

# C++ 벡터 및 클래스 활용 MFC 윈도우 응용 프로그램

작성자: 김진희

비록 간단한 프로젝트이지만 최소 이 정도의 윈도우 프로그램은 개발이 가능함을 보여주셨으면 하는 바램에  
이 자료를 만들었습니다.

# 목차

1.	프로그램 실행 화면	3
2.	프로그램의 기능	4
3.	Visual Studio 프로젝트 생성 설정	5
4.	Visual Studio 리소스 뷰 – Dialog	6
5.	연결할 장치 정보 정의	7
6.	프로그램 구조	8
7.	이 프로젝트의 의의	9

# 프로그램 실행 화면

신호복조장치	상태	설정	적용	설정 가져오기	연결 확인	닫기
광대역 수신판 #1	■	▼	적용	설정 가져오기	연결 확인	
협대역 수신판 #1	■	▼	적용	설정 가져오기		
협대역 수신판 #2	■	▼	적용	설정 가져오기		
협대역 수신판 #3	■	▼	적용	설정 가져오기		
협대역 수신판 #4	■	▼	적용	설정 가져오기		
방향탐지기 수신판 #1	■	▼	적용	설정 가져오기		닫기

이 프로젝트는 제가 대한민국 대표 방산업체 LIG 넥스원에 3개월 단기파견근무 당시 계약 만료 3일 전부터 개발한 프로그램입니다.

계약기간 종료 퇴사 이후 이 프로그램을 사용할 연구원들이 유연하게 수정할 수 있도록 클래스와 벡터를 활용하였습니다.

# 프로그램의 기능

## 프로그램 실행 화면



## 프로그램 기능

1. 프로그램을 처음 실행하거나, 실행 중에 연결 확인 버튼을 누르면, TCP 연결로 장치의 연결 여부를 확인하고, 연결 여부에 따라 장치 이름 옆에 빨간 불 혹은 초록 불을 보여줍니다.
2. 드롭다운 박스를 열면 정해져 있는 설정 목록이 나와 적용 버튼을 눌러서 선택한 설정으로 변경할 수 있게 하였습니다.

예: 광대역 수신판 #1 -> 협대역 수신판 #1

설정 변경은 SFTP로 설정 파일 전송하는 방식으로 구현하였습니다.

# Visual Studio 프로젝트 생성 설정

## ■ MFC, 대화상자 기반, Windows 소켓 사용

MFC 애플리케이션  
애플리케이션 종류 옵션

애플리케이션 종류  
대화 상자 기반

문서 템플릿 속성

사용자 인터페이스 기능

고급 기능

생성된 클래스

애플리케이션 종류(U)  
대화 상자 기반

애플리케이션 종류 옵션:  
☐ 탭 문서(B)  
☐ 문서/뷰 아키텍처 지원(V)

대화 상자 기반 옵션(O)  
<없음>

복합 문서 지원  
<없음>

문서 지원 옵션:  
☐ 활성 문서 서버(A)  
☐ 활성 문서 컨테이너(O)  
☐ 복합 파일 지원(U)

프로젝트 스타일  
MFC standard

비주얼 스타일 및 색(V)  
Windows Native/Default

비주얼 스타일 전환 사용(C)

리소스 언어(L)  
ko-KR

MFC 사용  
공유 DLL에서 MFC 사용

이전 다음 마침 취소

MFC 애플리케이션  
고급 기능 옵션

애플리케이션 종류

문서 템플릿 속성

사용자 인터페이스 기능

고급 기능

생성된 클래스

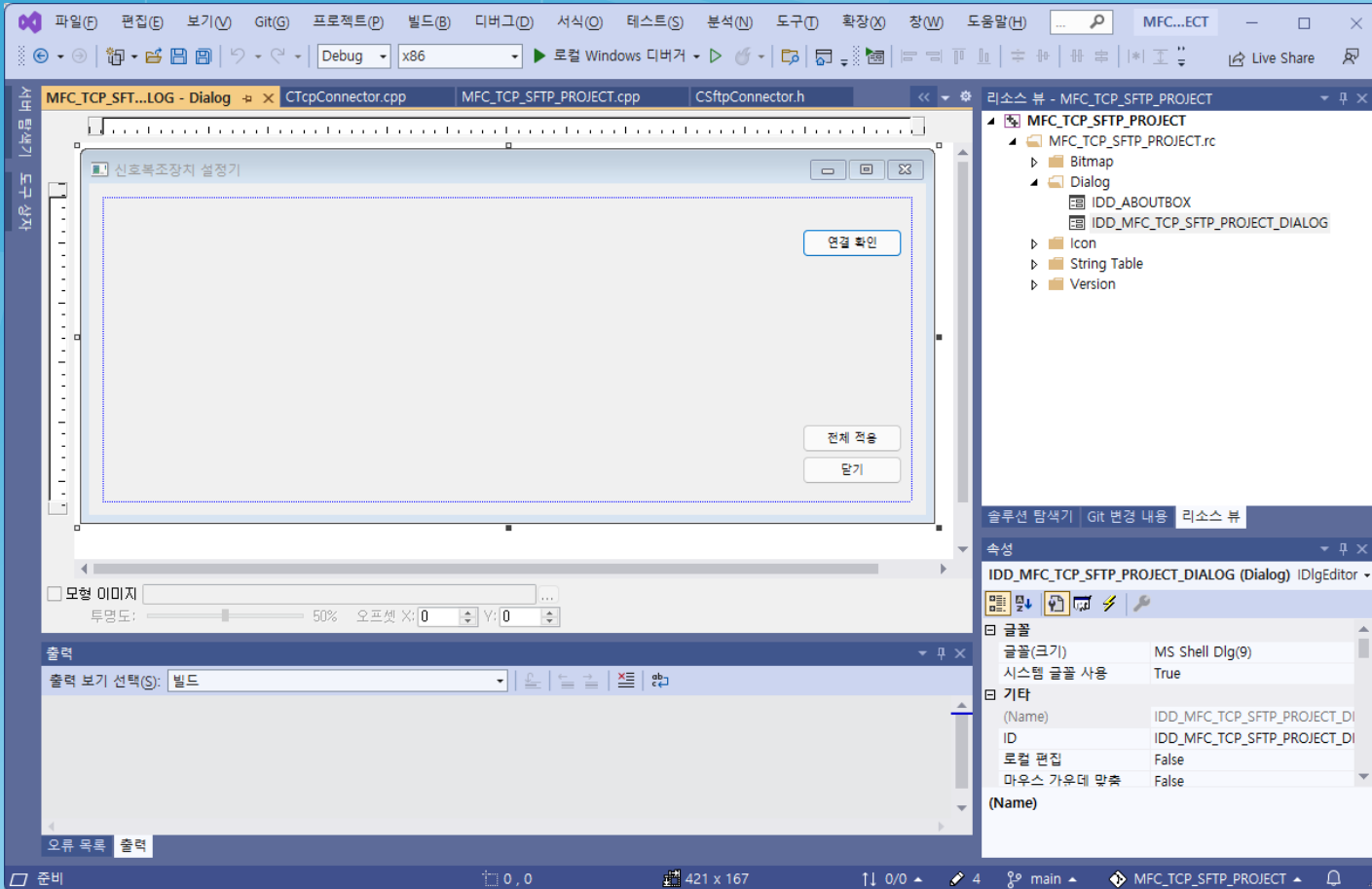
고급 기능:  
☒ 인쇄 및 인쇄 미리 보기(P)  
☐ 자동화(U)  
☒ ActiveX 컨트롤(R)  
☐ MAPI(메시징 API)(I)  
☒ Windows 소켓(W)  
☐ Active Accessibility(A)  
☒ 공용 컨트롤 매니페스트(M)  
☒ 다시 시작 관리자 지원(S)  
☒ 이전에 열려 있던 문서 다시 열기(U)  
☒ 애플리케이션 복구 지원(V)

고급 프레임 장:  
☐ 탐색기 도킹 창(D)  
☐ 출력 도킹 창(O)  
☐ 속성 도킹 창(S)  
☐ 탐색 창(T)  
☐ 캡션 표시줄(B)  
☐ 창을 표시하거나 활성화하는 고급 프레임 메뉴 항목(F)

최근 파일 목록의 파일 수(N)  
4

이전 다음 마침 취소

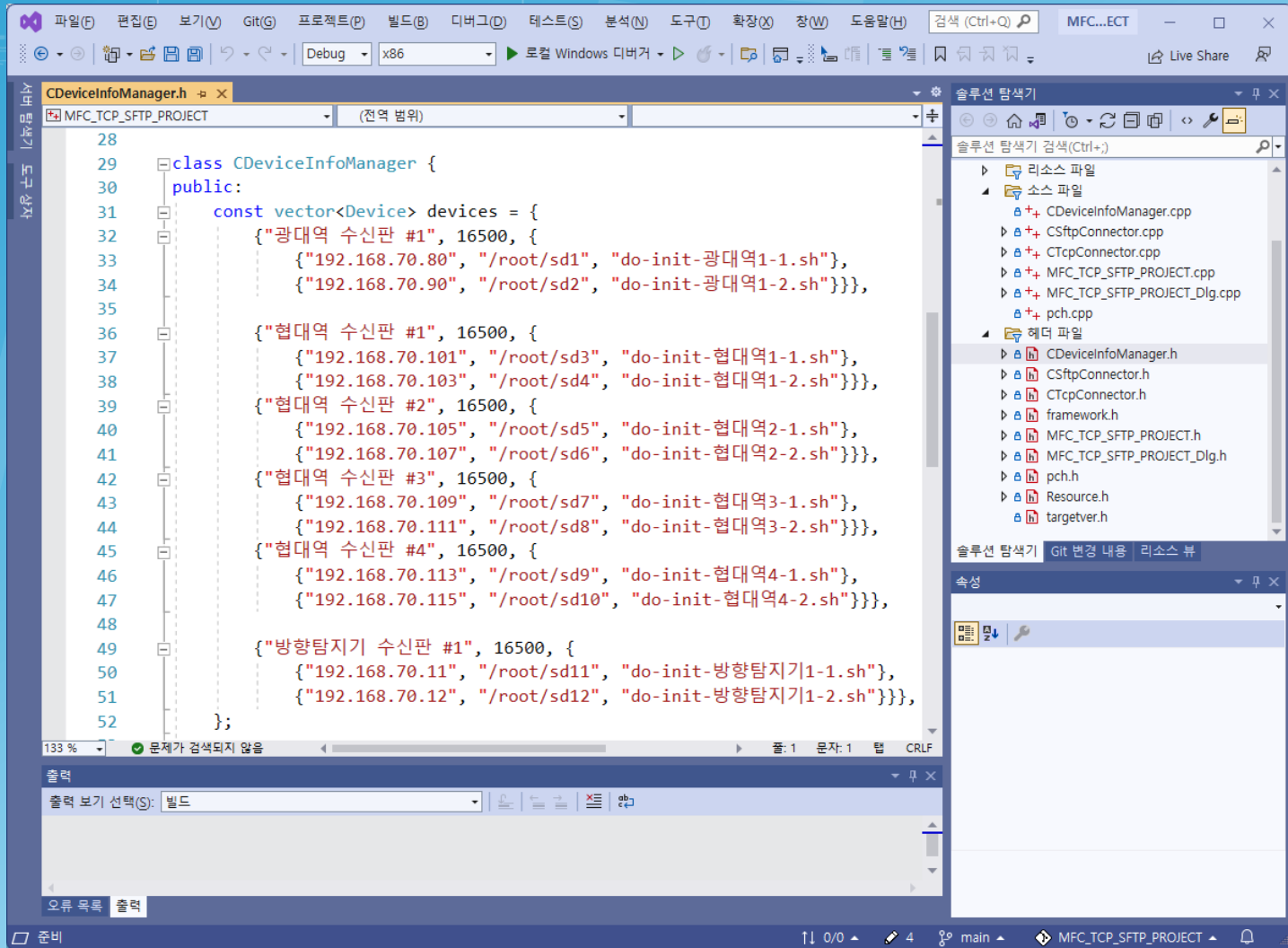
# Visual Studio 리소스 뷰 – Dialog



리소스 뷰의 다이얼로그에는  
연결 확인, 전체 적용, 닫기 버튼만 넣었고,

나머지 컨트롤은 정해 놓은 장치 정보에 따라  
자동 생성되게 하였습니다.

# 연결할 장치 정보 정의



```
28
29 class CDeviceInfoManager {
30 public:
31     const vector<Device> devices = {
32         {"광대역 수신판 #1", 16500, {
33             {"192.168.70.80", "/root/sd1", "do-init-광대역1-1.sh"},
34             {"192.168.70.90", "/root/sd2", "do-init-광대역1-2.sh"}},
35
36         {"협대역 수신판 #1", 16500, {
37             {"192.168.70.101", "/root/sd3", "do-init-협대역1-1.sh"},
38             {"192.168.70.103", "/root/sd4", "do-init-협대역1-2.sh"}},
39         {"협대역 수신판 #2", 16500, {
40             {"192.168.70.105", "/root/sd5", "do-init-협대역2-1.sh"},
41             {"192.168.70.107", "/root/sd6", "do-init-협대역2-2.sh"}},
42         {"협대역 수신판 #3", 16500, {
43             {"192.168.70.109", "/root/sd7", "do-init-협대역3-1.sh"},
44             {"192.168.70.111", "/root/sd8", "do-init-협대역3-2.sh"}},
45         {"협대역 수신판 #4", 16500, {
46             {"192.168.70.113", "/root/sd9", "do-init-협대역4-1.sh"},
47             {"192.168.70.115", "/root/sd10", "do-init-협대역4-2.sh"}},
48
49         {"방향탐지기 수신판 #1", 16500, {
50             {"192.168.70.11", "/root/sd11", "do-init-방향탐지기1-1.sh"},
51             {"192.168.70.12", "/root/sd12", "do-init-방향탐지기1-2.sh"}},
52     };
```

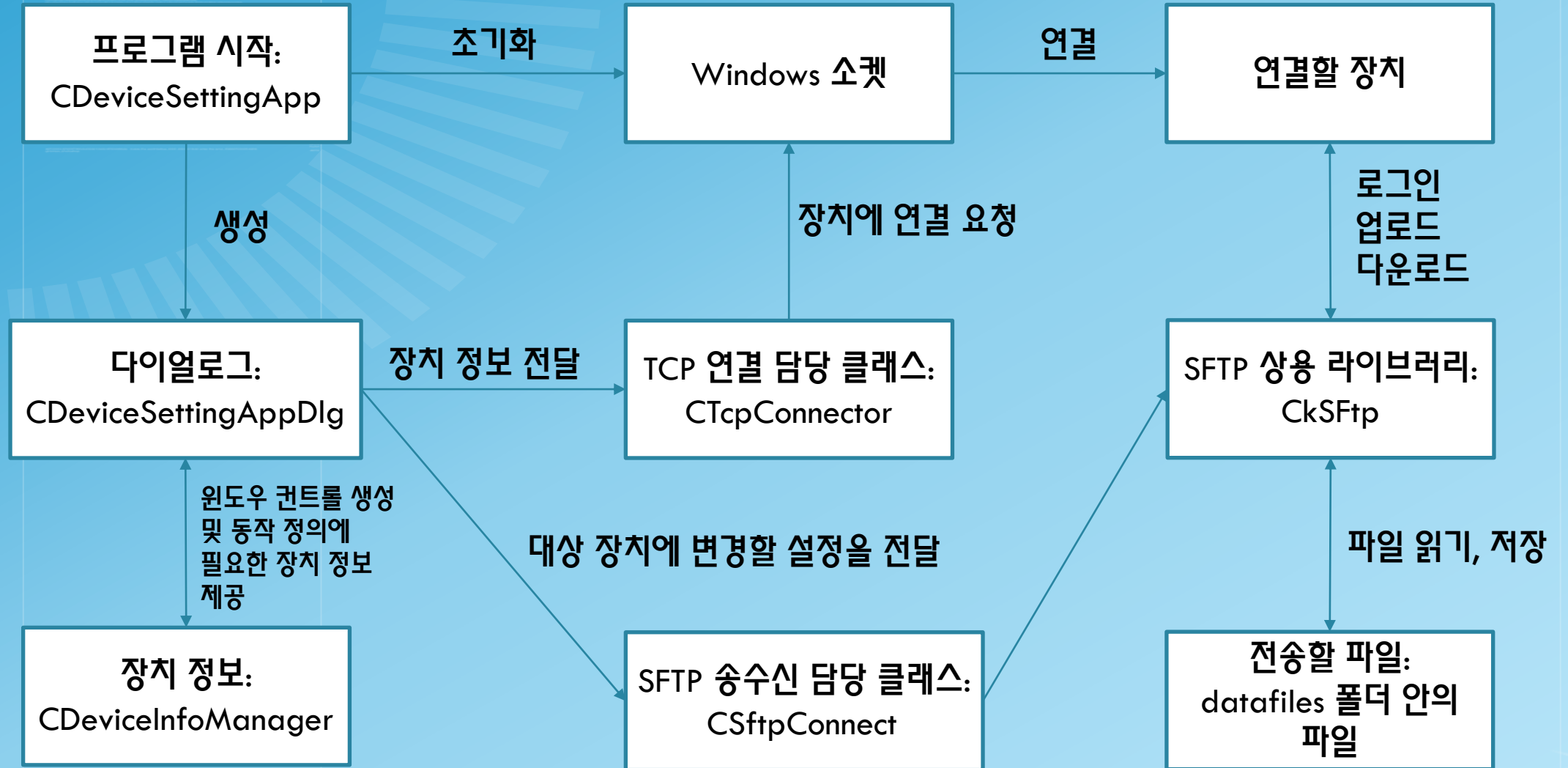
연결할 장치 정보는 CDeviceInfoManager.h  
파일에 왼쪽 사진과 같이 정의하였으며

앞서 리소스 뷰에서 정의한 다이얼로그의  
컨트롤 이외의 모든 컨트롤은 이 정보를 읽어서  
자동으로 생성하도록 구현하였습니다.

이것은 웹 앱에서 볼 수 있는 JSON과 모양이  
비슷하며, 이번 프로젝트를 제작하면서  
vector와 구조체를 이런 식으로 활용할 수  
있음을 알게 되었습니다.



# 프로그램 구조





# 이번 프로젝트의 의의 및 앞으로 나아갈 길

- 이 때까지 쌓아온 C++ 언어에 대한 지식을 활용한 첫 MFC 프로젝트입니다.
- 클래스와 벡터를 활용해 유지보수를 용이하게 하였습니다.
- 앞으로 C++ STL에 대해 더 공부하고, 유용하게 활용할 것입니다.

감사합니다.