JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 23, No. 10: 1216~1227, Oct. 2019

IoT 애플리케이션의 효율적인 테스팅을 위한 개발자 지원 도구

이의혁[1] · 김동우[1] · 강승우[2*]

A Developer Support Tool for Efficient Testing of IoT Applications

Euihyeok Lee[1] · Dongwoo Kim[1] · Seungwoo Kang[2*]

[1]Graduate Student, Department of Computer Engineering, Graduate School, KOREATECH, Cheonan, 31253 Korea

[2*]Assistant Professor, School of Computer Science and Engineering, KOREATECH, Cheonan, 31253 Korea

요 약

본 논문에서는 IoT 서비스 개발 과정에서 효율적인 IoT 애플리케이션 테스팅이 가능하도록 지원하는 도구인 TITAN(Tool for IoT ApplicatioN testing)을 제안한다. TITAN은 IoT 서비스 개발자가 개발 중인 애플리케이션 로직을 테스팅 하는 데 필요한 물리적 환경 및 사용자 행동에 제약받지 않고, 개발 환경에서 애플리케이션 실행 및 확인이 가능하도록 고안되었다. 개발자는 TITAN을 이용하여 개발 과정에서 반복적인 테스팅에 소모되는 시간과 노력을 줄일 수 있을 것이다. 본 논문에서는 TITAN의 시스템 아키텍처와 현재까지 구현된 프로토타입을 제시한다. 또한 소규모 유저 스터디를 통해 TITAN의 유용성과 사용성을 평가한다. 유저 스터디 참가자들은 TITAN의 유용성에 대해 긍정적으로 생각하였다. 마지막으로 현재 연구의 한계와 향후 연구 방향에 대해서 논의한다.

ABSTRACT

We propose TITAN (Tool for IoT ApplicatioN testing), a developer support tool that enables efficient testing of IoT applications. TITAN is designed to allow IoT application developers to run their applications under the development environment without being restricted by physical environments and users' behaviors required to test application logic being developed. Using TITAN, IoT application developers can save the time and effort needed to repeatedly perform the testing of application logic while they develop their applications. In this paper, we present the system architecture of TITAN and its current prototype implementation. We evaluate the usefulness and usability of TITAN through a small user study with two example IoT applications. The study participants show their positive perception about the usefulness of TITAN. We further discuss the limitations of the current study and future research directions.

키워드 : 사물인터넷, 개발자 도구, 테스팅, IoT 애플리케이션

Keywords : IoT, developer tool, testing, IoT application

Received 27 June 2019, **Revised** 26 July 2019, **Accepted** 13 August 2019

*** Corresponding Author** Seungwoo Kang(E-mail:swkang@koreatech.ac.kr, Tel:+82-41-560-1406)
Assistant Professor, School of Computer Science and Engineering, KOREATECH, Cheonan, 31253 Korea

Open Access **http://doi.org/10.6109/jkiice.2019.23.10.1216** print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

IoT 애플리케이션의 핵심은 각종 센서로 주변 환경 및 사용자의 상태, 행동 등을 인식하고 그에 따라 알맞게 동작하여 액추에이터를 구동하거나 사용자에게 필요한 정보 및 피드백을 주는 것이다. 예를 들면, 전력 효율을 위해 집이나 사무실에서 사람 유무나 사람의 행동 등에 따라 조명이나 냉난방을 제어하는 스마트 홈/오피스 [1-2], 천식 환자의 주변 상황(예, 미세먼지 농도)과 호흡기의 사용을 분석하여 천식 증상을 미리 예방하거나 적절한 시점에 사용을 유도하는 스마트 호흡기 [3] 등 다양한 애플리케이션이 있다. 이렇게 사람의 행동 혹은 주변의 환경을 인지하여 동작하는 IoT 애플리케이션의 특성상 이런 애플리케이션이 사용자의 실생활에서 구동되는 상황은 매우 다양할 수 있다. 따라서 개발자는 자신이 구현한 애플리케이션 로직이 그 동작이나 실행 결과가 달라질 수 있는 다양한 사용자 상황에서도 오류 없이 잘 동작하는지 확인하기 위해서 반복적인 테스트를 수행해야 한다.

하지만 IoT 애플리케이션의 특성상 다양한 실생활의 사용 상황에서 반복적인 테스팅을 한다는 것은 매우 어려운 일이다. 첫째, IoT 애플리케이션이 사용하는 센서 디바이스 자체의 문제(예를 들면, 센서 하드웨어의 오동작, 통신 불능 등)가 발생한 경우 개발자는 그 문제를 해결하느라 애플리케이션 로직을 테스팅 하는 것이 지연되거나 그것에 시간을 많이 쓰지 못 하는 상황에 빠질 수 있다. 둘째, 미세먼지, 온습도 등의 사용자 주변 환경 데이터를 이용하는 애플리케이션의 경우 다양한 조건 하에서 테스팅을 해보려면 직접 해당 조건을 만족하는 환경에서 실행을 해봐야 한다. 그러나 그런 조건에 도달하는 데 시간이 오래 걸리는 경우, 혹은 개발자가 주변 환경을 쉽게 바꾸기 어려운 경우 테스팅이 제한적일 수 밖에 없다. 셋째, 사용자가 책상 앞에 앉으면 조명이 더 밝아지고, 오피스 실내 인원수에 따라 환기 장치 작동이 달라지는 등 사용자의 물리적인 행동에 따라 동작이 달라지는 애플리케이션의 경우 로직이 제대로 실행되는지 확인하기 위해서는 테스팅 과정에서 직접 그런 행동을 수행해야만 한다. 이것은 그만큼 개발자의 시간과 노력을 요하는 일이다. 넷째, 스마트 홈이나 스마트 오피스처럼 여러 사람이 공유하는 공간에서 사용되는 애플리케이션의 경우는 여러 사람의 상태와 행동이 애플리케이션의 결과에 영향을 줄 수 있다. 테스팅을 위해서 여러 사용자가 존재하는 환경을 만드는 것 또한 개발자에게 힘든 일이다.

본 논문에서는 IoT 서비스 개발 과정에서 효율적인 IoT 애플리케이션 테스팅이 가능하도록 지원하는 도구인 TITAN(Tool for IoT ApplicatioN testing)을 제안한다. TITAN은 애플리케이션이 실행될 수 있는 다양한 사용자의 실생활 상황에 대응하여 다양한 시나리오에 대해 개발 환경에서 테스팅 할 수 있도록 지원한다. 따라서 개발 중인 애플리케이션 로직을 테스팅 하는 데 필요한 물리적 환경 및 사용자 행동에 제약받지 않고 애플리케이션의 실행 및 결과 확인이 가능하다. 개발자는 TITAN을 이용하여 개발 과정에서 필요한 반복적인 테스팅에 소모되는 노력과 시간을 줄일 수 있다.

TITAN의 핵심 아이디어는 가상의 IoT 환경을 제공하여 IoT 애플리케이션이 실행될 실제 사용자 상황과 유사한 다양한 환경 하에서 개발 중인 애플리케이션 로직을 실행하고 테스트할 수 있도록 하는 것이다. 본 논문에서는 TITAN이 가상의 IoT 환경을 제공할 수 있게 만들기 위하여 IoT 애플리케이션의 다음과 같은 특징을 고려하였다. 첫째, 많은 IoT 애플리케이션이 IoT 플랫폼을 기반으로 개발된다 [4-5]. 애플리케이션 구동에 필요한 센서, 액추에이터 디바이스를 IoT 플랫폼에 등록하고, 이를 통해 데이터를 수집하여 처리하고 디바이스를 제어하는 방식이다. 따라서 이러한 구현 및 실행 방식이 TITAN에서도 그대로 지원되어야 한다. 둘째, IoT 애플리케이션이 실행될 사용자 환경은 애플리케이션에 입력되는 센서 데이터에 의해서 정해진다. 다양한 상황에서 애플리케이션 로직을 테스트 해보기 위해서는 그에 대응하는 각종 데이터를 제공할 수 있어야 한다.

TITAN에서 가상 IoT 환경을 제공하는 것은 개발자가 애플리케이션 구현을 위해 필요한 IoT 디바이스를 가상으로 생성하고 이 가상 디바이스들이 애플리케이션 로직의 입력이 되는 데이터를 전송할 수 있도록 함으로써 가능해진다. TITAN은 이러한 가상 디바이스와 개발 중인 애플리케이션의 실행 환경을 제공해주어, 이들이 개발자에 의해 지정된 IoT 플랫폼과 인터랙션 할 수 있도록 한다. 서드 파티 IoT 플랫폼 입장에서는 실제 IoT 디바이스와 차이 없이 TITAN에서 생성한 가상 디바이스와 데이터를 주고받게 된다. 가상 디바이스의 동작은 record-and-replay 기법에 기반 하여 이루어진다.

개발자가 사전에 다양한 애플리케이션 실행 시나리오에 따라 수집하거나 생성한 데이터를 TITAN의 데이터셋 저장소에 업로드 하면, TITAN은 이를 기반으로 개발자가 애플리케이션 실행을 원할 때마다 가상 디바이스를 생성하고 저장된 데이터가 전송될 수 있도록 한다.

2장에서는 관련 연구에 대해서 논의하고, 3장에서 TITAN의 시스템 디자인과 프로토타입 구현 내용을 기술한다. 그리고 4장에서 TITAN 프로토타입을 이용한 사용자 스터디를 통해 TITAN의 유용성에 대한 평가 결과를 서술한다. 5장에서 현재 제시된 TITAN의 한계점과 향후 연구 계획을 기술하고 논문을 끝맺는다.

Ⅱ. 관련 연구

사물 인터넷과 관련된 기술 및 인프라가 발전하면서 스마트 홈, 스마트 빌딩, 스마트 팩토리 등 다양한 분야에서 많은 IoT 애플리케이션들이 개발되고 있다 [1-3, 6-7]. 이와 더불어 IoT 애플리케이션 및 시스템 테스팅과 관련된 연구도 활발히 진행되고 있다 [8-10].

이 가운데 논문 [8-9]에서는 IoT 솔루션 및 시스템의 테스팅과 관련된 여러 이슈와 도전 과제들에 대한 분석 내용을 제시하고 있다. 다양한 이기종의 디바이스, IoT 디바이스의 제한된 컴퓨팅 자원, 분산된 노드들, 다양한 환경의 네트워크 연결성, 이종의 플랫폼 등 IoT의 특성으로 인해 IoT 시스템의 테스팅의 어려움을 기술한다. 그리고 IoT 시스템 테스팅에서 상호운용성 테스팅, 제한된 네트워크 조건에서의 IoT 시스템의 테스팅 등의 이슈를 중요하게 언급하고 있다. 이러한 문제를 극복하기 위해서 [9]에서는 크라우드 컴퓨팅의 개념을 기반으로 다양한 참여자가 공유된 분산 테스트 플랫폼을 만들 수 있도록 지원하는 시스템 아키텍처를 제안한다. [10]에서는 IoT의 이종성, 분산성으로 인한 IoT 기반 솔루션 테스트의 어려움을 극복하기 위해 패턴 기반의 테스트 자동화 프레임워크를 제안한다. IoT 시스템에서 일반적으로 나타나는 주기적인 센서 리딩, 커맨드에 의한 액추에이터 구동, 이벤트 기반 액션 등의 패턴을 정의하여 효과적인 테스팅을 할 수 있도록 지원한다.

위 연구에서는 IoT 시스템을 구성하는 여러 가지 요소들 IoT 디바이스, 센서, 서버 등의 컴포넌트들이 모두 연동하여 동작하는 실제 테스트베드 상에서 테스트를 지원하는 솔루션을 제안하고 있다. IoT 애플리케이션을 최종적으로 출시하기 위해서는 실제 테스트베드 상에서 테스팅을 수행하는 것은 반드시 필요하다. 하지만 이 논문에서는 IoT 애플리케이션 로직의 구현 및 디버깅을 목적으로 실제 IoT 테스트베드 상에서 반복적인 테스트를 수행하는 데 있어 발생하는 문제점을 제시하며, 개발 환경에서 편리하게 테스팅을 할 수 있는 지원 도구를 제안한다. 이러한 도구는 IoT 애플리케이션 개발 과정 중 실제 테스트베드 상에서 테스트를 하기 전 초기 단계에서 소프트웨어의 유닛 테스팅이나 통합 테스팅에 도움을 줄 수 있을 것이다. 따라서 우리 연구는 기존 연구와 상호보완적으로 활용될 수 있을 것이다.

이 논문에서 제안하는 TITAN에서 record-and-replay 기법 기반의 가상 IoT 실행 환경을 제공하는 것과 유사하게 record-and-replay 기법을 모바일 애플리케이션 테스팅을 위한 도구에 적용한 연구가 있었다 [11,12]. 이 연구에서는 모바일 디바이스 상에서 애플리케이션의 입력으로 사용될 수 있는 센서 데이터, 터치 제스처 데이터, GPS 데이터 등을 저장하고 이를 리플레이하여 모바일 애플리케이션을 테스트한다. 우리 연구는 다양한 IoT 센서, IoT 플랫폼과 연동하는 애플리케이션 등 모바일 애플리케이션과는 다른 환경을 다룬다.

TITAN을 기반으로 테스트를 수행하기 위해서는 가상 IoT 디바이스 동작을 가능하게 하기 위한 센서 데이터셋이 필요하다. TITAN에서는 시뮬레이션 환경에서 생성된 데이터셋이나 실제 IoT 디바이스 및 센서들이 구축된 테스트베드 환경에서 수집한 센서 데이터셋 모두 활용할 수 있다. 시뮬레이션 환경에서 데이터 수집은 기존 연구에서 제안한 스마트 홈 시뮬레이터 [13,14] 등의 도구를 활용할 수 있을 것이다. 실제 테스트베드 환경에서 데이터 수집을 위해서는 그에 따른 시간과 비용이 수반될 수 있지만, 일단 데이터 수집이 되면, 이를 이용하여 반복적인 테스팅을 할 수 있기 때문에 전체적인 소요 시간과 비용 측면에서는 이점이 있을 것이다.

Ⅲ. TITAN 디자인 및 구현

3.1. TITAN 디자인 고려 사항

IoT 애플리케이션 개발 중 수반되는 반복적인 서비스 로직의 실행, 결과 확인 및 로직 수정 과정에서 발생

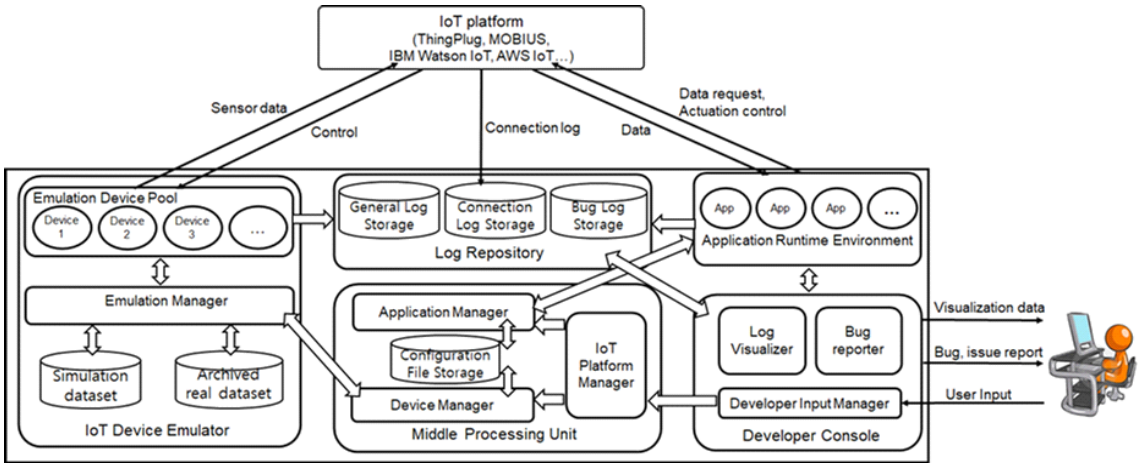

Fig. 1 TITAN architecture

하는 시간 소모와 비용을 최소화할 수 있게 하기 위하여 TITAN을 디자인하였다. 다음과 같은 고려 사항이 중점적으로 디자인에 반영되었다.

- 다양한 테스트 시나리오 지원: 애플리케이션 로직에 영향을 주는 다양한 실행 시나리오에 대한 센서 데이터를 이용하는 것을 지원할 수 있어야 한다.
- 커스터마이즈 된 테스트 환경 제공: 개발자가 테스트하고자 하는 IoT 애플리케이션 실행 환경을 제공할 수 있어야 한다. 다양한 IoT 플랫폼과 연동을 지원할 수 있어야 하며, IoT 애플리케이션의 실행에 필요한 센서 데이터가 개발자의 명세에 맞게 제공될 수 있어야 한다.
- 테스트 실행의 편의성 제공: IoT 애플리케이션에서 사용하는 디바이스가 다양하고 많을 수 있고 실행되어야 할 애플리케이션의 수도 많을 수 있는데 테스팅 과정의 편의성을 위해 이에 대한 제어가 쉽게 이루어질 수 있어야 한다.
- 테스팅과 관련된 풍부한 정보 제공: IoT 디바이스와 애플리케이션 간의 데이터 송수신 상태, 런타임 에러, 실행 결과에 대한 정보 등 테스트 실행 진행 상황 혹은 완료 후 결과를 개발자가 쉽게 확인할 수 있도록 제공하여야 한다.

3.2. TITAN 아키텍처

TITAN은 개발자가 구현한 애플리케이션 코드의 테스팅을 위해 센서를 비롯한 IoT 디바이스가 설치되어 구동 중인 실제 환경을 이용하지 않고도 개발 환경에서 테스팅이 가능하도록 디자인되었다. 따라서 개발 중인 애플리케이션 로직을 테스팅 하는데 수반되어야 하는 물리적 환경 및 사용자 행동에 제약받지 않고 애플리케이션의 실행 및 결과 확인이 가능하다.

그림 1은 TITAN의 아키텍처를 보여준다. TITAN은 Developer Console, Middle Processing Unit, IoT Device Emulator, Application Runtime Environment, Log Repository 총 5개의 주요 컴포넌트로 구성되어 있다. 이러한 컴포넌트는 단순한 사용자 인터페이스를 통해 개발자가 IoT 애플리케이션을 테스팅 할 수 있는 환경을 쉽게 셋업할 수 있도록 하고, 편리하게 반복적인 테스팅을 수행할 수 있도록 하며, 테스팅 과정 및 결과 정보를 제공하는 기능을 지원한다.

3.2.1. TITAN 사용 시나리오 및 시스템 동작

개발자는 IoT 애플리케이션 개발 시 TITAN을 이용하여 구현 중인 애플리케이션 로직의 테스트를 간편하게 수행할 수 있다. 다음은 TITAN 사용 시나리오와 그에 따른 시스템 동작을 간략히 기술한다.

(1) 개발자는 애플리케이션을 테스팅 하기 위한 케이스를 정의하고 이에 상응하는 센서 데이터를 수집하여 저장한다. 그리고 이 데이터를 이용하여 개발자가 원하는 테스팅 시나리오에 맞게 데이터셋으로 만들어 데이터셋 저장소에 저장한다. 센서 데이터는 실제 센서 디바이스를 설치하고 구동하여 수집하거나,

시뮬레이션 환경에서 수집하여 사용할 수 있다.

(2) 개발자는 Developer Console에서 개발하고자 하는 애플리케이션을 위한 센서 디바이스와 액추에이터 디바이스를 IoT 플랫폼과 연결하기 위해 필요한 디바이스 ID 혹은 토큰값과 같은 명세 데이터를 입력한다. 명세 데이터를 얻기 위해 IoT 플랫폼에 디바이스를 등록하는 과정은 사전에 수행되어야 한다.

(3) 개발자는 각 플랫폼에 맞춰 애플리케이션 코드와 IoT 디바이스(센서, 액추에이터)에서 실행될 코드를 구현한다.

(4) 개발자가 Developer Console에 입력한 데이터는 Middle Processing Unit에 전달되어 개발할 IoT 애플리케이션을 위해 사용할 플랫폼을 선택하고, 플랫폼에 맞는 설정 파일을 저장하여 가상 IoT 디바이스와 애플리케이션 실행 시 사용할 수 있도록 한다.

(5) 개발자가 구현한 코드를 Developer Console을 통하여 입력하면, IoT Device Emulator에서는 개발자의 명세에 맞게 사용할 센서 데이터를 선택하고 개발자가 구현한 IoT 디바이스 코드를 이용하여 가상 디바이스 인스턴스를 생성한다.

(6) Application Runtime Environment에서는 개발자가 업로드 한 애플리케이션 코드를 실행하고 애플리케이션 실행 상태를 모니터링하여 관련 로그 정보를 Log Repository에 저장하고 Developer Console로 전달한다.

(7) Developer Console은 로그 정보를 바탕으로 이슈 리포트를 생성하거나 실행 결과에 대한 요약 정보를 가시화하여 개발자에게 제공한다.

(8) 개발자는 이러한 정보를 바탕으로 필요에 따라 애플리케이션을 수정하고 이전과 동일한 데이터셋 혹은 새로운 데이터셋에 대해서 테스트를 수행한다.

3.2.2. 주요 구성 요소

Developer Console: 개발자가 TITAN을 사용할 수 있는 GUI 기반 인터페이스를 제공하며, 개발자로부터 테스팅에 필요한 입력을 받고, 테스팅 과정 및 결과를 출력하는 컴포넌트이다. Developer Input Manager는 개발자로부터 받은 IoT 서비스 환경에 대한 명세 데이터를 관련된 TITAN의 각 구성 요소로 전달한다. Log Visualizer는 가상 IoT 서비스 환경에서 동작하는 디바이스, 애플리케이션이 생성하는 로그 파일을 이용하여 입출력 데이터, 실행 로그 등을 가시화하여 제공한다. Bug Reporter는 실시간으로 로그를 출력해 테스팅 상황에서 발생하는 문제를 개발자에게 알려준다.

Middle Processing Unit: 개발자가 입력한 IoT 플랫폼에 맞게 필요한 설정 데이터를 셋팅하거나 제공해주는 컴포넌트이다. IoT Platform Manager는 개발자가 사용하고자 하는 IoT 플랫폼에 대한 정보를 전달받아 플랫폼에 맞는 설정 파일을 작성하고 불러올 수 있도록 한다. Device Manager는 개발자의 입력과 IoT Device Emulator의 데이터셋을 기반으로 가상으로 생성할 디바이스의 리스트를 관리한다. Application Manager에서는 개발자가 테스팅 하고자 하는 애플리케이션의 목록을 관리한다. Configuration File Storage는 개발자가 입력한 디바이스/애플리케이션의 설정 데이터를 저장하고 Device/Application Manager의 요청에 따라 사용될 플랫폼에 맞게 필요한 설정 데이터를 제공한다.

IoT Device Emulator: 개발자가 애플리케이션 테스팅을 위해 필요한 가상 IoT 디바이스 인스턴스를 생성하여 실행하고 관리하는 컴포넌트이다. Emulation Manager는 Middle Processing Unit의 Device Manager로부터 받은 디바이스 리스트를 기반으로 가상 디바이스 인스턴스를 생성하는 역할을 한다. 데이터셋 저장소에 있는 관련 데이터 파일에서 개발자의 명세에 맞게 사용할 데이터를 선택하고 개발자가 업로드한 디바이스 코드를 이용하여 에뮬레이션 디바이스 인스턴스를 생성한다. Emulation Device Pool에서는 생성된 하나 혹은 다중의 가상 디바이스들이 관리되며 개발자에게 받은 명령을 기반으로 디바이스들의 실행을 제어할 수 있다. 센서 디바이스가 실행되면 개발자가 선택한 IoT 플랫폼과 연결을 맺고 센서 데이터를 전송한다. 액추에이터 디바이스가 실행되는 경우에는 애플리케이션이 전송하는 제어 명령을 대기하는 상태가 된다. 각 디바이스는 데이터를 송수신할 때마다 Log Repository에 데이터를 기록한다.

Application Runtime Environment: 개발자가 테스팅을 위해 업로드 한 애플리케이션 코드를 실행하고 관리하는 컴포넌트이다. 개발자로부터 전달받은 실행/중지 명령으로 애플리케이션의 실행을 제어할 수 있고, 필요에 따라 다수의 애플리케이션을 동시에 실행할 수 있다. 애플리케이션이 실행 상태에 있는 경우에는 각 애플리케이션이 IoT 플랫폼과 통신을 유지하며 센서 데이터를

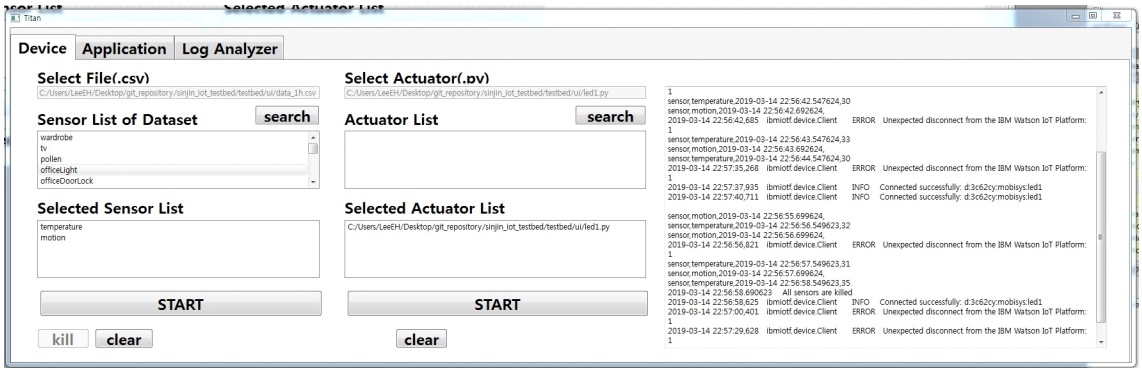

Fig. 2 TITAN screenshot (Device control tab)

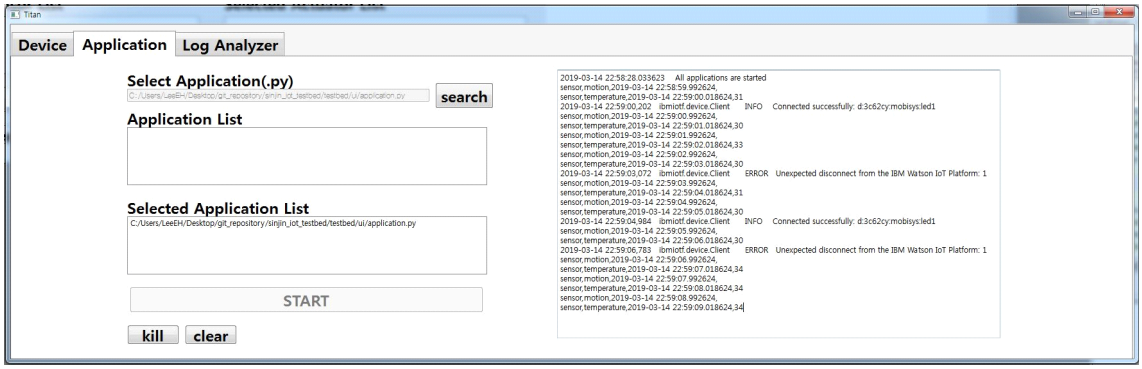

Fig. 3 TITAN screenshot (Application control tab)

요청하고, 로직에 따라 액추에이터에 제어 명령을 내릴 수 있다. 애플리케이션에서는 데이터를 송수신하거나 에러가 발생한 경우 Log Repository에 로그 데이터를 기록한다.

Log Repository: 다양한 종류의 로그 파일들이 저장되는 공간이다. General Log Storage는 실행되고 있는 디바이스, 애플리케이션의 실행 상태에 대한 로그가 저장된다. Log Visualizer에서 로그 데이터를 요청하는 경우 해당 데이터를 전달해준다. Bug Log Storage는 테스팅 대상이 되는 현재 실행 중인 애플리케이션에서 발생하는 에러 로그가 저장된다. 이는 필요시 Bug Reporter에 전달된다. Connection Log Storage는 사용 중인 IoT 플랫폼과의 네트워크 연결 상태에 대한 로그가 저장된다. 실시간으로 연결 관련 정보를 출력할 때 이 로그를 사용한다.

3.3. TITAN 프로토타입

본 논문에서는 그림 1에서 제시한 TITAN 아키텍처를 기반으로 TITAN의 프로토타입을 구현하였다. 현재 TITAN 프로토타입은 Windows OS 상에서 파이썬 3을 이용하여 구현되었으며, Developer Console에서 개발자가 사용하는 GUI는 PyQt5를 기반으로 구현되었다. 그리고 Log Visualizer에서 그래프로 가시화하는 것은 Plotly를 기반으로 구현되었다. IoT 애플리케이션 구동을 위한 플랫폼으로는 현재 IBM Watson IoT Platform과 ThingPlug를 지원한다.

그림 2와 3은 구현된 TITAN 프로토타입의 스크린 샷을 보여준다. TITAN의 Developer Console에서 제공되는 개발자 인터페이스는 크게 디바이스 관리, 애플리케이션 관리, 로그 분석 세 가지 기능으로 구분된다. 그림 2는 IoT 디바이스 실행과 관련된 Device 탭으로, 화면의 좌측에서는 개발자가 IoT 서비스 실행에 사용할 센서 및 액추에이터 디바이스를 선택하고 실행할 수 있는 인터페이스를 제공한다. 또한, 우측에서는 선택된 디바이스를 가상 환경에서 실행하는 과정에서 이루어지는 데이터 송수신 등의 상태 정보 로그를 실시간으로 확인할

수 있다. 그림 3은 IoT 애플리케이션 실행과 관련된 Application 탭으로, 개발자가 테스팅 하고자 하는 애플리케이션 코드를 선택하여 실행을 시작하거나 종료할 수 있다. 화면 우측에서는 애플리케이션 실행 시 출력되는 로그 정보를 실시간으로 확인할 수 있다. 그림으로 보여지진 않았지만, Log Analyzer 탭의 경우 로그 저장소에 저장된 로그를 분석하여 요약 정보 텍스트와 그래프로 가시화된 정보를 보여준다.

Ⅳ. 실 험

구현된 TITAN 프로토타입을 평가하기 위한 실험을 진행하였다. 이 논문에서는 소규모의 참가자를 대상으로 초기 실험을 통해 IoT 애플리케이션 개발과 테스팅 과정에서 TITAN의 유용성을 살펴보고자 하였다.

4.1. 실험 참가자

실험을 위하여 총 4명의 실험 참가자를 모집하였다. 참가자들은 모두 컴퓨터공학을 전공하는 4학년 학생들로, 4명 중 2명의 참가자는 IoT 애플리케이션 개발 경험이 있었으며, 나머지 2명은 개발 경험이 없었다. 개발 경험이 있는 참가자의 경우 라즈베리파이와 충격감지센서, 유해가스 센서 등을 이용하여 창문 및 문 파손 감지 시스템이나 화재 감지 시스템 같은 IoT 애플리케이션을 만들어본 경험이 있었다.

4.2. 실험 설계

4.2.1. 피험자 내 설계

본 논문에서는 실험 참가자가 IoT애플리케이션을 개발하고 테스팅 하는 과정에서 TITAN을 사용하지 않고 진행하는 경우와 TITAN을 사용하여 진행하는 경우에 어떤 차이점이 있는지 비교할 수 있도록 실험을 설계하였다. 이러한 목적을 달성하기 위해서는 실험자들이 각 경우에 대해 객관적으로 평가할 수 있도록 두 가지 경우를 모두 경험하는 것이 필요하다고 판단했다. 이를 만족하기 위해, 실험자가 TITAN을 사용하지 않고 개발 및 테스팅을 진행하는 경우와 TITAN을 사용하여 진행하는 경우를 실험의 독립 변인으로 설정하고, 독립 변인의 모든 조건에 실험자들을 할당하는 Within-subject [15] 형태의 실험으로 설계하였다.

4.2.2. 실험용 IoT 애플리케이션

실험용 IoT 애플리케이션의 기능 구현 요구사항을 정의하는데 다음과 같은 2가지 사항을 중요하게 고려하였다. 첫째, 복수개의 센서를 사용하여 센싱을 하고 이를 활용하는 IoT 애플리케이션을 개발하는 것을 목표로 했다. 둘째, 애플리케이션 구현의 난이도가 적절한 수준이 될 수 있도록 했다. 실험에 시간 제약이 존재하므로 난이도가 높으면, 실험자들은 프로그램 작성에 많은 시간을 써야 할 것이기 때문에 구현한 프로그램을 여러 시나리오에 대해 테스트할 시간을 확보하지 못할 것이다. 그래서 적절한 난이도의 애플리케이션을 태스크로 부여함으로써 실험자들이 충분히 코드를 짜고, 이에 대한 테스팅을 충분히 할 수 있도록 하였다.

피험자 내 설계 형태로 설계된 실험에서 TITAN 사용과 TITAN 미사용 두 가지 경우에서 개발해야 하는 애플리케이션이 동일한 경우, 연습 효과 때문에 실험 결과에 대한 신뢰성에 문제가 생길 수 있다. TITAN의 사용 여부에 대한 독립 변인 이외의 다른 변인들의 영향을 최소화하기 위해 실험자들이 비슷한 난이도의 2가지 다른 애플리케이션을 개발하는 것으로 미션을 부여하였다. 실험에 사용한 애플리케이션은 다음과 같다.

VDT(Visual Display Terminal) 증후군 예방 애플리케이션: 이 애플리케이션은 책상 앞에 사람이 40분 이상 앉아 있다면, 휴식을 권하는 알림을 주는 기능과 모니터와 사람의 거리가 50센치 이하라면 모니터로부터 멀어지기를 권하는 기능을 수행한다. 이를 구현하기 위해서 모니터와 사람의 거리를 측정하는 초음파 센서 1개와 사람의 움직임을 감지하는 PIR 센서를 활용하도록 하였다. 부가적인 기능으로, 초음파 센서는 사람이 책상 앞에 있다고 판단될 때만 동작하여 센서 값을 발생하도록 하며, PIR 센서는 15분 이상 사람의 움직임이 감지되지 않을 경우 사람이 없는 것으로 간주한다.

실내 인원수 모니터링 애플리케이션: 이 애플리케이션은 한 오피스 내에 몇 명의 사람이 있는지를 모니터링 하는 기능을 수행한다. 이를 구현하기 위하여, 오피스의 출입문을 기준으로 외부와 내부의 움직임을 파악하는 PIR 센서 2개와 문 열림을 감지하는 초음파 센서 1개를 활용하도록 하였다. 외부 움직임 감지, 문 열림, 내부 움직임 감지가 연속적으로 발생한 경우 오피스에 사람이 들어온 것으로 계산하고, 내부 움직임 감지, 문 열림, 외부 움직임 감지가 연속적으로 발생한 경우 오피스 내의

사람이 밖으로 나간 것으로 간주하도록 하였다.

제한된 실험 시간을 고려하여, 실험 참가자들이 두 가지 IoT 애플리케이션의 핵심 로직을 구현하고 테스트하는 것에 집중할 수 있도록 애플리케이션의 스켈레톤 코드를 미리 작성하여 제공하였다.

4.2.3. 데이터셋

TITAN을 이용한 테스팅을 수행하기 위해서는 가상의 센서 디바이스에서 애플리케이션에 전송해 주기 위한 데이터가 필요하다. 실험용 IoT 애플리케이션에 필요한 데이터를 몇 가지 케이스로 구분하여 정의하였고 이를 아래에 제시하였다. VDT 증후군 예방 애플리케이션의 경우 세 가지 케이스로 구성되어 있으며, 실내 인원수 모니터링 애플리케이션 데이터셋은 네 가지 케이스로 구성되어 있다. 이러한 데이터셋은 실험을 위해 실제 센서 디바이스를 설치한 오피스 환경에서 각 케이스에 따라 연구자들이 직접 액션을 취하여 수집하였다.

- VDT 증후군 예방 애플리케이션 데이터셋
  1. 사람이 의자로 와서, 5초간 앉았다가 일어나서 나간 경우
  2. 사람이 의자로 와서 10분 이상 앉아서 작업하는 경우 (중간에 모니터와 가까워지는 경우를 포함)
  3. 사람이 의자로 와서 5분간 앉아서 작업하다가, 20분 동안 휴식을 취하고 다시 30분 이상 작업한 경우 (중간에 모니터와 가까워지는 경우를 포함)
- 실내 인원수 모니터링 애플리케이션 데이터셋
  1. 오피스 외부의 사람이 문 앞을 지나가는 경우
  2. 오피스 내부의 사람이 오피스 내에서 문 앞을 지나가는 경우
  3. 외부의 사람이 문을 열고 내부로 들어오는 경우
  4. 내부의 사람이 문을 열고 외부로 나가는 경우

4.2.4. 실험 태스크

실험을 위해 참가자들에게 부여된 태스크는 표 1에 정리된 것과 같다. 앞서 언급한 것과 같이 피험자 내 설계로 실험이 진행되어, 각 참가자는 2개의 애플리케이션에 대해 하나는 TITAN을 사용하지 않는 방식으로, 다른 하나는 TITAN을 사용하는 방식으로 애플리케이션 테스팅을 수행한다. 4명의 참가자 중 2명(P1, P2)은 IoT 애플리케이션 개발에 경험이 있었고, 나머지 2명(P3, P4)은 없었기 때문에, 참가자를 두 그룹으로 나누어 태스크를 부여하였다. P1과 P2는 우리가 제공한 스켈레톤 코드를 기반으로 애플리케이션을 작성하는 것부터 시작하여 테스팅 하고 수정하는 것을 모두 수행하도록 하였고, P3와 P4는 구현된 애플리케이션을 제공하여 이를 테스팅 하고 필요에 따라 수정하는 것을 수행하도록 하였다.

4.2.5. 실험 절차

실험은 총 세 단계로 이루어진다. 첫째는 준비 단계로, 약 30분 동안 사전 인터뷰와 실험에 대한 설명을 진행했다. 사전 인터뷰에서는 기존에 경험했던 IoT 애플리케이션 개발 사례와 개발 과정, 그때 겪었던 어려움에 대해 알아보았다. 그리고 실험 방법과 실험에서 수행해야 하는 태스크에 대해 설명하고 TITAN의 사용법을 설명하였다. 또한, TITAN을 사용하는 경우 애플리케이션에서 사용할 데이터셋에 대해 설명하였다. 둘째, 본 실험 단계로, 참가자들은 약 3시간 동안 자신에게 부여된 태스크를 수행하였다. 각 참가자에 부여된 태스크는 위에서 언급한 2개의 애플리케이션, VDT 증후군 예방 애플리케이션과 실내 인원수 모니터링 애플리케이션을 개발 및 테스팅 하는 것이고, 각 애플리케이션에 1시간 반씩 시간을 할당하였다. 이 과정에서 연구자들은 개발 및 테스팅 중인 실험자들을 관찰하면서 테스팅 수행 횟수를 카운트하였다. 셋째, 마지막 단계로 태스크 수행 후 인터뷰를 진행하였다. 여기서는 TITAN을 사용하여

Table. 1 Experimental tasks

Task	Participant	Scope	VDT Syndrome Prevention Application		Room Occupancy Monitoring Application	
			W/ TITAN	W/O TITAN	W/ TITAN	W/O TITAN
Task1	P1	Coding/Testing		O	O	
Task2	P2	Coding/Testing	O			O
Task3	P3	Testing		O	O	
Task4	P4	Testing	O			O

개발 및 테스팅을 할 때와 사용하지 않았을 때의 경험과 TITAN의 유용성에 대해 알아보았다.

실험 과정에서 참가자가 테스팅을 수행하였다는 것은 애플리케이션 로직의 실행 결과에 영향을 주는 다양한 케이스에 대해 데이터를 입력하여 그 결과를 확인해 보는 것으로 정의하였다. TITAN을 사용하지 않는 경우는 실험참가자가 직접 문을 열거나 의자에 앉는 등의 실제 행동을 하여야 했고, TITAN을 사용하는 경우는 해당되는 데이터셋을 이용하여 데이터가 입력되도록 하였다. 연구자는 실험자들을 관찰하면서 위와 같이 테스팅을 수행하는 횟수를 기록하였다. 문법적인 오류를 확인하기 위한 단순한 테스트나 TITAN과 연동하는 IoT 플랫폼과의 데이터 흐름을 파악하기 위한 테스트의 경우는 TITAN을 사용한 효과를 보여주는 것으로는 적합하지 않다고 판단하여 제외하였다.

실험에서 센서 데이터 수집 및 전송을 위해 Raspberry Pi3을 사용하였고, IoT 플랫폼으로는 IBM Watson IoT Platform을 사용했다. 제한된 실험 시간을 감안하여 사전에 애플리케이션 개발에 사용되는 센서를 오피스에 설치하고 Watson Platform 사용을 위한 센서 디바이스 및 애플리케이션 등록을 완료하였다. 그리고 실제 구동 시 플랫폼과 연결에 필요한 토큰이나 키값을 담고 있는 설정 파일을 제작하여 실험 참가자들에게 제공했다.

4.3. 실험 결과

4.3.1. 테스팅 횟수

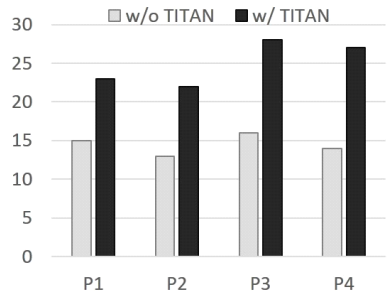

Fig. 4 The number of test runs

그림 4를 보면 실험자 모두 TITAN을 사용하지 않고 테스팅을 할 때보다 TITAN을 사용하였을 때 더 많은 횟수의 테스팅을 진행하였음을 알 수 있다. 실험자 모두 TITAN을 사용하지 않는 경우 처음에는 직접 행동을 해가며 테스팅 하는 것이 재미있었지만, 시간이 지날수록 점점 귀찮아져서 최대한 코드를 눈으로 확인한 후 테스팅을 하려고 했다고 답을 하였다. 그리고 TITAN을 사용한 경우에는, 코드를 수정하고 TITAN에서 몇 번의 클릭만으로도 테스팅을 할 수 있어서 더 많이 반복해서 테스팅을 했던 것 같다는 공통적인 답변을 하였다. 다만 현재 실험에서는 TITAN 사용 시 테스트 수행 횟수가 공통적으로 증가하였다는 것 이외의 측면에 대해서는 평가가 이루어지지는 않았다. TITAN으로 인한 테스팅의 용이성이 구현 시간 단축이나 코드 품질 향상 같은 측면으로 효과가 있는지에 대해 평가하는 것은 추후 연구로 진행할 것이다.

4.3.2. TITAN의 유용성

전체적으로, 실험에 참가한 모든 참가자가 TITAN의 유용성에 대해 만족해했다.

테스팅을 위해 반복적으로 액션을 취해야 하는 어려움 개선: 사후 인터뷰에 따르면, 실험에 참가한 모든 참가자가 TITAN을 사용함으로써, IoT 애플리케이션을 테스팅 하는 데 필요했던 노력을 줄일 수 있었다고 답했다. P1, P3는 TITAN을 사용하지 않고 개발 및 테스팅을 수행하면서 프로그램의 기능 테스팅을 위해 센서의 값을 변화시키고자 할 때 자신이 직접 문을 열고 나가거나 하는 등의 액션을 취했어야 했는데, TITAN을 사용하는 경우에는 자리에 앉아서도 센서 데이터를 변경하여 테스팅 할 수 있어서 좋았다고 답을 했다. P2, P4는 TITAN을 사용하는 개발 및 테스팅을 수행할 때는 자리에만 앉아서 테스팅 해서 몰랐지만, 이후의 TITAN을 사용하지 않은 경우에는 계속해서 움직여야 해서 귀찮았다고 답을 했다. 추가로 P3, P4는 테스팅을 할 때 직접 행동을 해야 하는 것이 귀찮아서, 코드의 이상 유무를 파악하는데 프로그램을 직접 실행해 보지 않고, 코드를 보며 머릿속으로 생각하는 것에 더 오랜 시간을 소모했다고 답했다.

IoT 애플리케이션 개발 경험이 있었던 P1과 P2는 이전 경험에 비추어 TITAN의 유용성을 높이 평가하기도 하였다. P1은 "지난번에 했던 프로젝트에서 GPS 센서를 쓰면서 계속 직접 걸어 다녀야 해서 귀찮았는데 TITAN을 썼었으면 안 걸어도 돼서 좋았을 것 같아요." 라고 언급하였다. P2는 파손 감지 도어락과 창문을 만들

때, 어느 정도의 세기로 힘이 가해졌을 때 파손으로 감지해야 하는가를 파악하기 위해 도어락과 창문에 여러 번의 충격을 가하는 것이 필요했다고 했다. 그 과정에서 많은 소음이 발생하여 주변에 피해를 주는 것 같아서 미안했었는데, 충격에 따른 센서 데이터를 저장해두고 TITAN을 활용했다면 그런 문제가 없었을 것 같다는 얘기를 하였다. 이 외에도 실험 참가자들은 TITAN을 사용할 때는 자신들이 직접 행동을 반복적으로 하지 않아도 테스팅을 할 수 있기 때문에 그 과정에 소요되는 시간을 절약하고 코드 수정하는데 시간을 더 쓸 수 있어서 시간적으로도 도움이 된 것 같다는 답을 하였다.

하드웨어 이슈에 영향 받지 않고 애플리케이션 로직 검증에 집중 가능: TITAN을 이용하면 실제 센서 하드웨어를 사용하지 않고도 애플리케이션을 테스팅 하는 것이 가능하므로, 참가자들은 다른 문제에 신경 쓰지 않고 애플리케이션 로직을 테스팅 하는 데 집중할 수 있다는 점 또한 장점으로 인식하였다. TITAN과 같은 테스팅 도구가 없는 경우에는 센서 하드웨어의 오동작과 같은 이슈가 발생하면 그것을 해결해야만 테스팅을 수행할 수 있다. P2는 "제가 전에 했던 프로젝트가 센서가 좀 많았는데, 중간에 센서 하나가 고장 나면 다시 확인해서 고치고 켜야 되고 해서 너무 힘들었어요. 근데 TITAN은 저장해놓은 데이터를 쓰니까 센서가 고장 나는 것을 걱정 안 해도 돼서 좋은 거 같아요."라고 말을 했다. P1의 경우 TITAN을 사용하지 않고 테스팅을 하는 도중 초음파 센서가 갑자기 반응하지 않아서 시간을 좀 허비하였다. 이런 상황을 경험하였던 P1 또한 "갑자기 센서가 고장이 나서 시간을 좀 허비했는데, 길어졌다면 많이 힘들었을 것 같아요. 두 번째 애플리케이션을 개발할 때는 센서가 고장 나는 걸 신경 안 써도 돼서 좋았어요."라고 말을 했다.

다양한 테스트 시나리오 생성 가능: 실험 참가자들은 TITAN에서 테스팅을 위해 제공된 데이터셋을 활용하여 자신이 생각한 애플리케이션 동작 시나리오를 직접 만들 수 있다는 것에도 긍정적인 반응을 보였다. 실험자들은 데이터셋을 이용하여, 자신이 생각한 여러 가지 시나리오에 대한 데이터셋을 만들어 테스팅에 활용하기도 하였다. P1은 이전에 화재 감지 시스템을 개발할 때, 연기 센서를 사용했었는데, 테스팅을 할 때마다 원하는 센서의 값을 얻기 위해 직접 연기를 피웠어야 했다고 얘기했다. 만약 당시에 TITAN을 사용할 수 있었다면 한 번에 연기의 양에 따른 여러 케이스의 센서 데이터를 모으고 이를 기준으로 여러 데이터셋을 만들어 테스팅에 이용할 수 있었을 것 같다고 하였다. 그리고 그랬다면 테스팅이 훨씬 수월했을 것 같다고 하였다.

4.3.3. TITAN의 사용성

실험 참가자 모두 처음 접하는 TITAN을 사용하는 것에 대해 어려워하지 않았다. 그리고 처음에는 어색한 점이 있었지만 금방 적응이 될 정도로 사용하기가 쉬웠다고 답을 하였다. 또한 TITAN의 장점 중 하나로, 몇 번의 클릭만으로도 여러 센서와 애플리케이션을 쉽게 다룰 수 있는 것이라고 하였다. P2는 자신이 했던 IoT 애플리케이션 개발 프로젝트에서 본 실험보다 더 많은 센서를 사용했었는데, 그때는 cmd창을 이용해 여러 센서를 일일이 제어하는 것이 귀찮았지만, TITAN은 클릭 몇 번만으로 제어가 가능해서 좋았다고 답을 하였다.

다만 현재 TITAN 프로토타입은 안정화가 더 필요한 부분이 있어, 실험 참가자들이 사용 중 다운되어 다시 실행해야 하는 불편함을 언급하기도 하였다. 그리고 TITAN에서 디바이스와 애플리케이션이 송수신한 데이터를 그래프로 가시화하여 보여주는 부분에서 너무 많은 데이터 포인터들이 존재하는데 그 구분이 불명확한 점을 지적하기도 하였다. 이러한 부분에 대해서는 향후 보완을 할 계획이다.

Ⅴ. 한계 및 향후 연구 방향

TITAN이 지원하는 테스팅의 범위: TITAN은 가상의 IoT 디바이스를 기반으로 애플리케이션 테스팅이 가능하도록 지원한다. 이는 애플리케이션 코드의 유닛 테스팅이나 통합 테스팅 과정에서 효과적으로 적용할 수 있을 것이다. 그러나 IoT 애플리케이션의 경우 궁극적으로는 실제 IoT 디바이스 하드웨어와 연동하여 동작할 것이기 때문에 모든 요소가 통합되어 구동되어야 하는 시스템 테스팅 과정에는 적용이 어려울 것이다. 하지만 시스템 테스팅 단계에 들어가기 전 소모되는 개발 및 테스팅 시간을 단축하여 IoT 애플리케이션 개발의 전체적인 프로세스에 도움이 될 것으로 예상한다.

실험의 한계 및 향후 계획: TITAN의 평가를 위해 진행했던 본 논문의 실험은 TITAN의 유용성을 평가하는 의미가 있지만, 그 한계도 있기에 추가적인 실험 및 연구가 필요하다. 첫째, 본 실험에서는 TITAN 프로토타입에 대한 기초적인 평가에 중점을 두어 컴퓨터공학 전공 학생을 대상으로 실험을 수행하였다. TITAN의 효과를 실질적으로 잘 보이기 위해서는 현업 IoT 애플리케이션 개발자 및 테스터를 대상으로 한 실험이 필요하다. 둘째, 많은 수의 개발자 및 테스터를 대상으로 하는 실험이 진행되어야 한다. 현재의 실험에서는 태스크 당 1명의 실험자가 배치되어 총 4명의 참가자를 대상으로 TITAN을 평가하였다. 현재 실험에서 TITAN의 유용성에 대해 긍정적인 평가를 얻었지만, 다수의 실험자를 대상으로 하는 추가 실험을 통하여 좀 더 신뢰성 있고 일반화할 수 있는 결과를 도출하고자 한다. 마지막으로 실험용 애플리케이션을 실제 상용 IoT 애플리케이션과 비슷한 수준의 스펙으로 정의하여 태스크를 부여하고, TITAN의 유용성에 대해 평가를 할 계획이다.

테스팅 시간 단축을 위한 에뮬레이션 가속화: TITAN은 사전에 준비된 데이터셋을 이용하여 가상의 센서 디바이스를 에뮬레이션함으로써 테스팅을 지원한다. 테스팅을 위한 데이터셋이 장시간 데이터라면 테스팅 결과를 확인하기 위해서 오랜 시간 동안 실행이 완료되는 것을 기다려야 한다. 이러한 문제를 해결하기 위해서 실행 결과에 영향을 주지 않으면서도 빠른 시간 내에 실행이 완료될 수 있도록 하기 위한 가속화 기술에 관한 연구를 수행할 계획이다.

센서 데이터셋 수집을 위한 지원: TITAN을 사용하여 테스팅을 하기 위해서는 애플리케이션 실행에 필요한 센서 데이터셋이 사전에 수집되어 있어야 한다. 현재는 실제 센서 디바이스를 설치하고 구동하여 수집하거나, 시뮬레이션 환경에서 수집하여 데이터셋을 만들 수 있다. OpenSHS[13]와 같은 스마트홈 시뮬레이터를 이용하여 데이터셋을 만드는 경우 시뮬레이터를 구동하는 것은 TITAN과 독립적으로 이루어져야 한다. 이러한 시뮬레이터와 연동하여 데이터 생성 및 수집 과정을 TITAN을 통해서 처리할 수 있도록 기능을 확장한다면 개발자의 편의성이 더 향상될 것이다. 그리고 실제 센서 디바이스들이 구동되는 환경에서 데이터를 수집하는 경우에도 데이터 수집 및 레이블링 과정을 최대한 자동화 할 수 있도록 지원하는 것도 필요할 것이다.

VI. 결 론

사람의 행동 혹은 주변의 환경을 인지하여 동작하는 IoT 애플리케이션의 개발자는 자신이 구현한 애플리케이션 로직이 그 동작이나 실행 결과가 달라질 수 있는 다양한 사용자 상황에서도 오류 없이 잘 동작하는지 확인하기 위해서 반복적인 테스트를 수행해야 한다. 하지만 IoT 애플리케이션의 특성상 다양한 실생활의 사용 상황에서 반복적인 테스팅을 한다는 것은 매우 어려운 일이다. 이러한 문제를 해결하기 위해, 본 논문에서는 IoT 애플리케이션 개발 과정에서 효율적인 테스팅을 지원하는 도구로 TITAN을 제안하였다. 개발자들은 TITAN을 이용하여 개발 과정에서 반복적인 테스팅에 소모되는 시간과 노력을 줄일 수 있을 것이다.

ACKNOWLEDGEMENT

This work was supported in part by the National Research Foundation of Korea(NRF) grant funded by the Korea government (2017R1C1B1010619, NRF-2017M3C4A7066473).

References

[1] OpenEnergyMonitor [Internet]. Available: https://guide.openenergymonitor.org/applications/home-energy/.

[2] X. Jin, G. Wang, Y. Song, and C. Sun, “Smart building energy management based on network occupancy sensing,” *Journal of International Council on Electrical Engineering*, 8(1), pp. 30-36, 2018.

[3] Propeller [Internet]. Available: https://www.propellerhealth.com/.

[4] IBM Watson IoT [Internet]. Available: https://www.ibm.com/internet-of-things/.

[5] Google Cloud IoT Core [Internet]. Available: https://cloud.google.com/iot-core/.

[6] M. Chae, Y. Kim, S. Kim, S. Kim, and S. Jung, “Study on Building Smart Home Testbed for Collecting Daily Health Condition based on Internet of Things,” *KIISE Transactions on Computing Practices*, vol. 23, no. 5, pp. 284-292, 2017. (in Korean)

[7] H. Lee, S. Moon, R. Y. Kim, and H. Son, “Constructing

Effective Smart Crosswalk Traffic Light Mechanism Through Simulation Technique," *KIISE Transactions on Computing Practices*, vol. 22, no. 2, pp. 113-118, 2016. (in Korean)

[8] M. Bures, T. Cerny, and B.S. Ahmed, "Internet of Things: Current Challenges in the Quality Assurance and Testing Methods," in *Proceedings of Information Science and Applications 2018*. ICISA 2018. Lecture Notes in Electrical Engineering, vol. 514, Springer, Singapore.

[9] P. Rosenkranz, M. Wahlisch, E. Baccelli, and L. Ortmann, "A Distributed Test System Architecture for Open-source IoT Software," in *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*, pp. 43-48, 2015.

[10] P. M. Pontes, B. Lima, and J. P. Faria, "Izinto: a pattern-based IoT testing framework," in *Proceedings of ISSTA'18 Companion Proceedings for the ISSTA/ECOOP 2018 Workshops*, pp. 125-131, 2018.

[11] Z. Qin, Y. Tang, E. Novak, and Q. Li, "MobiPlay: a remote execution based record-and-replay tool for mobile applications," in *Proceedings of the 38th International Conference on Software Engineering*, pp. 571-582, May. 14 - 22, 2016.

[12] C. Min, S. Lee, C. Lee, Y. Lee, S. Kang, S. Choi, W. Kim, and J. Song, "PADA: power-aware development assistant for mobile sensing applications," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM, pp. 946-957, 2016.

[13] N. Alshammari, T. Alshammari, M. Sedky, and J. Champion, "OpenSHS: Open Smart Home Simulator," *Sensors*, 17(5), 1003, 2017.

[14] J. Synnott, C. Nugent, and P. Jeffers, "Simulation of Smart Home Activity Datasets," *Sensors*, 15, 14162-14179, 2015.

[15] G. Charness, U. Gneezy, and M. A. Kuhn, "Experimental methods: Between-subject and within-subject design," *Journal of Economic Behavior & Organization*, 81(1) pp. 1-8, 2012.

이의혁(Euihyeok Lee)

2017년 한국기술교육대학교 컴퓨터공학부 학사
2019년 한국기술교육대학교 대학원 컴퓨터공학과 석사
2019년 9월-현재: 한국기술교육대학교 대학원 컴퓨터공학과 박사과정
※관심분야 : IoT/모바일 시스템, 모바일/웨어러블/유비쿼터스 컴퓨팅, 딥러닝

김동우(Dongwoo Kim)

2018년 한국기술교육대학교 컴퓨터공학부 학사
2018년-현재: 한국기술교육대학교 대학원 컴퓨터공학과 석사과정
※관심분야 : IoT/모바일 시스템, 모바일/웨어러블/유비쿼터스 컴퓨팅

강승우(Seungwoo Kang)

2010년 한국과학기술원 전산학과 공학박사
2015년-현재: 한국기술교육대학교 컴퓨터공학부 조교수
※관심분야 : IoT/모바일 시스템, 모바일/웨어러블/유비쿼터스 컴퓨팅