DJ\_PROJECT – 제품 디자인

2025년 11월 28일

20193031 최민성

동의대학교 컴퓨터공학과

kycten8402@naver.com

|  |
| --- |
| 요약 – 본 논문은 기존 Portable DJ Controller보다 더 작은 하드웨어를 구현하고 이에 의해 제어될 수 있는 DJ 소프트웨어의 구현을 제시한다. 기존 장비에서 크기를 2배 줄여 스마트폰과 비슷한 크기가 되었다. 하드웨어 크기를 줄였지만 대부분의 기능을 구현할 수 있는 소프트웨어적 환경을 마련했다. 현대 DJ 작업 및 퍼포먼스 방식을 고려했을 때 본 제품은 유의미한 휴대 편의성 향상 경험을 제공할 수 있을 것으로 보인다. |
| Keywords: Portable DJ Controller |

서론

초기 DJ 퍼포먼스는 아날로그 턴테이블을 기반으로 성장했다. 컴퓨터의 발전으로 하드웨어적으로 구현된 각종 DSP를 소프트웨어적으로 처리할 수 있게 되었다. 전문 아날로그 음향 장비였던 DJ 장비가 이퀄라이저와 같은 몇몇 DSP를 제외한 대부분이 소프트웨어로 구현되었고 하드웨어는 PC에 연결 가능한 HID 장치에 가까운 형태로 변했다. 더 나아가 이제는 공연 퍼포먼스뿐 아니라 평소에 휴대하며 시공간의 제약을 초월한 DJ-ing 작업이 가능해진 시대다. 이는 Portable DJ Controller의 등장으로 인해 가능해졌다.

본 논문에서 제시하는 새로운 제품은 기존 Portable DJ Controller의 특징을 더 극대화하는 디자인으로 만들었다. 기존 DJ Controller의 기능은 유지하면서 크기는 더욱 소형화하는 것이다.

2장에서는 제품 디자인의 목표와 현황 조사를 수행한다. 3장에서는 제품 디자인을 위해 고려한 핵심 요소를 살펴본다. 4장에서는 제품 디자인이 목표를 달성했는지 평가한다.

제품 디자인 준비

본 장에서는 제품 디자인의 목표를 제시하고 현황을 분석해 새로운 제품의 디자인을 준비한다.

목표 제시

본 제품의 디자인 목표와 상세한 설명을 제시한다.

1. 기존 Portable DJ Controller보다 좋은 휴대성
2. 언제, 어디서나 DJ-ing을 하더라도 큰 이목을 끌지 않도록 함
3. 기존 사용자가 본 제품에 적응하는 데 시간이 오래 걸리지 않도록 함

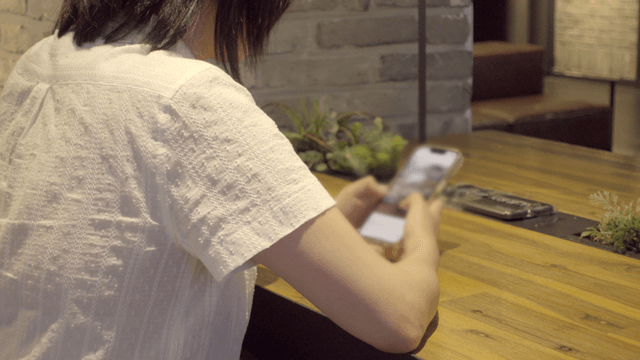
높은 수납 공간 활용도

기존 Portable DJ Controller보다 휴대성이 좋아야 한다. 현재 알려진 Portable DJ Controller는 가로 길이가 세로 길이보다 월등히 길다. 가로 길이를 줄여 더 작은 사이즈의 하드웨어를 만들 수 있다면 수납 공간에 배치하는 데 있어 위치나 방향에 제약이 줄어들 것이다.

스마트폰을 보는 듯한 자연스러움

이제는 스마트폰 없이 살기는 어려운 시대다. 모두가 스마트폰을 하나 이상은 들고 있다. 만약 스마트폰과 비슷한 크기로 하드웨어를 구현할 수 있다면 주변 사람들로 하여금 스마트폰을 보는 것 같은 느낌을 주게 해 이목이 덜 집중될 것이다.

아래 그림은 스마트폰을 보는 자세와 Portable DJ Controller를 사용하는 자세를 보여준다. 두 자세는 앉은 자세와 고개의 각도가 유사하다. 만약 DJ Controller가 스마트폰 크기와 비슷하다면 얼핏 보고서는 스마트폰을 보는 자세와 비슷하다고 오해할 수 있어 이목을 비교적 덜 끌게 될 것이다.

****

**그림1**. 책상에서 스마트폰을 보는 자세



**그림2**. Portable DJ Controller를 사용하는 자세

직관적인 레이아웃

기존 DJ Controller를 사용하던 사람이 쉽게 적응할 수 있도록 디자인할 필요가 있다. 아래 그림은 입문자용 모델인 Pioneer DDJ-FLX4와 프로용/공연용 모델인 Pioneer CDJ-3000이다.

****

**그림3**. Pioneer DDJ-FLX4

****

**그림4**. Pioneer CDJ-3000

기존 DJ Controller에서 볼 수 있는 구조적 특징 몇 가지는 다음과 같다. 이외에도 공통점을 더 찾을 수 있다. 이러한 구조적 특징을 크게 훼손하지 않는 선에서 새로운 제품을 개발할 필요가 있다.

1. 중앙 Controller 영역을 기준으로 양쪽에 Deck이 존재한다.
2. Play/Cue 버튼과 템포 슬라이더가 Deck의 하단 코너에 존재한다.
3. 중앙 Controller 영역에는 EQ → FX → 볼륨 슬라이더 → 크로스페이더 순으로 존재한다.
4. 퍼포먼스 패드가 8개 존재한다.

한편, 현재 DJ Controller를 연결해 사용할 수 있는 RekordBox 소프트웨어의 GUI는 아래와 같다. 큰 구조는 파형 뷰어(A), 퍼포먼스 패드 영역(B), 재생 제어부(C), 플레이리스트(D)로 구성되어 있다. 소프트웨어 역시 구조적 특징을 크게 훼손하지 않는 선에서 개발할 필요가 있다.

****

**그림5**. RekordBox 소프트웨어 GUI (2채널)

Portable DJ Controller 규격 조사

잘 알려진 Portable DJ Controller로 Numark사의 DJ2GO2 모델과 Hercules사의 DJControl Mix 모델이 있다. 장비의 규격은 아래 표와 같다.

|  |  |  |
| --- | --- | --- |
| **모델** | **크기 (가로\*세로\*높이, cm)** | **크기 (가로\*세로\*높이, inch)** |
| Numark DJ2GO2 | 31.4\*8.6\*1.6 | 12.4\*3.4\*0.6 |
| Hercules DJControl Mix | 31.5\*10\*4.9 | 13.4\*3.9\*1.9 |

**표1**. Portable DJ Control 규격



**그림6**. Numark DJ2GO2



**그림7**. Hercules DJControl Mix

백팩에서의 Portable DJ Controller 배치

참고문헌[1]의 Table 4.에 제시된 백팩 크기의 평균 (높이 \* 폭) = (44cm \* 35cm)이다[1]. 이를 기준 백팩 크기로 가정하자. 체감용량은 더 작기 때문에 기존 Portable DJ Controller는 높은 확률로 세로 또는 세로가 긴 대각선으로 배치되어야 한다. 따라서 기존 장비는 백팩에서의 배치 자유도가 낮은 편이라 볼 수 있다.

****

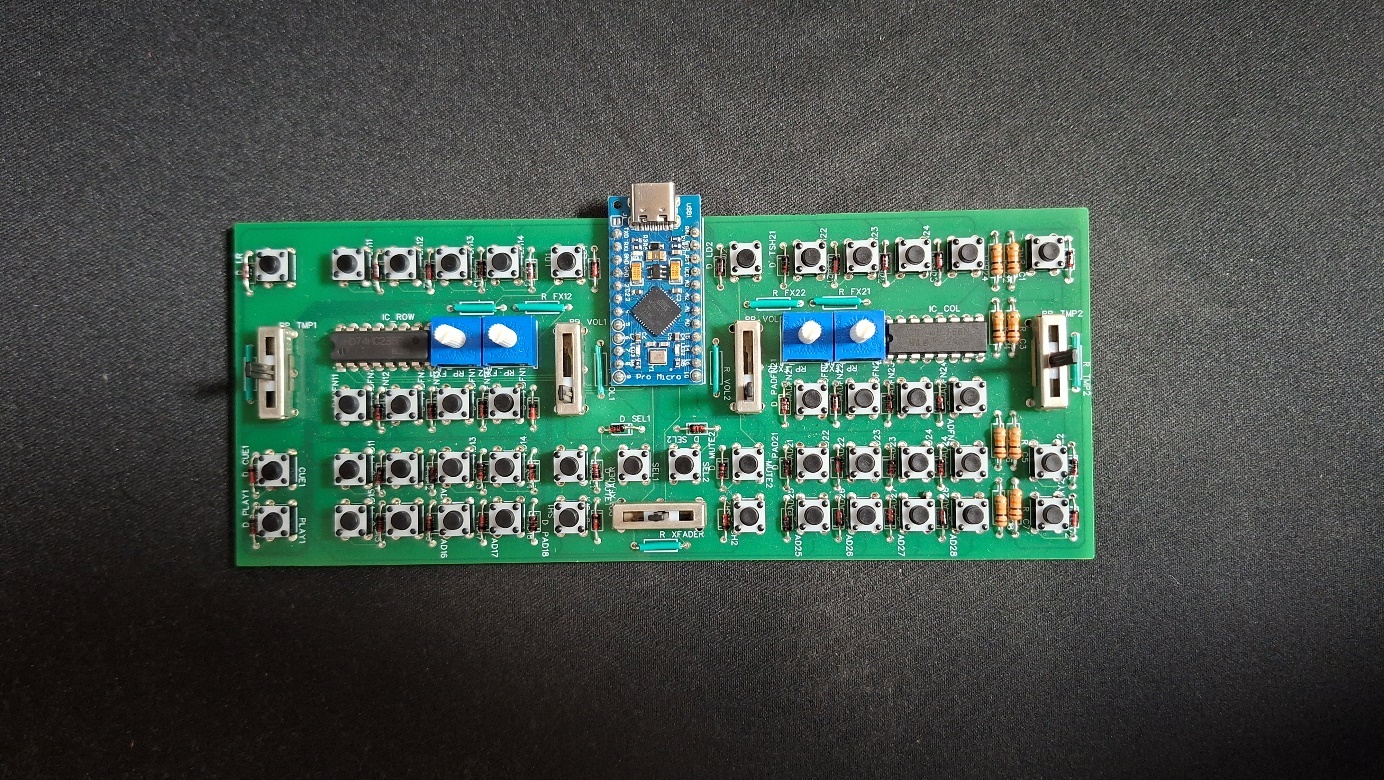
**그림8**. Portable DJ Controller의 일반적인 백팩상 배치

제품 디자인 요소

본 장에서는 이전 장에서 파악한 현황을 바탕으로 새로운 제품의 디자인을 위해 핵심적으로 고려한 사항들을 살펴본다.

제품 레이아웃 구성

본 제품은 하드웨어의 레이아웃 유지를 통해 기존의 DJ-ing 경험을 훼손하지 않고자 했다. 가장 큰 변화는 조그 휠의 제거와 이퀄라이저 노브의 개수 감소이다.

****

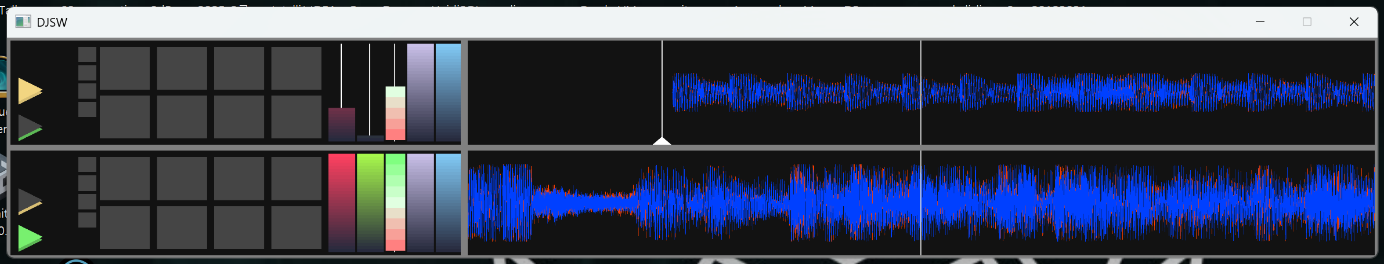
**그림9**. 구현한 DJ Controller의 레이아웃

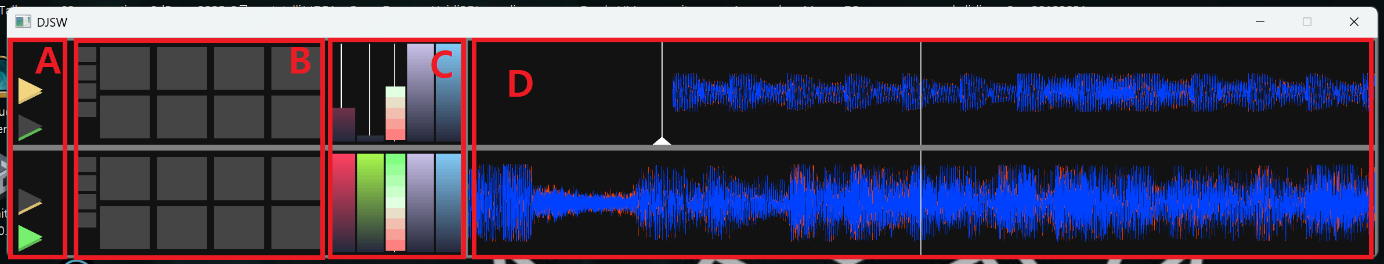
아래 사진은 입문자용 DJ Controller로 많이 사용되는 모델 중 하나인 Pioneer DDJ-FLX2 모델이다. 중앙 Controller 영역을 기준으로 양쪽에 Deck이 하나씩 존재한다. 각 Deck은 조그 휠이 제일 큰 영역을 차지하고 있고 Play/Cue 버튼, 퍼포먼스 패드, 템포 슬라이더, Shift 버튼이 존재한다. Controller 영역은 3채널 이퀄라이저, 다목적 FX 노브, 볼륨 슬라이더, 크로스페이더로 구성되어 있다. 대부분의 2-Deck DJ Controller는 이와 비슷한 하드웨어 레이아웃을 따른다.

****

**그림10**. Pioneer DDJ-FLX2 Controller 레이아웃

한편, 소프트웨어는 재생 제어부(A), 퍼포먼스 패드 영역(B), 믹서 뷰어(C), 파형 뷰어(D)로 이루어져 있다. 플레이리스트 영역의 기능 미구현으로 인해 완성된 형태는 아니다.

****

****

**그림11**. 구현 소프트웨어의 GUI

각 요소별 위치 변화가 존재하지만 기존 소프트웨어가 가진 구성 요소를 모두 포함하고 있다. 믹서 뷰어를 제외한 각 요소는 기존 소프트웨어의 레이아웃에서 크게 벗어나지는 않았다. 믹서 뷰어는 독자적으로 구현한 것으로, 왼쪽부터 FX2 노브, FX1 노브, 템포 슬라이더, 크로스페이더, 볼륨 슬라이더다.

같은 번호의 퍼포먼스 패드 기능 번호(B 영역 작은 사각형 버튼)를 토글하여 퍼포먼스 패드를 완전히 비활성화 할 수 있는 기능은 기존 소프트웨어가 가지고 있지 않아 이 점에서 차별점이 있다.

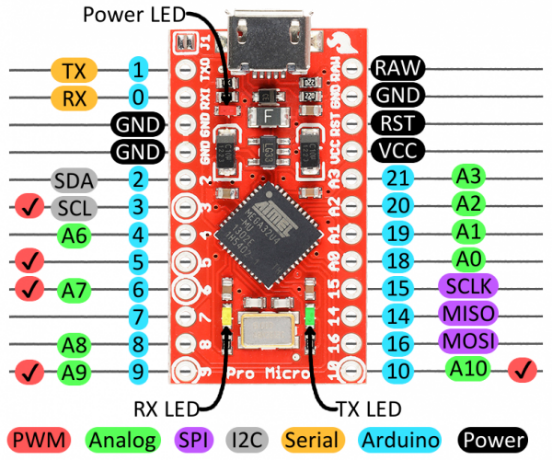
전자회로 구현 시 고려사항

MCU 선정

입력 장치로서 동작시키기 위해 PC와 통신을 수행할 수 있는 적절한 MCU를 선택할 필요가 있었다. MCU는 제품 디자인 목표를 달성할 수 있도록 아래 기준에 따라 선정되었다.

1. 개발 편의성을 위한 MCU 보드
2. 매우 작은 크기, 사각형 모양보다는 세로로 긴 형태
3. 충분한 수의 데이터 핀

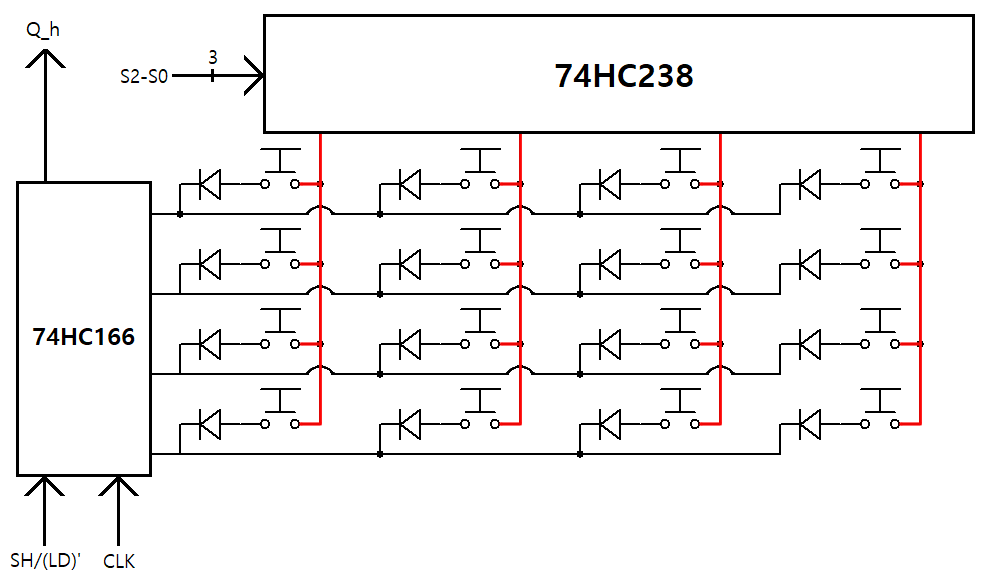
이를 만족하는 MCU로서 Sparkfun사의 Pro Micro를 사용했다. 이는 Arduino Leonardo 호환 MCU 보드다. Arduino IDE를 이용한 개발이 가능해 편의성이 높다. (38mm \* 18mm) 크기의 작은 보드지만 디지털 핀 7개와 아날로그 핀 9개를 사용할 수 있을 만큼 핀의 수가 충분하고 각 핀을 유동적으로 사용할 수 있어 해당 보드를 선택했다.

****

**그림12**. Sparkfun Pro Micro Pin-Out

핵심 회로 구조

6개의 디지털 핀으로 64개의 버튼을 제어할 수 있는 구조를 만들었다. 키보드 매트릭스 회로, 멀티플렉서, 병렬입력-직렬출력 시프트 레지스터를 이용해 구현했다. 이를 통해 하드웨어를 최소화하면서 많은 디지털 입력을 구현할 수 있었다.

****

**그림13**. 디지털 입력을 위한 키보드 매트릭스 회로, 완전하지 않은 추상화된 회로 구조다.

Jog Wheel 제거

조그 휠은 음성 신호를 시간축 이동시키는 장치다. 2가지의 사용 방법이 있다.

1. 조그 휠의 윗면을 회전시켜 음성 신호의 위치를 빠르게 이동시킬 수 있다. (위치 탐색/스크래칭 목적)
2. 조그 휠의 옆면을 회전시켜 음성 신호의 위치를 정밀하게 이동시킬 수 있다. (비트매칭 목적)

Portable DJ Controller는 휴대성이 좋아 기차, 비행기 등 장소에서 이동하면서 사용할 수 있다. 그러나 이와 같은 제한된 환경에서 스크래칭 퍼포먼스는 수행하기 어렵다. 그렇다면 조그 휠은 음성 신호의 위치 이동 기능만을 가지게 되어 휠 형태의 필요성에 의문을 제기할 수 있을 것이다.

시간축 이동 기능만을 수행한다면 휠 형태가 굳이 필요하지 않다고 판단했다. 따라서 휠을 완전히 제거하고 음성 신호의 위치를 이동시킬 수 있는 버튼을 Deck 당 4개씩 포함시켰다. 이를 TSH(Time SHift) 버튼이라 정의했다. 각 TSH 버튼은 오디오 엔진 루프 당 1샘플, 10샘플, 100샘플, 1000샘플 이동시킬 수 있다. TSH 버튼을 누르면 버튼에 할당된 샘플 수만큼 앞으로 이동한다. Shift+TSH 버튼을 누르면 뒤로 이동한다.

****

**그림14**. TSH 버튼의 위치

조그 휠을 제거하고 버튼으로 변경함으로서 휠이 차지하던 영역을 대폭 줄일 수 있었고 하드웨어의 전체 크기가 작아졌다.

소프트웨어 병렬화

DJ 소프트웨어는 실시간 퍼포먼스를 수행할 수 있도록 프레임 시간이 매우 짧아야 한다. 따라서 소프트웨어를 여러 스레드로 나누고 이를 멀티 코어로 처리하는 오디오 처리 엔진을 구축했다. 아래 코드는 렌더링 스레드에서 호출되는 LoopInit() 함수다.

|  |
| --- |
| int WINAPI LoopInit(HINSTANCE hInstance, HINSTANCE, LPSTR, int nCmdShow)  {  BOOL execResult;  HRESULT hr;  /\* 현재 스레드 == 렌더링 스레드 \*/  // 실시간 멀티코어 처리를 위한 설정  SYSTEM\_INFO info;  GetSystemInfo(&info);  ULONG\_PTR defaultProcessAffinityMask;  ULONG\_PTR defaultSystemAffinityMask;  execResult = GetProcessAffinityMask(  GetCurrentProcess(),  &defaultProcessAffinityMask,  &defaultSystemAffinityMask  );  assert(execResult);  // 워커 스레드 생성  \_jobParams.loopBaseParams.threadAffinityMask = 0;  \_jobParams.loopBaseParams.interruptNumber = DJSW\_INT\_SYNC;  \_jobParams.loopBaseParams.threadHandle = CreateThread(  NULL,  0,  JobMain,  &\_jobParams,  0,  &\_jobParams.loopBaseParams.threadId);  while (\_jobParams.loopBaseParams.interruptNumber != DJSW\_INT\_NULL);  // 입력 스레드 생성  \_hidParams.loopBaseParams.threadAffinityMask = 1ULL << CORE\_INDEX\_HID; // 코어 번호 고정  \_hidParams.loopBaseParams.interruptNumber = DJSW\_INT\_SYNC;  \_hidParams.loopBaseParams.threadHandle = CreateThread(  NULL,  0,  HidMain,  &\_hidParams,  0,  &\_hidParams.loopBaseParams.threadId);  assert(\_hidParams.loopBaseParams.threadHandle != NULL);  while (\_hidParams.loopBaseParams.interruptNumber != DJSW\_INT\_NULL);  // 오디오 처리 스레드 생성  \_audioParams.loopBaseParams.threadAffinityMask = 1ULL << CORE\_INDEX\_AUDIO; // 코어 번호 고정  \_audioParams.loopBaseParams.interruptNumber = DJSW\_INT\_SYNC;  \_audioParams.loopBaseParams.threadHandle = CreateThread(  NULL,  0,  AudioMain,  &\_audioParams,  0,  &\_audioParams.loopBaseParams.threadId);  assert(\_audioParams.loopBaseParams.threadHandle != NULL);  while (\_audioParams.loopBaseParams.interruptNumber != DJSW\_INT\_NULL);  // 실행되는 코어 위치 고정  if (info.dwNumberOfProcessors >= 2)  {  SetThreadAffinityMask(\_hidParams.loopBaseParams.threadHandle, \_hidParams.loopBaseParams.threadAffinityMask);  SetThreadAffinityMask(\_audioParams.loopBaseParams.threadHandle, \_audioParams.loopBaseParams.threadAffinityMask);  }  // 코어 수가 충분할 때 완전 실시간 처리 스레드로 고정  if (info.dwNumberOfProcessors >= 8)  {  SetThreadPriority(\_audioParams.loopBaseParams.threadHandle, THREAD\_PRIORITY\_TIME\_CRITICAL);  SetThreadPriority(\_hidParams.loopBaseParams.threadHandle, THREAD\_PRIORITY\_TIME\_CRITICAL);  }  // 렌더링 스레드 초기화  RenderInit(hInstance, NULL, NULL, nCmdShow);  return 1;  } |

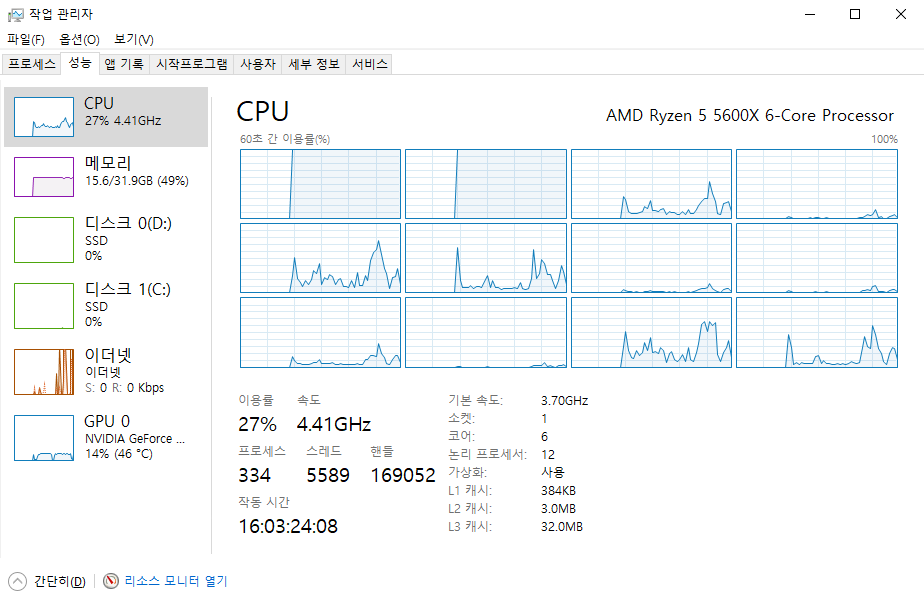
**코드1**. 멀티 코어 설정을 위한 코드

멀티 스레드 환경이 구성된 후 입력 스레드와 오디오 처리 스레드는 아래 루프를 반복하게 된다.

|  |
| --- |
| DWORD WINAPI HidMain(LPVOID lpParams)  {  HidParams\* hidParams = (HidParams\*)lpParams;  InputInit();  // 초기화 로직 동기화를 위함.  hidParams->loopBaseParams.interruptNumber = DJSW\_INT\_NULL;  while (hidParams->loopBaseParams.interruptNumber != 1)  {  InputUpdate();  }  InputFinal();  return 1;  }  DWORD WINAPI AudioMain(LPVOID lpParams)  {  AudioParams\* audioParams = (AudioParams\*)lpParams;  AudioInit();  // 초기화 로직 동기화를 위함.  audioParams->loopBaseParams.interruptNumber = DJSW\_INT\_NULL;  while (audioParams->loopBaseParams.interruptNumber != 1)  {  AudioUpdate();  }  AudioFinal();  return 0;  } |

**코드2**. 입력 스레드와 오디오 처리 스레드의 엔트리 포인트 함수

멀티 코어 설정 이후 입력 스레드와 오디오 처리 스레드의 CPU 점유율을 확인한 결과 코어를 완전히 점유하고 완전 실시간 스레드로 동작하고 있음을 확인했다.

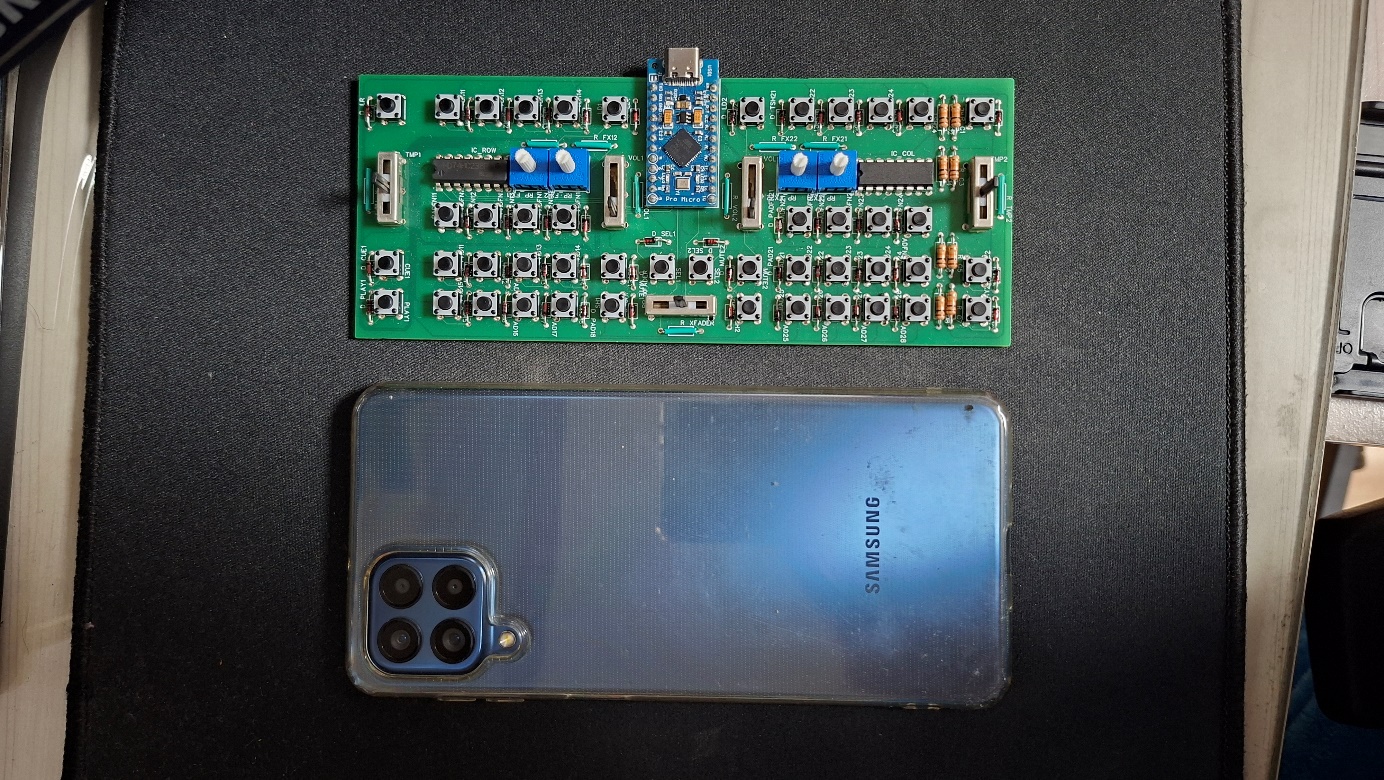
****

**그림15**. 완전 실시간으로 동작하는 입력 스레드와 오디오 처리 스레드

평가

하드웨어 평가

하드웨어는 스마트폰과 비슷한 크기로 제작되어 주머니에 넣고 다닐 수 있는 수준이다. 본 제품 디자인의 제 1목표인 높은 휴대성을 만족했다고 볼 수 있다. 회로 기판의 크기는 (가로 \* 세로) = (168mm \* 68mm)이다.

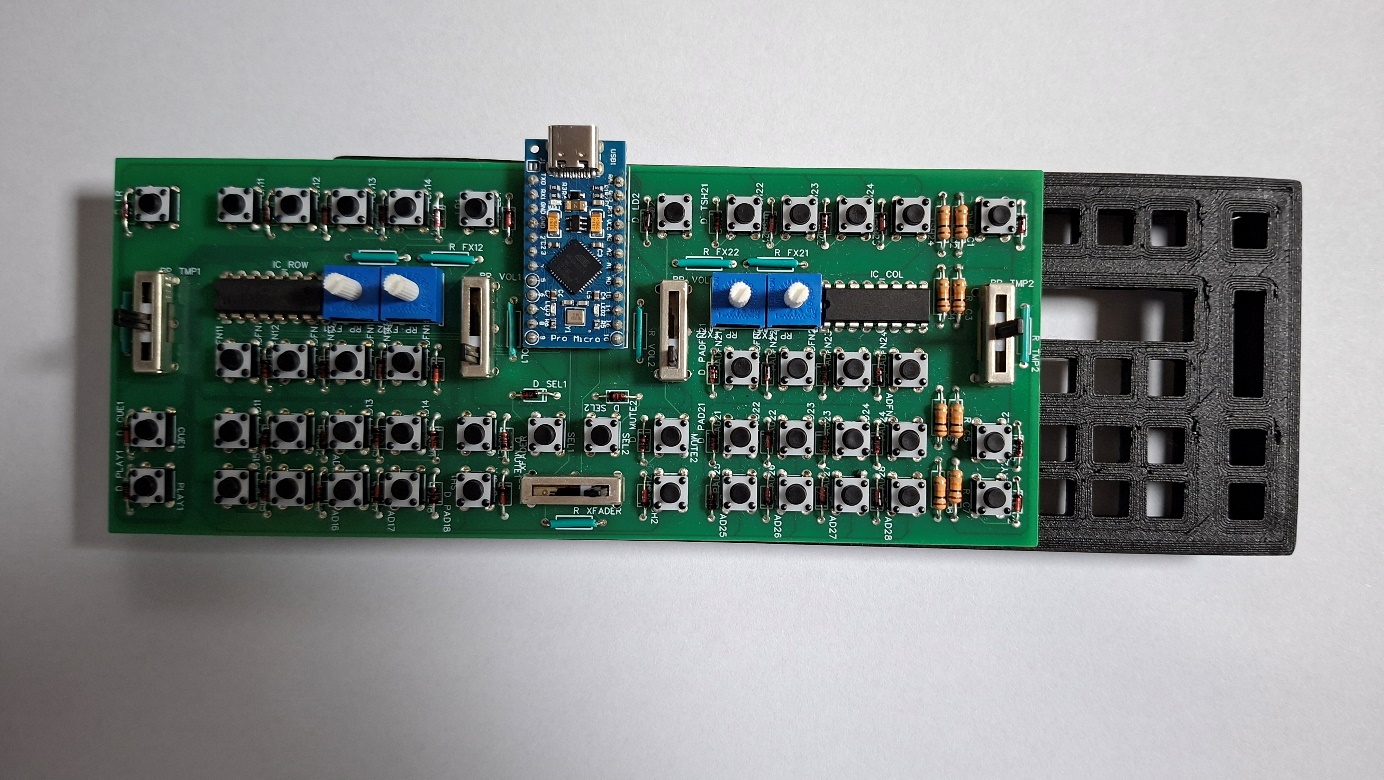
****

**그림16**. 스마트폰 크기와 비슷한 하드웨어 크기

****

**그림17**. 주머니에 들어가는 하드웨어

치명적인 문제점은, 회로 기판을 감싸는 하우징 케이스를 잘못 설계한 점이다. 가로축은 길이가 맞으나, 세로축의 길이가 길어 하우징 안에 회로 기판이 들어가지 않는다.



**그림18**. 잘못된 하우징 설계

하우징을 재설계한다고 가정하면 최종 하드웨어 사이즈는 (가로 \* 세로 \* 높이) = (180mm \* 80mm \* 19mm)가 될 것으로 예상하고 있다(노브 길이 미포함).

소프트웨어 평가

시간축 이동 시간 평가

조그 휠을 제거하며 THS 버튼이 추가되었다. 따라서 이 버튼의 기능 평가를 실시할 필요가 있었다.

아래 표는 THS 버튼을 누르고 있을 때 음원의 처음부터 끝까지 도달하는 데 걸리는 시간을 각 버튼별로 보여준다. 오디오 엔진 루프가 실시간으로 어떤 연산을 하느냐에 따라 처리 시간의 편차는 있을 수 있다. 1 (S/F), 10 (S/F)의 음원 탐색 시간은 추정치다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **음원 길이** | **음원 탐색 시간 (s)** | | | |
| **1 (Sample/Frame)** | **10 (Sample/F.)** | **100 (Sample/F.)** | **1000 (Sample/F.)** |
| 3m 26s | x10 of 10 (S/F) | x10 of 100 (S/F) | 44.5s – 46.7s | 4.6s – 4.8s |
| 3m 45s | x10 of 10 (S/F) | x10 of 100 (S/F) | 48.3s – 50.5s | 4.7s – 5.0s |

**표2**. TSH 버튼별 음원 전체 탐색에 소요되는 시간

각 버튼별 음원 탐색에 걸리는 시간을 고려하면 TSH 버튼을 이용해 빠른 음원 탐색과 정밀 음원 탐색을 모두 수행할 수 있다. 따라서 TSH 버튼은 조그 휠의 대체 기능으로서 적절하다고 볼 수 있다.

스레드별 처리 시간 평가

본 소프트웨어의 핵심 스레드는 입력, 오디오 처리, 렌더링 스레드이다. 따라서 3개의 스레드에 대해 실행 시간을 비교해 볼 필요가 있다.

|  |  |
| --- | --- |
| **스레드 유형** | **실행 시간 (us)** |
| 입력 (Non-Blocking IO 방식) | 15us – 3000us |
| 오디오 처리 | 2us – 3us |
| 렌더링 | 5800us – 11000us |

**표3**. 스레드별 실행 시간 측정

실시간성이 중요한 오디오 처리 엔진은 인간이 알아채지 못하는 수준의 지연 시간 이내에 프레임 처리가 이루어질 것을 요구한다. 인간이 통상 20ms(20000us) 이내의 속도를 분간하기는 어렵기 때문에 본 소프트웨어가 갖는 각 스레드별 프레임 실행 시간은 요구사항에 합치된다.

결론

본 논문에서는 새로운 Portable DJ Controller의 제품 디자인을 소개했다. 기존 장비보다 크기를 약 2배정도 줄여 스마트폰과 비슷한 크기가 되었다. 동시에 기존 DJ Controller가 가진 대부분의 기능은 유지했다.

또한, 오디오 처리 엔진의 구조를 설계했다. 이는 앞으로 추가될 다양한 기능을 개발할 수 있는 소프트웨어적 환경을 마련한 것이다.

현대 DJ 작업 및 퍼포먼스 방식을 고려했을 때 본 제품은 유의미한 휴대 편의성 향상 경험을 제공할 수 있을 것으로 보인다.

추후 플레이리스트 기능, GUI 이벤트 처리 기능은 반드시 구현해야 한다. 회로 기판의 하우징 역시 재설계해야 한다. 하드웨어의 레이아웃이 달라질 수 있는데, 하드웨어의 모델에 따른 소프트웨어의 처리 방식을 고민해볼 필요가 있다.

**참고 문헌 및 웹사이트**

1. R. A. Joy, M. I. Hossain, M. A. Sarker, M. N. H. Sarker, M. S. Rahman, “Evaluating backpack utilizing anthropometric measurements to ensure ergonomic fit and ease for Bangladeshi students”, International Journal of Industrial Engineering, vol. 12, no. 1, pp. 61-73, Jan.-Apr. 2025, doi: 10.14445/23499362/IJIE-V12I1P107.

**저자 정보**

**최민성**, 동의대학교 컴퓨터공학과 4학년, 2019학번

관심 있는 것: 게임 시스템 개발, 2D 게임 엔진, 유한상태기계, Parsing Algorithm, 음향 신호 처리