
                <string>File Open</string>
            </property>
        </widget>
            .
            .
            .
    </widget>
    <widget class="QMenuBar" name="menubar">
        <property name="geometry">
            <rect>
                <x>0</x>
```

```
                    <y>0</y>
                    <width>1300</width>
                    <height>22</height>
                </rect>
            </property>
        </widget>
        <widget class="QStatusBar" name="statusbar"/>
        </widget>
    <resources/>
    <connections/>
</ui>
```

표 4-26 Gui 기본 구조

hybriddvr.ui는 전체 프레임워크로 각종 버튼 등 여러 가지 QWidget을 갖는 GUI 이다.

QWidget *viewWidget
실시간 영상이 출력될 부분의 변수이다.
QPushButtion *FileOpen
XML을 읽어오게 할 버튼의 변수이다.
QComboBox *FileList
저장되는 FileList를 보여줄 변수이다.
QPushButtion *Setup
Setup 다이얼로그 창을 여는 버튼의 변수이다.
QPushButtion *PowerOFF
프로그램을 종료하는 버튼의 변수이다.
QPushButtion *Search
Search 다이얼로그 창을 여는 버튼의 변수이다.
QPushButtion *Record
녹화를 하는 버튼의 변수이다.
QPushButtion *Live
실시간 영상을 출력하게 하는 버튼의 변수이다.
QPushButtion *Other
다른 다이얼로그 창을 여는 버튼의 변수이다.

void setupUi(QMainWindow *HybridDVR)
각 Widget을 생성하는 함수이다.
void retranslateUi(QMainWindow *HybridDVR)
각 Widget의 위치 및 이름을 설정하는 함수이다.

표 4-27 hybirddvr.ui

dialog.ui는 동적인 GUI를 구성할 때 기본 바탕으로 사용되는 다이얼로그 창의 GUI이다.

QPushtButton *OKButton
OK 버튼의 변수이다.
void setUi(QDialog *Dialog)
각 Widget을 생성하는 함수이다.
void retranslateUi(QDialog *Dialog)
각 Widget의 위치 및 이름을 설정하는 함수이다.

표 4-28 dialog.ui

setupdil.ui는 동적인 GUI를 구성할 때 기본 바탕으로 사용되는 다이얼로그 창의 GUI이다.

QPushtButton *OKButton
OK 버튼의 변수이다.
QPushtButton *OTHERButton
Other 버튼의 변수이다.
QPushtButton *CANCELButton
Cancel 버튼의 변수이다.
void setUi(QDialog *Dialog)
각 Widget을 생성하는 함수이다.
void retranslateUi(QDialog *Dialog)
각 Widget의 위치 및 이름을 설정하는 함수이다.

표 4-29 setuidil.ui

search.ui는 녹화된 영상을 재생하기 위해서 필요한 다이얼로그 창으로 여러 가지 widget을 포함한 GUI이다.

QWidget *horizontallayoutWidget
horizontalLayout이 출력되는 부분의 변수이다.
WHBoxLayout *horizontalLayout
수평 형식의 레이아웃에 대한 변수이다.
QToolButton *searchButton
search의 툴 버튼의 변수이다.
QToolButton *leftEnd
많이 되감는 툴 버튼의 변수이다.
QToolButton *leftSmall
조금 되감는툴 버튼의 변수이다.
QToolButton *playButton
재생의 툴 버튼의 변수이다.
QToolButton *pauseButton
일시정지의 툴 버튼의 변수이다.
QToolButton *stopButton
정지의 툴 버튼의 변수이다.
QToolButton *rightSmall
조금 빨리 감는 툴 버튼의 변수이다.
QToolButton *rightEnd
많이 빨리 감는 툴 버튼의 변수이다.
QToolButton *soundOn
sound on의 툴 버튼의 변수이다.
QToolButton *soundOff
sound off의 툴 버튼의 변수이다.
QToolButton *quitButton
다이얼로그 종료의 툴 버튼의 변수이다.
QWidget *gridLayoutWidget
controlLayout이 출력되는 부분의 변수이다.
QGridLayout *controlLayout
격자형식의 레이아웃에 대한 변수이다.
QLabel *soundLabel
label에 대한 변수이다.
QLabel *speedLabel
label에 대한 변수이다.
QCheckBox *fitCheckBox
checkBox에 대한 변수이다.
QSlider *frameSlider
프레임을 제어할 slider에 대한 변수이다.
QSlider *soundSlider

소리를 제어할 slider에 대한 변수이다.
QSpinBox *speedSpin
재생 속도를 제어할 spinBox에 대한 변수이다.
QLabel *frameLabel
label에 대한 변수이다.
QListWidget *movieList
저장된 영상의 정보를 보여줄 listWidget 변수이다.
void setupUi(QDialog *Dialog)
각 Widget을 생성하는 함수이다.
void retranslateUi(QDialog *Dialog)
각 Widget의 위치 및 이름을 설정하는 함수이다.

표 4-30 search.ui

calendar.ui는 녹화된 영상을 찾기 위해서 필요한 다이얼로그 창으로 여러 가지 widget을 포함한 GUI이다.

QCalendarWidget *calendar
년, 월, 일 단위로 저장된 영상을 찾기 위한 calendar widget을 가지는 변수이다.
QGroupBox *resultGroupBox
결과를 표시하는 groupBox에 대한 변수이다.
QRadioButton *cam1
cam1을 선택하는 radio button 변수이다.
QRadioButton *cam2
cam2을 선택하는 radio button 변수이다.
QRadioButton *cam3
cam3을 선택하는 radio button 변수이다.
QRadioButton *cam4
cam4을 선택하는 radio button 변수이다.
QPushButton *selectButton
select button에 대한 변수이다.
QLabel *label
label에 대한 변수이다.
QPushButton *okButton
ok button에 대한 변수이다.
QPushButton *cancelButton
cancel button에 대한 변수이다.
QLabel *label_1

label에 대한 변수이다.
QLable *label_2
label에 대한 변수이다.
QTimeEdit *currentTimeEdit
시간에 대한 정보를 가지는 timeEdit에 대한 변수이다.
QTableView *tableVeiw
저장된 영상 목록을 보여 줄 tableView에 대한 변수이다.
void setupUi(QDialog *Dialog)
각 Widget을 생성하는 함수이다.
void retranslateUi(QDialog *Dialog)
각 Widget의 위치 및 이름을 설정하는 함수이다.

표 4-31 calendar.ui

3. GUI 구성 자료 구조 및 카메라 구성 자료

GUI 구성 자료 구조는 GUI를 동적으로 구성되는 GUI의 정보를 Capability list(XML)로부터 읽어와 저장하기 위한 자료 구조이다.

4. GuiData

기본적으로 세 가지(link, child, parent) GuiData가지고 이는 GUI 데이터의 구조를 link형식으로 구성할 수 있게 한다. 또한 GuiAction과 GuiWidget 클래스를 가지는데 각각은 action과 widget에 대한 정보를 가진다. 이 클래스를 상속받아 미리 정의된 GUI 그룹을 생성할 수 있다.

GuiData *link
GuiData의 수평 관계를 저장하는 변수이다.
GuiData *child
GuiData의 하위 관계를 저장하는 변수이다.
GuiData *parent
GuiData의 상위 관계를 저장하는 변수이다.
QString Tag
Xml 태그를 저장하는 변수이다.
QString widget
어떤 widget인지 저장하는 변수이다.
QString type

어떤 형식의 widget인지 저장하는 변수이다.
QString text
무엇이 쓰일지 저장하는 변수이다.
QString name
widget의 이름을 저장하는 변수이다.
QString parentTag
부모의 tag를 저장하는 변수이다.
GuiWidget *guiWidget
실제 표현될 QWidget에 대한 정보를 갖는 GuiWidget을 저장하는 변수이다.
GuiAction *guiAction
실제 액션에 대한 정보를 갖는 GuiAction을 저장하는 변수이다.
QWidget *widgetParent
부모 widget을 저장하는 변수이다.
int x, y, width, height
widget의 위치 및 크기 정보를 가지는 변수이다.
int actionNum
action 번호를 저장하는 변수이다
int min, max
최소, 최대 값을 저장하는 변수이다.
bool flag
widget의 타입을 구분되게 저장하는 변수이다.
void setLink(basedata *link)
link를 설정하는 함수이다.
GuiData* getLink()
link를 반환하는 함수이다.
void setChild(GuiData *child)
child를 설정하는 함수이다.
GuiData* getChild()
child를 반환하는 함수이다.
void setParent(GuiData *parent)
parent를 설정하는 함수이다.
GuiData* getParent()
parent를 반환하는 함수이다.
void setTag(QString tag)
tag를 설정하는 함수이다.
QString getTag()
tag를 반환하는 함수이다.
void setWidget(QString *widget)
widget을 설정하는 함수이다.

QString getWidget()
widget을 반환하는 함수이다.
void setType(QString *type)
type을 설정하는 함수이다.
QString getType()
type을 반환하는 함수이다.
void setText(QString text)
text를 설정하는 함수이다.
QString getText()
text를 반환하는 함수이다.
void setName(QString name)
name를 설정하는 함수이다.
QString getName()
name를 반환하는 함수이다.
void setParentTag(QString parentTag)
parentTag를 설정하는 함수이다.
QString getParentTag()
parentTag를 반환하는 함수이다.
void setGuiWidget(GuiWidget *guiWidget)
guiWidget를 설정하는 함수이다.
GuiWidget* getGuiWidget()
guiWidget를 반환하는 함수이다.
void setGuiAction(GuiAction *action)
guiAction를 설정하는 함수이다.
GuiAction* getGuiAction()
guiAction를 반환하는 함수이다.
void setParent(QWidget *parentWidget)
parnetWidget을 설정하는 함수이다.
void setSize(QString width, QString height)
width, height 위치를 설정하는 함수이다.
void setLocation(QString x, QString y)
x, y 위치를 설정하는 함수이다.
int getX()
x를 반환하는 함수이다.
int getY()
y를 반환하는 함수이다.
int getWidth()
width를 반환하는 함수이다.
int getHigth()
heigth를 반환하는 함수이다.

void setActionNum(QString action)
actionNum를 설정하는 함수이다.
int getActionNum()
actionNum를 반환하는 함수이다.
void setMin(QString min)
min을설정하는 함수이다.
void setMax(QString max)
max를 설정하는 함수이다.
int getMin()
min을 반환하는 함수이다.
int getMax()
max을 반환하는 함수이다.
void changeFlag(bool flag)
flag을 변경하는 함수이다.
bool getFlag()
flag을 반환하는 함수이다.

표 4-32 GuiData Class

그림4-11 은 GuiData의 link 구조로 parent와 child는 상하 관계를 가지고 link 관계는 수평관계를 가지는 것을 보여준다. 상하 관계는 포함 관계이고 수평 관계는 동등한 관계이다. 예로 widget안의 button과 label은 widget과 button 또는 label 상하 관계이고 button과 label은 수평 관계이다.

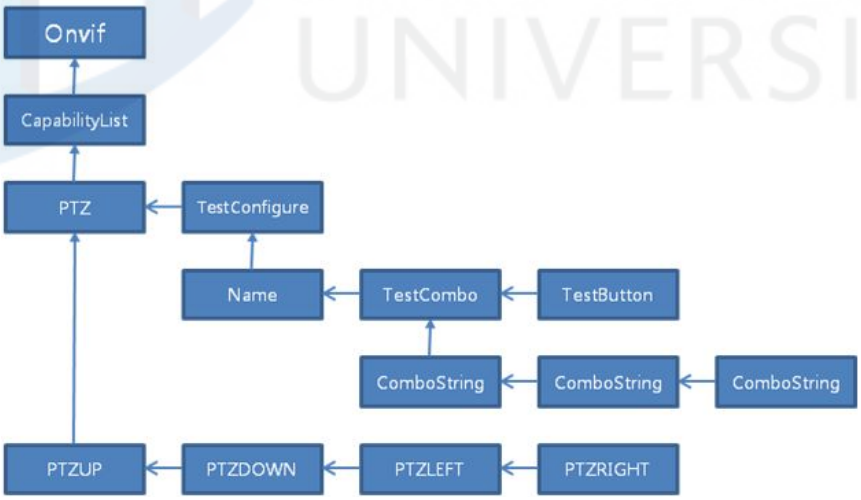

그림 4-11 GuiData link 구조

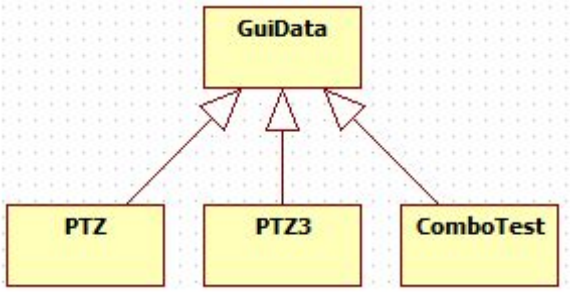

그림 4-12 GuiData 상속관계

그림4-12은 은 GuiData를 상속받은 클래스를 보여준다. Capability list(XML)을 이용한 동적인 GUI생성 시 컨포넌트 단위가 아닌 그룹 단위로 GUI를 생성하는 것을 가능하게 한다. 미리 정의해 놓은 그룹 단위의 GUI를 생성하고 필요한 그룹 단위의 GUI를 추가할 수 있다.

PTZ는 GuiData를 상속받은 클래스로 미리 정의된 GUI 그룹으로 PTZ에 대한 GUI 버튼에 대한 정보(GuiData, GuiAction, GuiWidget)를 가지는 클래스이다.

void setGeometryWidget(int x, int y, int w, int h)
widget의 위치 및 크기를 설정하는 함수이다.

표 4-33 PTZ Class

PTZ3는 GuiData를 상속받은 클래스로미리 정의된 GUI 그룹으로 PTZ에 대한 다른 형식의 GUI 버튼에 대한 정보(GuiData, GuiAction, GuiWidget)를 가지는 클래스이다.

void setGeometryWidget(int x, int y, int w, int h)
widget의 위치 및 크기를 설정하는 함수이다.

표 4-34 PTZ3 Class

ComboTest는 GuiData를 상속받은 클래스로미리 정의된 GUI 그룹으로 ComboButton에 대한 GUI 버튼에 대한 정보(GuiData, GuiAction, GuiWidget)를 가지는 클래스이다.

void setGeometryWidget(int x, int y, int w, int h)
widget의 위치 및 크기를 설정하는 함수이다.

표 4-35 ComboTest Class

GuiAction는 Qt 라이브러리의 QObject를 상속받았고 GUI 정보 중 액션 기능을 담당하는 클래스이다. 이 클래스를 상속받아 추가적으로 각 액션 기능을 담당하는 클래스를 생성할 수 있다.

QPushButton *button
PushButton을 저장하는 변수이다.
QLabel *label
Label을 저장하는 변수이다.
QComboBox *combo
ComboBox를 저장하는 변수이다.
QSlider *slider
Slider를 저장하는 변수이다.
QLineEdit *line
LineEdit을 저장하는 변수이다.
int actiontype
action type을 저장하는 변수이다.
Dialog *dial
Dialog를 저장하는 변수이다.
Dialog *setdial
Dialog를 저장하는 변수이다.
TabWidget *tab
TabWidget을 저장하는 변수이다.
int getActionType()
action 타입을 반환하는 함수이다.
void setActionType(int actiontype)
action 타입을 설정하는 함수이다.
void action()
설정된 action을 실행하는 함수이다.
void setDialog(Dialog *dial)
dialog를 설정하는 함수이다.
void setSetupDial(Dialog *setdial)

setup dialog를 설정하는 함수이다.
void setTab(TabWidget *tab)
tab을 설정하는 함수이다.
void setButton(QPushButton *button)
button을 설정하는 함수이다.
void setComboBox(QComboBox *combo)
comboBox를 설정하는 함수이다.
void setLabel(QLabel *label)
label을 설정하는 함수이다.
void setSlider(QSlider *slider)
slider을 설정하는 함수이다.
void setLineEdit(QLineEdit *lin)
lineEdit을 설정하는 함수이다.
void LabelVis()
설정된 Label을 보여주는 함수이다.

표 4-36 GuiAction Class

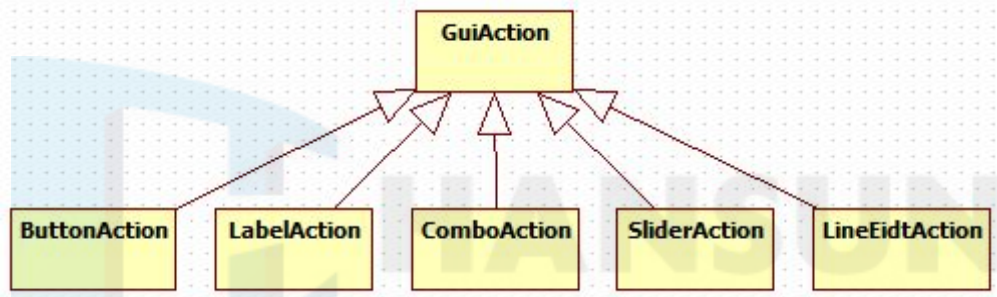

그림 4-13 GuiAction 상속관계

그림4-13은 GuiAction을 상속받아 생성되는 여러 액션을 보여주고 있다.

Class는 하나씩의 기능을 가지는 클래스로 구현되었다.

ButtonAction Class
GuiAction를 상속하여 작성된 ButtonAction은 Button에 대한 액션을 정의한 클래스이다.
LabelAction Class
GuiAction를 상속하여 작성된 LabelAction은 Label에 대한 액션을 정의한 클래

스이다.
ComboAction Class
GuiAction를 상속하여 작성된 ComboAction은 ComboButton에 대한 액션을 정의한 클래스이다.
SliderAction Class
GuiAction를 상속하여 작성된 SliderAction은 slider에 대한 액션을 정의한 클래스이다.
LineEditAction Class
GuiAction를 상속하여 작성된 LineEditAction은 lineedit에 대한 액션을 정의한 클래스이다.

표 4-37 GuiAction을 상속받는 각 Class

GuiWidget은 GUI 정보 중 widget 기능을 담당하는 클래스이다. 이 클래스를 상속받아 추가적으로 각 widget 기능을 담당하는 클래스를 생성할 수 있다.

int x, y, w, h
widget의 위치 및 크기에 대한 정보를 저장하는 변수이다.
GuiWidget *guiWidget
정보를 담고 있는 guiwidget을 저장하는 변수이다.
QWidget *parent
현재 위젯의 부모를 저장하는 변수이다.
QStringList *stringList
여러 개의 스트링을 저장하는 변수이다.
virtual void setGeometry(int x, int y, int w, int h)
virtual 함수로서 정의되지 않았다. 상속 받는 클래스가 정의하며, 위젯의 위치 및 크기를 설정하는 함수이다.
virtual void setParent(QWidget *parent)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, parent widget을 설정하는 함수이다.
virtual void setVisible(bool flag)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 설정된 widget을 보이게 하거나 보이지 않게 하는 함수이다.
virtual void setAction(baseaction *action)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, widget에 대한

action을 설정하는 함수이다.
virtual void setText()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, widget에서 표시할 text를 설정하는 함수이다.

표 4-38 GuiWidget Class

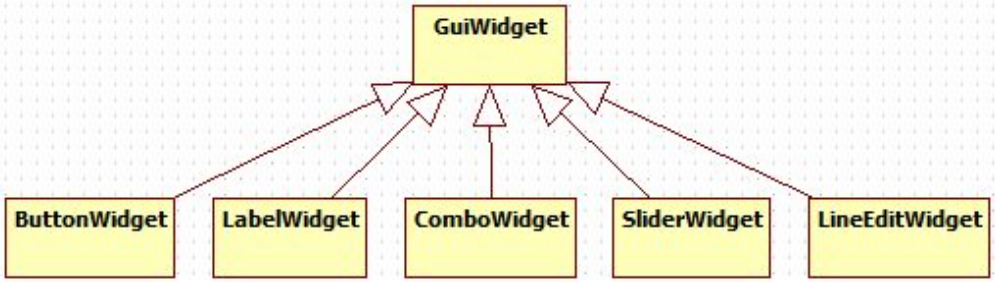

그림 4-14 GuiWidget 상속관계

그림4-14는 GuiWidget을 상속받아 생성되는 여러 widget을 보여주고 있다.

Class는 하나씩의 widget을 가지는 클래스로 구현되었다.

ButtonWidget Class
GuiWidget를 상속하여 작성된 ButtonWidget은 Button에 대한 widget을 정의한 클래스이다.
LabelWidget Class
GuiWidget를 상속하여 작성된 LabelWidget은 Label에 대한 widget을 정의한 클래스이다.
ComboWidget Class
GuiWidget를 상속하여 작성된 ComboWidget은 ComboButton에 대한 widget을 정의한 클래스이다.
SliderWidget Class
GuiWidget를 상속하여 작성된 SliderWidget은 Slider에 대한 widget을 정의한 클래스이다.
LineEditWidget Class
GuiWidget를 상속하여 작성된 LineEditWidget은 LineEdit에 대한 widget을 정의한 클래스이다.

표 4-39 GuiWidget을 상속받는 각 Class

CameraData는 카메라의 정보를 담는 클래스로 카메라의 인코딩, 디코딩, 저장 형식, 디스플레이 방식, 카메라 연결 방식, 제어 등 카메라를 사용하는데 필요한 요소들을 가진다. GUIGenerator가 동작할 때 Reader가 읽어온 Capability list(XML)에서 카메라 정보를 MappingCData를 이용해 저장하는 클래스이다.

QString *index
카메라 번호를 가지는 변수이다.
QString *camera
연결될 카메라 정보를 가지는 변수이다. (Local, USB, IP Camera 등)
QString *display
카메라의 영상을 출력하는 방식의 정보를 가지는 변수이다.
QString *encode
영상의 인코딩 정보를 가지는 변수이다.
QString *decode
영상의 디코딩 정보를 가지는 변수이다.
QString *iFile
읽어올 파일의 정보를 가지는 변수이다.
QString *oFile
저장할 파일의 정보를 가지는 변수이다.
QString *control
카메라 제어에 대한 정보를 가지는 변수이다. (V4l 등)
QString *device
카메라 장치의 위치 정보를 가지는 변수이다. (Local 정보, IP 정보 등)
QString *brightness
영상의 밝기에 대한 정보를 가지는 변수이다.
QString *saturation
영상의 채도에 대한 정보를 가지는 변수이다.
QString *hue
영상의 색조에 대한 정보를 가지는 변수이다.
QString *contrast
영상의 대비에 대한 정보를 가지는 변수이다.
void setIndex(QString value) / char* getIndex()
카메라 번호정보를 설정하고 가져오는 함수이다.
void setCamera(QString value) / char* getCamera()
카메라 연결 정보를 설정하고 가져오는 함수이다.
void setDisplay(QString value) / char* getDisplay()
카메라를 출력하는 방식에 대한 정보를 설정하고 가져오는 함수이다.

void setEncode(QString value) / char* getEncode()
영상을 인코딩하는 정보를 설정하고 가져오는 함수이다.
void setDecode(QString value) / char* getDecode()
영상을 디코딩하는 정보를 설정하고 가져오는 함수이다.
void setIFile(QString value) / char* getIFile()
읽어올 파일에 대한 정보를 설정하고 가져오는 함수이다.
void setOFile(QString value) / char* getOFile()
저장할 파일에 대한 정보를 설정하고 가져오는 함수이다.
void setControl(QString value) / char* getControl()
카메라 제어에 대한 정보를 설정하고 가져오는 함수이다.
void setDevice(QString value) / char* getDevice()
카메라 장치의 위치 정보를 설정하고 가져오는 함수이다.
void setBrightness(QString value) / char* getBrightness()
영상의 밝기 정보를 설정하고 가져오는 함수이다.
void setSaturation(QString value) / char* getSaturation()
영상의 채도 정보를 설정하고 가져오는 함수이다.
void setHue(QString value) / char* getHue()
영상의 색조 정보를 설정하고 가져오는 함수이다.
void setContrast(QString value) / char* getContrast()
영상의 대비 정보를 설정하고 가져오는 함수이다.

표 4-40 CameraData Class

Configuration은 저장된 CameraData 클래스를 관리하는 클래스로 최대 저장 가능한 카메라에 대한 정보 및 CameraData 클래스를 저장하고 관리하며 프로그램이 실행되면 한번만 생성되어 전역 함수를 이용하여 정보를 얻을 수 있다.

static Configuration config
정적인 Configuration 변수이다.
int count
저장된 카메라 수를 가지는 변수이다.
static const int CAMERA_INDEX_MAX
최대 카메라 수로 4로 정해진 변수이다.
int cameraIndex[CAMERA_INDEX_MAX]
카메라의 인덱스를 저장하는 공간의 변수이다.
QHash<int, CameraData*> hash
CameraData 클래스의 정보를 보관하는 QHash로 int로 구분하는 변수이다.

QMutex mutex
QHash에 접근할 때 중복을 막기 위한 mutex 변수이다.
Static Configuration *getInstance()
정적인 함수로 정적인 Configuration을 가져온다.
int setCamera(CameraData *camera) / CameraData* getCamera(int index)
CameraData 클래스를 설정하고 가져온다.
int setCameraIndex(int index) / int getCameraIndex(int index)
카메라의 인덱스를 설정하고 가져온다.
int getCameraIndexMax()
카메라 최대 인덱스 값을 가져온다.
int getCounter()
현재 저장된 카메라의 수를 가져온다.
void reset()
현재 저장된 정보를 초기화 한다.

표 4-41 Configuration Class

5. 동적 GUI 구조

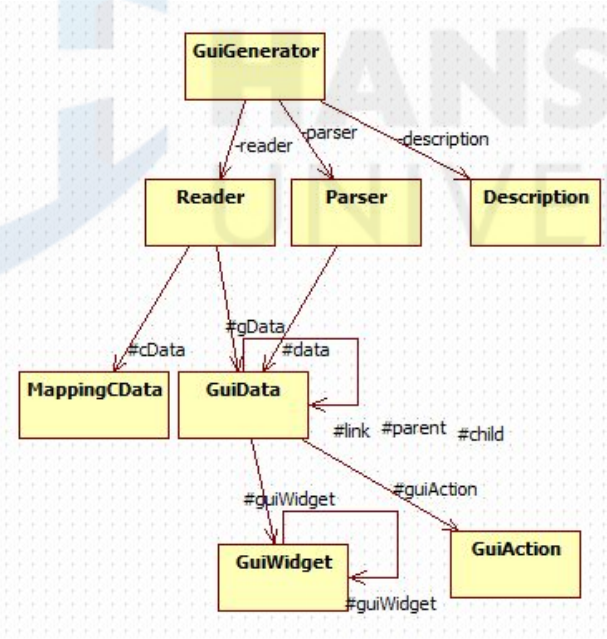

그림 4-15 GuiGenerator 구조

Capability list(XML)를 분석하여 GUI를 만드는 Reader, Parser, Description, MappingCData 클래스를 조합하는 클래스로 그림 4-15와 같은 구조를 가진다.

GUIGenerator Capability list(XML)의 정보를 읽어서 GUI를 동적으로 생성하는 기능을 가진 클래스이다. 이 클래스는 Reader, Parser, Description 클래스를 가진다.

Ui::HybridDVR *ui
상위 클래스의 GUI 포인터로 동적으로 적용될 GUI를 적용시키기 위한 변수이다.
QWidget *parent
상위 클래스의 포인터로 적동되는 동작을 연결시키기 위한 변수이다.
QWidget *widget
동적으로 생성될 GUI를 포함하여 상위 GUI에 적용시키기 위한 변수이다.
XMLReader *reader
Capability list(XML)을 읽어오는 기능을 가지는 클래스의 변수이다.
XMLParser *parser
읽어온 Capability list(XML)을 구분하는 기능을 가지는 클래스의 변수이다.
Description *description
구분된 정보에 widget과 action을 맵핑시키는 기능을 가지는 클래스의 변수이다.
MappingCAdta *mappingCData
읽어온 Capability list(XML)에서 카메라 정보를 저장하는 기능을 가지는 클래스의 변수이다.
QString filename
읽어올 파일의 위치 및 이름 정보를 갖는 변수이다.
Dialog *dial
비어 있는 다이얼로그의 변수이다.
Dialog *other
비어 있는 다이얼로그의 변수이다.
SetupDial *setupd
비어 있는 Setup 다이얼로그의 변수이다.
QHash <int, QDomElement> hash
읽어온 Capability list(XML) 정보를 구분하여 저장하기 위한 hash 변수이다.
void init()
초기화에 필요한 클래스들을 생성하는 함수이다.

void startNodeSet()
읽어올 파일을 reader와 연결하는 함수이다.
void deleteData(basedata *d)
생성되었던 정보를 제거하는 함수이다.
bool checkFile(QString file)
file list에 정보가 있는지 검사하는 함수이다.
void widgetSetVisible(basedata *d)
생성된 widget들을 보이게 하는 함수이다.
void buttionClick()
Capability list(XML) 파일을 읽어오게 하는 함수로 각각을 생성하고 연결하는 기능을 가지며 읽은 정보를 list box에 저장하는 함수이다.
void listChange(int index)
list가 바뀔 때 마다 Gui 및 카메라 정보를 새로 갱신하는 함수이다.

표 4-42 GuiGenerator Class

Reader는 Capability list(XML)를 읽어 Data로 저장하는 클래스이다. Virtual 클래스로서 비어있는 함수와 변수들로 이루어져 있다.

GuiData *gData
UI 정보를 저장하기 위한 변수이다.
QDomDocument *domDocument
읽어온 Capability list(XML)를 저장하기위한 변수이다.
MappingCData *cData
camera 정보를 구분하여 저장하는 클래스의 변수이다.
virtual void NodeDataSetup(QDomNode node, GuiData *data, GuiData *data2)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, node에서 데이터를 GuiData로 저장하는 함수이다.
virtual void NodeLoad()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 기본 Capability List(XML)을 읽어오는 함수이다.
virtual void NodeLoad(QString filename)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 기본 Capability List(XML) 이외에 다른 Capability List를 읽어오는 함수이다.
virtual GuiData* getData()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, GuiData를 반

환하는 함수이다.
virtual void setMappingCDatal(MappingCData *cData)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, MappingCData를 설정하는 함수이다.

표 4-43 Reader Class

Parser는 Reader에서 저장된 GuiData에 GUI를 생성하기 위한 정보를 입력하는 클래스이다. Virtual 클래스로서 비어있는 함수와 변수들로 이루어져 있다. GuiData의 내용을 구분하여 저장된 클래스에 적합한 API를 연결할 수 있도록 해야 한다.

GuiData *data
UI 정보를 저장하기 위한 변수이다.
QWidget *parent
상위 위젯을 저장하기 위한 변수이다.
QWidget *setupparent
setup widget을 저장하기 위한 변수이다.
virtual void setNodeData(GuiData *data, GuiData *befored)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, GuiData를 구분하는 함수이다.
virtual GuiData* setData(GuiData *data)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, GuiData를 설정하는 함수이다.
virtual void setParent(QWidget *parent)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 부모 위젯을 설정하는 함수이다.
virtual GuiData* getData()
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, GuiData를 반환하는 함수이다.
virtual void setSetupParent(QWidget *setupparent)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, setupParent를 설정하는 함수이다.

표 4-44 Parser Class

Description은 Parser에서 저장된 GuiData를 GUI의 widget, action을 설정하기 위

한 클래스이다. Virtual 클래스로서 비어있는 함수와 변수들로 이루어져 있다. Parser에서 만들어진 GuiData에 각 GUI들의 위젯 설정 및 액션 저장해야 한다.

Dialog *parent
다이얼로그를 저장하기 위한 변수이다.
Dialog *setupParent
setup 다이얼로그를 저장하기 위한 변수이다.
TabWidget *widget
TabWidget을 저장하기 위한 변수이다.
virtual void setParent(Dialog *parent)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, 다이얼로그를 설정하는 함수이다.
virtual void setSetupParent(Dialog *setupParent)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, setup 다이얼로그를 설정하는 함수이다.
virtual void dataDescription(GuiData *data)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, widget의 설정된 위치 값에 Gui를 보이게 설정하는 함수이다.
virtual void widgetDescription(GuiData *data, GuiData *parentnd, GuiData *parent)
virtual 함수로서 정의되지 않았다. 상속받는 클래스가 정의하며, widget에 데이터 값을 저장하고 액션을 설정하는 함수이다.

표 4-45 Description Class

MappingCData는 Capability list(XML)로부터 Camera 정보를 읽어 저장하는 기능을 담당하는 클래스이다.

void mappingData()
데이터를 매핑시키는 함수 함수이다.
void parseElement(QDomElement &element, CameraData *data, int count)
읽은 데이터를 구분하는 함수이다.
int insertItemValue(QDomElement node, CameraData *data, int count)
구분된 정보를 저장하는 함수이다.

표 4-46 MappingCData Class

그림 4-16은 Reader, Parser, Description을 상속하는 클래스 모습이다. virtual 클래스인 세 클래스를 상속 받아 실제 구현된 클래스이다.

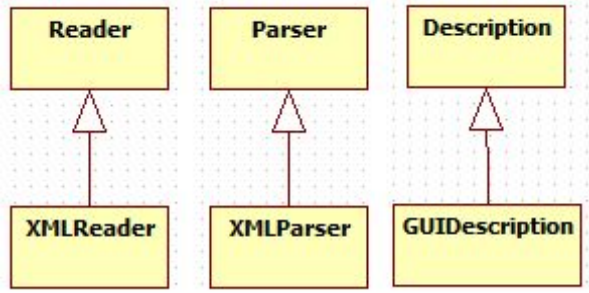

그림 4-16 Reader, Parser, Description 상속관계

XMLReader는 Virtual 클래스인 Reader 클래스를 상속받아 실제 함수를 구현하고 Capability list(XML) 파일을 읽어서 GuiData로 저장하는 클래스로 XML 형식으로 구성된 Capability List에 대해서만 읽을 수 있다.

GuiData* getData()
GuiData를 반환하는 함수이다.
void divideData(int i)
읽어온 Capability List(XML)를 gui 정보와 camera 정보로 나눈다.

표 4-47 XML의 각 Class

XMLParser는 Virtual 클래스인 Parser 클래스를 상속받아 실제 함수를 구현하고 저장된 데이터 클래스를 판단하고 적용해야 할 부분으로 구분하는 기능을 담당하는 클래스이다. XMLReader로 읽은 데이터인 GuiData의 내용을 구분한다.

GUIDescription는 Virtual 클래스인 Description 클래스를 상속받아 실제 함수를 구현하고 구분되어진 정보를 바탕으로 GUI를 생성하는 기능을 담당하는 클래스이다. Parser에서 만들어진 GuiData에 GUI들의 위젯 정보를 저장하고 해당 위치에 출력하고 각 액션을 저장한다.

Capability list가 XML 형식이 아닐 경우는 Reader를 상속 받는 OtherReader

Class를 생성하면 사용할 수 있다. 아래 그림4-17은 그 예를 보여주고 있다.

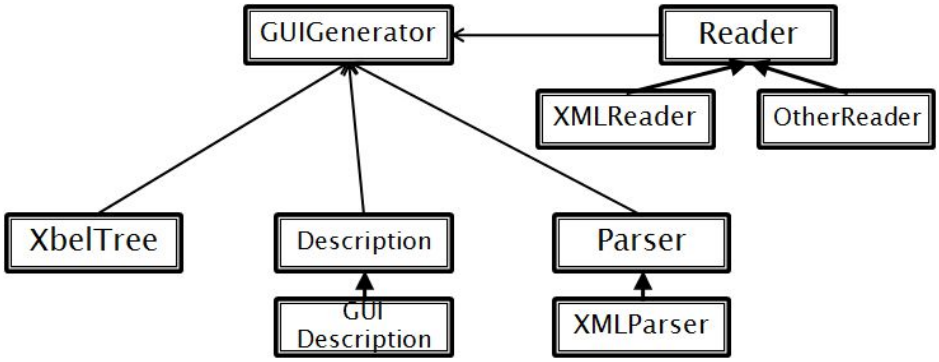

그림 4-17 Capability list가 XML이 아닐 경우

제 5 장 결과 및 토의

그림 5-1은 프로그램이 실행되고 아무런 동작도 하지 않은 초기 화면이다.

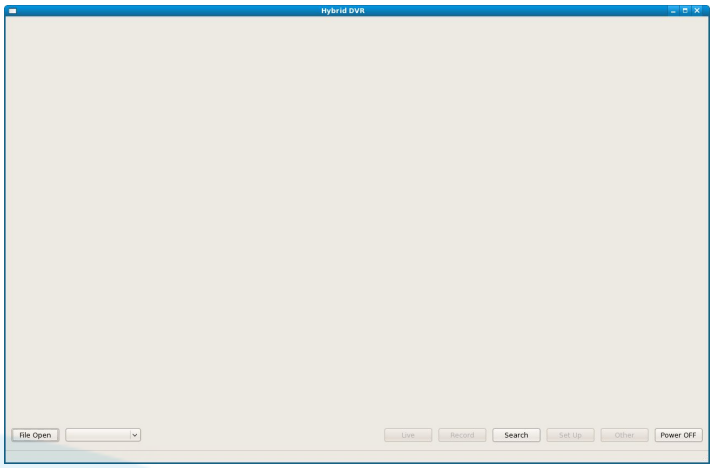

그림 5-1 초기화면

왼쪽 아래 File Open 버튼을 클릭하게 되면 XML 파일을 읽어서 연결 가능한 카메라의 정보와 적용 가능한 UI 정보를 프로그램에 적용한다. 그림 5-2은 적용된 모습

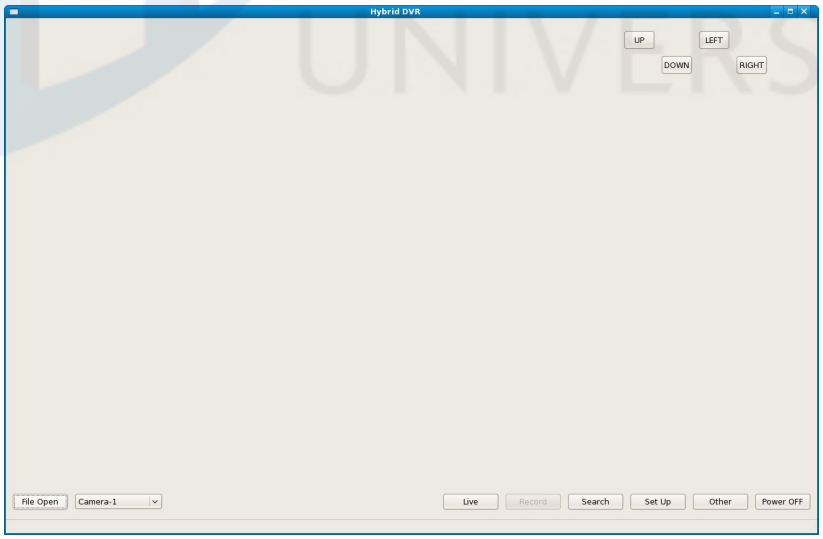

그림 5-2 UI가 적용된 화면

이고 왼쪽 아래 보이는 적용 가능한 리스트에서 선택하여 적용시킨다. 오른쪽 위 UI와 Setup, Other 버튼 등이 활성화되고 생성된다.

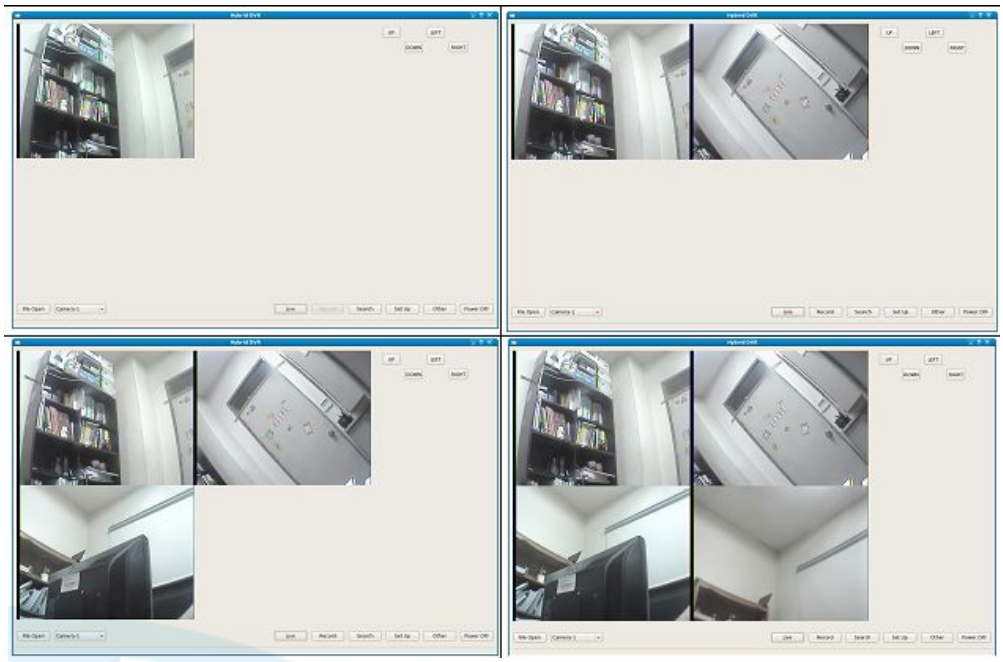

그림 5-3 라이브 뷰 실행 화면

그림 5-3는 카메라의 정보를 읽어 온 후 카메라수에 따라 LiveView를 보여준 모습이다. 이 프로그램은 1개부터 4개 까지의 카메라를 Live로 볼 수 있다. 카메라 정보가 적용된 뒤 Live 버튼을 누르면 실행된다.

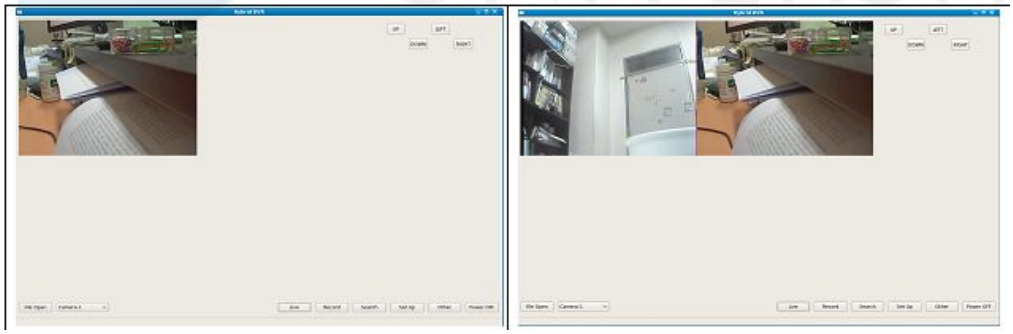

그림 5-4 로컬과 네트워크 동시 구동 모습

그림 5-3 로컬 카메라만 연결된 모습이고, 그림 5-4 네트워크 카메라를 연결한 모습과 네트워크 카메라와 로컬카메라가 연결된 모습이다. 구동방식은 동일하고

Configuration(XML)을 수정하여 원하는 카메라를 연결할 수 있다.

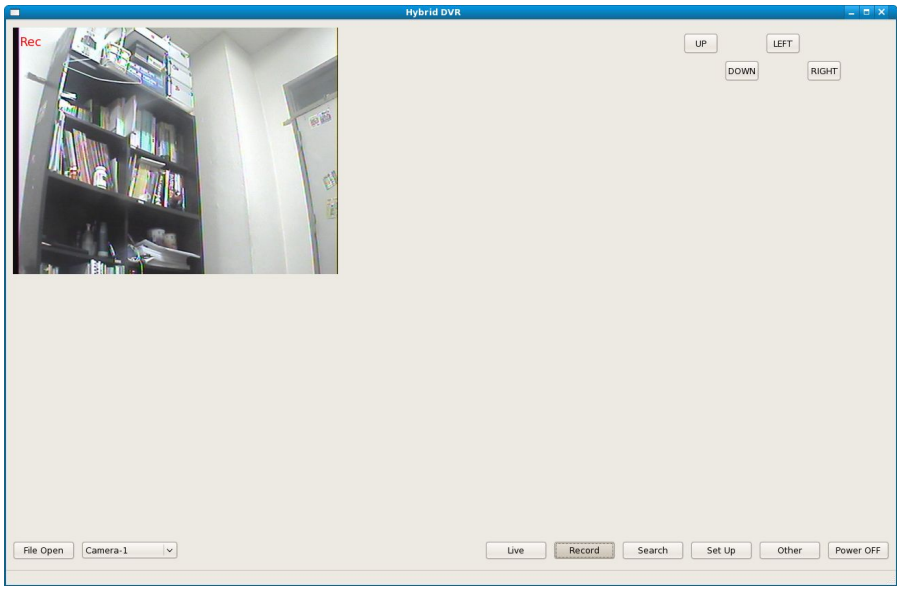

그림 5-5 녹화 모습

그림 5-5은 하나의 카메라가 녹화되는 모습이다. 녹화될 때 화면 왼쪽 위에 빨간 글씨로 Rec 라고 표시 해준다. 녹화는 LiveView를 보고 있을 때 실행 가능하며 Record 버튼을 클릭하면 녹화된다.

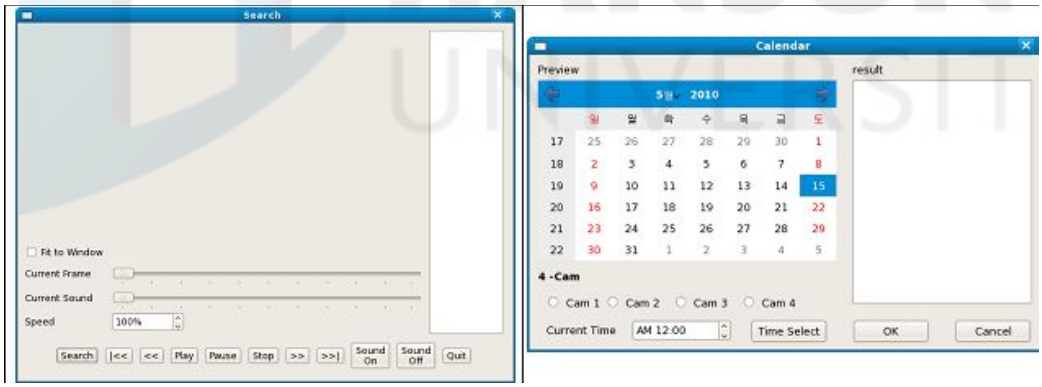

그림 5-6 Replay를 위한 화면

그림 5-6은 동적으로 생성된 UI는 오른쪽 위의 UI외에 다른 다이얼 창이 있고 다음은 그것들을 보인다. 현재 실제 기능을 가지 않고 어떻게 생성가능한지를 보여준다.

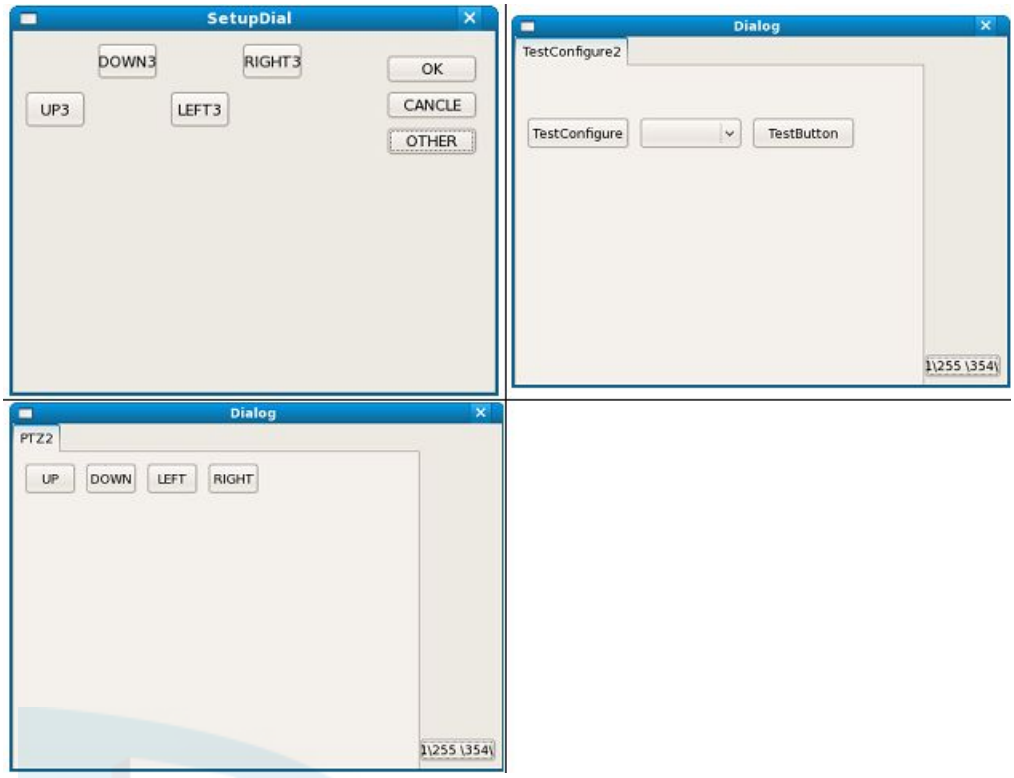

그림 5-7 동적 UI 적용 모습

그림 5-7는 녹화된 영상을 재생하여 다시 보기 위해 저장된 파일을 찾고 실행하는 모습을 보여 준다. 왼쪽은 영상이 재생되는 것을 보여주고 제어하는 부분이고 오른쪽 모습은 저장된 날, 시간, 카메라 번호를 기준으로 저장된 영상을 찾는 부분이다.

대규모 CMS 구성에 필요한 소프트웨어 플랫폼을 지향하는 Hybrid DVR을 위한 소프트웨어 플랫폼을 설계하고 프로토타입을 구현함으로써, 참여 기업이 보유하고 있던 IP Surveillance 시장에 필요한 요소 기술들을 효과적으로 활용할 수 있는 기반을 확보하였다.

다양한 네트워크 카메라를 위한 스트리밍 인터페이스 기술과 새로운 코덱 기술을 수용할 수 있는 유연한 시스템 구조를 확보하였다. 그래픽 사용자 인터페이스와 관련하여 문서화 방법론과 GUI 자동 생성에 관한 기술적 절차 등을 확보하였다.

제 6 장 결론 및 향후 연구

본 논문에서는 보안 영상 시장이 IP Surveillance 시스템으로 시장이 전환되는 과정에 독립적으로 DVR로서의 기능을 수행하고 IP 카메라, 네트워크 장치 등을 수용할 수 있는 하이브리드 DVR을 위한 소프트웨어 플랫폼을 설계하였다. 설계된 네트워크 기반 DVR 통합 관리 소프트웨어 플랫폼은 다양한 하드웨어, 소프트웨어 기능들을 표준 API 기반으로 모듈화하여, DVR 업체들이 이전에 가지고 있던 기술에 대한 재사용성을 높이고 새로운 기능을 빠르게 추가할 수 있으며, 다양한 제3자 기술을 적은 비용으로 수용할 수 있다. 만들어진 소프트웨어 플랫폼은 그대로 대규모 CMS에 적용될 수 있다.

【참고문헌】

1. 국내문헌

유장희, 「지능형 영상보안기술 현황 및 동향」, 전자통신동향분석, 2008년 8월

김선호, 조준래, 이민석, 「하이브리드 DVR을 위한 소프트웨어 플랫폼」, 임베디드 공학회, 2009년

2. 국외문헌

Jasmin Blanchette, Mark Summerfield. 2008. C++ GUI Programming with Qt4, 2nd edition.

3. 웹 사이트

Onvif, http://www.onvif.org

PSIA, http://www.psialliance.org

Ffmpeg, http://www.ffmpeg.org

Ffmpeg, http://ffmpeg.mplayerhq.hu

Ffmpeg, http://en.wikipedia.org/wiki/FFmpeg

V4L, http://linux.bytesex.org/v4l2

V4L, http://en.wikipedia.org/wiki/Video4Linux

AXIS, http://www.axis.co.kr

ABSTRACT

Software Platform for Hybrid DVR.

Junrae Cho
Major in Computer Engineering
Dept. of Computer Engineering
Graduate School, Hansung University

The global security video market is moving fast to IP surveillance systems from analog systems. IP surveillance systems consist of a massive number of high definition IP cameras, network video recorders, and centralized monitoring system. In the middle of the transition, DVR manufacturers are building hybrid DVR systems which incorporate both analog and IP cameras with HD, D1 video quality and more CMS functions. This study is to propose and prototype a software platform for hybrid DVRs. With this software platform which defines layers and modules, DVR manufacturers can easily reuse their legacy technologies and interface their DVRs with many other IP surveillance products. In this study, we also designed a automatic GUI generator to configure and control new devices without any coding.