JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 22, No. 5, pp. 889-896, May. 2021

스마트 IoT 기기의 동작 검증을 위한 패킷 분석 프로그램

이 세 율[1] · 신 지 영[2] · 김 지 용[3] · 강 수 용[4] · 한 혁[5*]
[1]한양대학교 컴퓨터·소프트웨어학과 박사과정
[2]삼성전자 LED 사업부 QA그룹 책임연구원 / [3]삼성전자 LED 사업부 QA그룹 수석연구원
[4]한양대학교 컴퓨터·소프트웨어학과 교수 / [5*]동덕여자대학교 컴퓨터학과 부교수

Packet Analysis Program for Verifying the Behavior of Smart IoT Devices

Seyul Lee[1] · Jiyoung Shin[2] · Jiyong Kim[3] · Sooyong Kang[4] · Hyuck Han[5*]
[1]Ph.D Course, Department of Computer Science, Hanyang University, Seoul, Korea
[2]Senior Engineer, QA Group, Samsung LED, Korea / [3]Principal Engineer, QA Group, Samsung LED, Korea
[4]Professor, Department of Computer Science, Hanyang University, Seoul, Korea
[5*]Associate Professor, Department of Computer Science, Dongduk Women's University, Seoul, Korea

[요 약]

최근 유비쿼터스 시대에 진입하며 스마트 IoT 기기의 동작 검증이 중요한 문제로 떠오르고 있다. 특히 보통의 테스트 환경에서는 놓치기 쉬운, 실제 사용 케이스에서 발생할 수 있는 예외들을 처리하기 위한 blackbox test가 필요하다. 이 논문에서는 스마트 IoT 기기의 통신에서 주로 사용되는 프로토콜인 ZigBee와 BLE를 중심으로 blackbox test를 수행하는 프로그램인 Packet Analyzer를 제시한다. Packet Analyzer는 ZigBee와 BLE 패킷, beacon 패킷, IoT 허브와 스마트폰의 로그 등을 수집하고 패킷을 기반으로 명령을 구분하며 패킷 / 로그를 명령에 매칭하여 각 명령에 대해 OK 또는 NG 판정을 내린다. 실제 스마트 IoT 장치를 대상으로 실험하여 Packet Analyzer가 다양한 상황에 대해 올바른 판정을 내릴 수 있음을 보인다.

[Abstract]

Verifying the behavior of smart Iot devices in the ubiquitous era is emerging as an important issue. Blackbox testing is required to handle exceptions or errors that are difficult to detect in typical test environments, but may occur in real-world use cases. In this paper, we present Packet Analyzer, a program that performs blackbox testing on ZigBee and Bluetooth Low Energy (BLE), which are protocols mainly used in communication among Iot devices. Packet Analyzer collects ZigBee and BLE packets, beacon packets, IoT hub logs and smartphone logs. Then it analyzes test commands based on collected packets and logs to make decisions for each test command. Our evaluation on real-world smart IoT devices shows that Packet Analyzer can make correct decisions on various test commands.

색인어 : IoT, 테스트 프레임워크. 패킷 분석, 지그비, 블루투스 저전력 프로토콜

Key word : IoT, Test framework, Packet analysis, ZigBee, Bluetooth Low Energy

http://dx.doi.org/10.9728/dcs.2021.22.5.889

Received 05 March 2021; **Revised** 14 April 2021
Accepted 25 May2021

***Corresponding Author; Hyuck Han**

Tel: 02-940-4580
E-mail: hhyuck96@dongduk.ac.kr

Ⅰ. 서 론

2010년대에 스마트폰이 대중에게 널리 보급되면서 현재는 대부분의 사람들이 언제 어디서나 스마트폰을 사용하고 있다. 이 흐름에 따라 최근에는 스마트폰에 설치된 앱을 통해 조작이 가능한 스마트 IoT 기기들 또한 다수 개발되며 유비쿼터스 시대에 진입하고 있다. 스마트폰을 통해 스마트 IoT 기기들을 조작하기 위해서는 스마트 IoT 기기들이 항상 네트워크에 연결되어 있어야 하며, 이와 관련한 Internet of Things (IoT) 연구 또한 활발하게 진행되고 있다.

서로 다른 장치 간의 통신에는 표준화된 무선 통신 프로토콜들이 사용된다. 그림 1과 같이 스마트폰과 스마트 IoT 기기들의 통신에 주로 사용되고 있는 프로토콜에는 ZigBee[1]와 BLE (Bluetooth Low Energy)[2]가 있다. ZigBee는 기기를 IoT 허브를 통해 제어할 때 주로 사용되는 프로토콜로, IoT 허브를 중심으로 구성된 ZigBee 네트워크상에서 통신이 이루어진다. 다수의 기기를 동시에 연결하여 안정적인 네트워크를 구성하고 장거리의 기기들을 일괄적으로 통제할 수 있다는 것이 장점이다. 반면에 BLE는 근거리에서 스마트폰이 직접적으로 기기를 제어할 때 주로 사용되며, 스마트폰과 장치가 1대 1로 페어링되어 통신한다.

한편 IoT 서비스가 점점 활성화됨에 따라 IoT 장치의 동작을 검증하는 과정이 중요한 문제로 떠오르고 있다. 특히 사용자의 A/S 요청 시에는 개발 단계에서 사용할 수 있는 유닛 테스트나 디버깅 도구를 활용하기 어렵고, IoT 프로토콜들은 무선 통신이기 때문에 장치를 사용하는 과정에서 예기치 못한 상황을 발생시키는 경우가 잦기 때문에 문제의 원인을 찾기 어렵다. 실제로 기기를 사용할 때에는 통신 과정에서 주변의 다른 기기들에 의한 간섭이 발생할 수도 있으며, 사용자가 의도되지 않은 방법으로 스마트 앱을 조작하여 잘못된 신호가 전달될 수도 있다. 또한 실내에서 조작하는 경우와 실외에서 조작하는 경우 등의 다양한 사용 시나리오도 존재한다.

따라서 제조사에게는 개발자의 소스 코드에 의존한 디버깅 뿐만 아니라, 장치를 사용하는 실제 상황과 최대한 유사한 실험 환경에서 무선 통신의 동작을 검증하기 위한 blackbox test 시스템 또한 필요하다. Blackbox test는 개발 도구를 사용하여 기기를 조작하지 않고 사용자의 입장이 되어 온전히 기기 자체만을 사용하는 것이므로 사용 환경에 의해 특수한 상황에서만 발생할 수 있는 예상하지 못한 문제점을 찾는 데에 큰 도움을 준다.

Blackbox test를 통한 검증의 주요 과정은 패킷을 수집 후 해독 및 분석하는 것이다. 스마트폰에서 내린 명령이 장치에 올바르게 전달되어 수행되고 그 결과가 올바르게 나타나는지 검증하는 과정이다. 이 과정에는 다음과 같은 issue들이 존재한다. i) 무선 패킷은 암호화된 상태로 교환되는 경우가 많기 때문에 수집할 수 있는 정보가 제한될 수 있다. ii) 개별 패킷들의 정보만으로 그 패킷들이 어떤 명령으로부터 발생한 것인지를 알아내어 명령을 구분해서 기기 동작 과정 및 결과를 분석해야 한다.

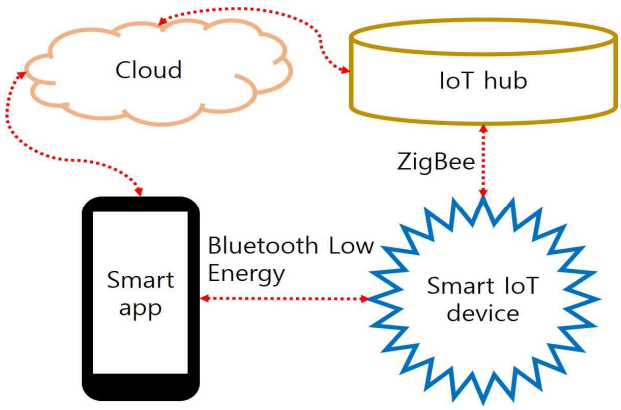


그림 1. 스마트 IoT 기기 네트워크
Fig. 1. Smart IoT device network

이 논문에서는 이러한 문제들을 해결하기 위해 통신을 검증하기 위한 blackbox 형식의 패킷 분석 도구인 Packet Analyzer를 제시한다. Packet Analyzer에서는 ZigBee 와 BLE 패킷들과 IoT 허브의 로그 및 스마트폰의 로그를 수집하고 이것들을 종합적으로 이용하여 명령 단위의 분석을 수행하고 각 명령에 대한 OK 또는 NG 판정을 내린다. 실험 평가를 통해 Packet Analyzer에 대한 여러 사용 시나리오에서의 테스트를 진행하였다.

이 논문의 구성은 다음과 같다. II장에서는 기존의 연구들에 대해 소개한다. III장에서는 패킷 분석 도구의 설계 및 분석 과정에서 발생할 수 있는 문제점들과 그에 대한 해결책을 제시하고, Packet Analzyer의 구현 상세를 기술한다. IV장에서는 실험을 통해 이 도구가 제시한 Packet Analyzer의 유효성을 보이고, V장에서는 이 논문에 대한 결론을 제시한다.

Ⅱ. 관련 연구

시스템 테스트는 완료 시 수행되는 테스트의 한 유형이며, 본 연구의 대상이 되는 IoT 장치의 동작 검증은 시스템 테스트에 해당한다. [3]의 연구에서는 휴리스틱을 기반으로 한 GUI 테스트에 대한 시스템 테스트 프레임워크를 제안했다. [4]에서는 컴퓨터 과학 분야의 테스트와 벤치마킹에 대한 프레임워크에 대해 기술되어 있다. [5]의 연구에서는 오픈 소스 소프트웨어(Apache HTTP Server, Mozilla Web Browser, NetBeans IDE)에서 세 가지 테스트 프로세스 및 테스트 프로세스 표준 ISO/IEC의 활동과 유사한 세 가지 활동에 대해 연구하였으며 오픈 소스 소프트웨어 테스트 프로세스 프레임워크(OSS-TPF)가 제안되었다.

[6]의 연구에서는 임베디드 소프트웨어 테스트 환경의 구성을 도입하기 위한 프레임워크를 제안했으며, 제안된 마이크로코어 플러그인을 통해 임베디드 소프트웨어 테스트 개발 환경을 설계할 수 있다. [10]의 연구에서는 실시간 임베디드 소프트웨어 시스템에서 온라인 모바일 기반 테스트(MBT)를 생성하고 배포하는 프레임워크를 테스트하기 위한 알고리듬을 제안한다.

표 1. 기존 도구들과의 비교
Table 1. Comparison to existing tools

Tool	Test on real device	Blackbox	Can use various information
Packet Analyzer	Y	Y	Y
PlatformIO[7]	Y	N	N
SimIoT[8]	N	Y	N
IoTIFY[9]	N	N	N

[11]의 논문은 지식 기반 시스템에서 시스템을 테스트하기 위한 프레임워크를 제안하였고, 이 프레임워크는 시스템 유효성 검사 노력을 줄이는 데 사용될 수 있다. 이러한 연구들은 네트워크 기능을 포함하지 않는 임베디드 시스템의 테스트 프레임워크를 제안하였다.

IoT 장치를 테스트하기 위한 연구들은 다음과 같다. [12]에서는 IoT 장치를 테스트하기 위한 기존의 도구와 테스트 방법들을 정리하고, 기존의 소프트웨어 개발 커뮤니티에서 널리 사용되던 최신 기술과 방법들을 IoT 개발에 도입할 필요성을 제시하였고, [13]에서는 IoT 네트워크의 모델과 사용 목적에 따라 각기 다른 효율적인 테스트 방법을 제시하였다.

표 1에서는 [12]에서 소개한 기존의 도구들과 Packet Analyzer를 비교하였다. 이 논문은 실제 IoT 장치의 동작을 패킷, 스마트폰 로그 등의 다양한 정보의 수집을 통해 종합하여 명령 단위의 무결성을 blackbox 방식으로 검증할 수 있게 한다는 점에서 기존 연구들과 차별된다.

III. Packet Analyzer의 설계 및 구현

이 장에서는 Packet Analyzer의 전체적인 설계를 설명하고, 통신 과정의 검증에서 발생할 수 있는 issue들과 Packet Analyzer에서 이들을 해결하기 위해 사용한 방법을 소개한다.

Packet Analyzer가 동작하는 시나리오는 그림 2와 같이 ZigBee를 사용할 때와 BLE를 사용할 때로 구분된다. 각각의 경우에 발생하는 패킷 (BLE 사용 시 beacon 포함)은 각각 nRF52840-Dongle과 nRF52840-DK를 통해 스니핑되어 Wireshark를 통해 JSON 형식으로 변환된 뒤 Packet Analyzer에 전달된다. ZigBee의 사용 과정에서 발생한 IoT 허브의 로그는 클라우드에 기록되고, 이를 Packet Analyzer가 크롤링을 통해 가져온다. 스마트폰에서는 ZigBee와 BLE를 사용할 때 각각 다른 종류의 로그가 발생하며 이 로그들도 Packet Analayer에 전달된다.

3-1 Packet Analyzer의 설계

1) 분석 정보의 수집

통신 과정의 검증을 blackbox 형식으로 수행하기 위해서는 무선 통신에서 발생하는 ZigBee와 BLE 패킷들을 스니핑을 통해 수집해야 한다.

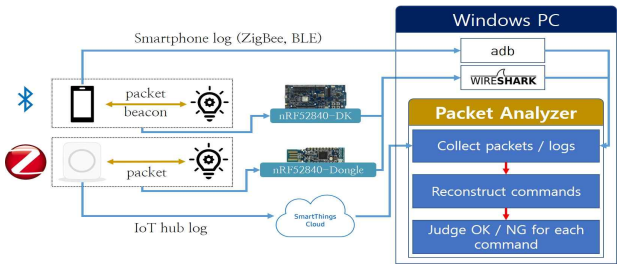


그림 2. Packet Analyzer 시스템
Fig. 2. Packet Analyzer system

그러나 스니핑은 통신하는 기기들의 허락을 받지 않고 중간에 끼어들어 엿듣는 것이기 때문에 스니핑으로 패킷 정보를 얻어내는 데에는 여러 가지 제약이 있다. 따라서 제안하는 Packet Analyzer는 효과적인 테스트를 위해 패킷 정보 외에도 IoT 허브가 남기기는 로그와 스마트폰이 남기는 로그를 최대한 활용한다.

ZigBee 패킷의 수집. ZigBee를 사용하여 IoT 허브와 연결할 때에는 ZigBeeAlliance09를 사용한 암호화 통신을 수행한다. 여기서는 미리 정해진 값의 키를 사용하기 때문에 해당 키를 사용하여 연결 과정에서의 패킷들을 해석할 수 있다. 연결 과정의 마지막에는 두 기기가 이후 장기적으로 통신에 사용할 키를 교환하는 단계가 있고 이 키의 값 또한 읽을 수 있으므로 연결된 이후의 통신에서 발생하는 패킷들의 내용을 모두 해석할 수 있다.

IoT 허브의 로그 수집. ZigBee를 사용하는 경우 IoT 허브가 각 명령에 대한 로그를 별도로 클라우드에 기록할 수 있다. 클라우드에 기록되는 정보로는 명령의 종류와 명령의 값, 그리고 명령을 수행한 시간 등이 있다. 이 기록을 API를 사용하거나 웹 페이지에 기록되는 경우 크롤링을 통하여 가져올 수 있다. IoT 허브의 로그로부터는 명령의 정확한 내용을 항상 알 수 있으므로 교환된 ZigBee 패킷 내역과 비교하여 분석의 정확성을 높인다. 만일 IoT 허브에서 기록한 로그가 캡처된 패킷과 일관성이 없다면 어딘가에서 오작동이 있었음을 유추할 수 있다.

BLE 패킷의 해석. BLE의 패킷을 완벽하게 해석하는 데에는 어려움이 있다. 기기들이 BLE로 페어링하는 과정에서는 초기에 short term key (STK)를 생성하는데, 이때 공격자가 키를 가로챌 수 없도록 Diffie-Hellman key exchange[14]를 사용하기 때문에 제3자는 이 키의 값을 알아낼 수 있는 방법이 없다. 페어링 과정의 나머지 부분의 패킷들은 모두 이 STK를 이용하여 암호화되어 교환되고, 이 과정에서 이후 두 장치가 장기적으로 통신에서 사용할 별도의 키가 교환되므로 그 값을 알아낼 수 없다.

따라서 BLE 패킷의 스니핑에서 획득할 수 있는 정보에는 한계가 있다. 그러나 값을 알아낼 수 없는 키를 통해 암호화되는 정보는 각 패킷의 데이터 필드에 제한되기 때문에 그 이외의 필드로부터 유의미한 정보를 얻어낼 수 있다. 암호화되지 않는 필드 중에는 패킷의 종류 (write request, write response 등), 명령의 universally unique identifier (UUID), 그리고 service UUID 등이 있고, BLE 프로토콜의 generic attribute profile (GATT)에 정의된 값을 참조하여 각 패킷이 어떤 종류의 명령을 내린 것인지, 또는 어떤 명령에 대한 응답을 한 것인지를 판단할 수 있다.

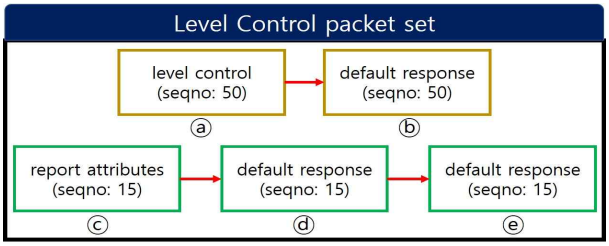


그림 3. Level Control 패킷 집합
Fig. 3. Packet Set of Level Control

예를 들어, 특정 BLE 패킷에서 밝기 수준의 알아낼 수 없지만 해당 패킷이 LED의 밝기를 조절하는 패킷이라는 것은 해석할 수 있다.

BLE beacon 패킷의 수집. BLE의 경우 구체적인 데이터 필드의 값을 명령 패킷으로부터 직접 알아내지 못하는 대신 beacon 패킷을 분석하는 우회적인 방법을 사용할 수 있다. 기기에서는 짧은 주기로 자신의 상태를 advertising하는 beacon 패킷을 전송하는데, beacon 패킷의 데이터 필드는 암호화 되어있지 않으므로 특정 명령이 내려진 전후의 상태 변화를 분석하여 각 명령의 내용을 구체적으로 파악할 수 있다. 예를 들어 LED 장치의 밝기가 명령 이전에 a였으나 명령 이후 b로 변한 것을 beacon에서 확인했다면 그 명령의 내용은 b로 밝기를 조절하는 명령임을 유추할 수 있다.

스마트폰 로그의 수집. 스마트폰에서의 로그는 스마트 어플리케이션이 BLE 통신을 수행할 때마다 사용한 API에서 디버깅용으로 남기는 것으로, 일반적으로 전송한 패킷에 대한 요약 정보가 포함되어 있다.

2) 명령의 구분

사용자가 명령을 발생시키면 스마트폰과 기기, 또는 IoT 허브는 정해진 수순에 따라 패킷들을 발생시키고 그에 대한 로그를 남긴다. 어떤 명령이 내려졌을 때 발생하는 패킷 및 로그의 목록을 그 명령에서의 패킷·로그 집합이라고 부른다. 검증 프로그램은 수집한 정보로부터 이러한 패킷·로그 집합들을 감지하여 그 시점에서 내려진 명령이 무엇이었는지를 파악해야 하고, 각 패킷이나 로그가 어떤 명령에 의해 발생했는지 알아내야 한다. 즉, 명령의 구분이 이루어져야 한다. 이 과정은 크게 명령의 감지와 매칭의 두 단계로 이루어진다.

명령의 감지. 분석의 주 대상인 ZigBee 또는 BLE 패킷을 기준으로 하여 명령을 감지하는 단계이다. 특정한 종류의 패킷은 특정한 명령에 의해서만 발생한다는 점을 이용하여, 어떤 명령의 패킷·로그 집합에 속한 패킷이 모두 캡처되었다면 그 명령이 내려진 것으로 간주할 수 있다. 예를 들어 명령 C의 패킷·로그 집합에 속한 패킷이 순서대로 P, Q, R이라고 할 때, P 패킷을 찾은 경우 이후 Q와 R 패킷도 캡처되었는지를 확인하여 모두 존재하는 경우 명령 C가 내려진 것으로 간주할 수 있다.

하나의 패킷·로그 집합에 속한 패킷들이 항상 일정한 순서로 캡처되지 않을 수도 있다. 이는 패킷이 모두 연속적으로 하나씩 교환되는 것이 아니라 서로 다른 sequence number를 가진 패킷들이 병렬적으로 동시에 교환될 수 있기 때문에 발생하는 문제이다. 예를 들어 level control 명령은 그림 3과 같은 패킷 집합을 가지고 있는데, IoT 허브는 첫 패킷인 ⓐ 패킷을 전송한 이후 응답 패킷인 ⓑ 패킷을 기다리지 않고 곧바로 ⓒ 패킷을 이어서 전송할 수 있다. 따라서 패킷의 발생순서는 ⓐ - ⓑ - ⓒ - ⓓ - ⓔ 일 수도 있고, ⓐ - ⓒ - ⓑ - ⓓ - ⓔ일 수도 있기 때문에 특정한 순서로 패킷을 탐색해서는 명령을 올바르게 구분하지 못할 수도 있다. Packet Analyzer에서는 이를 고려하여 서로 같은 sequence number를 가진 패킷들을 그룹화하고 그룹별로 개별적으로 탐색하여 명령을 구분한다. 그림 3의 예시에서 ⓐ, ⓑ 패킷을 순서대로 먼저 찾은 뒤, ⓐ가 발생한 시간으로부터 다시 ⓒ, ⓓ, ⓔ 패킷을 순서대로 찾는 방식이다.

매칭 프로시저. 감지된 명령 각각에 ZigBee 또는 BLE 패킷 이외의 다른 로그들을 대응시키는 단계이다. 매칭 프로시저의 목표는 크게 "시간적으로 가장 가까운 명령과 매칭하는 것"과 "실제와 다른 명령에 매칭되지 않게 하는 것"이다.

매칭 프로시저에서는 로그를 기록하는 장치들 사이의 timestamp 오차가 중요한 문제이다. 서로 다른 장치에서 로그를 남기기 때문에 각각의 장치가 사용하는 시계가 서로 동기화되지 않을 수 있어, 한 장치에서 시간 t에 기록한 로그가 실제로는 시간 t+a에 기록한 로그일 수도 있다. 따라서 검증 프로그램에서는 장치 사이의 timestamp 오차가 발생할 수 있음을 고려하여 매칭을 수행한다.

반면에 오차를 너무 크게 허용하는 경우 실제와 다른 명령에 잘못 매칭이 될 가능성도 높아진다. 만일 어떤 이유로 특정 명령에 대한 로그가 남지 않았다면 그 로그는 수집하지 못한 것으로 처리해야 하는데, 오차를 허용한 것 때문에 그 이후에 발생한 다른 명령에서 남긴 로그에 매칭을 하게 될 수도 있기 때문이다. 따라서 오차를 허용하는 정도는 각 로그를 기록하는 장치의 동작 방식에 따라 유연하게 조절해야 한다. 이와 같은 오차 허용은 패킷·로그 집합에 속한 패킷들을 찾는 데에도 동일하게 적용한다.

3-2 Packet Analyzer의 설계

Packet Analyzer는 스마트 LED 장치를 대상으로 구현하였다. IoT 허브로는 SmartThings Hub를 사용하였으며, 스마트폰으로는 Galaxy S8을 사용하였고, 스마트폰에서는 SmartThings 어플리케이션을 구동하였다. Packet Analyzer에서 분석하는 정보는 모두 이 기기들 사이에서의 통신에서 발생하는 정보들을 기반으로 하며, 인식할 수 있는 명령은 SmartThings 어플리케이션에서 발생시킬 수 있는 네 가지 명령인 On, Off, 밝기 조절, 색온도 조절이다. 프로그램의 구현을 위해 Python 3.8을 사용하였고, PyQt5로 GUI를 구현하였다. SmartThings Hub의 로그를 크롤링하기 위한 모듈로는 chromedriver[21]를 사용하였다.

1) 정보의 수집

스니핑을 수행하기 위해서는 ZigBee 및 BLE 프로토콜을 인

식할 수 있는 특수한 하드웨어와 펌웨어가 필요하다. ZigBee 패킷의 스니핑에는 nRF52840 Dongle[15]과 nRF Sniffer for 802.15.4[16]를 사용하였고, BLE 패킷과 beacon 패킷의 스니핑에는 nRF52840 DK[17]와 nRF Sniffer for Bluetooth LE[18]를 사용하였다. 패킷들은 모두 Wireshark[19]를 통해 수집하여 JSON 형태로 저장한 뒤 Packet Analyzer에서 사용하였다.

ZigBee의 경우 각 명령을 수행한 직후 SmartThings Hub에서 클라우드에 로그를 기록한다. 이 로그에서는 각 명령을 수행한 시간, 명령의 종류, 그리고 값 등의 정보를 확인할 수 있다. 이 정보는 SmartThings Groovy IDE[20]에서 웹 페이지의 형태로 제공되며, 이를 chromedriver를 통해 크롤링을 통해 정보를 받아온다.

스마트폰 로그는 SmartThings 어플리케이션에서 디버깅용으로 남기는 것을 사용하였다. 로그의 종류는 ZigBee를 사용할 때와 BLE를 사용할 때가 서로 다른데, ZigBee의 경우 명령을 네트워크를 통해 전송하고 올바른 응답을 받았다는 정보를 확인할 수 있으며, BLE의 경우 각 명령의 종류와 값을 확인할 수 있다. 이 정보는 스마트폰을 컴퓨터에 USB로 연결한 후 Android 디버그 브리지 (adb)[22]의 logcat 기능을 사용하여 얻어온 뒤 연관된 로그들만 필터링하여 수집하였다.

2) 명령의 구분

Packet Analyzer에서는 JSON으로 저장된 패킷 정보를 필터링하여 명령과 연관된 패킷들만을 사용한다. ZigBee 또는 BLE 패킷 목록을 순서대로 탐색하며 각 명령에서 처음으로 발생하는 패킷을 발견하면 그 시점으로부터 일정 시간 범위 내의 패킷들을 탐색하여 패킷·로그 집합에 속한 패킷들을 모아 그룹화한다. 서로 다른 명령들의 첫 패킷이 같은 경우 두 명령에 대한 패킷·로그 집합을 각각 탐색하여 더 가능성이 높은 쪽으로 그룹화한다.

Hub의 로그 및 BLE의 스마트폰 로그와 같이 명령 하나당 하나의 로그가 남는 경우에는 지정된 timestamp leniency 내에서 시간상 가장 가까우면서 명령의 종류가 일치하는 그룹과 일대일 매칭을 한다. ZigBee의 스마트폰 로그의 경우에는 명령 하나에 여러 로그가 함께 남으므로 이들을 모두 찾아서 묶은 뒤 매칭한다.

BLE beacon 패킷의 경우에는 각 패킷이 명령에 직접적으로 대응하는 것이 아닌, 명령의 수행 전후로 상태가 바뀌는 것을 관찰하는 것이 주목적이다. 연속적으로 같은 상태가 반복되는 패킷을 모아 그룹화하고, 각 명령이 발생한 시간 전의 그룹과 후의 그룹을 각각 찾아 둘을 모두 매칭시키는 방법을 사용하였다.

Ⅳ. 실험 및 평가

Packet Analyzer의 평가를 위해 4-1과 4-2에서는 스마트 LED 장치의 여러 사용 케이스에 대한 실험을 진행하였다. 실험은 SmartThings 어플리케이션에서 On, Off, 밝기 조절, 색온도 조절 명령을 내리고 이때 발생하는 패킷과 로그를 분석하여 상황에 맞는 판정을 내리는지 확인하는 방식을 사용하였다. 실외에서의 조작 실험을 제외한 나머지 실험은 모두 실내 (약 50cm 거리)에서 진행하였다. 기기의 상태가 Off일 때에는 On 명령만을 수행하였고, BLE 실험 중 상태 변화가 없는 명령 실험을 제외한 모든 실험은 현재 상태와 다른 상태가 되도록 하는 명령만을 수행하였다. 4-3에서는 Packet Analyzer가 패킷 정보의 결함을 발견했을 때 NG를 판정하고 문제에 대한 유용한 정보를 제공할 수 있는지에 대한 실험을 수행하였다.

4-1 ZigBee 검증 실험

ZigBee 검증 실험으로는 실외에서의 조작, 짧은 시간 간격의 명령, 그리고 timestamp leniency 실험의 세 가지 실험을 진행하였다. 모든 실험은 네 가지 명령 중 임의의 명령을 무작위로 선택하는 것을 반복하였다. 표 2에 ZigBee 검증 실험 결과를 기술하였다.

1) 실외에서의 조작

ZigBee의 통신은 스마트폰과 IoT 허브가 인터넷에 연결된 상태에서 이루어지기 때문에 실외에서도 온전한 조작이 가능해야 한다. 표 2의 실험 1의 결과는 스마트 IoT 기기와 약 30m 거리에서 내린 명령을 Packet Analzyer가 인식할 수 있음을 보여준다.

2) 짧은 시간 간격의 명령

짧은 시간 간격을 두고 연속적으로 명령을 내리는 경우 여러 명령에서 발생한 패킷들이 서로 순서가 섞일 수 있다. 표 2의 실험 2의 결과는 Packet Analyzer가 이러한 경우에도 각 패킷이 어떤 명령의 일부인지 찾아낼 수 있음을 보여준다.

3) Timestamp leniency 실험

로그 기록 장치와의 시간 오차가 크거나 불안정한 경우 timestamp leniency를 사용자가 지정하여 매칭의 정확성을 높일 수 있다. 표 2의 실험 3은 IoT 허브의 로그가 일부 늦게 기록되는 환경에서 timestamp leniency를 1초로 설정했을 때 일부 로그를 매칭하지 못해 NG로 판단한 것을 보여주며, 실험 4에서는 timestamp leniency를 3초로 설정하여 모든 명령을 OK로 판정한 것을 보여준다.

4-2 BLE 검증 실험

BLE 검증 실험으로는 실외 / 실내에서의 조작, 짧은 시간 간격의 명령, 그리고 상태에 변화가 없는 명령에 대한 실험을 진행하였다. 상태에 변화가 없는 명령에 대한 실험은 밝기 조절과 색온도 조절만을 측정하였고, 나머지 실험은 모두 무작위 명령을 사용하였다. 표 3에 BLE 검증 실험 결과를 기술하였다.

1) 실외 / 실내에서의 조작

표 3의 실험 1에서, BLE는 유효 거리가 약 10m로 짧기 때문에 거리가 멀어지면 연결이 끊겨 어플리케이션이 종료되어 실외에서는 명령을 수행할 수 없었다. 대신 실험 2와 같이 실내에서 실험했을 때에는 모든 명령에 대해 OK를 판정해내었다.

표 2. ZigBee 검증 실험 결과
Table 2. ZigBee verification test results

Exp. no.	Details	# of cmd.	OK	NG	Command type
1	Outdoor commands (~30m distance)	100	100	0	random
2	Commands with short time interval	100	100	0	random
3	Timestamp leniency 1 second	100	87	13	random
4	Timestamp leniency 3 seconds	100	100	0	random

표 3. BLE 검증 실험 결과
Table 3. BLE verification test results

Exp. no.	Details	# of cmd.	OK	NG	Command type
1	Outdoor commands (~30m distance)	N/A	N/A	N/A	random
2	Indoor commands	100	100	0	random
3	Commands with short time interval (wait time 0.5 seconds)	100	63	37	random
4	Commands with short time interval (wait time 0.2 seconds)	100	100	0	random
5	Commands that don't change device status	100	0	100	level control, color control

2) 짧은 시간 간격의 명령

짧은 시간 간격 (약 0.3초)으로 내리는 명령에 대해 beacon 상태의 변화를 올바르게 매칭하는지에 대한 실험이다. 명령이 전송된 후 기기의 상태가 변하기까지 다소 시간이 소요될 수 있기에 Packet Analyzer에서는 명령이 내려진 시간 이후의 beacon 패킷을 찾기 위해 명령 패킷의 시점보다 일정 시간 이후의 beacon 패킷을 명령 후 상태로 인식한다. 이 여유 시간을 실험 환경에서 0.5초로 설정했을 경우 실험 3과 같이 일부 명령을 NG로 판단한 것을 볼 수 있는데, 이는 연속된 두 명령이 기기의 상태를 S에서 다른 상태 T로 바꾸었다가 다시 S로 만들 경우 Packet Analyzer가 명령의 전후로 선택한 beacon 패킷의 상태가 둘 다 상태 S에 대한 패킷일 수 있기 때문이다. 여유 시간을 0.2초로 변경하여 실험한 결과 실험 4와 같이 모든 결과가 OK로 판정되었는데, 이 값은 실험 대상 기기의 동작에 따라 적절하게 조정되어야 할 것으로 보인다.

3) 상태에 변화가 없는 명령

밝기 조절과 색온도 조절 명령은 어플리케이션에서 현재 값과 동일한 값으로 설정하는 명령을 내리는 것이 가능하다. 실험 5에서는 이러한 명령들만을 수행했을 때 beacon에 의해 판정을 NG로 내리는 것을 확인하였다. 이는 beacon을 판정 기준에 사용할 경우 그 시점에 어떤 명령이 있었음을 판단할 수 있는 근거가 상태의 변화뿐이기 때문이다.

4-3 문제 분석 실험

캡처한 패킷 내역에 결함이 있을 때 이를 구체적으로 분석하는 기능을 실험하였다. 실험을 진행한 기기와 환경에서 결함을 의도적으로 발생시키기 어려운 관계로, OK로 판정된 정상적인 패킷 캡처 내역의 일부 중요 패킷을 변조 혹은 삭제하였을 때의 프로그램의 동작을 실험하였다.

그림 4는 정상적인 패킷 캡처 파일과 변조한 패킷 캡처 파일을 각각 Packet Analyzer를 통해 분석한 결과에 대한 보고서이다. 변조한 패킷 캡처 파일에서는 세 번째의 Level 명령에서 요구되는 Report Attributes 패킷과 Default Response 패킷을 삭제하고, 여섯 번째의 Color Temperature 명령에서 요구되는 Read Attributes Response 패킷을 삭제하였다. 두 명령에 대해 모두 명령 자체는 인식하였으나, 결여된 패킷이 있음을 인지하고 NG로 판정하며 그 패킷이 어떤 패킷인지에 대한 자세한 정보를 보고하였다. 이 실험을 통해 Packet Analyzer는 명령을 구성하는 패킷의 일부만으로도 어떤 명령이었는지 구분할 수 있으며 문제점의 세부 사항을 분석하는 데에 유용함을 볼 수 있다.

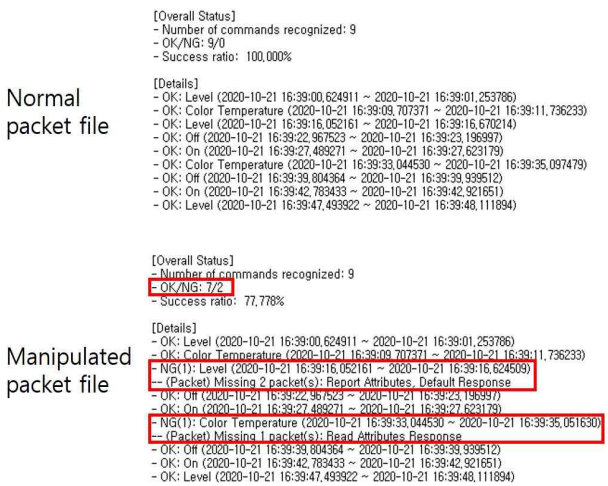


그림 4. 정상 패킷 파일과 변조된 패킷 파일에 대한 보고서
Fig. 4. Report for a normal and a manipulated packet file

Ⅴ. 결 론

이 논문에서는 스마트 IoT 장치의 동작을 blackbox 형식으로 검증하기 위한 ZigBee / BLE 패킷 분석 프로그램인 Packet Analyzer를 제시하였다. Packet Analyzer의 설계와 이 과정에서 마주한 문제점들 및 그 해결책을 제시하였고 실제 스마트 IoT 기기를 대상으로 한 테스트를 통해 이 프로그램이 다양한 사용 케이스에 대해 패킷을 비롯한 여러 정보들을 종합적으로 분석하여 합리적인 OK / NG 판정을 내릴 수 있음을 보였다. 또한 패킷 정보에 결함이 있는 경우 이를 찾아내어 문제의 원인을 파악하는 데에 도움이 될 수 있음을 보였다. 향후에는 제안하는 프로그램이 IoT 기기 개발사의 검증 절차 및 A/S 과정에 적용되어 다양한 검증 시나리오를 수행할 수 있도록 확장할 예정이다.

감사의 글

이 논문은 한국연구재단의 지원에 의하여 수행된 것임. (NRF-2020R1F1A1055489)

참고문헌

[1] ZigBee Alliance, ZigBee, https://zigbeealliance.org/solution/zigbee/

[2] Heydon, Robin, and Nick Hunn. “Bluetooth low energy.” *CSR Presentation, Bluetooth SIG https://www. bluetooth. org/DocMan/handlers/DownloadDoc. ashx*, 2012.

[3] S. McMaster and A. M. Memon, “An Extensible Heuristic-Based Framework for GUI Test Case Maintenance,” *2009 International Conference on Software Testing, Verification, and Validation Workshops*, pp. 251-254. 2009.

[4] D. Eisenbiegler and F. Feigenbutz, “Schwarzwälder — An online test framework,” *2012 6th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, pp. 1-6, 2012.

[5] T. Abdou, P. Grogono and P. Kamthan, “A Conceptual Framework for Open Source Software Test Process,” *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*, pp. 458-463, 2012.

[6] Y. Yin, B. Liu and G. Zhang, “On framework oriented embedded software testing development environment,” *8th International Conference on Reliability, Maintainability and Safety*, pp. 708-712, 2009.

[7] PlatformIO Labs, A Professional collaborative platform for embedded development · PlatformIO, https://platformio.org/

[8] S. Sotiriadis, N. Bessis, E. Asimakopoulou, and N. Mustafee, “Towards simulating the internet of things,” *IEEE 28th International Conference on Advanced Information Networking and Applications Workshops*, pp. 444–448, 2014.

[9] Ternary GmbH, IoTIFY - cloud based IoT simulator and IoT testing platform | IoTIFY is industry’s first cloud-based IoT simulation and Enterprise IoT testing platform, https://iotify.io/

[10] P. Iyenghar, “Test Framework Generation for Model-Based Testing in Embedded Systems,” *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 267-274, 2011.

[11] P. Bera and A. Pasala, “A Framework for Optimizing Effort in Testing of System of Systems,” *2012 Third International Conference on Services in Emerging Markets*, pp. 136-141, 2012.

[12] J. P. Dias, F. Couto, A. C. R. Paiva and H. S. Ferreira, “A Brief Overview of Existing Tools for Testing the Internet-of-Things,” *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Västerås, Sweden, pp. 104-109, 2018.

[13] S. Popereshnyak, O. Suprun, O. Suprun and T. Wieckowski, “IoT application testing features based on the modelling network,” *2018 XIV-th International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, Lviv, Ukraine, pp. 127-131, 2018.

[14] Diffie, Whitfield, and Martin Hellman. “New directions in cryptography.” *IEEE transactions on Information Theory 22.6*, pp. 644-654, 1976.

[15] Nordic Semiconductor, nRF52840-Dongle, https://www.nordicsemi.com/Software -and-tools/Development-Kits/nRF52840-Dongle

[16] Nordic Semiconductor, nRF-Sniffer-for-802154, https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Sniffer-for-802154

[17] Nordic Semiconductor, nRF52840-DK, https://www.nordicsemi.com/Software -and-Tools/Development-Kits/nRF52840-DK

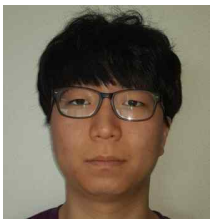
[18] Nordic Semiconductor, nRF-Sniffer-for-Bluetooth-LE, https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Sniffer-for-Bluetooth-LE

[19] Wireshark Foundation, WireShark, https://www.wireshark.org/

[20] SmartThings, Inc. SmartThings API, https://graph.api.smartthings.com/

[21] Chromium team, ChromeDriver, https://chromedriver.chromium.org/

[22] Google LLC, Android Debug Bridge (adb), https://developer.android.com/studio/command- line/adb



이세율(Seyul Lee)

2019년 : 한양대학교 컴퓨터전공 (공학사)

2019년～현　재: 한양대학교 대학원 석박사통합과정

※관심분야 : 분산시스템(Distributed System), 시스템 소프트웨어 개발(System Software Development) 등

신지영(Jiyoung Shin)

2010년 : 한양대학교 화학공학과 (학사졸업)

2010년～2011년 : LG display 모니터 패널설계 재직

2011년～현　재 : 삼성전자 LED사업팀 QA그룹 재직

2010년～2011년: LG

2011년～현　재: 삼성전자

※관심분야 : 품질평가기법(Quality Evaluation Methodology), LED, IoT

김지용(Jiyong Kim)

2001년 : 건국대학교 전기공학과 졸업

2001년～2010년 : (주)CLT 자동차용 LED 개발

2010년～현　재 : 삼성전자 LED사업팀 품질팀 QA그룹

2001년～2010년: (주)CLT

2010년～현　재: 삼성전자

※관심분야 : 품질평가기법(Quality Evaluation Methodology), LED, IoT

강수용(Sooyong Kang)

1998년 : 서울대학교 대학원 (공학석사)

2002년 : 서울대학교 대학원 (공학박사-분산시스템)

2003년～현　재: 한양대학교 소프트웨어학과　교수

※관심분야 : 운영체제(Operating System), 데이터베이스시스템(DBMS) 등

한혁(Hyuck Han)

2006년 : 서울대학교 대학원 (공학석사)

2011년 : 서울대학교 대학원 (공학박사-분산시스템)

1999년～2003년: 델타정보통신

2011년～2012년: 서울대학교

2012년～2014년: 삼성전자

2014년～현　재: 동덕여자대학교 컴퓨터학과 조교수/부교수

※관심분야 : 분산시스템(Distributed System), 운영체제(Operating System), 데이터베이스시스템(DBMS) 등