2023년도
학사학위논문

MFC를 이용한 네트워크 분석기 튜닝

Network Analyzer Tuning Using MFC

2023년 11월

순천향대학교 공과대학
전기공학과

전준영 이태윤
박태우

MFC를 이용한 네트워크 분석기 튜닝

Network Analyzer Tuning Using MFC

지도교수 안 달

이 논문을 공학사학위 논문으로 제출함

2023년 11월

순천향대학교 공과대학
전기공학과

전준영 이태윤
박태우

전준영 이태윤
박태우 의 공학사학위논문을 인준함

2023 11월

심사위원 인
심사위원 인

순천향대학교 공과대학

전기공학과

초 록

네트워크 튜닝은 시스템이나 응용 프로그램의 성능을 최적화하기 위해 네트워크 구성을 조정하는 프로세스를 나타낸다. 현재 네트워크 튜닝에 대한 기대는 여러 측면에서 나타난다. 1. 성능 향상, 2. 자동화와 효율성, 3. 보안 강화 4. 5G 전개, 5. 클라우드 기술의 통합에 대한 이러한 기대들은 현재와 미래의 비즈니스 및 기술 환경에서 네트워크 튜닝이 가지는 중요성을 강조하고 있다. 네트워크 튜닝은 계속해서 기술적으로 발전하고 있다. 몇 가지 주요 동향들을 살펴보면 1. 자동화 및 인공지능(AI), 2. SDN(소프트웨어 정의 네트워크), 3. 5G 기술, 4. 클라우드 네이티브 기술, 5. 보안 강화 6. IoT(사물인터넷)와 같은 이러한 기술적인 발전은 향후에도 계속될 것으로 예상되며, 네트워크 튜닝은 더욱 효율적이고 유연한 형태로 진화할 것이다.

이러한 튜닝 기술을 사용하려면 네트워크 분석기를 사용할 줄 알아야 하는데 전공자가 아닌 비전공자들은 네트워크 분석기의 사용법을 잘 모르기 때문에 본 논문에서는 비전공자들도 네트워크 튜닝을 쉽게 완료할 수 있도록 도와주는 프로그램에 대한 연구를 하였다.

본 작품을 구현하기 위해서 아두이노를 사용하였다. 아두이노와 필터 나사 역할을 대신해주는 가변저항과 6개의 스위치의 가변저항 값을 측정하여 화면에 그래프 파형으로 나타난다. 스위치를 회전하여 가변저항 값의 변화에 따른 그래프 변화를 보면서 원하는 골든샘플에 도달하면 튜닝이 완료된다. 아두이노 코딩 값을 변경하며 서로 같은 방향으로만 회전이 아닌 서로 각기 다른 방향으로 회전하며 튜닝할 수 있도록 설정했다.

주요어 : 네트워크 분석기, 네트워크 튜닝, 아두이노, 가변저항, 골든샘플

차 례

초 록 ····· i
차 례 ····· ii
그 림 목 차 ····· iv
제 1 장 서 론 ····· 1
1.1 연구 배경 및 목적 ····· 1
1.2 네트워크 튜닝이 필요한 이유 ····· 2
1.3 튜닝 관련 선행 연구 고찰 ····· 3
제 2 장 본 론 ····· 5
2.1 네트워크 튜닝 ····· 5
2.1.1 네트워크 튜닝 조건 ····· 5
2.1.2 네트워크 튜닝 기술적인 발전 ····· 6
2.1.3 네트워크 분석기 ····· 7
2.2 RF 회로 ····· 9
2.2.1 RF 회로의 정의 ····· 9
2.2.2 RF 회로와 네트워크 튜닝의 관계성 ····· 10
2.2.3 RF 회로를 이용한 네트워크 튜닝 ····· 11
2.3 MFC 프로그래밍 ····· 12
2.3.1 MFC 프로그래밍 정의 ····· 12
2.3.2 MFC 프로그래밍과 네트워크 튜닝의 관계성 ····· 13
2.3.3 MFC를 이용한 네트워크 튜닝 ····· 14

2.4 아두이노 ····· 15
2.4.1 아두이노 정의 ····· 15
2.4.2 아두이노 코딩 ····· 17
2.5 제작 ····· 33
2.5.1 제작과정 ····· 33
2.5.2 네트워크 튜닝 완성 ····· 34
제 3 장 결 론 ····· 35
3.1 성능 평가 ····· 35
3.2 활용방안 및 결론 ····· 35

참 고 문 헌 ····· 36

ABSTRACT ····· 37

감사의 글 ····· 38

그 림 목 차

그림 1. DV 리우팅 알고리즘 네트워크 튜닝 ····· 3
그림 2. 스칼라 네트워크 분석기 ····· 8
그림 3. 벡터 네트워크 분석기 ····· 8
그림 4. RF 회로 ····· 9
그림 5. 아두이노 우노 보드 ····· 16
그림 6. MFC 프로그래밍을 이용한 코딩 ····· 32
그림 7. 아두이노 필터 ····· 33
그림 8. 튜닝 완료 시 그래프 ····· 34

제 1 장 서 론

1.1 연구 배경 및 목적

네트워크 튜닝은 시간이 지남에 따라 계속 발전해왔다. 초기에는 주로 하드웨어의 성능 향상에 중점을 두었지만, 최근에는 소프트웨어와 알고리즘의 최적화에 더 많은 주목이 가고 있다. 딥러닝에서는 가중치 초기화, 학습률 조절, 배치 정규화와 같은 기술들이 모델 성능을 향상시키는 데 중요한 역할을 하고 있다. 또한, 하이퍼파라미터 튜닝 및 자동화된 튜닝 도구의 등장으로 모델 최적화가 더욱 효율적으로 이루어지고 있다. 튜닝은 컴퓨터 네트워크를 최적화하고 성능을 향상시키는 과정을 의미하며 주로 네트워크의 대역폭, 지연 시간, 패킷 손실 등과 관련이 있다. 기업 및 조직에서 더 많은 데이터를 빠르게 전송하고 처리해야 하는 현대의 환경에서 더욱 중요해졌다. 따라서 효율적이고 최적화된 네트워크는 업무 효율성과 사용자 경험에 따라 큰 영향을 미치게 된다.

이에 본 연구에서는 MFC 프로그래밍을 이용하여 네트워크 분석기의 필터에 대한 지식이 부족하거나 튜닝에 어려움을 겪는 사람도 화면에 나타나는 프로그램의 지시에 따라 쉽게 튜닝을 완료할 수 있도록 도와주는 프로그램을 목적으로 연구를 진행했다.

1.2 네트워크 튜닝이 필요한 이유

네트워크 튜닝은 네트워크의 성능을 최적화하기 위해 수행되는 과정이다. 이는 여러 이유로 필요할 수 있다.

1. 트래픽 증가에 대비하여 대역폭을 최대한 활용하고 병목현상을 줄이기 위해 필요하다.

2. 응답 시간을 최소화하여 사용자 경험을 향상시키기 위한 목적도 있다. 또한 네트워크의 안정성과 신뢰성을 향상시키는 것이 중요하다. 네트워크 튜닝은 주로 패킷 손실, 지연, 대역폭 제한과 같은 문제를 식별하고 해결하는 데 중점을 둔다. 이를 통해 효율적인 데이터 전송과 높은 성능을 달성할 수 있다.

3. 적절한 네트워크 튜닝은 보안 측면에서도 중요하다. 방화벽 및 암호화 구성, 접근 제어 설정 등을 통해 네트워크 보안을 강화할 수 있다.

4. 효율적으로 조정된 네트워크는 자원 사용을 최적화하고 비용을 절감할 수 있다. 불필요한 대역폭 소비를 줄이고 최적의 하드웨어 및 소프트웨어 구성을 찾는 것은 비용 효율성을 향상시킨다.

5. 클라우드 컴퓨팅 및 가상화 기술의발전으로 인해 네트워크는 더 유연하고 확장 가능한 형태로 조정될 수 있다. 튜닝은 이러한 유연성과 확장성을 최대한 활용할 수 있도록 도와준다.

6. 사용자들의 요구사항은 다양하게 변화한다. 효과적인 네트워크 튜닝은 이러한 다향한 요구사항을 충족시키고 새로운 기술과 서비스에 대응할 수 있도록 한다. 네트워크 튜닝은 비즈니스 운영과 기술 인프라의 핵심 부분으로 인식되며, 효과적으로 구현되면 조직이 효율적으로 운영되고 혁신에 적응할 수 있도록 지원하므로 네트워크 튜닝이 필요한 이유이다.

1.3 튜닝 관련 선행 연구 고찰

네트워크 튜닝과 관련된 선행연구는 다양한 분야에서 이루어졌다. 몇 가지 주요 주제와 관련된 연구들을 살펴보면 다음과 같다.

1. 연결관리, 패킷 스케줄링, 리우팅 알고리즘을 최적화하는 연구들이 있다.

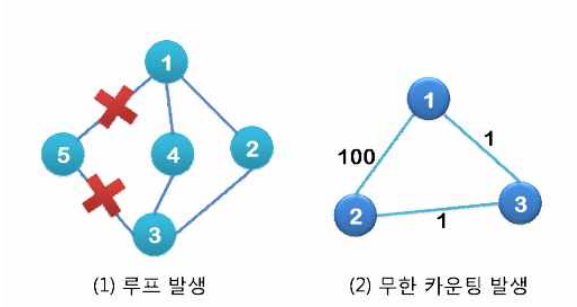

그림 1. DV 리우팅 알고리즘 네트워크 튜닝[1]

2. 대역폭을 효율적으로 활용하기 위한 알고리즘 및 매커니즘에 대한 연구들이 진행됐다.

3. Quality of Service(QOS) 향상시키기 위한 방법과 기술적인 측면에서의 연구가 이루어졌다.

4. 무선통신에서의 튜닝과 관련된 연구도 많이 이루어져 왔다. 이는 주로 무선 환경에서의 신호 간섭, 전력 소모 최적화 등을 다룬다.

5. 다양한 네트워크 시나리오에서의 성능을 시뮬레이션하고 평가하는 연구도 있다.

6. 시간 트래픽에 대한 효과적인 관리를 위한 연구도 있다.

7. Software-Defined Networking (SDN) 및 네트워크 가상화에 관한 연구에서는 네트워크의 소프트웨어 중심 제어 및 가상화 기술을 적용하여 유연하고 효율적인 네트워크를 구축하는 방법에 대한 연구가 진행됐다.

8. 네트워크에서 발생하는 플로우를 기반으로 한 동적인 관리 방법에 대한 연구가 이루어졌다. 특정 플로우의 특성을 파악하고 최적의 경로를 동적으로 설정하는 방법 등이 다루어졌다.

9. 클라우드 환경에서의 네트워크 튜닝과 데이터 센터 최적화에 대한 연구에서는 가상화, 컨테이너화, 자동 확장과 같은 기술을 활용하여 대규모 네트워크 인프라를 효율적으로 관리하는 방법에 중점이 둔 연구가 수행되었다.

10. 보안 측면에서의 네트워크 튜닝 연구에서는 침입 탐지, 트래픽분석, 암호화 및 인증 등을 통해 네트워크의 안전성을 강화하는 방법에 대한 연구가 이루어졌다.

11. 머신 러닝과 인공 지능을 활용하여 네트워크의 이상 감지, 예측 분석 및 자동화된 네트워크 관리에 관한 연구가 활발히 진행되고 있다. 이러한 선행 연구들은 네트워크 튜닝 분야에서의 다양한 도전과제에 대한 이해를 높이고, 향후 발전을 위한 기반을 제공한다.

제 2 장 본 론

2.1 네트워크 튜닝

2.1.1 네트워크 튜닝 조건

네트워크 튜닝은 다양한 조건과 맥락에서 이루어질 수 있다. 몇 가지 일반적인 네트워크 튜닝 조건은 다음과 같다.

1. 네트워크를 튜닝할 때 트래픽의 특성을 고려해야 한다. 예를 들어, 어떤 응용 프로그램이 주로 사용되는지, 피크 시간이 언제인지 등을 고려한다.

2. 네트워크 장비의 성능 및 용량은 튜닝에 영향을 미치는 중요한 요소이다. 라우터, 스위치, 서버 등의 성능을 고려하여 최적의 조건을 찾는다.

3. 네트워크를 사용하는 사용자 수와 그들의 요구사항은 튜닝에 영향을 미친다. 증가하는 트래픽에 대응하기 위해 조정이 필요할 수 있다.

4. 사용되는 응용 프로그램의 종류와 특성에 따라 다양한 튜닝이 필요할 수 있다. 예를 들어, 음성 통화와 데이터 전송을 처리하는 응용 프로그램은 다른 튜닝이 필요하다.

5. 보안 정책 및 요구사항에 따라 네트워크를 튜닝할 필요가 있다. 암호화, 인증, 방화벽 설정 등이 고려되어야 한다.

6. 예산 및 자원 제약은 네트워크 튜닝의 중요한 제약 사항 중 하나이다. 가용한 자원과 비용을 고려하여 최적의 튜닝 방법을 결정한다.

이러한 조건들을 고려하여 네트워크 튜닝을 수행하면 효과적으로 성능을 향상시키고 최적의 운영 환경을 조성할 수 있다.

2.1.2 네트워크 튜닝 기술적인 발전

네트워크 튜닝은 계속해서 기술적으로 발전하고 있다. 몇 가지 주요 동향은 다음과 같다.

1. 네트워크 튜닝에서는 자동화와 인공 지능 기술이 증가하고 있다. 이를 통해 시스템은 실시간으로 네트워크 성능을 모니터하고 조정할 수 있다.

2. SDN (Software-Defined Networking)은 네트워크 관리를 소프트웨어 중심으로 이동시키는 기술로, 네트워크 튜닝에 유연성을 부여하고 자원 할당을 효과적으로 관리할 수 있게 한다.

3. 5세대 이동통신 기술인 5G는 높은 대역폭, 낮은 지연 시간, 대규모 디바이스 연결을 제공하여 네트워크 성능 향상에 기여한다.

4. 클라우드 네이티브 기술은 가상화, 컨테이너화, 마이크로서비스 등을 통해 네트워크 구조를 더욱 유연하게 만들어준다.

5. 튜닝은 보안 측면에서도 진화하고 있다. 암호화 기술, 네트워크 모니터링, 위협 탐지 시스템 등이 향상되어 네트워크를 더욱 안전하게 만든다.

6. 다양한 디바이스 간의 연결이 증가함에 따라, 네트워크 튜닝은 대량의 데이터 및 다양한 디바이스를 지원할 수 있는 성능을 제공해야 한다.

이러한 기술적인 발전은 향후에도 계속될 것으로 예상되며, 네트워크 튜닝은 더욱 효율적이고 유연한 형태로 진화할 것이다.

2.1.3 네트워크 분석기

네트워크 분석기는 컴퓨터 네트워크에서 데이터 흐름과 패턴을 모니터링하고 분석하는 도구이다. 이 도구들은 네트워크 성능 감지, 보안 감지, 트래픽 분석 등 다양한 용도로 사용된다. 주로 패킷 수준에서 데이터를 캡처하고, 분석하여 네트워크 동작을 이해하고 문제를 해결하는 데 도움을 준다. 또한 RF 회로망을 분석해 주는 기계이고, 종류로는 진폭만을 분석하는 스칼라 네트워크 분석기와 진폭과 위상을 분석해주는 벡터 네트워크 분석기가 있다. 일반적으로 네트워크 분석기는 다음과 같은 주요 기능을 제공한다.

1. 패킷 캡처 및 분석에서는 네트워크 상에서 전송되는 패킷을 캡처하고 분석하여 데이터 전송의 세부사항을 확인한다.
2. 다양한 네트워크 프로토콜에 대한 분석을 통해 네트워크 동작을 이해하고 프로토콜 수준에서의 문제를 식별한다.
3. 트래픽 통계 및 성능 감지는 네트워크의 트래픽 통계를 수집하고 성능지표를 모니터링하여 병목현상이나 성능 저하를 식별한다.
4. 보안 감지에서는 이상 행동을 감지하고 보안 위협에 대한 트리거를 설정하여 네트워크 보안을 강화한다.
5. 로그 기록 및 분석에서는 네트워크 활동에 대한 로그를 기록하고 분석하여 잠재적인 문제나 보안 위협을 추적한다.
6. 수집된 데이터를 시각적으로 표현하여 사용하자 쉽게 이해하고 빠르게 대응할 수 있도록 도와준다. 주로 wireshark, tepdump, solarwinds 등이 널리 사용되는 네트워크 분석 도구이다.

그림 2. 스칼라 네트워크 분석기

그림 3. 벡터 네트워크 분석기

2.2 RF 회로

2.2.1 RF 회로의 정의

RF 회로는 Ridio Frequency 라디오 주파수로 다른 말로는 무선 주파수라고 불린다. 무선통신 시스템에서 사용된 회로이며, 이 회로는 주로 안테나, 필터, 증폭기, 믹서, 주파수 발생기 등의 다양한 구성 요소로 이루어져 있다. 안테나는 RF 신호를 전파로 변환하고, 필터는 원하는 주파수를 선택하거나 원하지 않는 신호를 차단하는 역할을 한다. RF 증폭기는 신호를 증폭하여 전송 범위를 확장하고, 믹서는 다양한 주파수를 혼합하여 새로운 주파수를 생성한다. 주파수 발생기는 정확한 주파수를 생성하여 통신의 안정성을 유지한다. 이러한 구성 요소들은 다향한 무선통신 시스템에서 사용되며, RF 회로의 설계는 신호의 안정성, 전송거리, 대역폭 등을 고려하여 이루어진다.

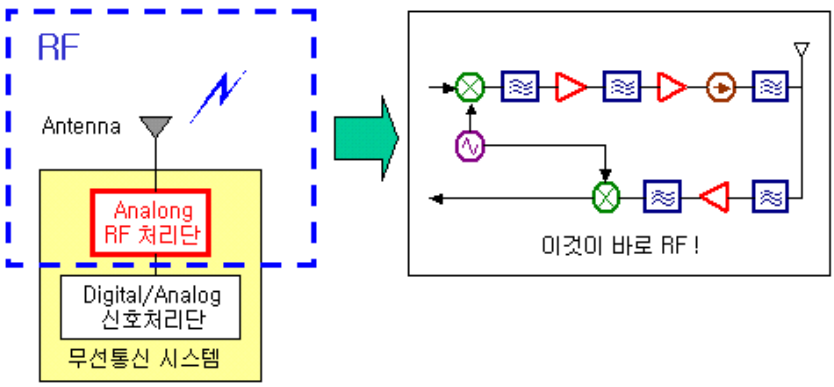

그림 4. RF 회로

2.2.2 RF 회로와 네트워크 튜닝의 관계성

RF 회로와 네트워크 튜닝은 물리적인 신호 처리와 네트워크 성능 최적화 간에 밀접한 관련이 있다. RF 회로와 네트워크 튜닝의 관계성을 보면 다음과 같다.

1. RF 회로는 무선통신에서 사용되는 회로로, 신호의 무결성과 대역폭 효율성이 매우 중요하다. 네트워크 튜닝은 이러한 RF 회로의 특성을 고려하여 대역폭을 효과적으로 관리하고 신호 무결성을 유지하는 데 기여할 수 있다.

2. RF 회로에서는 안테나의 튜닝이 중요하다. 네트워크 튜닝은 안테나의 방향성과 효율성을 높이기 위한 작업을 수행할 수 있다.

3. RF 회로는 주파수 스펙트럼에서 다양한 신호와의 간섭에 노출될 수 있다. 네트워크 튜닝은 이러한 간섭을 최소화하고 신호 간의 상호 작용을 최적화하는 데 도움을 줄 수 있다.

4. 무선 네트워크는 RF 신호를 기반으로 하며, 이를 최적화하는 것이 중요하다. 네트워크 튜닝은 무선 네트워크의 성능을 향상시키고 안전성을 유지하기 위해 RF 신호를 조절하고 최적화할 수 있다.

5. RF 회로의 튜닝을 통해 전송 속도를 최적화하고 신호 강도를 적절히 조절함으로써 네트워크 성능을 향상시킬 수 있다.

6. RF 회로에서 발생하는 소음과 왜곡은 신호의 정확성을 저해할 수 있다. 네트워크 튜닝은 이러한 소음과 왜곡을 최소화하고 신호 품질을 개선하는 데 기여할 수 있다.

이러한 이유로 RF 회로와 네트워크 튜닝은 무선통신 및 네트워크 기술 분야에서 밀접한 협력이 필요한 영역이다. 네트워크 튜닝은 RF 회로의 특성을 이해하고 최적화함으로써 전체적인 통신 시스템의 성능을 향상시킬 수 있다.

2.2.3 RF 회로의 네트워크 튜닝

RF 회로를 사용한 네트워크 튜닝은 무선통신 시스템에서 주파수 도메인에서의 성능을 최적화하는 과정을 나타낸다. 주로 무선통신에서 사용되는 RF 회로의 튜닝은 다음과 같은 몇 가지 측면에서 이루어질 수 있다.

1. 안테나는 RF 회로에서 핵심적인 역할을 한다. 안테나의 크기, 방향성, 고리 특성 등을 조절하여 무선 신호의 효율과 전송 거리를 최적화할 수 있다.

2. RF 회로에서는 발신기의 출력을 조절하여 전송 신호의 세기를 최적화한다. 이는 통신 거리 및 신호 대 잡음 비율을 개선하는 데 도움이 된다.

3. RF 회로에서는 특정 주파수 대역을 선택하고, 주파수 스펙트럼을 분석하여 간섭을 최소화하고, 최적의 무선 채널을 선택한다.

4. 무선통신에서는 신호와 노이즈의 비율을 최대화하고 신호의 왜곡을 최소화하는 튜닝이 중요하다. 이를 통해 통신 품질을 향상시킬 수 있다.

5. RF 회로에서는 반송파 주파수를 정확하게 튜닝하여 통신 시스템 간의 동기화를 유지하고 다중 접속을 관리한다.

6. RF 회로는 수신된 신호의 강도, 품질 및 기타 특성을 감지하고 평가하여 동적으로 네트워크에 대한 최적의 조건을 찾을 수 있다.

7. RF 회로에서는 증폭기 및 필터를 조절하여 신호의 증폭과 정제를 수행한다.

RF 회로를 사용한 네트워크 튜닝은 주로 무선통신 시스템에서 적용되며, 전송 품질, 신호 간섭 및 효율성을 향상시키기 위한 목적으로 이루어진다. 특히, 무선통신에서는 주파수 도메인에서의 튜닝이 중요하므로 RF 회로의 최적화는 전체적인 무선통신 성능에 큰 영향을 미친다.

2.3 MFC 프로그래밍

2.3.1 MFC 프로그래밍 정의

MFC(Microsoft Foundation Classes) 프로그래밍은 Microsoft가 제공하는 C++ 기반의 프레임워크를 사용하여 Windows 응용 프로그램을 개발하는 것을 말한다. MFC는 Windows API를 추상화하고 객체 지향적인 접근을 도입하여 프로그래머가 Windows 응용 프로그램을 쉽게 개발할 수 있도록 도와준다. MFC 프로그래밍에는 몇 가지 핵심 개념이 포함되어 있는데 다음과 같다.

1. MFC는 C++ 언어를 기반으로 하며, 클래스와 객체 지향 프로그래밍의 개념을 사용한다.
2. MFC는 윈도우와 다양한 컨트롤(버튼, 텍스트 상자 등)을 쉽게 생성하고 관리할 수 있는 클래스를 제공한다.
3. 사용자 입력, 버튼 클릭과 같은 이벤트에 대한 처리를 간편하게 구현할 수 있다.
4. MFC는 문서-뷰 아키텍처를 지원하여 응용 프로그램의 데이터와 사용자 인터페이스를 분리할 수 있도록 도와준다.
5. 다양한 언어로 응용 프로그램을 개발하고 다국어 지원을 제공할 수 있다.
6. Visual Studio와 통합되어 디버깅 및 성능 프로파일링을 쉽게 수행할 수 있다.

MFC는 오래된 기술이지만 여전히 많은 Windows 응용 프로그램에서 사용되고 있다. 그러나 현대적인 Windows 개발에서는 다른 기술들 특히, WPF 나 UWP 같은 기술들이 더 많이 사용되고 있다.

2.3.2 MFC 프로그래밍과 네트워크 튜닝의 관계성

MFC 프로그래밍과 네트워크 튜닝은 주로 다른 영역에 속하는 기술이지만, 특정 응용 프로그램이나 시스템에서 함께 사용될 수 있다. 아래는 이 두 가지 기술 간의 관계에 대한 몇 가지 예시이다.

1. MFC를 사용하여 Windows 응용 프로그램을 개발하면서, 해당 응용 프로그램이 네트워크와 상호 작용하는 경우가 있다. 예를 들어, 사용자가 네트워크 연결을 모니터하거나 제어하는 도구를 개발할 때 MFC를 사용할 수 있다.
2. MFC를 이용하여 개발한 응용 프로그램에서는 UI 상에 네트워크 상태를 시각적으로 표시하는 기능을 추가할 수 있다. 이는 사용자에게 네트워크 연결의 상태를 알려주거나, 문제를 해결하는 데 도움이 될 수 있다.
3. 네트워크 튜닝이 필요한 상황에서, MFC를 사용하여 사용자가 네트워크 설정을 구성하고 관리할 수 있는 사용자 인터페이스를 개발할 수 있다.
4. MFC를 활용하여 개발한 응용 프로그램이 네트워크 성능을 모니터링하고 로깅할 수 있다. 이를 통해 네트워크 튜닝에 필요한 데이터를 수집하고 분석할 수 있다.
5. MFC를 사용하여 클라이언트-서버 기반의 응용 프로그램을 개발할 때, 네트워크 튜닝이 필요한 경우가 있다. 특히 대용량 데이터 전송이나 효율적인 통신을 위한 조치가 필요한 경우 MFC를 사용하여 구현할 수 있다.

이러한 방식으로 MFC와 네트워크 튜닝이 연결될 수 있으며, 특히 Windows 환경에서 응용 프로그램을 개발하는 경우에 이 두 기술이 함께 사용될 수 있다.

2.3.3 MFC를 이용한 네트워크 튜닝

MFC를 사용한 네트워크 튜닝은 주로 Windows 환경에서 GUI 기반의 응용 프로그램을 개발하고, 해당 응용 프로그램에서 네트워크 설정을 모니터링하고 조절하는 기능을 구현하는 데 관련된다. 아래는 MFC를 이용하여 네트워크 튜닝을 수행하는 기본적인 단계와 예시이다.

1. Visual Studio에서 MFC 프로젝트를 생성한다. MFC Application 템플릿을 선택하고 기본 설정을 구성한다.

2. MFC는 리소스 에디터를 통해 UI를 디자인할 수 있는 기능을 제공한다. 사용자에게 보여질 다양한 컨트롤 및 창들을 디자인한다.

3. MFC에서는 Windows 소켓 API를 사용하여 네트워크 통신을 구현할 수 있다. 필요한 경우 소켓 클래스를 사용하여 서버와 클라이언트 사이의 통신을 설정한다.

4. UI에는 현재 네트워크 연결 상태, 대역폭, 지연 시간과 같은 정보를 표시하는 컨트롤을 추가한다.

5. 사용자가 네트워크 설정을 구성할 수 있는 다이얼로그 상자를 만들어 추가한다. 예를 들어 IP 주소, 포트 번호, 프로토콜 선택 등을 구성할 수 있다.

6. 사용자가 지정한 설정을 기반으로 소켓 통신을 조정하거나 다른 네트워크 매개변수를 조절하는 기능을 구현한다.

7. 네트워크 튜닝의 결과를 사용자에게 알리기 위해 적절한 이벤트 처리기를 추가하고, 로깅을 통해 네트워크 동작 및 변경 사항을 기록할 수 있다.

8. 프로젝트를 빌드하고 테스트하여 네트워크 설정 변경 및 튜닝이 의도한 대로 작동하는지 확인한다.

이러한 단계를 통해 MFC를 사용하여 Windows 환경에서 네트워크 튜닝을 수행하는 기본적인 구현을 할 수 있다. 이 프로세스는 응용 프로그램의 목적에 따라 추가적인 기능이나 세부적인 조정이 필요할 수 있다.

2.4 아두이노

2.4.1 아두이노 정의

아두이노는 오픈 소스 전자 프로토 타이핑 플랫폼으로, 간단한 프로그래밍 및 하드웨어 조립을 통해 다양한 프로젝트를 만들 수 있는 매우 인기 있는 도구이다. 주로 입문자부터 전문가까지 다양한 수준의 사용자들에게 활용되고 있다. 아래는 아두이노에 대한 몇 가지 중요한 특정과 정보이다.

1. 아두이노 개발은 Arduino IDE를 통해 이루어진다. 이 IDE는 사용자가 아두이노 보드에 프로그램을 업로드하고 제어할 수 있는 간단하고 사용하기 쉬운 환경을 제공한다.

2. 아두이노 보드는 다양한 모델이 있지만, 대표적으로 Arduino Uno가 널리 사용된다. 보드에는 디지털 및 아날로그 입력/출력 핀, USB 연결 포트, 전원 연결 포트 등이 포함되어 있다.

3. 아두이노는 C 및 C++ 프로그래밍 언어를 기반으로 한다. 사용자는 간단한 스케치를 작성하여 다양한 하드웨어 모듈 및 센서와 상호 작용할 수 있다.

4. 아두이노는 다양한 확장 모듈 및 쉴드를 지원하여 사용자가 프로젝트에 필요한 다양한 하드웨어를 연결할 수 있도록 한다. 이로써 센서, 모터, 디스플레이, 무선통신 모듈 등을 추가될 수 있다.

5. 아두이노 플랫폼은 오픈 소스로 개발되어 다양한 사용자들이 개발에 참여하고 기여할 수 있다. 이는 커뮤니티의 활발한 지원과 다양한 라이브러리, 예제 코드를 제공한다.

6. 아두이노는 교육용으로도 널리 활용되며, 프로그래밍과 전자 공학의 기초를 학습하는 데에 적합하다. 수많은 교육 기관 및 강의에서 아두이노를 활용한 교육이 이루어지고 있다.

7. Arduino IDE는 Windows, macOS, Linux와 호환되어 다양한 운영체제에서 사용할 수 있다.

아두이노는 프로토타입 제작, 로봇 공학, IoT 프로젝트, 예술 작품 등 다양한 분야에서 활용되며, 진입 장벽이 낮고 다양한 자원이 제공되어 초보자부터 전문가까지 다양한 사용자에게 인기를 끌고 있다.

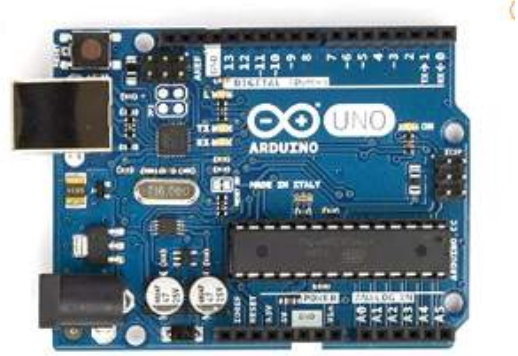

그림 5. 아두이노 우노 보드

2.4.2 아두이노 코딩

```
#include "pch.h"
#include "framework.h"
// SHARED_HANDLERS는 미리 보기, 축소판 그림 및 검색 필터 처리기를 구현하는 ATL 프로젝트에서 정의할 수 있으며
// 해당 프로젝트와 문서 코드를 공유하도록 해 줍니다.
#ifndef SHARED_HANDLERS
#include "SCHtuning.h"
#endif

#include "SCHtuningDoc.h"
#include "SCHtuningView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

UINT ThreadProc(LPVOID lParam)
{
    CSCHtuningView* pView = (CSCHtuningView*)lParam;

    while (true)
    {
        if (pView->m_thread_flag == 1)
            pView->proc_unit();
    }
    return 0;
}
```

```
IMPLEMENT_DYNCREATE(CSCHtuningView, CFormView)

BEGIN_MESSAGE_MAP(CSCHtuningView, CFormView)
    ON_WM_CONTEXTMENU()
    ON_WM_RBUTTONUP()
    ON_BN_CLICKED(IDC_BTN_START, &CSCHtuningView::OnBnClickedBtnStart)
    ON_BN_CLICKED(IDC_BTN_END, &CSCHtuningView::OnBnClickedBtnEnd)
    //ON_BN_CLICKED(IDC_BTN_INIT, &CSCHtuningView::OnBnClickedBtnInit)
    ON_BN_CLICKED(IDC_BTN_CW_01, &CSCHtuningView::OnBnClickedBtnCw01)
    ON_BN_CLICKED(IDC_BTN_CW_02, &CSCHtuningView::OnBnClickedBtnCw02)
    ON_BN_CLICKED(IDC_BTN_CW_03, &CSCHtuningView::OnBnClickedBtnCw03)
    ON_BN_CLICKED(IDC_BTN_CW_04, &CSCHtuningView::OnBnClickedBtnCw04)
    ON_BN_CLICKED(IDC_BTN_CW_05, &CSCHtuningView::OnBnClickedBtnCw05)
    ON_BN_CLICKED(IDC_BTN_CW_06, &CSCHtuningView::OnBnClickedBtnCw06)
    ON_BN_CLICKED(IDC_BTN_CCW_01, &CSCHtuningView::OnBnClickedBtnCcw01)
    ON_BN_CLICKED(IDC_BTN_CCW_02, &CSCHtuningView::OnBnClickedBtnCcw02)
    ON_BN_CLICKED(IDC_BTN_CCW_03, &CSCHtuningView::OnBnClickedBtnCcw03)
    ON_BN_CLICKED(IDC_BTN_CCW_04, &CSCHtuningView::OnBnClickedBtnCcw04)
    ON_BN_CLICKED(IDC_BTN_CCW_05, &CSCHtuningView::OnBnClickedBtnCcw05)
    ON_BN_CLICKED(IDC_BTN_CCW_06, &CSCHtuningView::OnBnClickedBtnCcw06)
    //ON_BN_CLICKED(IDC_BTN_SERVER_START, &CSCHtuningView::OnBnClickedBtnServerStart)
    //ON_BN_CLICKED(IDC_BTN_SERVER_STOP, &CSCHtuningView::OnBnClickedBtnServerStop)
    //ON_BN_CLICKED(IDC_RADIO_MAT, &CSCHtuningView::OnBnClickedRadioMat)
    //ON_BN_CLICKED(IDC_RADIO_MFC, &CSCHtuningView::OnBnClickedRadioMfc)
    //ON_EN_CHANGE(IDC_EDIT_RECV, &CSCHtuningView::OnEnChangeEditRecv)
    //ON_LBN_SELCHANGE(IDC_LIST_01, &CSCHtuningView::OnLbnSelchangeList01)
    ON_BN_CLICKED(IDC_BTN_USB_OPEN, &CSCHtuningView::OnBnClickedBtnUsbOpen)
    ON_BN_CLICKED(IDC_BTN_USB_CLOSE, &CSCHtuningView::OnBnClickedBtnUsbClose)
    ON_MESSAGE(WM_COMM_READ, OnCommunication)
```

```
64    END_MESSAGE_MAP()
65
66    // CSCHtuningView 생성/소멸
67
68    CSCHtuningView::CSCHtuningView() noexcept
69        : CFormView(IDD_SCHTUNING_FORM)
70    {
71        // TODO: 여기에 생성 코드를 추가합니다.
72
73        m_pThread = NULL;
74
75        m_SerialComm = NULL;
76
77    }
78
79    CSCHtuningView::~CSCHtuningView()
80    {
81        if (m_pThread != NULL)
82        {
83            delete m_pThread;
84            m_pThread = NULL;
85        }
86
87        if (m_SerialComm != NULL)
88        {
89            delete m_SerialComm;
90            m_SerialComm = NULL;
91        }
92    }
```

```
93
94    void CSCHtuningView::DoDataExchange(CDataExchange* pDX)
95    {
96        CFormView::DoDataExchange(pDX);
97    //  DDX_Control(pDX, IDC_LIST_01, m_List);
98    }
99
100   BOOL CSCHtuningView::PreCreateWindow(CREATESTRUCT& cs)
101   {
102       // TODO: CREATESTRUCT cs를 수정하여 여기에서
103       //  Window 클래스 또는 스타일을 수정합니다.
104
105       pView = this;
106
107       return CFormView::PreCreateWindow(cs);
108   }
109
110   void CSCHtuningView::OnInitialUpdate()
111   {
112       CFormView::OnInitialUpdate();
113       GetParentFrame()->RecalcLayout();
114       ResizeParentToFit();
115
116       RECT rect_win;
117       AfxGetMainWnd()->GetWindowRect(&rect_win);
118       AfxGetMainWnd()->MoveWindow(rect_win.left, rect_win.top, 1920, 1200);
119
120       GetParentFrame()->SetWindowText(_T("Tuning"));
121
```

```
122     m_endian_flag = 1;
123
124     CButton* pBtn = NULL;
125
126     //pBtn = (CButton*)GetDlgItem(IDC_RADIO_MFC);
127     //pBtn->SetCheck(false);
128
129     //pBtn = (CButton*)GetDlgItem(IDC_RADIO_MATLAB);
130     //pBtn->SetCheck(true);
131
132
133     CWnd* pWnd = NULL;
134
135     //pWnd = GetDlgItem(IDC_BTN_SERVER_START);
136     //pWnd->EnableWindow(true);
137
138     //pWnd = GetDlgItem(IDC_BTN_SERVER_STOP);
139     //pWnd->EnableWindow(false);
140
141     m_dc_w = 840;
142     m_dc_h = 200+50;
143
144     m_dc_x = 600;
145     m_dc_y = 30;
146
147     m_pt_filter = CPoint(40, 40);
148
149     m_pt_screw[0] = CPoint(3, 2);
150     m_pt_screw[1] = CPoint(130, 2);
```

```
151     m_pt_screw[2] = CPoint(257, 2);
152     m_pt_screw[3] = CPoint(384, 2);
153     m_pt_screw[4] = CPoint(511, 2);
154     m_pt_screw[5] = CPoint(638, 2);
155     //m_pt_screw[6] = CPoint(1048, 50);
156
157     m_thread_delay = 200;
158
159     m_cur = 0;
160
161     m_screw_id = 0;
162     m_rot_id = 0;
163
164     init_dc_buffer(this);
165
166     m_dc_x_image = 200;
167     m_dc_y_image = 300;
168     m_dc_w_image = 1110;
169     m_dc_h_image = 470;
170
171     init_dc_buffer_image(this);
172
173
174     m_thread_flag = 0;
175
176     m_Image_filter.Load(_T("filter_image.png"));
177
178     CString str;
179
```

```
180     for (int i = 0; i < 60; i++)
181     {
182         str.Format(_T("arrow_cw_0%d.png"), i + 1);
183         m_Image_cw[i].Load(str);
184
185         str.Format(_T("arrow_ccw_0%d.png"), i + 1);
186         m_Image_ccw[i].Load(str);
187     }
188
189     proc_unit();
190
191     m_port = 1;
192     m_baud_rate = 9600;
193
194     search_usb_port();
195
196     load_config();
197
198     m_graph_id_cur = 0;
199     m_graph_id_prev = 0;
200 }
201
202 void CSCHtuningView::OnRButtonUp(UINT /* nFlags */, CPoint point)
203 {
204     ClientToScreen(&point);
205     OnContextMenu(this, point);
206 }
207
208 void CSCHtuningView::OnContextMenu(CWnd* /* pWnd */, CPoint point)
```

```
210 #ifndef SHARED_HANDLERS
211     theApp.GetContextMenuManager()->ShowPopupMenu(IDR_POPUP_EDIT, point.x, point.y, this, TRUE);
212 #endif
213 }
214
215
216 // CSCHtuningView 진단
217
218 #ifdef _DEBUG
219 void CSCHtuningView::AssertValid() const
220 {
221     CFormView::AssertValid();
222 }
223
224 void CSCHtuningView::Dump(CDumpContext& dc) const
225 {
226     CFormView::Dump(dc);
227 }
228
229 CSCHtuningDoc* CSCHtuningView::GetDocument() const // 디버그되지 않은 버전은 인라인으로 지정됩니다.
230 {
231     ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CSCHtuningDoc)));
232     return (CSCHtuningDoc*)m_pDocument;
233 }
234 #endif //_DEBUG
235
236
237 // CSCHtuningView 메시지 처리기
238
```

```
322  void CSCHtuningView::OnBnClickedBtnStart()
323  {
324      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
325
326      m_thread_flag = 1;
327      m_cur = 0;
328
329      if (m_pThread == NULL)
330          m_pThread = AfxBeginThread(ThreadProc, this);
331  }
332
333
334  void CSCHtuningView::OnBnClickedBtnEnd()
335  {
336      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
337
338      if (m_pThread != NULL)
339      {
340          proc_stop();
341
342          m_thread_flag = 0;
343
344          HANDLE nThread = m_pThread->m_hThread;
345          DWORD nExitCode = 0;
346          ::TerminateThread(nThread, nExitCode);
347
348          delete m_pThread;
349          m_pThread = NULL;
350      }
```

```
368  void CSCHtuningView::OnBnClickedBtnCw01()
369  {
370      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
371
372      change_screw();
373
374      m_screw_id = 1;
375      m_rot_id = 0;
376
377      m_cur = 0;
378
379      m_screw_value[0] = "시계 방향으로 회전하세요";
380  }
381
382
383  void CSCHtuningView::OnBnClickedBtnCw02()
384  {
385      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
386
387      change_screw();
388
389      m_screw_id = 2;
390      m_rot_id = 0;
391
392      m_cur = 0;
393
394      m_screw_value[0] = "시계 방향으로 회전하세요";
395  }
396
```

```
398  void CSCHtuningView::OnBnClickedBtnCw03()
399  {
400      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
401
402      change_screw();
403
404      m_screw_id = 3;
405      m_rot_id = 0;
406
407      m_cur = 0;
408
409      m_screw_value[0] = "시계 방향으로 회전하세요";
410  }
411
412
413  void CSCHtuningView::OnBnClickedBtnCw04()
414  {
415      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
416
417      change_screw();
418
419      m_screw_id = 4;
420      m_rot_id = 0;
421
422      m_cur = 0;
423
424      m_screw_value[0] = "시계 방향으로 회전하세요";
425  }
426
```

```
428  void CSCHtuningView::OnBnClickedBtnCw05()
429  {
430      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
431
432      change_screw();
433
434      m_screw_id = 5;
435      m_rot_id = 0;
436
437      m_cur = 0;
438
439      m_screw_value[0] = "시계 방향으로 회전하세요";
440  }
441
442
443  void CSCHtuningView::OnBnClickedBtnCw06()
444  {
445      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
446
447      change_screw();
448
449      m_screw_id = 6;
450      m_rot_id = 0;
451
452      m_cur = 0;
453
454      m_screw_value[0] = "시계 방향으로 회전하세요";
455  }
456
```

```
458  void CSCHtuningView::OnBnClickedBtnCcw01()
459  {
460      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
461
462      change_screw();
463
464      m_screw_id = 1;
465      m_rot_id = 1;
466
467      m_cur = 0;
468
469      m_screw_value[0] = "시계 반대 방향으로 회전하세요";
470  }
471
472
473  void CSCHtuningView::OnBnClickedBtnCcw02()
474  {
475      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
476
477      change_screw();
478
479      m_screw_id = 2;
480      m_rot_id = 1;
481
482      m_cur = 0;
483
484      m_screw_value[0] = "시계 반대 방향으로 회전하세요";
485  }
486
```

```
488  void CSCHtuningView::OnBnClickedBtnCcw03()
489  {
490      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
491
492      change_screw();
493
494      m_screw_id = 3;
495      m_rot_id = 1;
496
497      m_cur = 0;
498
499      m_screw_value[0] = "시계 반대 방향으로 회전하세요";
500  }
501
502
503  void CSCHtuningView::OnBnClickedBtnCcw04()
504  {
505      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
506
507      change_screw();
508
509      m_screw_id = 4;
510      m_rot_id = 1;
511
512      m_cur = 0;
513
514      m_screw_value[0] = "시계 반대 방향으로 회전하세요";
515  }
516
```

```
518  void CSCHtuningView::OnBnClickedBtnCcw05()
519  {
520      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
521
522      change_screw();
523
524      m_screw_id = 5;
525      m_rot_id = 1;
526
527      m_cur = 0;
528
529      m_screw_value[0] = "시계 반대 방향으로 회전하세요";
530  }
531
532
533  void CSCHtuningView::OnBnClickedBtnCcw06()
534  {
535      // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
536
537      change_screw();
538
539      m_screw_id = 6;
540      m_rot_id = 1;
541
542      m_cur = 0;
543
544      m_screw_value[0] = "시계 반대 방향으로 회전하세요";
545  }
546
```

```
656  void CSCHtuningView::OnDraw(CDC* pDC)
657  {
658      // TODO: 여기에 특수화된 코드를 추가 및/또는 기본 클래스를 호출합니다.
659
660      if (m_dc)
661          pDC->BitBlt(m_dc_x, m_dc_y, m_dc_w, m_dc_h, &m_dc, 0, 0, SRCCOPY);
662
663      if (m_dc_image)
664          pDC->BitBlt(m_dc_x_image, m_dc_y_image, m_dc_w_image, m_dc_h_image, &m_dc_image, 0, 0, SRCCOPY);
665
666  }
667
668  void CSCHtuningView::init_dc_buffer(CWnd* pWnd)
669  {
670      // TODO: 여기에 구현 코드 추가.
671
672      CDC* pDC = pWnd->GetDC();
673
674      m_dc.CreateCompatibleDC(pDC);
675      m_bitmap.CreateCompatibleBitmap(pDC, m_dc_w, m_dc_h);
676      CBitmap* pOldBitmap = (CBitmap*)m_dc.SelectObject(&m_bitmap);
677
678      CRect rect(0, 0, m_dc_w, m_dc_h);
679      CBrush Brush;
680      Brush.CreateSolidBrush(RGB(255, 255, 255));
681
682      m_dc.FillRect(rect, &Brush);
683  }
684
```

```
685  void CSCHtuningView::init_dc_buffer_image(CWnd* pWnd)
686  {
687      // TODO: 여기에 구현 코드 추가.
688
689      CDC* pDC = pWnd->GetDC();
690
691      m_dc_image.CreateCompatibleDC(pDC);
692      m_bitmap_image.CreateCompatibleBitmap(pDC, m_dc_w_image, m_dc_h_image);
693      CBitmap* pOldBitmap = (CBitmap*)m_dc_image.SelectObject(&m_bitmap_image);
694
695      CRect rect(0, 0, m_dc_w_image, m_dc_h_image);
696      CBrush Brush;
697      Brush.CreateSolidBrush(RGB(255, 255, 255));
698
699      m_dc_image.FillRect(rect, &Brush);
700  }
701
702  void CSCHtuningView::proc_unit()
703  {
704      // TODO: 여기에 구현 코드 추가.
705
706      if (m_cur > 100)
707          m_cur = 0;
708
709      //m_Image_filter.BitBlt(m_dc, m_pt_filter.x, m_pt_filter.y);
710      m_Image_filter.BitBlt(m_dc, 0, 0);
711
712
713      CPen pen;
```

```
714      pen.CreatePen(PS_SOLID, 1, RGB(255, 255, 255));
715      m_dc.SelectObject(&pen);
716
717      CBrush brush;
718      brush.CreateSolidBrush(RGB(255, 255, 255));
719      m_dc.SelectObject(&brush);
720
721      m_dc.FillRect(CRect(250, 200, 350+300, 250), &brush);
722
723
724      CFont font;
725      font.CreatePointFont(50000, _T("굴림체"));
726      CString str_screw[7];
727      str_screw[0].Format(_T("%s"), m_screw_value[0]);
728
729      m_dc.TextOutW(320, 200, str_screw[0]);
730
731      if (m_screw_id == 0)
732      {
733          InvalidateRect(CRect(m_dc_x, m_dc_y, m_dc_x + m_dc_w, m_dc_y + m_dc_h), false);
734
735          m_cur++;
736
737          Sleep(m_thread_delay);
738
739          return;
740      }
741
742      int a_cur = m_cur % 8;
```

```
744         if (m_rot_id == 0)
745             m_Image_cw[a_cur].TransparentBlt(m_dc, m_pt_screw[m_screw_id - 1].x,
746                 m_pt_screw[m_screw_id - 1].y, m_Image_cw[a_cur].GetWidth(),
747                 m_Image_cw[a_cur].GetHeight(), RGB(255, 255, 255));
748         else
749             m_Image_ccw[a_cur].TransparentBlt(m_dc, m_pt_screw[m_screw_id - 1].x,
750                 m_pt_screw[m_screw_id - 1].y, m_Image_ccw[a_cur].GetWidth(),
751                 m_Image_ccw[a_cur].GetHeight(), RGB(255, 255, 255));
752
753         m_cur++;
754
755         InvalidateRect(CRect(m_dc_x, m_dc_y, m_dc_x + m_dc_w, m_dc_y + m_dc_h), false);
756
757         Sleep(m_thread_delay);
758     }
759
760     void CSCHtuningView::change_screw()
761     {
762         // TODO: 여기에 구현 코드 추가.
763
764         m_thread_flag = 0;
765         Sleep(m_thread_delay * 2);
766
767         m_thread_flag = 1;
768     }
769
770     void CSCHtuningView::proc_stop()
771     {
772         // TODO: 여기에 구현 코드 추가.
```

```
774         m_screw_id = 0;
775         m_Image_filter.BitBlt(m_dc, m_pt_filter.x, m_pt_filter.y);
776         Invalidate(true);
777     }
778
779     LRESULT CSCHtuningView::OnCommunication(WPARAM wParam, LPARAM lPararm)
780     {
781         display(m_SerialComm->m_RecvMessage);
782         return TRUE;
783     }
784
785     void CSCHtuningView::display(CString arg_msg)
786     {
787         // TODO: 여기에 구현 코드 추가.
788
789         CString str_unit;
790
791         int cur_01 = 0;
792         int cur_02 = 2;
793
794         int count = 0;
795         while (1)
796         {
797             cur_02 = arg_msg.Find(_T("₩n"), cur_01);
798
799             if (cur_02 >= 0)
800             {
801                 cur_01 = cur_02 + 1;
802                 count++;
```

```
803             }
804             else
805             {
806                 break;
807             }
808         }
809
810         cur_01 = 0;
811         int len = 0;
812         for (int i = 0; i < count; i++)
813         {
814             cur_02 = arg_msg.Find(_T("₩n"), cur_01);
815             str_unit = arg_msg.Mid(cur_01, cur_02 - cur_01);
816
817             len = str_unit.GetLength();
818
819             if (len >= 0)
820                 receive_serial(str_unit);
821
822
823             cur_01 = cur_02 + 1;
824         }
825     }
826
827     CString CSCHtuningView::load_file(CString filename)
828     {
829         // TODO: 여기에 구현 코드 추가.
830
831         CString str;
```

```
833         CFile file;
834         if (!file.Open(filename, CFile::modeRead))
835         {
836             MessageBox(_T("파일을 열지 못했습니다."), _T("경고!"), MB_OK | MB_ICONHAND);
837             return str;
838         }
839
840         UINT len = (UINT)file.GetLength();
841         char* buf = new char[len];
842
843         file.Read(buf, len);
844         file.Close();
845
846         buf[len] = '₩0';
847         len++;
848
849         str = (LPSTR)buf;
850
851         return str;
852     }
853
854     void CSCHtuningView::load_config()
855     {
856         // TODO: 여기에 구현 코드 추가.
857
858         CString str_total = load_file(_T("01_config.txt"));
859
860         int len = str_total.GetLength();
861
```

```
862     if (len == 0)
863     {
864         CString str;
865
866         str.Format(_T("%d"), m_baud_rate);
867
868         CWnd* pWnd_02 = GetDlgItem(IDC_EDIT_BAUD_RATE);
869         pWnd_02->SetWindowText(str);
870
871         return;
872     }
873
874     CString str_column;
875     CString str_data;
876     int cur_01, cur_02;
877
878     cur_01 = 0;
879     cur_02 = 0;
880
881     for (int i = 0; i < 2; i++)
882     {
883         cur_02 = str_total.Find(_T(":"), cur_01);
884         str_column = str_total.Mid(cur_01, cur_02 - cur_01);
885         cur_01 = cur_02 + 1;
886
887         cur_02 = str_total.Find(_T("₩n"), cur_01);
888         str_data = str_total.Mid(cur_01, cur_02 - cur_01);
889         cur_01 = cur_02 + 1;
```

```
891         str_data.TrimLeft();
892
893         if (i == 0)
894         {
895             add_usb_port(_ttoi(str_data));
896         }
897         else if (i == 1)
898         {
899             CWnd* pWnd_02 = GetDlgItem(IDC_EDIT_BAUD_RATE);
900             pWnd_02->SetWindowText(str_data);
901
902             m_baud_rate = _ttoi(str_data);
903         }
904     }
905
906 }
907
908 void CSCHtuningView::save_file(CString filename, CString arg_msg)
909 {
910     // TODO: 여기에 구현 코드 추가.
911
912     int len = arg_msg.GetLength();
913
914     char buf[500];
915     WideCharToMultiByte(CP_ACP, 0, arg_msg, -1, buf, len, NULL, NULL);
916
917     CFile file;
918     if (!file.Open(filename, CFile::modeCreate | CFile::modeWrite))
919     {
```

```
1481        if (m_screw[0] < 4 * (1024 / 10))
1482        {
1483            m_screw_id = 1;
1484            m_rot_id = 1;
1485
1486            m_screw_value[0] = "1번 나사를 시계 반대 방향으로 회전하세요";
1487
1488            return;
1489        }
1490
1491        if (m_screw[0] > 5 * (1024 / 10))
1492        {
1493            m_screw_id = 1;
1494            m_rot_id = 0;
1495
1496            m_screw_value[0] = "1번 나사를 시계 방향으로 회전하세요";
1497
1498            return;
1499        }
1500
1501        if (m_screw[0] > 4 * (1024 / 10) && m_screw[0] < 5 * (1024 / 10))
1502        {
1503            m_screw_value[0] = "2번 나사로 넘어가세요.";
1504
1505            m_screw_id = 0;
1506            m_rot_id = 0;
1507
1508    //      proc_unit();
```

```
1512        if (m_screw[1] < 4 * (1024 / 10))
1513        {
1514            m_screw_id = 2;
1515            m_rot_id = 1;
1516
1517            m_screw_value[0] = "2번 나사를 시계 반대 방향으로 회전하세요";
1518
1519            return;
1520        }
1521
1522        if (m_screw[1] > 5 * (1024 / 10))
1523        {
1524            m_screw_id = 2;
1525            m_rot_id = 0;
1526
1527            m_screw_value[0] = "2번 나사를 시계 방향으로 회전하세요";
1528
1529            return;
1530        }
1531
1532        if (m_screw[1] > 4 * (1024 / 10) && m_screw[0] < 5 * (1024 / 10))
1533        {
1534            m_screw_value[0] = "3번 나사로 넘어가세요";
1535
1536            m_screw_id = 0;
1537            m_rot_id = 0;
1538
1539    //      proc_unit();
```

```
1543    if (m_screw[2] < 4 * (1024 / 10))
1544    {
1545        m_screw_id = 3;
1546        m_rot_id = 1;
1547
1548        m_screw_value[0] = "3번 나사를 시계 반대 방향으로 회전하세요";
1549
1550        return;
1551    }
1552
1553    if (m_screw[2] > 5 * (1024 / 10))
1554    {
1555        m_screw_id = 3;
1556        m_rot_id = 0;
1557
1558        m_screw_value[0] = "3번 나사를 시계 방향으로 회전하세요";
1559
1560        return;
1561    }
1562
1563    if (m_screw[2] > 4 * (1024 / 10) && m_screw[0] < 5 * (1024 / 10))
1564    {
1565        m_screw_value[0] = "4번 나사로 넘어가세요";
1566
1567        m_screw_id = 0;
1568        m_rot_id = 0;
1569
1570        //      proc_unit();
```

```
1574    if (m_screw[3] < 4 * (1024 / 10))
1575    {
1576        m_screw_id = 4;
1577        m_rot_id = 1;
1578
1579        m_screw_value[0] = "4번 나사를 시계 반대 방향으로 회전하세요";
1580
1581        return;
1582    }
1583
1584    if (m_screw[3] > 5 * (1024 / 10))
1585    {
1586        m_screw_id = 4;
1587        m_rot_id = 0;
1588
1589        m_screw_value[0] = "4번 나사를 시계 방향으로 회전하세요";
1590
1591        return;
1592    }
1593
1594    if (m_screw[3] > 4 * (1024 / 10) && m_screw[0] < 5 * (1024 / 10))
1595    {
1596        m_screw_value[0] = "5번 나사로 넘어가세요";
1597
1598        m_screw_id = 0;
1599        m_rot_id = 0;
1600
```

```
1605    if (m_screw[4] < 4 * (1024 / 10))
1606    {
1607        m_screw_id = 5;
1608        m_rot_id = 1;
1609
1610        m_screw_value[0] = "5번 나사를 시계 반대 방향으로 회전하세요";
1611
1612        return;
1613    }
1614
1615    if (m_screw[4] > 5 * (1024 / 10))
1616    {
1617        m_screw_id = 5;
1618        m_rot_id = 0;
1619
1620        m_screw_value[0] = "5번 나사를 시계 방향으로 회전하세요";
1621
1622        return;
1623    }
1624
1625    if (m_screw[4] > 4 * (1024 / 10) && m_screw[0] < 5 * (1024 / 10))
1626    {
1627        m_screw_value[0] = "6번 나사로 넘어가세요";
1628
1629        m_screw_id = 0;
1630        m_rot_id = 0;
1631
1632        //      proc_unit();
```

```
1636    if (m_screw[5] < 4 * (1024 / 10))
1637    {
1638        m_screw_id = 6;
1639        m_rot_id = 1;
1640
1641        m_screw_value[0] = "6번 나사를 시계 반대 방향으로 회전하세요";
1642
1643        return;
1644    }
1645
1646    if (m_screw[5] > 5 * (1024 / 10))
1647    {
1648        m_screw_id = 6;
1649        m_rot_id = 0;
1650
1651        m_screw_value[0] = "6번 나사를 시계 방향으로 회전하세요";
1652
1653        return;
1654    }
1655
1656    if (m_screw[5] > 4 * (1024 / 10) && m_screw[0] < 5 * (1024 / 10))
1657    {
1658        m_screw_value[0] = "튜닝이 완료되었습니다";
1659
1660        m_screw_id = 0;
1661        m_rot_id = 0;
1662
```

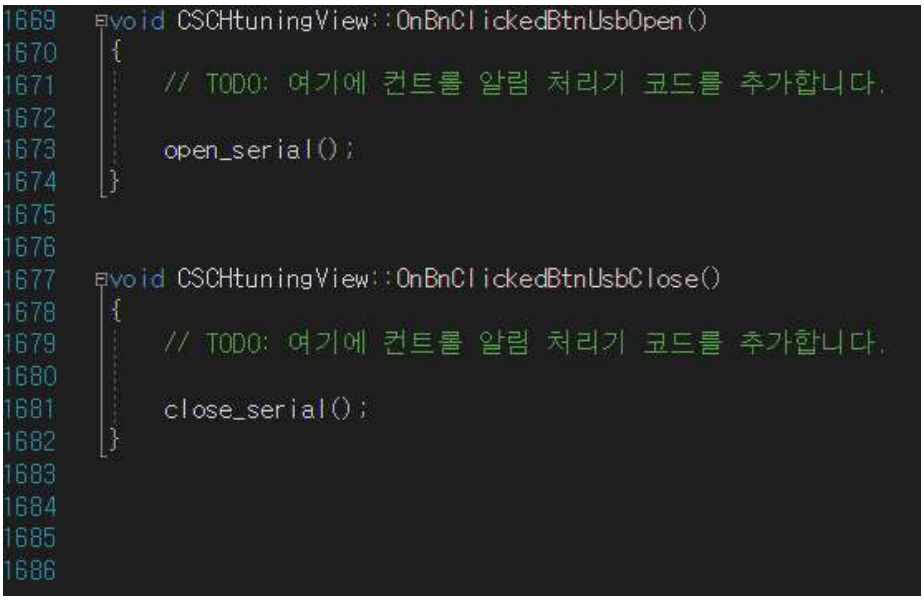

그림 6. MFC 프로그래밍을 이용한 코딩

아두이노를 이용하여 필터 나사 역할을 하는 가변저항의 값을 받아 그 값이 변화함에 따라 그래프의 파형이 나타나 그래프 화면을 보며 나사를 회전함으로써 튜닝을 완료할 수 있게 하는 코딩을 하였다.

2.5 제작

2.5.1 제작과정

그림 7. 아두이노 필터

2.5.2 네트워크 튜닝 완성

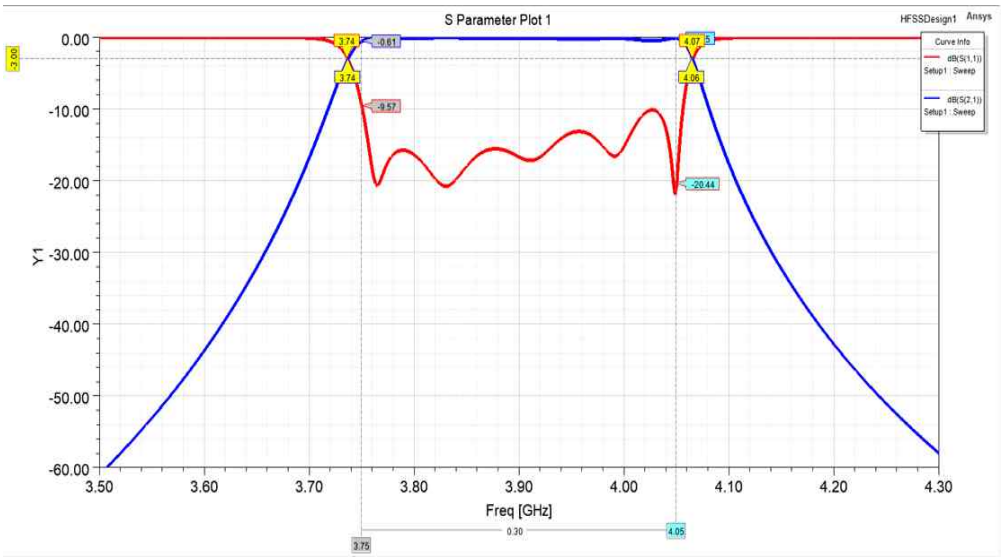

그림 8. 튜닝 완료 시 그래프

제 3 장 결 론

3.1 성능 평가

아두이노를 이용하여 필터 나사 역할을 하는 가변저항의 값을 받아 그 값이 변화함에 따라 그래프의 파형이 화면에 나타내었고 나사를 회전시켜 가변저항 값이 변화하면서 그래프 파형이 변화하여 튜닝이 완료될 수 있도록 코딩하였는데 정상적으로 작동됐다. 여기서 단방향 회전이 아닌 다양한 방향으로 나사를 회전시켜 주기 위해 코딩 값을 수정하였고 정상적으로 작동됐다. 그러나 아쉬운 점은 아두이노로 측정하는 것이 아닌 실제 네트워크 분석기로 측정하여 튜닝을 했으면 좀 더 골든샘플에 가까운 파형 값을 얻을 수 있던 점이 아쉬웠다.

3.2 활용방안 및 결론

네트워크 튜닝은 성능 최적화, 보안 강화, 무선 네트워크 최적화, QOS 관리, 트래픽 관리 등 다양한 분야에서 활용이 된다.

MFC를 이용한 네트워크 튜닝은 Windows 환경에서의 GUI 기반 응용 프로그램의 네트워크 성능을 최적화하는 데 도움이 된다. 몇 가지 활용 방안으로는 프로토콜 최적화, 비동기 통신, 다중 스레드 처리, 데이터 압축 및 암호화, 네트워크 에러 핸들링, 자동 재접속 및 회복 기능, 네트워크 성능 모니터링 등 다양한 네트워크 응용 프로그램을 개발할 수 있다.

본 논문에서는 전공자가 아니면 다루기 어려운 네트워크 분석기를 통한 튜닝을 비전공자들도 쉽게 튜닝을 완료할 수 있도록 도움을 줄 수 있다. 위에 언급한 것과 같이 실제 네트워크 분석기로 튜닝을 했으면 더 좋은 결과값을 얻을 수 있었을 것이다.

참 고 문 헌

[1] 곽종욱, "Ad-hoc 네트워크 환경에서 DSDV 라우팅 알고리즘을 이용한 위치 정보 시스템 및 사용자 맵핑 시스템의 설계 및 구현", 영남대학교, 3월, 2014년

[2] 노민기, 길준민, 최장원, 안성진, "차세대 그리드 네트워크 성능향상을 위한 동적 TCP 튜닝에 대한 연구", 한국정보기술학회지, p.2, 12월, 2005년

[3] IBM, "네트워크 튜닝", 11월, 2022년

[4] 최남대, "네트워크 시스템의 성능 파라미터 조정", 석사학위논문, 숭실대학교 정보과학대학원, p.5, 2월, 1995년

[5] 기형서, "네트워크 감시 도구를 이용한 성능분석 및 개선", 석사학위논문, 숭실대학교 정보과학대학원, p.7, 2월, 1994년

[6] 손준우, "지능제어기법에 의한 공장자동화용 네트워크의 성능관리에 관한 연구", 석사학위논문, 부산대학교 대학원, pp.8-10, 2월 1996년

[7] 권우창, "안전한 대용량 과학데이터 전송을 위한 네트워크 전송구조", 박사학위논문, 한남대학교 대학원, pp.12-17, 8월 2021년

ABSTRACT

Networking provides a process for adjusting network configuration to optimize the performance of a system or application. Expectations for the current network still exist. Expectations for 1. improved performance, 2. with integration, 3. enhanced security, 4. 5G bearing, and 5. integration of cloud technologies are emphasizing integration to withstand current and future business and technology environments. Network addresses continue to evolve technologically. Some key trends include 1. Automation and Artificial Intelligence (AI), 2. Software Defined Networks (SDN), 3. 5G technologies, 4. Encrypted technologies, 5. Enhanced security, 6. Internet of Things (IoT) and the like. no see. Technological advancements are expected to continue in maintenance, which will evolve into more comfortable and flexible forms.

To use this tuning technology, you must know how to use a network analyzer, but since non-majors do not know how to use a network analyzer, this paper studied a program that helps non-majors easily complete network tuning.

Arduino was used to implement this work. The variable resistance values of the Arduino, the variable resistor that acts as a filter screw, and the six switches are measured and displayed as a graph waveform on the screen. Tuning is completed when the desired golden sample is reached by rotating the switch and watching the graph change according to the change in variable resistance value. By changing the Arduino coding value, we set it so that it can be tuned by rotating in different directions instead of just rotating in the same direction.

Key words: network analyzer, network tuning, Arduino, variable resistor, golden sample

감사의 글

4년간의 학사과정을 마치고 학사 논문을 쓰면서 프로그램 코딩 과정에서 많은 시행착오와 어려움도 있었고 부족한 점이 많지만 끝까지 하여 논문을 마칠 수 있게 되어 감사의 말씀을 전하고 싶습니다.

주제로 뭘 해야할지 어떤 방식으로 나아가야 하는지 졸업논문 방향성에 관하여 조언해주신 안 달 교수님께 감사의 말씀을 드리고 싶습니다. 작품 과정 동안 함께해 온 팀원들한테 감사드리고 또한 그동안 같이 배우고 함께해 온 동기, 교수님, 선후배님들에게도 감사드립니다.