|  |
| --- |
| **USB Host Driver** |
| 구조설계서 |
|  |
|  |
| **2022-01-14** |
| **김성후** |

이 문서는 **USB Host Driver** 개발을 위한 구조설계서이다.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 0.1 | 2022-01-14 | 김성후 | 초기 문서 생성 |
| 0.2 | 2022-01-16 | 김성후 | Interim 작성 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[1. 시스템 개요 3](#_Toc479150191)

[2. 요구사항 4](#_Toc479150192)

[2.1. 기능적 요구사항 4](#_Toc479150193)

[2.2. 비기능적 요구사항 4](#_Toc479150194)

[2.3. 품질 속성 5](#_Toc479150195)

[부록 8](#_Toc479150198)

[A. 도메인 모델 9](#_Toc479150199)

[B. 품질 시나리오 10](#_Toc479150200)

[C. 품질 시나리오 분석 11](#_Toc479150201)

# 시스템 개요

* 1. **시스템 개요**

USB란 Universal Serial Bus의 약어로, CTI(Computer Telephony Integration) 산업의 성장을 위해 인텔, 마이크로소프트, 컴팩, IBM, NEC, DEC 및 Nortel 등의 회사가 개발하였다. 플러그 앤 플레이를 위한 PC 주변장치의 Bus 규격으로서, 새로운 주변기기가 접속되었을 때 재부팅이나 셋업 과정 없이 자동인식으로 최대 127개의 장치를 연결할 수 있으며, 기존 흔히 사용되던 시리얼 포트에 비해 데이터 전송속도도 빠르게 향상되었다. 규격이 마련됨과 동시에 인텔과 VIA, SiS, ALi 등과 같은 메인보드용 칩셋 제작사들은 USB 포트를 지원하는 칩셋을 발표하고 그들을 주축으로 지금처럼 PC에 USB 포트가 기본적으로 장착되기 시작됐다.

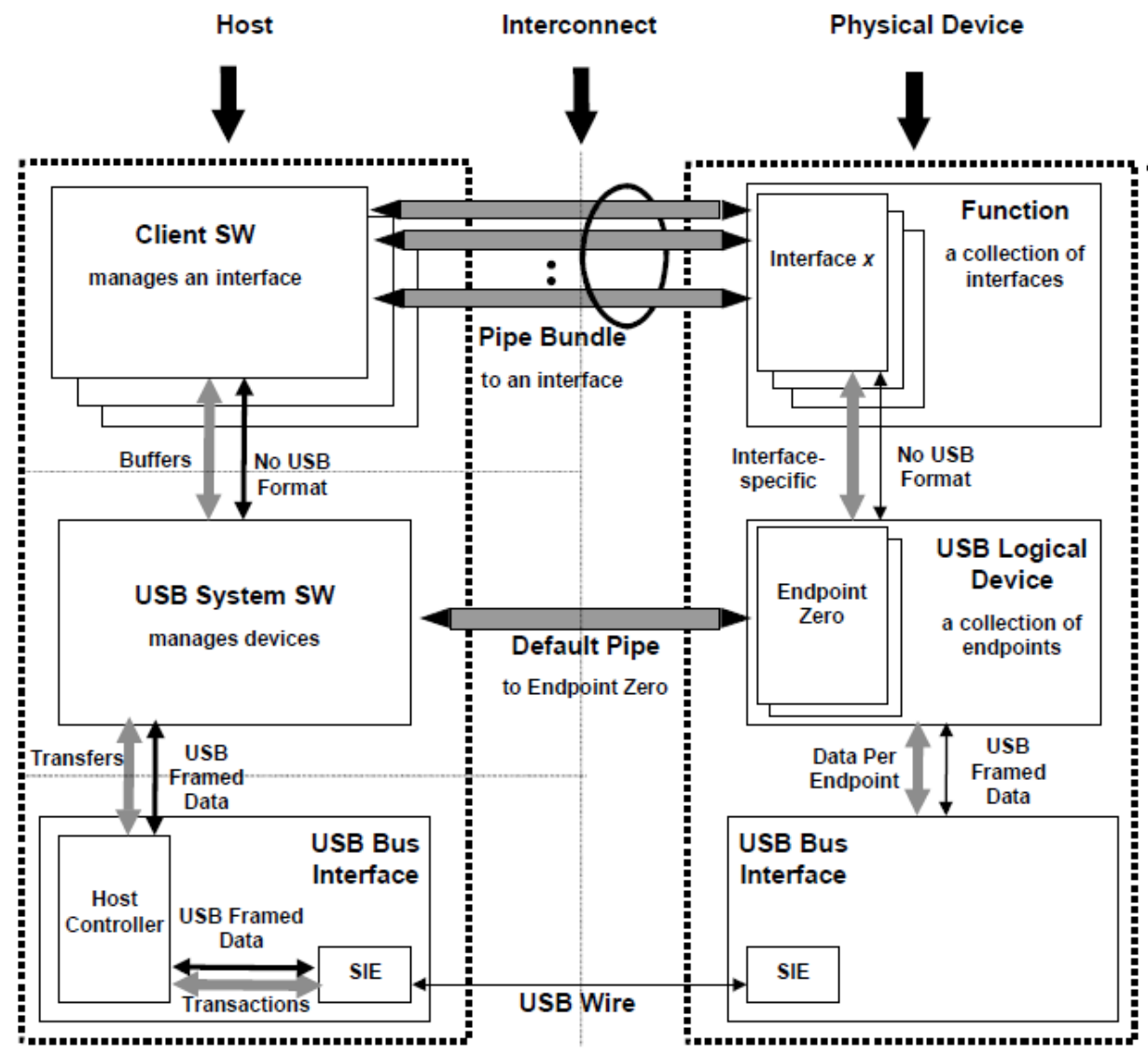


그림 - USB 시스템의 구조

USB는 하나의 Host와 다수의 USB Device를 연결한다. 이들은 물리적인 USB Wire를 통해 연결되어 서로 통신하며 그 위에 논리적인 계층을 가진다. 우선 물리적 계층의 USB Bus Interface의 Host Controller는 통신을 위한 하드웨어 적인 요소를 담당한다. 그리고 그 위의 레이어의 USB System SW는 USB 장치들을 관리하기 위한 Logic을 담당하며 Peer 개념의 USB Logical Driver와 논리적인 연결을 가지고 있다. USB Logical driver의 통신을 위한 접점을 Endpoint Zero라고 부르며, 논리적 개념의 연결을 Default Pipe라고 명명하고 있다. 또 그 상위 레이어는 Client Software와 USB Device의 Function과의 논리적인 연결을 가지게 된다. Client Software는 Host의 운영체제 내의 소프트웨어 혹은 Host에 탑재된 소프트웨어를 의미하며, USB Device의 Function은 해당 장치의 고유한 기능을 구현한 소프트웨어를 의미한다. 각각의 논리적 연결은 실제로는 물리적 계층의 USB Wire를 통해 전기적인 신호로 전달되지만 논리적 개념의 추상화를 통해 복잡도를 낮춰주는 역할을 한다.



그림 2 - USB 허브 제품의 예(Belkin社 멀티포트 허브)

USB 허브는 하나의 USB 포트를 여러 개로 확장하여 장치 연결을 위해 더 많은 포트를 호스트 시스템에 사용할 수 있게 하는 장치이다. USB 허브는 컴퓨터, 자판, 모니터, 프린터 따위의 장비에 기본 포함되어 있기도 하다. 이러한 장치가 수많은 USB 포트를 지니고 있으면, 각 포트가 독립적으로 USB 우회를 거치는 것이 아니라, 하나 이상의 외부 USB 허브로부터 연결된다. 물리적으로 구분된 USB 허브는 다양한 폼팩터로 되어 있다. 긴 케이블이 제공되는 네트워크 허브와 비슷한 모양의 외부 박스 형태에서부터 USB 포트에 직접 연결할 수 있는 조그마한 디자인에 이르기까지 다양하다.

* 1. **사업 환경**

USB Device가 PC 업계의 인터페이스 표준으로 확고한 지위를 가지면서, 매해 수십억 개의 새로운 USB Device들이 시장에 출시되고 있다. USB Device의 사용이 활성화되면서 PC 제품 들에서 기본적으로 제공되는 USB 슬롯을 감당하기 어려워졌고 특히나 최근 수요가 많은 Note PC 제품에서는 상대적으로 슬롯이 더 적어 다수의 장치를 이용하기가 어렵다. 또한 Desktop PC의 경우 제품 후면에 슬롯이 많이 존재하여 USB Device를 삽입하기 까다로운 경우가 많으며, Note PC의 경우에는 다수의 USB Device를 연결할 경우 휴대성이 떨어지는 단점이 있다. 그러한 환경하에서 USB 허브 장치에 대한 수요가 예상되며 따라서 본 과제에서는 PC에 연결하여 5개의 추가적인 USB 연결을 지원하는 장치를 설계한다.

* + 1. **Stakeholder**

|  |  |
| --- | --- |
| Stakeholder | Interest |
| Operation System  개발사 | 불특정 다수의 USB Device에 대해서 최대한 정보를 파악할 수 있길 바라며, 그에 따른 최적의 데이터 전송을 기대한다. |
| USB Device  개발사 | 개발한 장치가 타겟 운영체제에서 개발의도대로 동작하기를 기대한다. 그러기 위해 제품의 특성에 맞는 최대한의 리소스(전력, 대역폭 등)를 할당 받기를 기대한다. |
| LED Controller  개발사 | 다양한 제품에 범용적으로 사용될 수 있는 LED 제품을 개발하고 납품하고 싶어한다. |
| USB Device  사용자 | 가능한 많은 장치를 연결하여 사용하고 싶어한다. 합리적인 가격에 안정적인 속도와 충전 기능을 제공받고 싶어한다. |

* + 1. **시스템 제약사항**
* 제품은 Host와의 연결을 위한 하나의 케이블과 5개의 USB 슬롯을 제공한다.
* 제품은 외부에 LED가 6개 존재하며, Host와의 연결과 연결된 5개의 USB Device를 표현한다. 파란색 불빛을 낼 수 있으며 점멸할 수 있고 점멸 속도도 조절이 가능하다.
* USB 허브에는 자체 전원 공급장치가 있어 서브 시스템에 충분한 전력을 공급할 수 있음을 가정한다.
* USB Device Driver를 가진 운영체제가 있는 PC 환경을 Host로 가정한다.
* USB의 물리적 통신 프로토콜은 USB Host Controller가 담당한다.
* USB Host Controller는 CRC(Cyclic Redundancy Check) 등을 통해 전송 데이터에 대한 무결성을 보장한다고 가정한다.
* USB Host Controller는 Host 시스템과 USB Device와의 물리적인 통신을 담당하며 각각에 대해 별도의 인터페이스를 가진다고 가정한다.
* USB 허브에 계층적으로 (127 - Host가 사용중인 USB Device 개수)만큼의 USB Device 연결이 가능하며, 물리적으로 30미터 이내에 연결되어 있음을 가정한다.
* 루트 허브라 함은 본 과제에서 개발하는 허브에 해당하는 위상을 의미힌다.
* USB 2.0 버전을 지원하는 장치를 가정한다. 차기 제품에서 상위 버전에 대한 하드웨어 지원을 할 수 있으며 그에 따른 펌웨어 업데이트가 가능하다.
  1. **시스템 정의**

본 과제에서 개발하는 USB Hub System의 개요 및 개발 범위는 아래 그림 3과 같다.

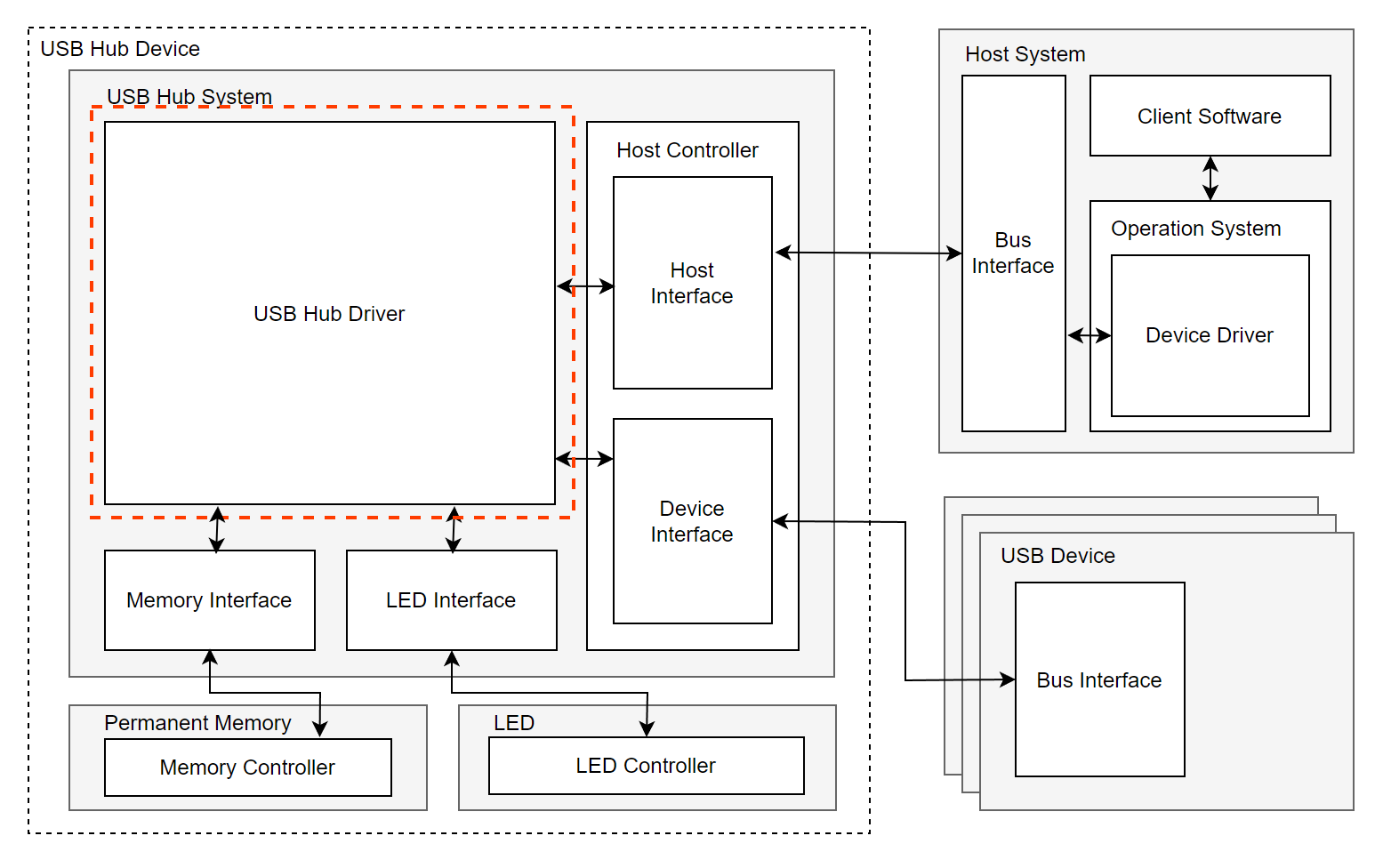


그림 3 - 시스템 전체 구조와 개발 범위

본 과제의 설계 범위는 USB Hub Driver까지이다. USB Hub System 내에 Host Controller, Permanent Memory와 LED 등의 구성요소가 있다. USB Hub Driver는 이 구성요소들과 통신할 수 있는 인터페이스를 가진다. Host Controller에는 Host와 USB Device를 위한 별도의 인터페이스를 가지고 있다. 본 시스템이 PC 제품에 포함되는 제품이 아니라 USB를 통해 Host에 연결되는 제품이므로 별도의 인터페이스가 필요하기 때문이다.

* + 1. **External Interface**

|  |  |
| --- | --- |
| External Interface | Description |
| Host Interface | Host 시스템과 통신할 수 있는 인터페이스이다. 데이터를 주고받을 수 있는 두 개의 버퍼와 제어 정보를 주고받을 수 있는 인터럽트 회선이 있다. USB Hub Device의 구성요소로서 USB Host Driver에 의해 직접 접근이 가능하다. |
| Device Interface | USB Device와 통신할 수 있는 인터페이스이다. 데이터를 주고받을 수 있는 두 개의 버퍼와 제어 정보를 주고받을 수 있는 인터럽트 회선이 있다. USB Hub Device의 구성요소로서 USB Host Driver에 의해 직접 접근이 가능하다. |
| Led Interface | Third-party 하드웨어인 LED 장치를 제어할 수 있는 인터페이스이다. |
| Memory Interface | 시스템에 전원이 들어오지 않아도 정보가 저장되는 영구적인 메모리를 제어할 수 있는 인터페이스이다. |

* + 1. **시스템 경계**

본 과제에서 다루고자 하는 USB Hub Driver의 경계는 아래 그림 4와 같다.

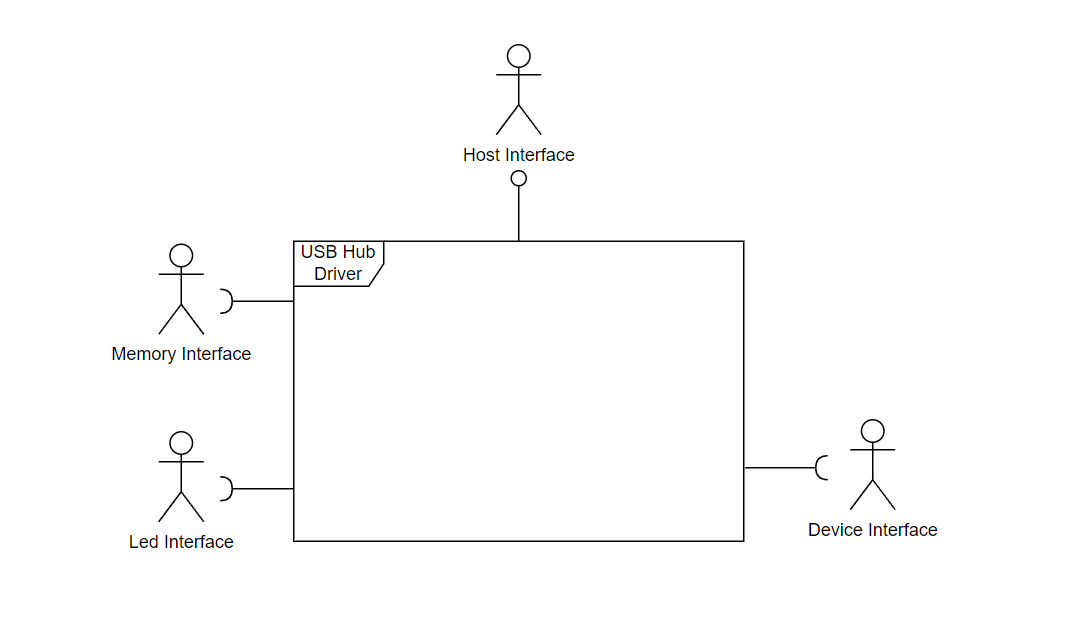


그림 4 - 시스템 경계

본 시스템(USB Hub Driver)은 Host Interface와 Device Interface를 통해 각각 연결된 장치로부터 데이터 송수신을 요청할 수 있다. Device Interface는 USB Device의 상태를 통보할 수 있고, Host Interface는 USB Device의 상태를 요구하고 제어를 본 시스템에게 요청할 수 있다. 본 시스템은 LED 하드웨어에 대한 제어를 Led Interface에게 요청할 수 있다.

* + 1. **시스템 동작**
* USB Hub의 Host에 대한 연결 및 해제 처리를 수행한다.
* USB Device 연결 및 해제에 대한 처리를 수행한다.
* USB Device에 대한 정보를 수집하고 요청에 따라 전달한다.
* USB Device에 대한 읽기 및 쓰기를 수행한다.

# 요구사항

## 기능적 요구사항

* + 1. **Use Case Diagram**

시스템 경계에 따른 Use Case Diagram은 아래 그림 5와 같다.

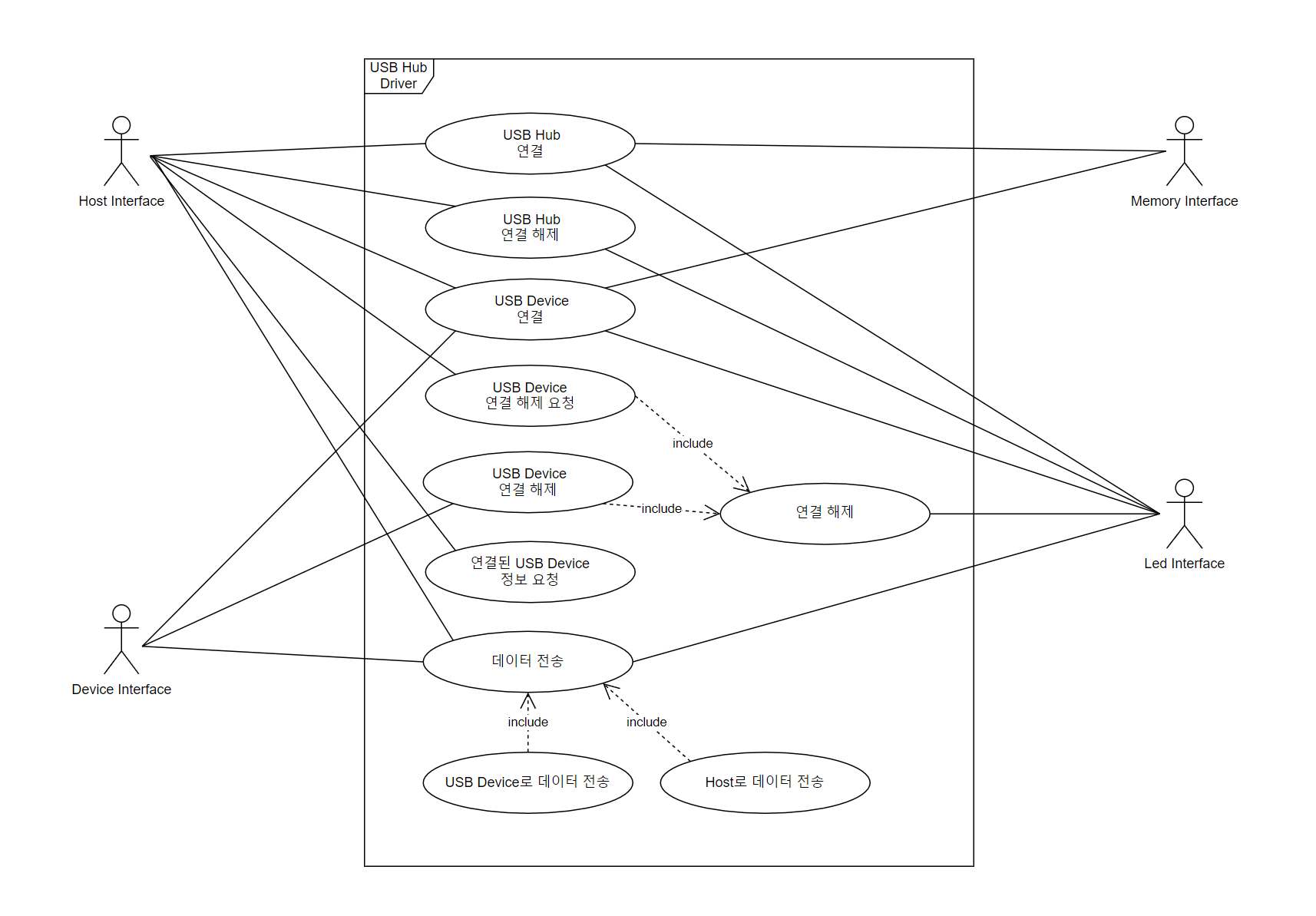


그림 5 – Use Case Diagram

* + 1. **Use Case 명세**
       1. **USB Hub 연결**

|  |  |
| --- | --- |
| **UC\_01** | USB Hub 연결 |
| 설명 |  |
| 행위자 |  |
| 선행조건 |  |
| 후행조건 |  |
| 기본 동작 |  |
| 추가 동작 |  |

## 비기능적 요구사항

// 활동5. 품질 속성 선정

// 점검5-2. 품질 요구사항의 명세가 적절한가?

// 점검5-3. 품질 요구사항의 측정이 가능한가?

성능 – 디바이스의 연결, 제어 등의 성능이 빠를 수록 좋다. ▪ 변경 용이성 / 확장성 – 하드웨어 사양의 변경이 용이할 수록 좋다. – 다양한 클래스로의 확장이 용이할 수록 좋다

* 뎁스 7 이하
* 총 연결 개수 127개 이하
* 펌웨어 업데이트를 통해 클래스를 추가할 수 있어야 한다.
* 전원 차단시 데이터 보호 => 클라이언트 소프트웨어에 영향 안가도록
* 한번 연결된 장치는 더 빨ㄹ ㅣ연결
* 여러 디바이스에 독립성 제공(스트림 파이프 별개)

// 점검5-4. 비기능적 요구사항의 제약이 명확한가?

|  |  |  |
| --- | --- | --- |
| **NFR\_01** |  |  |
| 설명 |  | |
| 환경 |  | |
| 자극 |  | |
| 반응 |  | |
| 측정 |  | |
| **제약** |  | |

## 품질 속성

// 활동5. 품질 속성 선정

// 점검5-2. 품질 요구사항의 명세가 적절한가?

// 점검5-3. 품질 요구사항의 측정이 가능한가?

|  |  |  |
| --- | --- | --- |
| **QA\_01** |  |  |
| 설명 |  | |
| 환경 |  | |
| 자극 |  | |
| 반응 |  | |
| 측정 |  | |

* 디바이스가 요구하는 전력, 대역폭등은 베스트 이포트로 제공해야 한다.

(네고시에이터)

* - 유에스비 클래스에 적합한 전송속도로 제공해야 한다.
* => 시나리오 키보드 로우 스피드, 오디오 풀스피드, 비디오 하이스피드
* 영상과 오디오와 같이 품질보다 등시성이 중요한 경우 등시성을 지킬 수 있어야 한다.
* => 스트림 파이프 열 때 특성 지정
* 조작된 유에스비 클래스 인식가능? 배드유에스비인가 무시긴가 막을수있나
* 다른 버전에 대한 하위호환
* => 네고시에이터가 상하위 버전 디바이스 입력시 버전도 정해준다고 가정

부록

[A. 도메인 모델 9](#_Toc479150206)

[B. 품질 시나리오 10](#_Toc479150207)

[C. 품질 시나리오 분석 11](#_Toc479150208)

1. 도메인 모델

본 과제에서 USB Hub Driver에 대한 도메인 모델은 그림 6과 같다. 본 도메인 모델은 Boundary/Control/Entity 모델로 작성되었다. Boundary는 외부의 Interface와 상호 작용하고, Control은 Boundary와 Entity를 연결해주며, Entity는 시스템에 필요한 데이터를 저장한다.



그림 - Domain Model

**A1. Domain Model List**

|  |  |  |
| --- | --- | --- |
| Domain Model | Category | Description |
| HostInterfaceProxy | Boundary | Host 시스템과 통신하는 Host Interface에 대한 Boundary 도메인 모델이다. Host Interface와 인터럽트를 주고받으며 명령을 해석하거나 명령으로 변환하는 역할을 한다. |
| DeviceInterfaceProxy | Boundary | USB Device와 통신하는 Device Interface에 대한 Boundary 도메인 모델이다. Device Interface와 인터럽트를 주고받으며 명령을 해석하거나 명령으로 변환하는 역할을 한다. |
| LedInterfaceProxy | Boundary | LED Controller와 통신하는 LED Interface에 대한 Boundary 도메인 모델이다. LED Interface와 인터럽트를 주고받으며 명령을 해석하거나 명령으로 변환하는 역할을 한다. |
| MemoryInterfaceProxy | Boundary | Memory Controller와 통신하는 Memory Interface에 대한 Boundary 도메인 모델이다. Memory Interface와 인터럽트를 주고받으며 명령을 해석하거나 명령으로 변환하는 역할을 한다. |
| StreamPipe | Control | Host 시스템과 USB Device 간의 데이터 전송을 담당하는 도메인 모델이다. USB 2.0 Specification에서 Host의 Client SW와 USB Device의 Function 간의 Endpoint에 해당하는 역할이다. |
| MessagePipe | Control | Host 시스템과 USB Device 간의 제어 신호를 전송하는 도메인 모델이다. USB 2.0 Specification에서 Endpoint Zero에 해당하는 역할이다. |
| Enumerator | Control | USB Device에 대한 열거 절차를 담당하는 도메인 모델이다. Host와 USB Device가 서로 인지할 수 있도록 Configuration 절차를 주도하며 Device Entity를 생성하고 Pipe를 개설한다. |
| SystemInitiator | Control | 시스템의 Boot 절차와 Halt 절차를 관리하는 도메인 모델이다. |
| ResourceObserver | Control | HostController의 상태를 체크하여 가용 가능한 자원에 대한 정보를 주기적으로 얻어와 저장하는 역할을 하는 도메인 모델이다. |
| ResourceNegotiator | Control | 가용 가능한 자원을 조회하고 USB Device에서 요청하는 자원에 대한 가용 범위를 결정하는 도메인 모델이다. |
| KnownDeviceList | Entity | 연결된 적 있는 USB Device에 대한 식별자를 저장하는 도메인 모델이다. USB Device의 VID, PID를 조합하여 USB Device의 고유한 값을 만들어 저장한다. |
| UsbClassPermission | Entity | 허용가능한 USB Class를 저장하는 도메인 모델이다. Host 시스템에게 질의하여 수용가능한 USB Class를 얻어와 해당 Entity에 저장한다. |
| BandwidthResource | Entity | 가용 가능한 대역폭 자원에 대한 정보를 저장하는 도메인 모델이다. |
| PowerResource | Entity | 가용 가능한 전력 자원에 대한 정보를 저장하는 도메인 모델이다. |
| DirectConnectedDevice | Entity | USB 허브에 직접 연결된 장치를 추상화한 도메인 모델이다. 직접 연결된 장치는 LED 장치와 Mapping되므로 플로우가 달라져 간접 연결된 장치와 구분하였다. 본 시스템은 직접적으로 USB Device와 연결되지 않기 때문에 USB Device에 대한 정보를 저장하는 역할을 한다. |
| IndirectConnectedDevice | Entity | USB 허브에 연결된 서브 허브를 통해 연결된 장치를 추상화한 도메인 모델이다. 본 시스템은 직접적으로 USB Device와 연결되지 않기 때문에 USB Device에 대한 정보를 저장하는 역할을 한다. |

**A2. Sequence Diagram**

**A2.1. USB Hub 연결**

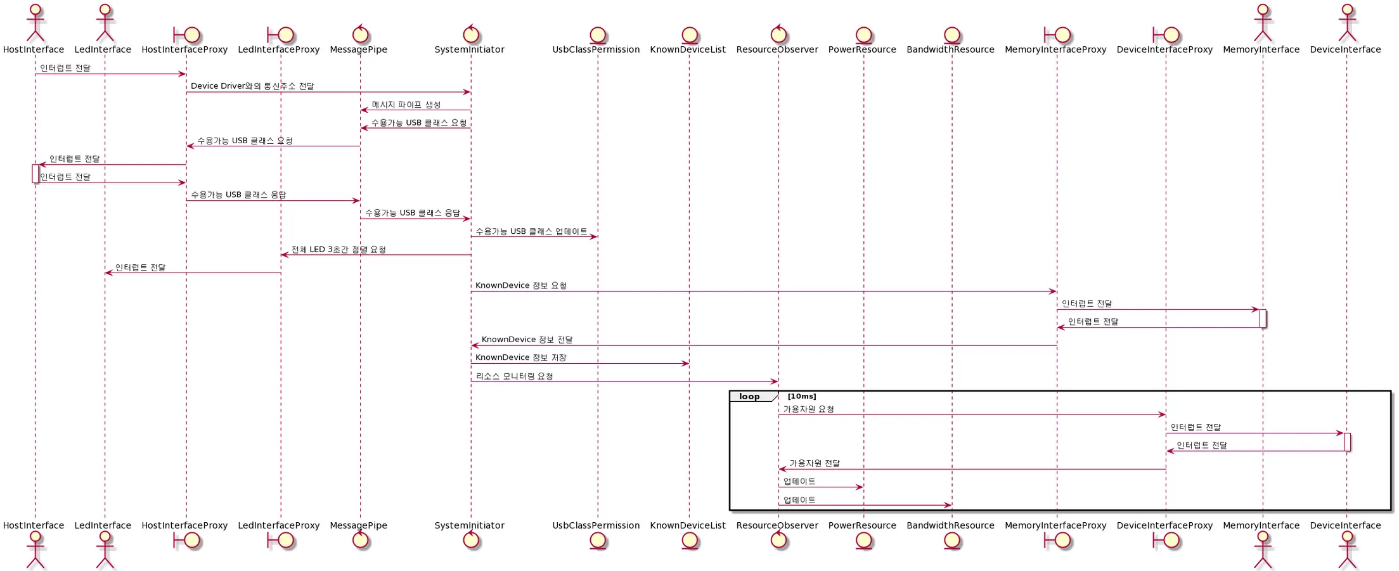
****

그림 - USB Hub 연결

**A2.2. USB Hub 연결 해제**

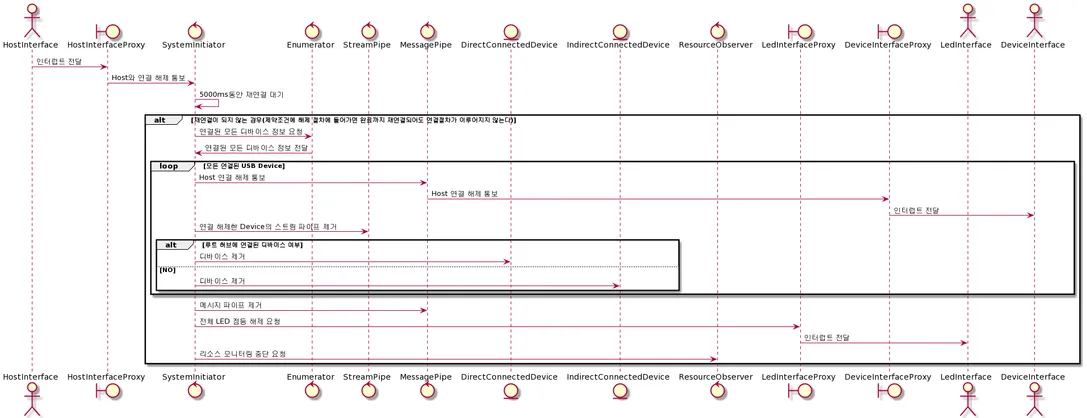
****

그림 - USB Hub 연결 해제

**A2.3. USB Device 연결**

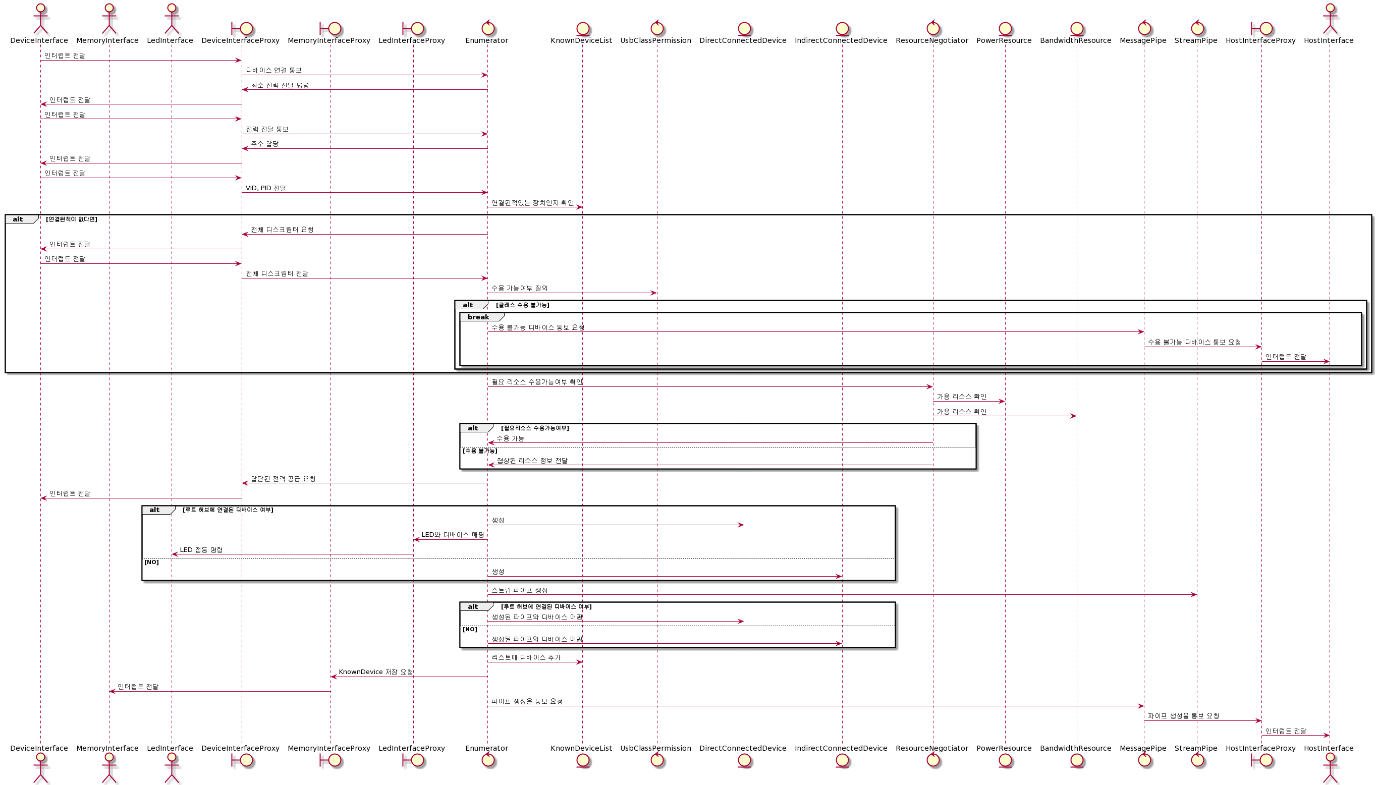


그림 - USB Device 연결

**A2.4. USB Device 연결 해제 요청**

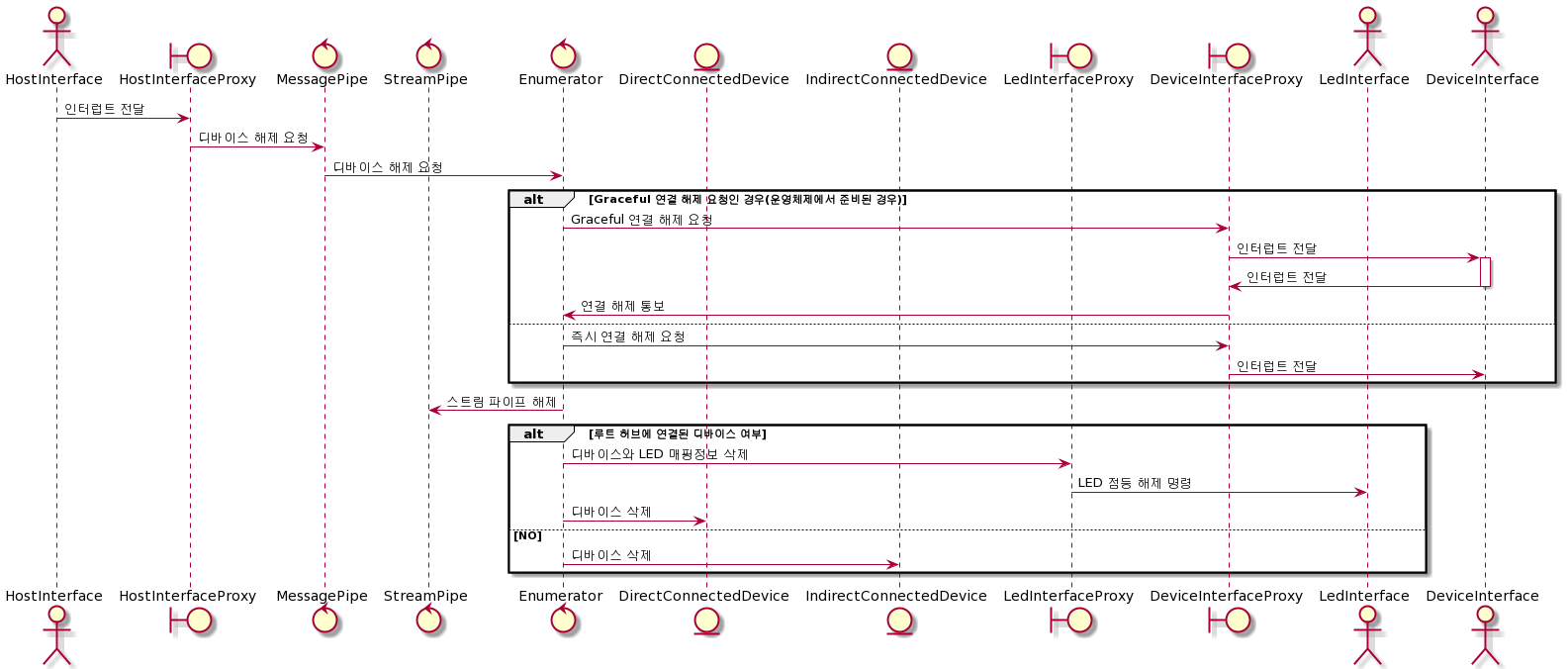


그림 - USB Device 연결 해제 요청

**A2.5. USB Device 연결 해제**

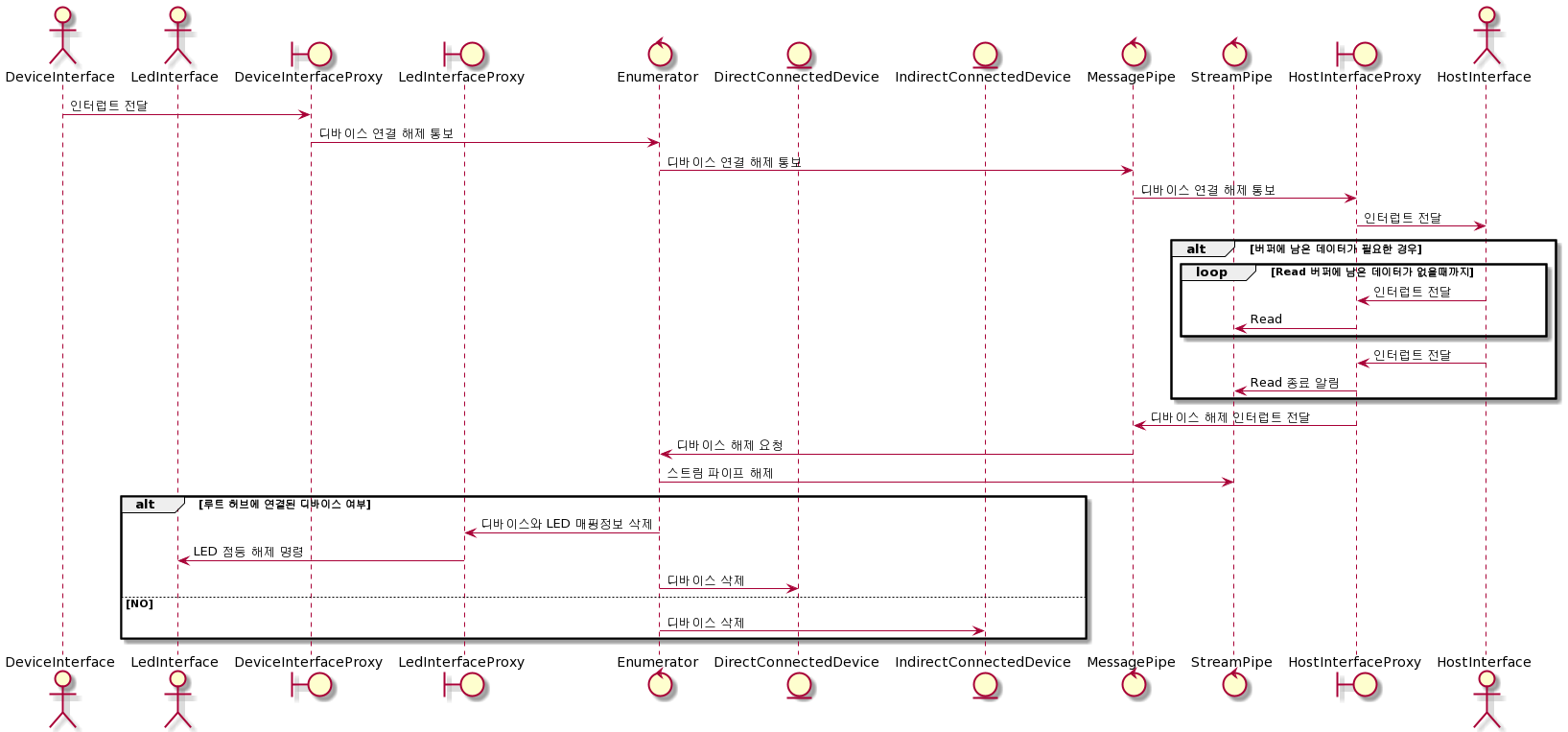


그림 - USB Device 연결 해제

**A2.6. 연결된 USB Device 정보 요청**

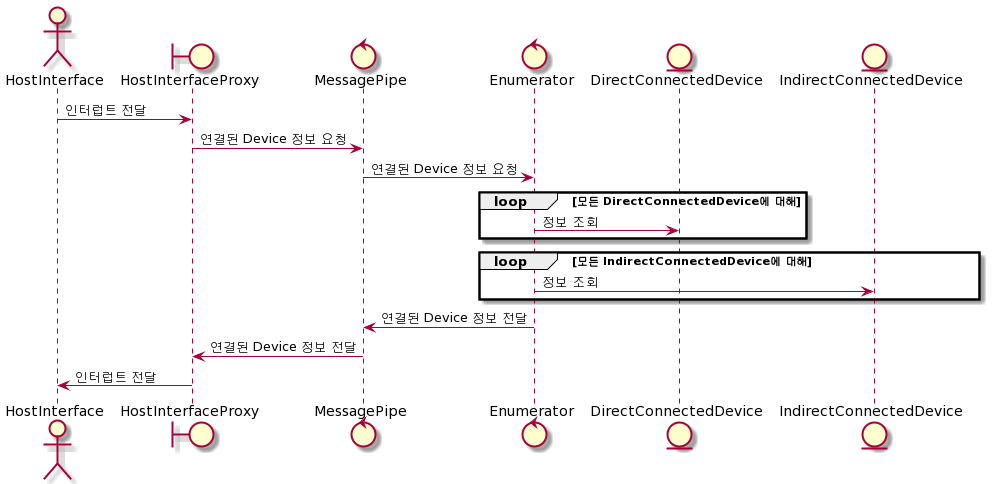


그림 - 연결된 USB Device 정보 요청

**A2.7. Host로 데이터 전송**

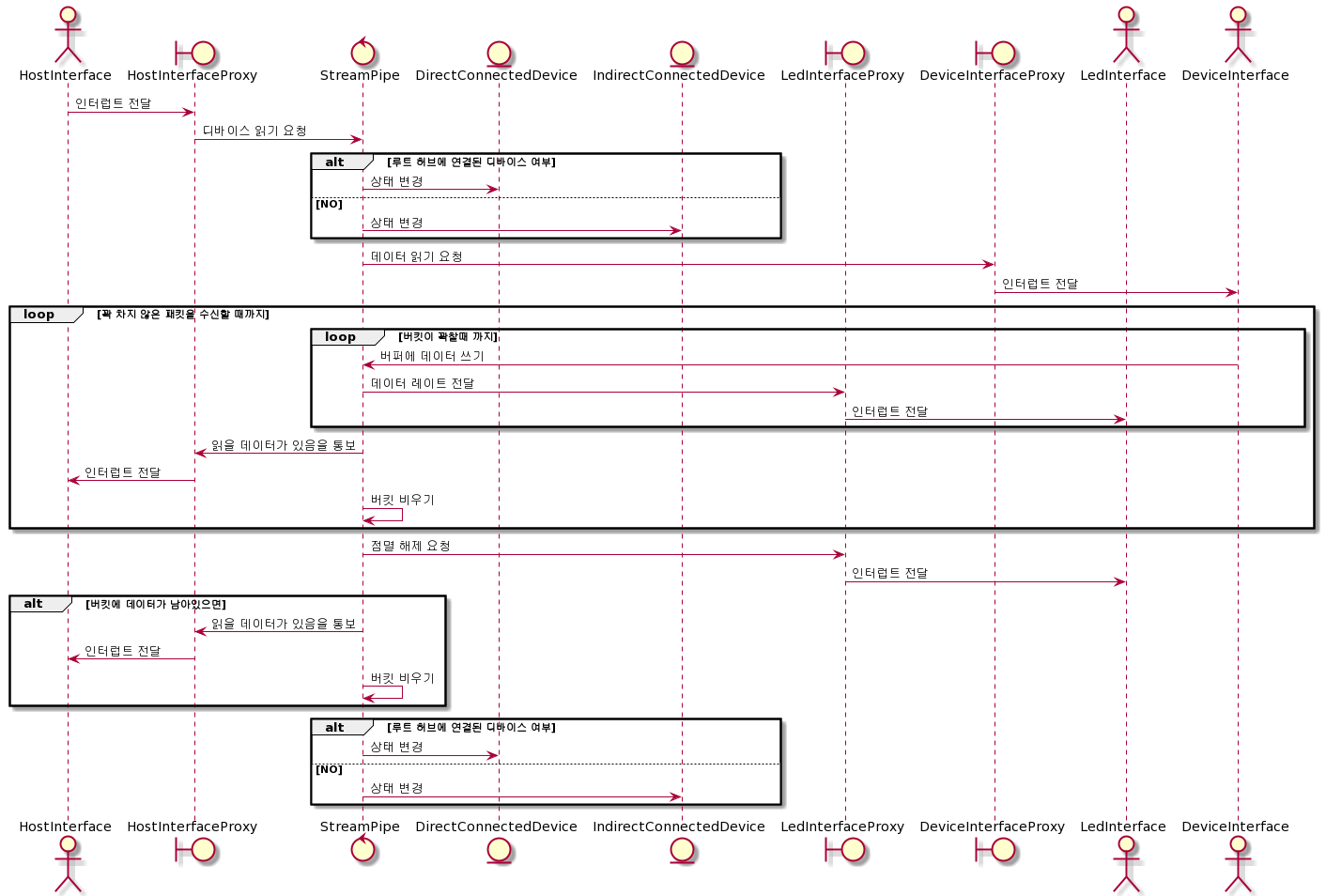


그림 - Host로 데이터 전송

**A2.7. USB Device로 데이터 전송**

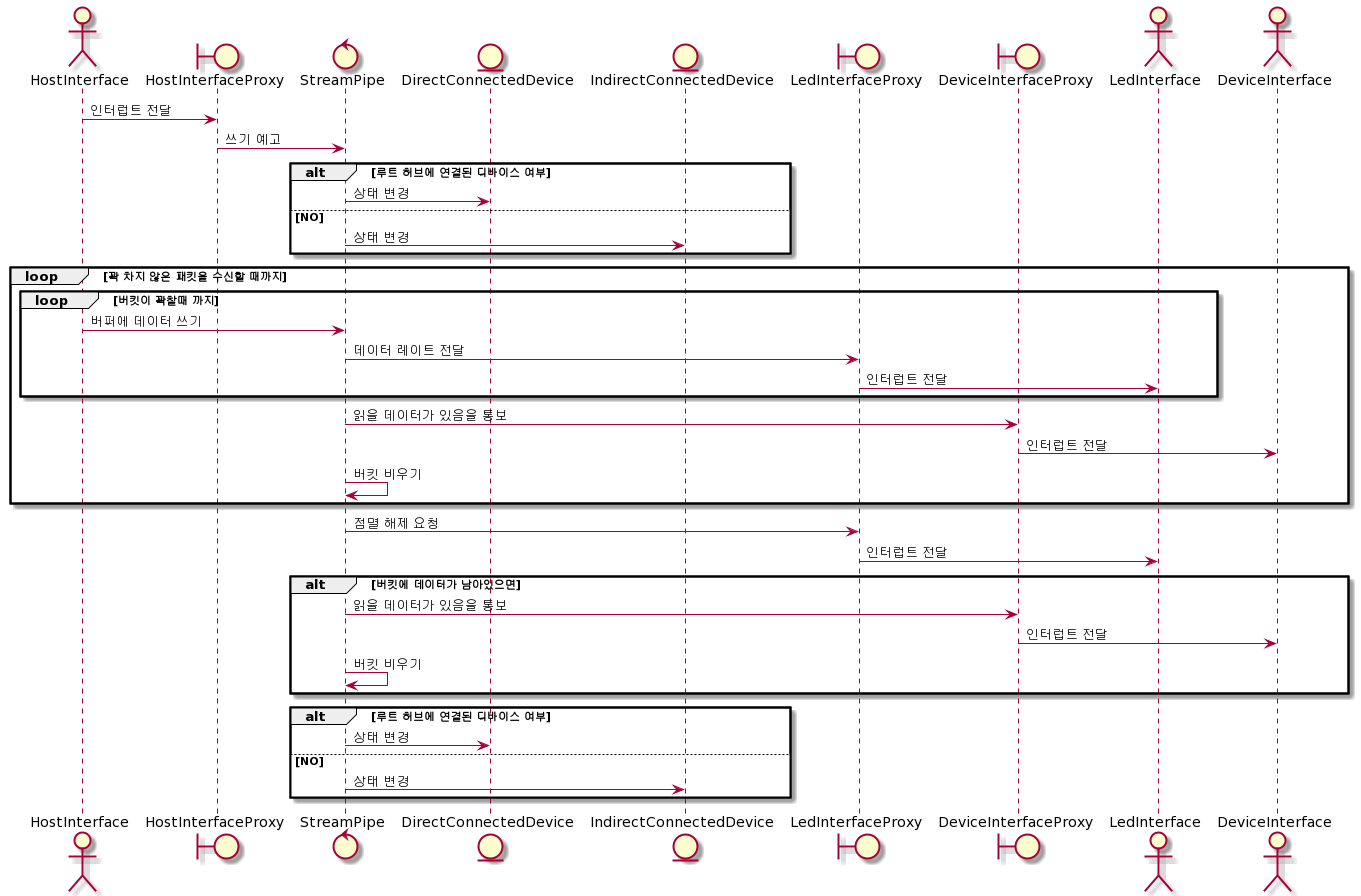


그림 - USB Device로 데이터 전송

1. 품질 시나리오

// 활동4. 품질 시나리오 생성

// 점검4-1. 구조에 영향을 미치는 품질에 대한 검토가 충분한가?

// 점검4-2. 성능에 대한 검토가 충분한가?

// 점검4-3. 변경용이성/확장성에 대한 검토가 충분한가?

Enumerator -> MessagePipe: 디바이스 연결 해제 통보(데이터 손실에 주의할 수 있도록 하기 위함 => 비기능요구사항)

OperationSystem -> StreamPipe: Read(파이프 버퍼에 남은 데이터까지는 리드할 수 있도록 하기 위함 => 비기능 요구사항)

1. 품질 시나리오 분석

// 활동5. 품질 속성 선정

// 점검5-1. 품질 시나리오 분석이 적절한가? (근거)