웨어러블 어플리케이션 개발을 위한 안드로이드 BLE 에뮬레이터†

(An Android BLE Emulator for Developing Wearable Apps)

문 현 아 ‡ 박 수 용 § 최 광 훈 ¶

(Hyeonah Moon) (Sooyong Park) (Kwanghoon Choi)

요 약 사물 인터넷 환경에서 모바일 어플리케이션과 웨어러블 기기를 연동하기 위해 BLE (Bluetooth Low Energy) 기반 통신을 많이 활용하고 있다. 특히 BLE 연동 안드로이드 어플리케이션을 개발할 때 개발 환경에서 BLE 에뮬레이션을 지원하지 않아 반드시 웨어러블 기기가 필요한 제약이 있다. 본 연구에서는 처음으로 안드로이드 BLE 에뮬레이터를 설계 및 구현하였다. 이를 활용하여 웨어러블 기기가 없어도 BLE 연동 어플리케이션을 개발할 수 있음을 확인하였다. 그리고 그래프 모델 기반의 안드로이드 BLE 시나리오 자동 생성 방법을 제안하고 자동 생성한 시나리오들을 제안한 안드로이드 BLE 에뮬레이터 상에서 실행하여 어플리케이션의 BLE 응용 프로토콜을 체계적으로 테스트하는데 유용함을 보였다.

키워드 : 저전력 불루투스, 웨어러블 기기, 안드로이드, 어플리케이션, 에뮬레이터

Abstract BLE (Bluetooth Low Energy) has been extensively used for communication between mobile applications and wearable devices in IoT (Internet of Things). In developing Android applications, wearable devices, on which the applications can run, should be available because the existing Android SDK does not support any BLE emulation facility. In this study, we have designed and implemented the first Android BLE emulator. Using this, we are able to develop and test BLE-based Android applications even when without wearable devices. We have also proposed an automatic generation method of Android BLE scenarios based on graph model. We have shown that the method is useful for systematically testing BLE application protocols by running the generated scenarios on the Android BLE emulator.

Key words : BLE, wearable device, Android, application, emulator

1. 서 론

사물인터넷 환경에서 다양한 기기와 모바일 어플리케이션을 연결하기 위해 BLE(Bluetooth Low Energy) 통신 방법을 많이 사용하고 있다. 예를 들어, 웨어러블 기기와 모바일 어플리케이션을 연동할 때도 주로 BLE 통신 방법을 사용한다.

웨어러블 기기를 위한 어플리케이션을 개발할 때 BLE 통신으로 이 기기와 연동해서 테스트하는 어려움이 있다. 특히 시장 선점을 위해 빨리 출시하려는 경우 웨어러블 기기와 어플리케이션을 동시에 개발하게 되므로 테스팅이 더욱 복잡해진다.

이 논문에서 다루고 있는 상용 웨어러블 기기인 맥박 측정기(HRM3200)를 위한 전용 안드로이드 어플리케이션을 실제 개발하면서 미완성 기기의 안정적이지 못한 BLE 통신 상황에서 테스트하고, 자주 통신 프로토콜이 변경되어 테스트에 큰 어려움이 있었다.

기존 개발 방법에서는 불완전한 웨어러블 하드웨어 기기로 연동 테스트하거나, 또는 BLE 테스트 키트를 이용하여 최소한의 기능만 테스트했다. 예를 들어, BLE 통신에 의존하는 사용자 인터페이스를 테스트하려면 아직 완

† 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원(No.201739028.01, 블록체인 기반 IoT 신뢰성 제어 및 관리 기술 개발)과 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No.2017R1A2B4005138)

‡ 학생회원 : 서강대학교 대학원 컴퓨터공학과
hamoon@sogang.ac.kr
§ 종신회원 : 서강대학교 컴퓨터공학과 교수
sypark@sogang.ac.kr
¶ 정 회 원 : 전남대학교 전자컴퓨터공학부 교수
kwanghoon.choi@jnu.ac.kr

논문접수 : 2017년 00월 00일
심사완료 : 2017년 00월 00일

정보과학회논문지: 정보통신 제31권 제6호(2004.12)

성되지 않은 BLE 기능을 대체하는 코드를 매번 작성해야 한다.

본 연구에서는 웨어러블 하드웨어 없이도 독립적으로 안드로이드 어플리케이션을 개발할 수 있도록 지원하는 안드로이드 BLE 에뮬레이터를 처음으로 설계하고 구현하였다. 기존의 안드로이드 어플리케이션 개발 환경에서는 BLE 에뮬레이터를 지원하지 않고 있다.

이 에뮬레이터는 두 가지 방식으로 활용할 수 있다. 첫째, 이 논문에서 개발한 안드로이드 BLE 에뮬레이터를 확장하여 웨어러블 어플리케이션에서 원하는 BLE 기반 응용 프로토콜을 쉽게 에뮬레이션 할 수 있다. 둘째, 첫 번째 활용 방법에서 단일 통신 시나리오를 에뮬레이션 하는 점을 보완하기 위하여 그래프 기반 BLE 응용 프로토콜 모델을 작성하여 다수의 통신 시나리오들을 자동 생성 하도록 지원한다. 이 시나리오들에 따라 웨어러블 어플리케이션을 안드로이드 BLE 에뮬레이터 상에서 체계적으로 테스트할 수 있다.

제안한 안드로이드 BLE 에뮬레이터의 첫번째 활용 방법을 검증하는 실험을 위하여 (그림 1)의 상용 맥박 측정기(HRM3200), BLE 테스트키트(BoT-CLE110)와 각각 연동하는 안드로이드 어플리케이션 개발에 적용하였다. 실험 결과, 이 에뮬레이터를 활용하여 하드웨어 기기들 없이도 각각의 BLE 기반 응용 프로토콜에 따라 안드로이드 어플리케이션이 오류없이 동작함을 확인하였다. 또한, 이 에뮬레이터와 연동한 안드로이드 어플리케이션을 실제 하드웨어 기기와 연동하는 어플리케이션으로 전환하기 위해서 수정해야 하는 소스 코드 라인 수가 매우 적음 (0.2%)을 확인하였다.

그림 1 HRM3200과 BoT-CLE110 테스트 키트
(Fig. 1 HRM3200 and BoT-CLE110 Test Kit)

두번째 활용 방법을 검증하는 실험을 위하여 상용 맥박 측정기(HRM3200)의 BLE 기반 응용 프로토콜의 그래프 기반 모델을 작성하였다. 이 그래프의 기본 경로에 해당하는 15개 시나리오들을 자동 생성하였다. 자동 생성한 시나리오들을 테스트한 결과, 상용 맥박 측정기(HRM3200) 어플리케이션 BLE 응용 프로토콜 소스코드의 85%를 테스트하였음을 확인하였다. 그래프 모델을 통해 다수의 시나리오들을 쉽게 자동 생성하여 효과적으로 테스트 할 수 있음을 확인하였다.

이 논문에서 제안한 안드로이드 BLE 에뮬레이터의 활용도는 매우 높을 것이다. 왜냐하면, 모바일 운영체제 시장의 점유율이 가장 높은 안드로이드에 BLE 통신 방법이 기본 탑재되어 이미 많이 사용되고 있고, 사물 인터넷과 웨어러블 기기의 통신 방법으로 많이 활용되고 있기 때문이다.

본 논문의 2장에서 관련 연구를 살펴본다. 3장에서는 안드로이드 BLE 에뮬레이터를 설계 및 구현하여, 두 가지 BLE 기반 하드웨어 기기를 에뮬레이션한 실험 결과를 제시한다. 4장에서는 3장의 BLE 에뮬레이터를 활용하여 그래프 모델 기반의 BLE 시나리오 자동 생성 방법을 제안하고, 이 방법을 활용하여 상용 맥박 측정기 응용 어플리케이션을 체계적으로 테스트한 사례를 제시한다. 마지막으로 5장에서는 결론과 향후 연구 방향에 대해 논의한다.

2. 관련 연구

BLE(Bluetooth Low Energy)는 모바일 폰과 같은 중심 기기와 하나 이상의 주변 기기들 사이의 통신을 위한 무선 통신 프로토콜이다. 기존 블루투스 프로토콜보다 단순한 구조를 갖는 블루투스 저전력 프로토콜은 전송 데이터양이 많지 않은 IoT(사물인터넷) 분야와 저전력이 필수적인 무선 웨어러블 장치에 많이 이용되고 있다[1,2,3,4].

BLE는 범용속성(GATT: Generic ATTribute) 프로파일을 통해 배터리 레벨, 맥박 수치 등의 다양한 데이터를 주고받는다. GATT프로파일 기반 BLE통신은 일반적으로 5단계로 이루어진다.

- 주변 BLE 기기 스캔
- 찾은 BLE 기기에 연결
- 연결된 BLE 기기가 제공하는 서비스 검색
- 통지(Notification) 방식으로 데이터 수신을 반복
- BLE 기기 연결 해제

이 BLE 통신 기능은 안드로이드와 같은 모바일 플랫폼에 기본 탑재되어 안드로이드 어플리케이션에서 사용할 수 있지만 이 BLE 기능을 테스트하기 위한 개발 환경이 부족하다. BLE 기반 안드로이드 어플리케이션을 개발할 때 연동할 하드웨어가 준비되어 있지 않은 경우에 BLE 기능을 전혀 테스트 할 수 없다. 반면에 문자 송수신, GPS 기능 등과 같은 통신 방법의 경우 안드로이드 어플리케이션 개발환경에서 각각의 에뮬레이터를 지원하기 때문에 특별한 하드웨어가 없어도 해당 통신 기능을 테스트 할 수 있다. 이러한 이유로 안드로이드 BLE 에뮬레이터를 개발하게 되었다.

소프트웨어 기반 에뮬레이션에 대한 다양한 연구가 특히 테스팅 분야에서 진행되었다[5,6,7,8,9]. 예를 들어, 소프트웨어 기반 에뮬레이션에 관한 대표적인 연구 결과인 CMock[6]과 Mockito[7]는 단위 테스트를 위한 모의 객

체(Mocking Object) 생성 프레임워크이다. 단위 테스트 대상 함수가 다른 함수를 호출하는 경우 호출되는 함수를 테스트 대상 함수와 분리해 주어야 한다. 테스트 대상 이외에 의존하는 함수까지 모두 테스트할 수 없으므로 의존하는 함수들을 에뮬레이션 하는 모의 객체로 대체해 주어야 한다.

C언어에서 사용 가능한 모의 객체 생성 프레임워크로 CMock 이 있다[6]. 이 도구를 활용하여 C헤더 화일을 분석해서 필요한 모의 객체들을 자동으로 생성한다. 이렇게 생성된 코드를 이용해 테스트 대상 함수를 준비하여 실행하고 그 결과 값을 예상 값과 비교하여 테스트한다.

Mockito는 Java 언어에서 사용 가능한 Junit 기반 모의 객체 생성 프레임워크이다[7]. 이 프레임워크는 테스트 대상 클래스와 협업하는 클래스가 아직 구현되지 않은 경우 유용하다. 협업 클래스의 코드에 의해 테스트 대상 클래스의 실행 경로가 달라지는 경우, 테스트할 때 협력 클래스의 특정 기능이 호출되었는지 확인이 필요한 경우, 실제 협업 클래스의 객체를 사용해 테스트하면 비용이 큰 경우 등에 활용한다.

이와 같이 테스트 분야에서 소프트웨어 기반 에뮬레이션에 대한 연구가 활발하게 이루어지고 있다. 하지만, 현재 안드로이드 개발 환경에서 BLE 에뮬레이터를 지원하고 있지 않다. 이러한 점을 개선하고자 크리스 라슨은 안드로이드 에뮬레이터에서 실행하는 안드로이드 어플리케이션에서 BLE 프로토콜을 통해 웨어러블 기기와 연동하는 방법을 제안하였다[10]. 이 방법을 사용하려면 데스크톱 컴퓨터에 BLE USB 어댑터를 미리 설치해야 한다. 이 방법을 통해 모바일 기기가 없어도 에뮬레이터 위에서 웨어러블 기기와 연동하도록 안드로이드 어플리케이션을 개발하고 및 테스트할 수 있는 장점이 있지만 웨어러블 하드웨어 기기는 반드시 존재해야 한다. 본 논문에서 제안한 안드로이드 BLE 에뮬레이터를 사용하면 웨어러블 기기가 없어도 안드로이드 어플리케이션을 실행하고 테스트 할 수 있고, 또한 USB 어댑터도 필요로 하지 않는다.

표 1 크리스 라슨 방법과의 비교
(Table 1 Comparison with Chris Larson's Work)

Hardware Dependency	Chris Larson [10]	Android BLE Emulator
Mobile Device	No	No
Wearable Device	Yes	No
USB Adapter	Yes	No

웨어러블 기기가 없는 상황에서 BLE 기반 안드로이드 어플리케이션 개발을 위한 BLE 에뮬레이터는 아직 연구 개발 되지 않았다[11]. 크리스 라슨이 제안한 방법도 본 논문의 연구 목적이 중요하다는 것을 나타내는 사례로 볼 수 있다.

3. 안드로이드 BLE 에뮬레이터

이 장에서 안드로이드 BLE에뮬레이터의 구조 및 특징을 설명하고, 제안한 에뮬레이터를 두 가지 BLE 기반 하드웨어 기기와 연동하는 안드로이드 어플리케이션에 실제 적용한 실험 결과를 논의한다.

3.1. 구조 및 특징

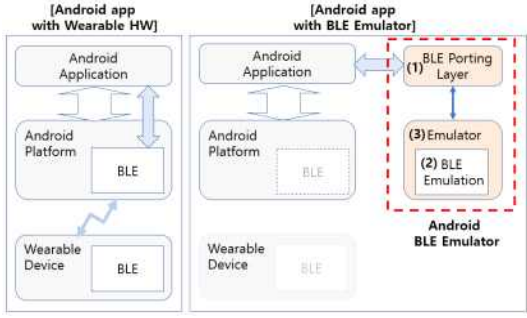

그림 2 웨어러블 기기 또는 BLE 에뮬레이터와 어플리케이션의 연동
(Fig. 2 Wearable Device vs. BLE Emulator)

이 논문에서 설계한 안드로이드 BLE 에뮬레이터의 구조는 (그림 2)와 같다.

제안한 구조는 크게 BLE 포팅 레이어, BLE 에뮬레이션 확장 모듈, 에뮬레이터로 구성된다. (그림 2)의 (1) BLE 포팅 레이어는 안드로이드 어플리케이션이 안드로이드 플랫폼의 BLE API 대신 구현한 에뮬레이션 API와 통신하기 위해 필요하다. (2) BLE 에뮬레이션 확장 모듈은 각 웨어러블 기기별로 정한 BLE 기반 응용 프로토콜을 소프트웨어로 구현한 모듈이다. (3) 에뮬레이터는 BLE 에뮬레이션 확장 모듈을 마치 하드웨어 장치인 것처럼 포팅 레이어를 통해 어플리케이션과 연결시킨다.

제안한 안드로이드 에뮬레이터는 웨어러블 기기의 BLE 응용 프로토콜에 따라 (그림 2)의 (2) BLE 에뮬레이션 모듈을 자유롭게 확장할 수 있는 구조를 가지고 있다. HRM3200 또는 BoT-CLE110 테스트 키트에서 사용하는 BLE 응용 프로토콜에 맞추어 (2)번 BLE 에뮬레이션 확장 모듈을 개발하면 각 하드웨어 기기의 에뮬레이터가 된다. 어떤 확장을 하더라도 (1)과 (3)은 변경없이 재사용 가능한 장점이 있다

(그림 3)은 BLE 에뮬레이터의 기본 동작을 정의하는 기반 클래스 BluetoothLE와 이 클래스를 상속하여 두 가지 BLE 하드웨어 기기의 응용 프로토콜을 에뮬레이션하는 HRM3200 클래스와 BoTCLE110 클래스를 보여준다. BLE 프로토콜의 기본 동작을 정의한 기반 클래스의 메서드를 각 응용 프로토콜에 맞추어 다시 정의하여 각 기

기를 위한 BLE 확장 에뮬레이터를 구현한다.

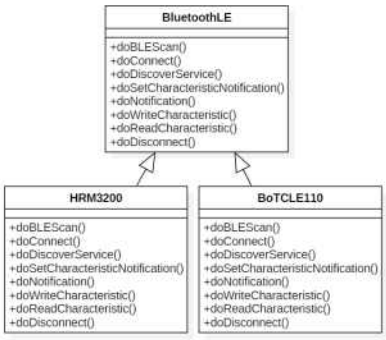

그림 3 클래스 상속으로 확장된 BLE 기기 에뮬레이션 (Fig. 3 Two Extended BLE Emulation Classes for HRM3200 and BoTCLE110 BLE Test Kit)

(그림 4)의 순차 다이어그램은 확장된 BLE 에뮬레이터의 일반적인 동작을 설명한다. (그림 2)에서 보인 것과 같이 안드로이드 어플리케이션, BLE 포팅 레이어, 에뮬레이터를 통해 BLE의 GATT 프로파일 기반의 데이터 통신을 에뮬레이션 하는데, 이 순차 다이어그램은 어플리케이션, 포팅 레이어, 에뮬레이터 세 가지 객체의 상호작용을 보여준다. 그 중 어플리케이션과 포팅 레이어가 동일한 스레드에서 동작하고, 에뮬레이터는 다른 스레드에서 실행된다. 어플리케이션에서 주변의 BLE 기기를 검색할 때 에뮬레이터 스레드를 처음 생성하고, 두 스레드는 서로 메시지를 주고받으며 비동기적 BLE 통신을 에뮬레이션한다.

이 순차 다이어그램에서 언급한 BluetoothAdapter 클래스는 BLE 기기를 찾는 기능을 제공하고, BluetoothGatt 클래스는 BLE의 GATT 프로파일을 통해 연결된 BLE 기기에 데이터를 보낼 때 사용하고, BluetoothGattCallback 클래스는 반대로 데이터를 받을 때 사용한다. 이 세가지 클래스는 안드로이드 플랫폼에서 정의한 클래스와 동일한 클래스 이름과 메서드로 구성한다. 마지막으로, BluetoothLE 클래스는 BLE 에뮬레이터를 구성하는 클래스로 스레드 간 통신을 제공한다. BLE 확장 에뮬레이터에서 상세한 통신 구현 방법에 의존하지 않도록 추상화하여 이 클래스를 구현하였다.

(그림 4)의 순차 다이어그램에서 BLE의 GATT 프로파일 기반의 데이터 통신을 에뮬레이션하는 과정은 다음과 같다.

- 주변 BLE 기기 스캔: 1~5번 단계
- 찾은 BLE 기기에 연결: 6~10번 단계
- 연결된 BLE기기가 제공하는 서비스 검색: 11~15번 단계
- 통지(Notification)방식으로 데이터 수신을 반복: 16~17번, 18~23번 단계
- BLE 기기 연결 해제: 24~28번 단계

예를 들어, 주변 BLE 기기를 스캔할 때 BluetoothAdapter 클래스의 startLeScan 메소드를 호출하면(1번) BLE 포팅 레이어에서 에뮬레이터의 BluetoothLE의 doBLEScan 메소드가 호출되어(2번) 에뮬레이션 대상 BLE 기기의 이름과 주소를(3번) 포팅 레이어를 거쳐(4번) BluetoothAdapter.LeScanCallBack 인터페이스의 onLeScan 을 통해 어플리케이션에 전달한다(5번). 찾은 BLE 기기에 연결, 서비스 검색, 기기 연결 해제 과정도 이와 유사하게 진행한다.

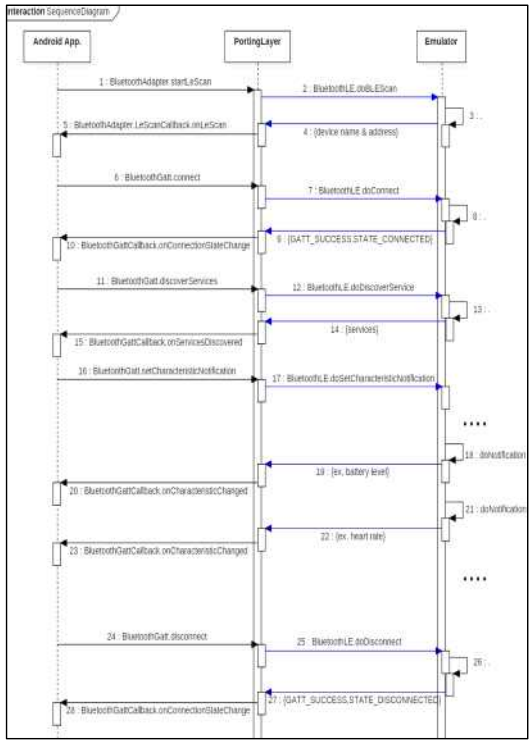

그림 4 안드로이드 BLE 에뮬레이터의 순차다이어그램 (Fig. 4 Sequence Diagram of Android BLE Emulator)

통지(Notification) 방식으로 BLE 기기로부터 데이터를 수신하기 위해서 먼저 포팅 레이어의 BluetoothGatt 클래스의 setCharacteristicNotification 메소드를 호출하면(16번) 에뮬레이터의 BluetoothLE 클래스의 doSetCharacterisitcNotification 메소드가 호출되어 BLE 기기에서 데이터가 변경될 때 마다 Notification하도록 설정한다(17번). BLE 기기 에뮬레이터는 데이터가 변경되면 BluetoothLE 클래스의 doNotification 메소드를 호출하고(18번) 포팅 레이어를 거쳐(19번) 어플리케이션의 BluetoothGattCallback 클래스의 onCharacteristicChanged 메소드를 호출하여 변경된 데

이터를 전달한다(20번). 이 과정을 데이터가 변경될 때마다 반복한다.

아래의 웹 주소에 이 논문에서 구현한 BLE 에뮬레이터 전체 소스 코드가 있다. 이 소스 코드를 통해 이 순차 다이어그램에 대한 보다 상세한 구현 내용을 확인할 수 있을 것이다.

http://github.com/■■■/■■■/

제안한 방법은 다음과 같은 장점이 있다. 첫째, BLE 연동 안드로이드 어플리케이션을 상대 하드웨어 기기가 없어도 안드로이드 기기 에뮬레이터에서 개발할 수 있다. 일반적으로 안드로이드 어플리케이션을 개발할 때에는 개발 툴(Android SDK)에 포함된 안드로이드 기기 에뮬레이터를 활용한다. 하지만 BLE 연동 기능이 포함된 경우 이 안드로이드 기기 에뮬레이터를 사용할 수 없었다.

둘째, BLE 통신 웨어러블 하드웨어 개발을 안드로이드 어플리케이션 소프트웨어 개발과 동시에 병행할 수 있다. 이러한 환경에서 제안한 BLE 에뮬레이터를 사용해 완성되지 않은 하드웨어에서 제공해야 하는 응용 프로토콜을 에뮬레이션 하여 어플리케이션 개발을 동시에 진행할 수 있다.

셋째, 제안한 BLE 에뮬레이터를 사용하여 개발한 안드로이드 어플리케이션은 향후 실제 기기와 연동할 때 어플리케이션의 소스 코드를 수정해야 하는데 이때 수정해야하는 코드의 양은 전체 소스 코드와 비교해 매우 적으므로 최종 완성을 위한 시간을 절약할 수 있다. 다음 절에서 관련된 실험 결과를 제시한다.

3.2. 적용 예

제안한 안드로이드 BLE 에뮬레이터를 (표 2)의 BLE 통신을 사용하는 두 가지 하드웨어 기기에 적용한 실험 결과를 설명한다.

표 2 BLE 기반 기기
(Table 2 BLE based Devices)

HRM3200	- H3 System Co., Ltd. - Heart rate monitor - ARM-band type wearable - Measures heart rate at upper forearm during exercise and sends wirelessly - Communication devices : Bluetooth 4.0 (Bluetooth smart)
BoT-CLE110	- Chipsen Inc. - Sensors (Temperature, etc.) - BLE based Test kit - Ultra Small 4.1 BLE Module - Bluetooth v4.0 or v4.1 (BLE), Class1.5(+8dBm) - AT command

두 가지 BLE 기기의 응용 프로토콜에 따라 (그림 2)의 (2)번 BLE 에뮬레이션 모듈을 각각 작성하였다. HRM3200의 경우, 실시간으로 맥박을 측정하는 시나리오와 측정기에 저장된 맥박 데이터를 모바일로 다운로드하는 시나리오를 지원하는 통신 규약을 구현하였다. BoT-CLE110의 경우 테스트 키트에 부착된 센서로부터 온도, 스위치 눌림 상태 등을 읽는 시나리오를 지원하는 통신 규약을 구현하였다.

비록 두 가지 하드웨어 기기의 BLE 응용 통신 프로토콜을 에뮬레이션 하는 실험이었지만, 제안한 BLE 에뮬레이터를 확장하여 임의의 하드웨어 기기의 BLE 응용 프로토콜을 구현할 수 있는 가능성을 확인하였다. BLE 에뮬레이터에서 기본적으로 지원하는 기반 BLE 통신 프로토콜을 각 응용 프로토콜에 맞추어 BLE 에뮬레이션 모듈로 확장 구현하면 원하는 하드웨어 기기의 BLE 기능을 에뮬레이션 할 수 있다.

안드로이드 BLE 에뮬레이터와 연동해 안드로이드 어플리케이션을 개발한 후, 실제 하드웨어와 연동하도록 하기 위해서는 어플리케이션의 매우 적은 소스 코드 만을 수정하면 된다. (표 3)은 두 BLE 기기에 대한 관련 실험 결과를 보여준다. BLE 에뮬레이터를 기반으로 개발한 어플리케이션 코드 대비 약 0.2%만 수정해서 실제 하드웨어 기기용 어플리케이션으로 전환할 수 있었다. 이는 전체 소스 코드 대비 매우 낮은 비중이다.

표 3 안드로이드 BLE 에뮬레이터 사용하는 경우와 하드웨어를 사용하는 경우 소스 코드 수정이 필요한 라인 수에 대한 실험 결과
(Table 3 The Number of Modified Lines of Application on Emulator for Application on Hardware)

BLE Devices	LOCs of Applications	# of Modified Lines		
		Total	Java Stmts	Import
HRM3200	7503	14	2	12
BoT-CLE110	5982	16	2	14

(표 3)은 BLE 에뮬레이터를 활용하여 개발한 안드로이드 어플리케이션의 전체 라인 수와 이 어플리케이션을 실제 하드웨어와 연동하도록 전환할 때 수정해야하는 라인 수를 보여준다. 이 표에서 보여주는 바와 같이 에뮬레이터로 개발한 어플리케이션을 실제 하드웨어용으로 변경하기 위해 단 두 개의 Java 문장과 12~14개의 import문만 수정하면 되었다. 두 개의 Java 문장을 수정한 이유는 제안한 에뮬레이터의 BLE 서비스 대신 안드로이드 플랫폼에서 제공하는 BLE 서비스를 사용하도록 전환이 필요하기 때문이다. import 문을 수정한 것은 어플리케이션이 사용하는 안드로이드 플랫폼 BLE API를 (그림 2)의 (1)번 BLE 포팅 레이어에서 지원하는 해당 API로 대체하기

위해 필요하였다. 이러한 수정은 불가피하지만 전체 소스 라인 수 대비 최소 수준으로 변경해도 가능함을 확인하였다.

BLE 에뮬레이터를 사용하면 하드웨어 없이 안드로이드 어플리케이션을 개발할 수 있다는 장점 이외에도 안드로이드 어플리케이션의 BLE 응용 프로토콜 기능을 테스트하는 목적으로도 활용할 수 있을 것이다. 첫째, 복잡한 BLE 응용 프로토콜을 고려할 때 에뮬레이터를 활용하면 더 쉽게 반복적으로 테스트할 수 있다. 복잡한 시나리오도 제안한 BLE 에뮬레이터를 확장하여 구현하면 이 시나리오를 테스트하는 환경을 구축할 수 있을 것으로 기대한다. 둘째, 웨어러블 기기와 안드로이드 어플리케이션의 연동에 대한 스트레스 테스트를 할 때 사람이 직접 착용하고 테스트하는 대신 소프트웨어적으로 쉽게 대체할 수 있다. 상대적으로 테스트 비용도 낮고, 다양한 스트레스 테스트를 진행 할 수 있을 것이다.

4. 모델 기반 안드로이드 BLE 시나리오 자동 생성 방법

앞 장에서 안드로이드 BLE 에뮬레이터 클래스를 작성하여 어플리케이션의 BLE 기반 응용 프로토콜의 시나리오를 테스트할 수 있음을 확인하였는데, 이 클래스는 일반적으로 단일 시나리오만을 에뮬레이션 한다. 따라서 어플리케이션에서 사용하는 모든 BLE 시나리오를 테스트하려면 각 시나리오 별로 하나씩 클래스를 작성해야 한다. 만일 하나의 클래스로 여러 시나리오들에 대응하도록 만들려면 공통적인 부분과 다른 부분을 일일이 조합해야 하기 때문에 매우 복잡해진다.

이러한 제한점을 보완하기 위해 안드로이드 어플리케이션의 BLE 응용 프로토콜에서 정의한 많은 시나리오들을 단 하나의 클래스로 작성하면서도 자동으로 많은 시나리오들을 조합해서 만드는 모델 기반 시나리오 자동 생성 방법을 제안한다.

4.1 모델 기반 BLE 시나리오 자동 생성 방법

기본적인 아이디어는 안드로이드 어플리케이션의 BLE 기반 응용 프로토콜을 그래프로 모델링하고, 이 그래프의 각 기본 경로(basic path)를 하나의 시나리오로 정의하며, 그래프를 탐색하여 모든 기본 경로를 빠짐없이 커버하는 시나리오를 생성하는 것이다.

첫째, 그래프 기반 모델은 어플리케이션의 BLE 기반 응용 프로토콜의 주요 상태를 노드로 하고 어플리케이션 또는 기기에 의하여 현재 상태에서 다음 상태로 변경되는 경우 마다 두 상태들 간에 에지를 연결하여 만든다. 이 응용 프로토콜의 시작하는 상태를 시작 노드로, 정상적으로 또는 비정상적으로 종료하는 상태들을 종료 노드들로 분류한다. 각 노드에 수행할 액션을 속성으로 붙인다.

둘째, 시작 노드에서 출발하여 종료 노드까지 동일한 에지를 중복되지 않도록 따라가는 경로를 기본 경로로 정의하고, 기본 경로를 따라가면서 거치는 각 노드의 BLE 프로토콜 액션을 수행하는 것을 이 기본 경로에 대한 시나리오라고 정의한다.

셋째, 모델로 정의한 그래프를 탐색하여 모든 기본 경로들을 생성하고 어플리케이션의 BLE 응용 프로토콜의 모델 기반 기본 경로 커버리지(basic path coverage)를 충족하는 시나리오 집합을 자동 생성한다.

먼저 안드로이드 어플리케이션의 BLE 기반 응용 프로토콜 모델을 그래프로 표현하는 방법에 대해 설명한다. 그래프의 각 노드는 이 응용 프로토콜의 단일 상태를 표현하고, 각 에지는 상태 간 변이를 표현한다. BLE 고유의 프로토콜의 상태를 표현하는 클래스들을 BLE 에뮬레이터에서 (그림 5)의 기반 클래스들로 제공하고, 이 클래스들을 상속받아 응용 프로토콜의 상태를 표현하는 클래스들을 만든다.

BLE 고유의 프로토콜 상태를 표현하는 클래스는 모두 기반 클래스 BLE State를 상속받도록 설계하였다. BLE 프로토콜의 상태를 표현하는 클래스로 BLE Scan State, BLE Connect State에서 BLE Disconnect State까지 BLE 에뮬레이터의 라이브러리로 제공한다. 각 클래스는 자신이 표현하는 상태에 도달했을 때 해야 할 일을 액션으로 정의한다.

BLE Scan State 클래스에는 연결 가능한 BLE 기기의 주소와 이름 목록을 안드로이드 어플리케이션에 보내는 역할을 하도록 액션을 정의하였다. 예를 들면, HRM3200 기기 모델의 경우 이 클래스를 사용하여 블루투스 주소 ("00:11:22:AA:BB:CC")와 자신의 이름 ("HRM3200")을 어플리케이션에 보내는 상태 노드를 만들었다.

BLE Connect State 클래스를 활용하여 안드로이드 어플리케이션이 검색된 BLE 기기에 연결을 시도할 때 성공하는 상태 노드와 연결에 실패하는 상태 노드를 만든다. BLE Disconnect State 클래스도 연결 해제에 관하여 비슷하게 수행하도록 정의하였다.

BLE Service Discover State 클래스는 BLE 기기에 연결을 성공한 후 연결된 기기에서 제공하는 BLE 서비스 목록을 어플리케이션에 전달하는 역할을 담당한다.

BLE Write Characteristic State클래스는 어플리케이션에서 기기로 패킷을 보낼 때 활용한다. 이 클래스의 액션에서 어플리케이션이 제대로 패킷을 보냈는지 확인한다. 예를 들어, (그림 5)와 같이 BLE Write Characteristic State 클래스를 상속받아 App Time State 클래스를 만들었다. 이 클래스의 액션은 어플리케이션에서 HRM3200 기기에 실시간 맥박 정보를 요청했는지 확인한다. 다른 예로, Req Download State 클래스도 BLE Write Characteristic State 클래스를 상속받아 만든다. 이 클래스는 어플리케이션이 기기에 저장된 맥박 정보를 보내 달라고 요청하는 상태를 표현하고, 클래스

액션은 이러한 점을 확인한다.

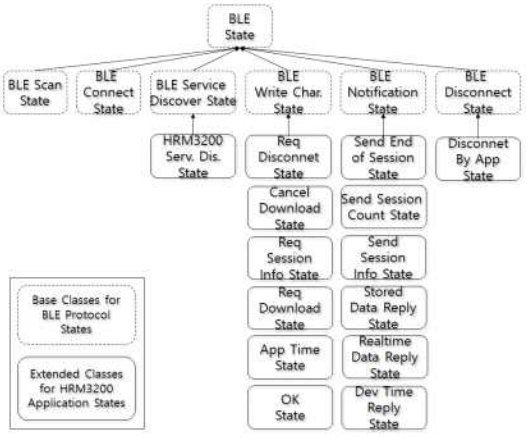

그림 5 BLE 프로토콜 기반 클래스들과 이를 상속한 HRM3200의 상태 클래스들
(Fig. 5 Base Classes for BLE Protocol States and Their Extended Classes for HRM3200)

BLE Notification State 클래스는 반대 방향 즉, 기기에서 어플리케이션으로 패킷을 보낼 때 사용한다. 이 클래스의 액션에서는 기기에서 어플리케이션으로 정상적인 또는 비정상적인 패킷을 보낸다. 기기가 보낸 정상 또는 비정상 패킷에 대해 어플리케이션이 어떻게 동작하는지 확인하는데 활용할 수 있다.

BLE Notification State 클래스를 상속받아 정의한 Real-time Data Reply State 클래스는 HRM3200 기기에서 측정한 맥박 정보를 한 번 어플리케이션에 보내주는 액션을 구현하였다. Stored Data Reply State 클래스는 HRM3200 기기에 (어플리케이션과 연결되지 않은 동안 측정해 저장해 놓은) 저장된 맥박 정보를 한 번 어플리케이션에 보내주도록 액션을 구현하였다.

이 확장된 클래스들을 활용하여 HRM3200 응용 프로토콜의 그래프 모델을 (그림 7)과 같이 구성한다. 이 그래프 모델에 대해서는 다음 절에서 구체적으로 설명한다.

HRM3200 응용 프로토콜의 그래프 모델이 주어지면, 깊이 우선 탐색(Depth-first search) 방법으로 이 그래프의 시작 상태(BLE Scan State 클래스)에서 종료 상태(BLE Disconnect State 클래스)까지의 모든 기본 경로들을 자동으로 찾는다. 각 기본 경로를 구성하는 노드들의 액션을 에뮬레이터 상에서 차례로 수행하면 이 기본 경로로 표현한 시나리오를 에뮬레이션 할 수 있다. 궁극적으로 HRM3200 응용 프로토콜 모델에서 찾은 모든 기본 경로들에 해당하는 BLE 시나리오들을 에뮬레이션 할 수 있다.

이러한 기본 경로 커버리지를 만족하는 에뮬레이션을 통해 해당하는 BLE 시나리오가 어플리케이션에서 모두 오류없이 동작하는지 체계적으로 테스트할 수 있다. 다음 절에서 HRM3200 어플리케이션의 모델에서 생성한 모든 시나리오들에 대한 화이트박스 테스트 결과 및 소스 코드 커버리지 결과를 제시하여 이를 확인한다.

본 연구에서는 BLE 에뮬레이터와 함께 BLE 프로토콜 상태 클래스와 그래프 모델을 만들고 기본 경로를 찾는 라이브러리를 설계 및 구현하였다. 이 라이브러리는 재사용 가능하다. 즉, 새로운 BLE 응용 프로토콜의 모델을 일단 만들면 모든 기본 경로들을 찾아 해당 시나리오를 에뮬레이션 할 수 있다.

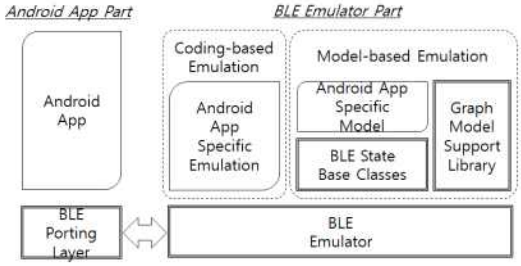

그림 6 BLE 에뮬레이터의 구조
(Fig. 6 Structure of BLE Emulator)

앞 장에서 BLE 에뮬레이터를 확장한 클래스를 직접 작성하여 특정 안드로이드 어플리케이션의 BLE 프로토콜의 특정 단일 시나리오를 에뮬레이션 하는 방법을 설명하였고, 이 장에서 특정 BLE 프로토콜을 그래프 모델로 구현하여 다양한 시나리오들을 자동 생성하여 이를 에뮬레이션 하는 방법을 설명하였다. 두 가지 모두 이 논문에서 제안한 BLE 에뮬레이터를 활용하는 방법이다. (그림 6)에서는 두 가지 방법을 모두 포함한 BLE 에뮬레이터의 구조를 보여준다. 이 그림에서 이중 박스로 표시한 것은 특정 프로토콜과 독립적인 재사용 가능한 부분이고, 나머지 부분은 안드로이드 어플리케이션의 BLE 응용 프로토콜에 의존적인 모듈이다.

4.2 Case Study: HRM3200 어플리케이션의 BLE 응용 프로토콜 테스트 커버리지 측정

앞 절에서 소개한 HRM3200 BLE 응용 프로토콜 그래프 모델에서 자동 생성한 기본 경로 커버리지 시나리오에 따라 이 어플리케이션을 체계적으로 테스트한 실험 결과를 제시하고, 제안한 모델 기반 시나리오 생성 방법이 테스트 방법으로 유용함을 확인한다.

4.2.1 HRM3200 BLE 응용 프로토콜을 위한 모델 및 시나리오 자동 생성

실험을 위해 만든 HRM3200 어플리케이션의 응용 프로토콜을 위한 그래프 모델은 47개의 노드와 58개의 에지로 구성되어 있고 주석과 공백 라인을 제외한 200여

라인으로 구성된 Java 메서드로 작성하였다. (그림 7)에서 이 그래프 모델을 보여준다. 일부 반복적인 서브 그래프들을 생략하였다.

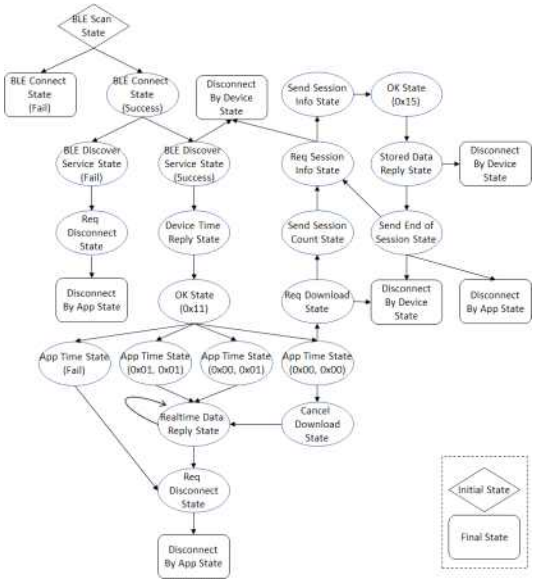

그림 7 HRM3200의 상태다이어그램
(Fig. 7 State Diagram of HRM3200 Protocol)

이 모델에서 구현한 것은 (그림 7)에서 보인 것처럼 HRM3200 응용 프로토콜의 세 가지 기본 시나리오를 포함한다. 어플리케이션이 HRM3200 기기와 연결해서 실시간으로 맥박 정보를 받아오는 시나리오, 이 기기에 저장된 맥박 정보를 다운 받아 저장하는 시나리오, 기기에 저장된 맥박을 다운 받지 않고 실시간 맥박 정보를 받아보는 모드로 전환하는 시나리오이다.

이 외에도 전형적인 비정상 시나리오를 포함한다. BLE 기기 연결에 실패하는 시나리오, 연결되었지만 기기의 BLE 서비스 목록을 가져오지 못하는 시나리오, 기기에 저장된 맥박 정보를 다운 받는 도중 배터리 부족 등의 이유로 기기가 강제 종료되는 시나리오를 모델에 포함시켰다.

실험 결과, HRM3200 응용 프로토콜을 구현한 그래프 모델에서 그래프의 기본 경로 커버리지를 만족하는 15가지 시나리오를 자동 생성하였다.

4.2.2 HRM3200 어플리케이션 테스팅 결과 및 오류 분석

모델로부터 자동 생성한 15가지의 시나리오에 따라 HRM3200 어플리케이션을 각각 테스트 실행한 결과 12개의 시나리오의 경우는 정상 실행 후 종료했고, 나머지 3가지 시나리오에서 어플리케이션이 비정상 종료하였다.

비정상 종료를 분석한 결과는 다음과 같다. 먼저, 기기에 연결된 다음, 기기가 제공하는 서비스 정보를 받지 못하는 경우에 대한 처리가 어플리케이션에서 빠져 있음을 발견하였다. 두번째로, 기기 연결이 이루어지고 시간 정보를 서로 주고받은 다음 프로토콜에 정의되지 않은 데이터를 어플리케이션이 받아 처리하지 못하고 비정상 종료한다. 어플리케이션과 기기가 동시에 개발되는 상황에서 계속 수정되는 프로토콜을 제대로 반영하지 못하면서 이러한 오류가 발생하였다. 마지막으로 기기에 저장된 데이터를 다운로드 받는 중에 통신 두절 등 여러 이유로 기기 연결이 해제되면 안정된 데이터를 전송하지 못하는데 이런 상황을 반영하여 구현하지 않아 어플리케이션이 비정상 종료되었다.

15가지 시나리오에 대해 안드로이드 어플리케이션을 테스트하고 분석한 결과, HRM3200 기기와 연동되는 기본적인 정상 시나리오에 대해서는 어플리케이션이 오류없이 동작하였지만 이 시나리오를 벗어나는 프로토콜에 대해서는 빠짐없이 처리하지 못해서 발생하는 오류가 있음을 확인하였다. 자동 생성된 시나리오를 사용함으로써 기본적인 시나리오 이외의 다양한 시나리오를 테스트 할 수 있었고, 그 결과 이러한 문제점을 쉽게 파악할 수 있었다.

4.2.3 HRM3200 어플리케이션의 커버리지 분석

HRM3200 어플리케이션 테스트 실험을 통해 그래프 모델의 기본 경로 커버리지를 만족하는 시나리오들로 이 어플리케이션의 BLE 응용 프로토콜 소스 코드를 높은 커버리지 비율로 테스트할 수 있음을 확인하였다.

HRM3200 그래프 모델의 기본 경로 커버리지를 만족하는 15가지 시나리오에 따라 어플리케이션을 실행시켰을 때 어플리케이션의 BLE 프로토콜을 구현하는 소스프로그램의 커버리지가 85%임을 확인하였다. JaCoCo 자바 커버리지 라이브러리를 활용하여 각 시나리오를 실행하면서 커버리지 결과를 누적하여 분석하였다. HRM3200 BLE 응용 프로토콜을 구현한 BluetoothLeService.java 파일을 분석하였다.

커버하지 못한 15% 부분에 대한 분석 결과는 다음과 같다. 주요 원인으로, HRM3200 응용 프로토콜 모델을 더 상세하게 만들면 커버할 수 있는 경우가 58%이다. 예를 들어 현재 모델은 정상적인 길이의 패킷만 다루었는데 비정상적인 길이의 패킷도 다루도록 모델을 상세화하면 커버될 영역이 많다. 기타 다양한 이유로 불필요한 코드로 남아있는 경우가 42%이다. HRM3200 응용 프로토콜이 결정되기 전에 시험용으로 작성했던 코드나 어플리케이션 아키텍처 설계 변경으로 삭제되어야 할 코드였다.

5. 결론 및 향후 연구

본 논문에서 안드로이드 BLE 에뮬레이터를 처음으로 설계 및 개발하였고, 두 가지 BLE기기 용 안드로이드 어

플리케이션 사례를 통해 실제 하드웨어가 없이 개발할 수 있는 장점을 확인하였다. 이후에 하드웨어 기기와 연동하더라도 에뮬레이터로 개발한 어플리케이션의 소스 코드를 거의 수정하지 않아도 되고, BLE 기반 서비스가 복잡하더라도 BLE 에뮬레이터를 확장하여 쉽게 구현할 수 있는 장점이 있다. 그리고, 안드로이드 BLE 에뮬레이터 상에서 어플리케이션의 BLE 응용 프로토콜의 다양한 시나리오들을 체계적으로 테스트하는데 유용한 그래프 모델 기반 시나리오 자동 생성 방법을 제안하였다.

모바일과 사물인터넷에서 BLE 통신을 이용하는 사례가 많아서 이 연구 결과의 활용도가 높을 것으로 기대한다. 향후 BLE 기기의 통신 프로토콜 명세로부터 그래프 모델을 자동 생성 하는 연구와 퍼지 테스트(Fuzz Testing)를 기반으로 BLE 응용 프로토콜의 강건성을 테스트하는 연구를 수행하려 한다.

참 고 문 헌

[1] Bluetooth Low Energy Specification AdoptedDocuments [Online]. Available: https://www.bluetooth.org/en-us/specification/adopted-specifications/ (downloaded 2012, Sep. 10)

[2] McGrath, Will; Etemadi, Mozziyar; Roy, Shuvo; Hartmann, Bjoern, Fabryq: Using Phones as Gateways to Prototype Internet of Things Applications Using Web Scripting, EICS '15 Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 164-173, 2015.

[3] Ju-Hyeong Lee, Moon-Seog Han, "Signal Sensing System Design for Pedestrian Safety using Beacon Service", KIISE Transactions on Computing Practices, Vol. 22, No. 11, pp. 576-582, Nov. 2016. (in Korean)

[4] Juwon Lee, Woosaeng Kim, "Location-based Mobile IoT Control System by utilizing the BLE Technique", Proc. of the KSCI Winter Conferences, Vol. 23, No. 1, pp. 231-233, Jan. 2015. (in Korean).

[5] Hyungkeun Song, Junho Yoon, Jonghyun Lee, "A Test Practice of Web Server extended Module by mocking external systems", Proc. of the KIISE Fall Conferences, Vol.38, No.2(B), pp. 128-131, Nov. 2011. (in Korean)

[6] CMock Mock/stub generator for C [Online]. Available: https://github.com/ThrowTheSwitch/CMock (downloaded 2017, Sep. 6)

[7] Mockito Tasty mocking framework for unit tests in Java [Online]. Available: http://site.mockito.org/ (downloaded 2017, Sep. 10)

[8] Freeman, Steve and Mackinnon, Tim and Pryce, Nat and Walnes, Joe, "jMock: Supporting Responsibility-based Design with Mock Objects", Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications, pp. 4-5, 2004

[9] Xiaoyin Wang, "An Empirical Study on the Usage of Mocking Frameworks in Software Testing Shaikh Mostafa", 14th International Conference on Quality Software, pp. 127-132, 2014

[10] http://chrislarson.me/blog/emulate-android-and-bluetooth-le-hardware.html

[11] http://stackoverflow.com/questions/38436370/hermatic-testing-for-ble-based-android-application?rq=1

문 현 아

1994년 서강대학교 컴퓨터공학과 학사

1996년 서강대학교 컴퓨터공학과 석사

2016년~현재 서강대학교 컴퓨터공학과 박사과정

관심분야는 소프트웨어개발 교육, 소프트웨어 안전성, 블록체인

박 수 용

1986년 서강대학교 컴퓨터공학과 학사

1988년 플로리다 주립대학교 컴퓨터 및 정보과학 석사

1995년 조지메이슨 대학교 소프트웨어 공학 박사

1998년~현재 서강대학교 컴퓨터공학과 교수

2012년~2014년 정보통신산업진흥원 원장

관심분야는 요구공학, 동적 아키텍처, 블록체인, 핀테크

최 광 훈

1994년 KAIST 전산학과 학사

1996년 KAIST 전산학과 석사

2003년 KAIST 전산학과 박사

2006년~2010년 LG전자 책임연구원

2011년~2016년 연세대학교 원주 컴퓨터정보통신공학부 조교수

2016년~현재 전남대학교 전자컴퓨터공학부 부교수

관심분야는 프로그래밍언어, 컴파일러, 프로그램 분석, 소프트웨어공학