KIPS Tr. Software and Data Eng.
Vol.4, No.3 pp.99~110 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2015.4.3.99

Technical Issues and Solutions for Developing IoT Applications

Dong Ha Shin† · Seung Ho Han† · Hyun Jung La†† · Soo Dong Kim†††

ABSTRACT

Internet-of-Things(IoT) is the computing paradigm converged with different technologies, where diverse devices are connected via the wireless network, acquire environmental information from their equipped sensors, and actuated. IoT applications typically provide smart services to users by interacting with multiple devices connected to the network and are designed by integrating multiple technologies such as sensor network, communication technologies, and software engineering. Moreover, since the concept of IoT has been introduced recently, most of the researches are in the beginning step, which is too early to be practically applied. Due to these facts, developing IoT application results in unconventional technical challenges which have not been observed in typical software applications. And, it is not straightforward to apply conventional project guidelines to IoT application development projects. Hence, there can be many difficulties to successfully complete the projects. Therefore, for successful completion of the projects, we analyze technical challenges occurring in all phases of the project lifecycle, i.e. project preparation stage and development stage. And, we propose the effective solutions to overcome the issues. To verify identified issues and presented solutions, we present the result of applying the solutions to an IoT application development. Through the case study, we evaluate how reasonable the unconventional technical issues are generated and analyze effectiveness of applying the solutions to the application.

Keywords : Internet-of-Things, IoT Devices, IoT Application Development, Technical Issues, Solutions

IoT 애플리케이션 개발의 기술적 이슈 및 솔루션

신 동 하† · 한 승 호† · 라 현 정†† · 김 수 동†††

요 약

사물 인터넷(Internet-of-Things, IoT) 컴퓨팅은 무선 인터넷으로 다양한 디바이스를 연결하고 센서를 통해 획득한 사용자 주변 환경 정보를 이용하여 디바이스를 제어하는 여러 기술의 융합 기술이다. IoT 애플리케이션은 기존 소프트웨어와는 달리 다수 개의 IoT 디바이스와 협업을 통해 사용자에게 기능을 제공하고, 센서 네트워크, 통신 기술, 소프트웨어 공학 등 여러 기술들을 활용하여 설계된다. 그리고 최근에 소개된 신 기술이기 때문에, 대부분의 연구는 시작 단계에 있다. 이런 이유로, IoT 애플리케이션 개발 프로젝트는 기존의 소프트웨어 개발 프로젝트에서 관찰되지 않은 기술적 이슈들이 발생할 수 있고, 기존의 프로젝트 수행 가이드라인을 그대로 적용하는 것이 제한되어 성공적으로 프로젝트를 수행하는 데 어려움이 따른다. 따라서 본 논문에서는 IoT 애플리케이션을 효율적으로 개발하기 위해, 프로젝트 준비 및 계획 단계와 설계 및 개발 단계로 구분하여 각 단계별로 기술적 이슈를 나열하고 효과적인 솔루션을 제시하고자 한다. 또한 IoT 디바이스 중 AR.Drone과 Sphero Ball을 활용한 애플리케이션 개발에서 본 논문의 솔루션에 대한 적용 및 활용 사례를 보여줌으로써, 연구의 실효성을 검증한다.

키워드 : 사물인터넷, IoT 디바이스, IoT 애플리케이션 개발, 기술적 이슈, 솔루션

1. 서 론

사물 인터넷(Internet-of-Things, IoT) 컴퓨팅은 무선 인터넷으로 다양한 디바이스를 연결하고 센서를 통해 획득한 사용자 주변 환경 정보를 이용하여 디바이스를 제어하는 여러 기술의 융합 기술이다. IoT 컴퓨팅은 미국의 시장조사기관인 가트너에서 조사한 바에 따르면, 2020년에 인터넷 인구는 50억 명, 1000억 대의 인터넷 접속기기가 인터넷에 연결될 것으로 전망할 만큼 빠르게 성장할 것으로 주목되는 분야 중 하나이다[1-2].

IoT 컴퓨팅 기술의 발전을 위해서 IoT 컴퓨팅을 실현할 수 있는 여러 기술 및 기법들이 요구된다. 그러나 다음과 같은 이유들 때문에 IoT 환경에서 운영되는 애플리케이션의 설계 및 개발에 어려움이 따른다. 첫째, IoT 애플리케이션은

※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2012R1A1B3004130).
† 준 회 원 : 숭실대학교 컴퓨터학부 학사과정
†† 종신회원 : (주)스마티랩 대표
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
Manuscript Received : October 06, 2014
First Revision : December 29, 2014
Accepted : February 06, 2015
* Corresponding Author : Seung Ho Han(gks8030@gmail.com)

네트워크에 연결된 여러 IoT 디바이스들로부터 환경 정보를 획득하고 사용자에게 유용한 형태의 서비스를 제공하기 위해 관련 디바이스를 액추에이션(Actuation)한다. 즉, IoT 애플리케이션은 소프트웨어 기능 모듈로만 구성된 기존 소프트웨어와는 달리 IoT 디바이스들 간의 협업을 통해 사용자에게 서비스를 제공한다는 특징을 가지고 있다. 둘째, IoT 컴퓨팅 환경을 구성하거나 IoT 애플리케이션을 개발하기 위해서 통신, 센서, 소프트웨어 공학 등 여러 기술들이 집약되어 개발 난도가 매우 높아진다. 셋째, IoT 기술은 최근에 소개된 신기술이기 때문에, 이와 관련된 연구는 시작 단계에 불과하고 대다수의 연구는 현재 진행 중이어서 관련 서적, 기법 등이 부족하므로 IoT 애플리케이션을 성공적으로 개발하는 것이 어렵고, 기존의 프로젝트 수행 가이드라인을 그대로 적용하기에 제한이 따른다.

본 논문에서는 IoT 애플리케이션 개발 프로젝트를 성공적으로 수행하기 위해, 프로젝트를 수행하는 전체 기간 동안 발생할 수 있는 이슈를 식별하고, 이를 해결할 수 있는 지침 및 솔루션을 제안한다. 프로젝트 수행 기간은 프로젝트 준비 및 계획 단계와 설계 및 개발 단계로 구분된다[3]. 성공적인 프로젝트 수행을 위해서는 프로젝트 준비부터 애플리케이션 운영 단계에 이르기까지 전체 단계에 적용될 수 있는 가이드라인이 필요하다. 그리고 개발될 애플리케이션의 특징을 잘 반영하는 구체적인 가이드라인이 필요하다. 예를 들어, IoT 애플리케이션 개발은 IoT 디바이스의 종류 특성, 네트워크 등을 고려한 개발 지침이 필요하다.

이를 위해, 먼저 IoT 개발 프로젝트의 각 단계에서 발생할 수 있고, IoT 애플리케이션에서만 관찰될 수 있는 이슈들을 나열한다. 그리고 도출된 이슈들을 해결하기 위해, 적용할 수 있는 효과적인 솔루션을 제공한다. 제시된 솔루션은 네트워크, 보안 등 IoT 컴퓨팅 관련 여러 기술 중 소프트웨어를 체계적으로 개발할 수 있는 소프트웨어 공학을 기반으로 한다.

3절에서는 프로젝트 준비 및 계획 단계에서 관찰될 수 있는 IoT 프로젝트 프로세스 적용 관련 이슈들을 나열하고, 이를 위한 솔루션을 제시한다. 4절에서는 IoT 애플리케이션의 설계와 개발 및 운영 시 직면할 수 있는 이슈 및 솔루션을 제안한다. 마지막으로, 5절에서는 앞서 제시된 솔루션을 바탕으로 이동 기반의 IoT 디바이스 중 AR.Drone과 Sphero Ball을 활용한 사례연구를 통해 이슈들이 해결됨을 보이고, 결과를 비교하여 본 논문이 제안한 기법의 실효성을 검증한다. 제시된 이슈들에 대한 해결책을 이용하면, IoT 애플리케이션 개발 시 발생할 수 있는 이슈들에 대해 효과적으로 해결할 수 있고, 높은 품질의 IoT 애플리케이션을 적시에 개발할 수 있을 것으로 기대한다.

2. 관련 연구

IoT 개념은 최근에 소개된 개념이기 때문에 앞으로의 연구 방향을 제시하기 위해, IoT 개념을 이용하여 애플리케이션을 개발할 때 고려해야 하는 여러 연구 이슈를 다룬 논문들이 게재되었다.

Miorandi의 연구는 먼저 IoT 개념을 구현할 때 시스템 수준에서 제공되어야 하는 주요 기능과 이와 관련된 기술적 이슈를 나열하였다[4]. 본 논문에서 제기된 이슈는 디바이스 식별 관련 이슈, 분산된 디바이스 관리 관련 이슈, 분산된 데이터 처리 관련 이슈, 보안 관련 이슈 등이다. 그리고 기존 연구 기법들이 해당 이슈들을 어느 정도 해결할 수 있는지를 기술하였다. 이 연구는 IoT 애플리케이션 개발 시 발생할 수 있는 이슈를 위주로 언급하였지만, 이슈에 대한 구체적인 솔루션을 제시하고 있지 않다.

Chen의 연구에서는 디바이스 초기화부터 사용자에게 유용한 서비스 제공에 이르기까지 IoT 애플리케이션이 수행해야 하는 전반적인 기능과 이와 관련된 이슈를 제기하였다 [5]. 제기된 이슈에는 디바이스 설치 및 관리 노력 최소화, 디바이스의 제한된 전원, 수많은 디바이스 연결 관련 문제, 보안 및 프라이버시 이슈, 방대한 양의 효율적인 데이터 처리 이슈, 인터페이스 표준화 등이 있다. 그리고 제기된 이슈들에 대한 해결책을 언급하였다. 이 연구는 개발 단계에서 직면할 수 있는 이슈에 국한하여 해결책을 제시하였기 때문에, 이슈에 대한 직접적인 해결 방안 제시가 부족하다. 또한 소프트웨어뿐 아니라 하드웨어 관련 이슈까지 제기하고 있어, 소프트웨어 공학 관점에서 솔루션을 제기하는 데 한계점이 있다.

Chaqfeh의 연구는 이기종 디바이스 간의 상호연동성을 지원하는 IoT 미들웨어 개발에 관련된 다양한 이슈들과 솔루션을 제시하고 있다[6]. 이기종 디바이스에 대한 높은 상호 연동성, 많은 디바이스를 관리할 수 있는 높은 확장성, 잘 설계된 API, 이동성 있는 디바이스 관리 및 요청에 맞게 적절한 디바이스 및 서비스 바인딩 등을 이슈로 도출하였다. 그리고 시멘틱 웹, 센서 네트워크, 로봇 공학에서 사용되는 솔루션이 해당 이슈들을 효과적으로 해결할 수 있는지 분석하였다. 본 연구에서는 IoT 애플리케이션 개발보다는 IoT 미들웨어 개발 관련 이슈들에 초점이 맞추어져 있어, IoT 애플리케이션 개발과 직접적인 관련이 없는 이슈들도 포함되어있다. 또한 제기된 이슈에 대한 구체적인 해결 방안 제시가 부족하다.

Khan의 연구는 IoT 애플리케이션을 위한 표준 아키텍처를 제시하고, IoT 애플리케이션 개발과 관련한 기술적 이슈를 제기하였다[7]. 제기된 이슈에는 IoT 디바이스의 네이밍 관리, 이질적인 디바이스 간의 상호 연동성 및 표준화, 정보 보안, 데이터 비밀 보장 및 암호화, 네트워크 보안, IoT 디바이스의 전원 제약성 등이 있다. 본 논문에서 제기된 이슈 역시 IoT 애플리케이션 개발에 국한된 것들이며, 개발 단계 중, 특히 IoT 아키텍처 설계 시에 직면할 수 있는 이슈들 위주로 나열되어있다. 그리고 제기된 이슈들의 솔루션에 대한 언급은 없다.

La의 연구는 IoT 애플리케이션을 개발할 때 발생할 수 있는 비전형적인 기술적 이슈를 나열하고 이에 대한 해결책을 제시하고 있다[8]. 특히, 기존의 소프트웨어 시스템 개발

시 발생하지 않았던 IoT 디바이스 간의 이질성, 자체 이동성 그리고 물리적 제약성과 같은 비전형적인 이슈에 대한 솔루션을 제시하여 IoT 프레임워크 개발을 위한 설계의 기반이 되고 효율적인 IoT 애플리케이션 개발을 돕는다. 그러나 IoT 애플리케이션 개발 시 직면할 수 있는 IoT 디바이스 관련 이슈들에 국한되어있어, 성공적인 프로젝트 수행을 위해 전체 생명주기를 고려한 이슈 제기가 필요하다.

3. 프로세스 준비와 계획 관련 이슈 및 솔루션

3.1 높은 기술적 난도를 반영한 프로세스 선정 및 적용

이슈 : IoT 애플리케이션 개발은 기존의 소프트웨어 개발보다 기술적 난도가 높으며, 이에 따라 개발 시 이를 고려한 프로세스 선정이 필요하다. 기술적 난도를 높게 만드는 다양한 원인은 다음과 같다.

- **IoT 디바이스의 증가 :** IoT에 대한 관심이 커지고, IoT 애플리케이션 개발의 활성화에 따라 다양한 종류의 디바이스가 소개되고, 이로 인한 디바이스 분석 및 활용에 많은 비용이 든다.
- **관련된 연구의 부족 :** 최근 IoT에 대한 연구가 활발히 진행되고 있지만, IoT 디바이스의 계속적인 증가로 관련된 연구 정보가 부족하다. 따라서 IoT 애플리케이션 개발 시 기존 소프트웨어 개발과 다르게 프로세스를 그대로 적용하기에 높은 난도를 가진다.
- **높은 복합 기술의 필요 :** IoT 소프트웨어는 소프트웨어, 하드웨어, 네트워크 등 다양한 분야의 기술적 복합체이기 때문에 높은 복합 기술이 필요하며, 이에 따라 개발 복잡도가 증가한다.

제시된 원인들로 IoT 소프트웨어 개발은 높은 기술적 난도를 가지며, 이에 따라 개발 비용 증가와 품질 저하 등 다양한 문제가 발생한다. 따라서 이러한 문제를 줄이기 위한 적절한 프로세스 선정 및 적용이 필요하다.

솔루션: 높은 기술적 난도에 대한 효과적인 개발 프로세스는 Agile 방법론이다. 짧은 스프린트를 반복하여 점진적으로 개발함으로써 IoT 디바이스의 증가와 관련 연구의 부족으로 인해 발생하는 문제를 효과적이고 빠르게 대처할 수 있다 [9]. 즉, 지속적인 요구사항 변경을 수용하여 변화에 대한 대처를 계획의 수행보다 우선 시 하기 때문에 IoT 애플리케이션과 같이 높은 기술적 난도를 가지는 소프트웨어 개발에 효과적으로 적용될 수 있다.

3.2 실현되기 어려운 클라이언트의 요구사항

이슈 : 다양한 IoT 디바이스의 증가로 IoT 애플리케이션 개발 진행 시 개발자 또는 클라이언트가 대상 디바이스에 대한 지식이 부족할 수 있다. 이에 따라 클라이언트는 디바이스 API가 제공하지 않는 기능성이나 하드웨어 자체가 제공하지 않는 기능성 등 실현되기 어려운 요구사항을 제시할 수 있으며, 개발자는 클라이언트 요구사항에 대한 정확한 분석이나 실현 가능성 등을 판단하기 어렵다. 예를 들면, Sphero Ball은 구 형태의 지상 디바이스이지만 장애물 감지 센서가 장착되어있지 않아 자체적인 장애물 우회가 불가능하다. 그러나 클라이언트는 자체적인 장애물 우회 기능성을 요구할 수 있으며, 만약 개발자가 디바이스에 대한 지식이 부족할 경우, 이러한 요구사항을 수락할 수도 있다. 개발자나 클라이언트의 디바이스 지식 부족은 실현되기 어려운 요구사항을 만들어내며, 이에 따라 개발 중 여러 문제 발생이나 전체적인 개발에 영향을 미친다.

솔루션 : 실현되기 어려운 요구사항을 효과적으로 대처하기 위해서 철저한 디바이스 분석을 기반으로 클라이언트를 설득해야 한다.

먼저, 개발 팀이 협상에 필요한 정도의 IoT 디바이스 관련 기술적 지식을 학습하기 위한 기술 분석 기간을 확보한다. 기간은 개발 팀의 동일 디바이스에 대한 개발 경험 여부나 요구사항과 디바이스 사이의 의존도, 제공되는 API 기능 및 활용된 샘플 코드 등을 고려해서 정한다. 협상 전 기술 분석 기간은 짧게 정하는 것이 중요하며, 최대 일주일을 넘지 않도록 한다.

그리고 IoT 디바이스의 개발에서 필요로 하는 기술적 지식을 습득하여, 이를 문서화해야 한다. 특히 요구사항에서 요구하는 IoT 디바이스 기능 및 특성, 환경 등과 비교 분석해야 한다. 예를 들면, 지원 언어, 네트워크 프로토콜, 디바이스의 하드웨어적 사양, 장착된 센서의 종류, 수행할 수 있는 기능, 지원되는 스마트 기기 등이 될 수 있다. Table 1은 요구사항 기능별 가용 IoT 디바이스, API 지원 여부, 필요 API 함수 등에 대한 표이다. 여기에서 'X'는 API에서 지원되지 않는 기능을 나타내며, '△'는 API가 존재하되 기능을 완벽히 지원할 수 없음을 의미한다. Table 1과 같은 API 분석 템플릿 문서 작성 시, API를 완벽히 파악할 필요는 없으며, 개략적으로 API 분석을 통해 실현되기 어려운 요구사항의 유무를 파악할 수 있다.

Table 1. A Template for Analyzing IoT APIs

Requirement Items	IoT Device	Supported by APIs?	Applicable APIs	Description
Auto Crusing	Sphero Ball		drive (heading, speed)	Move device
Avoid obstacles	Sphero Ball	X	-	-
Return	Sphero Ball	△	drive (heading, speed)	Move device
…	…	…	…	…

마지막으로, 대상 디바이스의 기능, 특성 및 역량 분석 데이터와 API와 요구사항의 연관성 분석 데이터, 그리고 요구사항 실현 시 대상 IoT 디바이스의 제약사항 분석 데이터를

기반으로 클라이언트와 협상을 진행한다.

이와 같은 지침을 통해, 개발자는 개발 전 IoT 디바이스에 대한 지식 습득을 통한 실현되기 어려운 요구사항을 가려낼 수 있으며, 대상 IoT 디바이스 분석 데이터를 기반으로 클라이언트와 협상을 진행함으로써 개발이 어렵다고 판단된 요구사항을 정제할 수 있다.

3.3 프로젝트 태스크의 정확한 도출 난이도

이슈 : 프로젝트 태스크 선정은 요구 기능 분할이나 필요 지식 습득 등 전체 프로젝트 진행에 필요한 일들을 프로젝트 초기에 태스크 단위로 도출하는 것이다. 프로젝트 태스크 선정 방법은 여러 소프트웨어 개발 모델에 따라 약간씩 상이한 방법을 갖는데, 그중 Agile 모델에서의 프로젝트 태스크 선정 방법은 PBI(Product Backlog Item)를 선정하는 단계에 해당된다. Agile 모델의 특징상 점진적으로 PBI를 정제해나갈 수는 있지만, 초기의 PBI 선정은 전체 개발 효율에 많은 영향을 미치며 이에 따라 정확한 PBI 선정이 요구된다. 그러나 Agile에서의 IoT 애플리케이션 개발은 일반 소프트웨어 개발과는 다르게 디바이스를 사용하며, 대부분의 기능이 디바이스에 대한 이동을 기반으로 수행되기 때문에 각 기능 사이의 의존도가 높아 기능의 모듈화가 어렵다. 이에 따라 PBI 선정 시, 항목 분할과 항목 사이 우선순위를 결정하기 어렵다. 이러한 원인으로 인하여 개발 중 추가적인 비용과 노력이 발생하여 Agile 모델의 장점을 살리기 어렵다. 또한, 프로젝트 기간이 고정된 경우 기간 부족 문제로 인한 일부 기능 포기나 무리한 요구사항 변경 등에 따른 프로젝트 품질 저하 문제가 발생한다.

솔루션 : IoT 애플리케이션의 PBI 선정은 각 기능 사이의 선행 관계와 우선순위를 고려하여 선정해야 하며, 다음의 4단계를 통해 정확한 PBI를 선정하는 방법을 제시한다.

첫 번째 단계에서는 Fig. 1과 같이 요구사항으로부터 기능을 추출하고 유사한 기능을 그룹핑(Grouping) 한다. 기능 간 그룹핑은 서로 다른 기능 사이에 유사한 기능성을 갖거나 중복되는 API를 사용하는 등의 경우에 수행한다. 예를 들면, AR.Drone의 기능 중 자동 비행 모드나 사용자 조종 모드, 복귀 비행 기능 등은 AR.Drone이 이동한다는 유사한 기능성을 가지며, 움직임 제어와 관련된 API를 공통적으로 사용하므로 하나의 그룹이 될 수 있다.

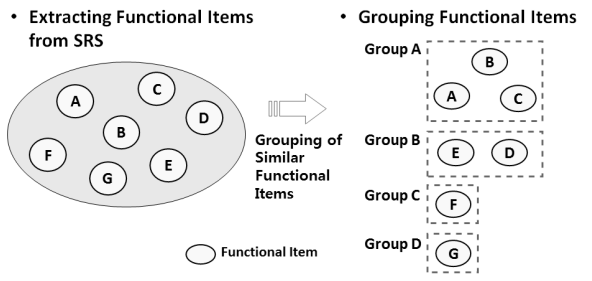


Fig. 1. Extracting Functional Items and Grouping Them

두 번째 단계에서는 Fig. 2와 같이 그룹 안의 기능을 PBI 단위로 분할하고, 각 기능과 분할된 항목의 우선순위를 할당한다. 항목 분할은 기능 중 한 스프린트 안에 개발하기 힘든 경우나 여러 작은 기능들이 하나의 기능으로 합쳐진 경우 수행한다. 또한, 분할 된 항목은 결과물이 나올 수 있는 패키지 단위이어야 하고, 항목 사이의 의존도가 적어야 하며, 각 기능과 항목의 우선순위는 클라이언트가 중요시하는 기능이나 먼저 개발되어야 되는 기능 등에 대해 낮은 값을 할당한다. 예를 들면, AR.Drone의 자동 비행 모드 기능은 제어와 시퀀스 생성 등의 PBI 단위로 분할이 가능하며, 두 항목 중 AR.Drone제어 항목이 먼저 개발되어야 하므로 낮은 값을 할당해야 한다.

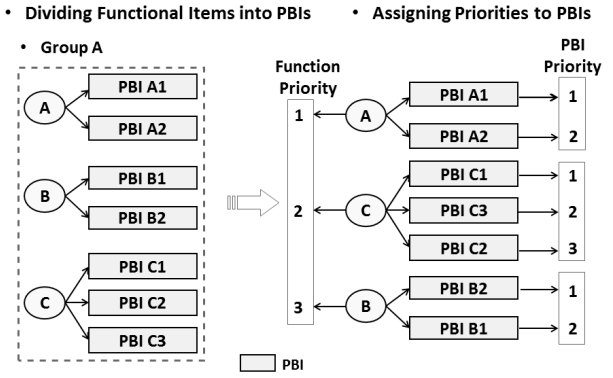


Fig. 2. Extracting PBIs from Requirement Items and Assigning Priorities to PBIs

세 번째 단계에서는 각 PBI에 대한 최종 우선순위를 계산하고 나열한 후 Equation (1)을 이용하여 한 그룹에서 FP가 낮은 기능 순으로 계산한다. Equation (1)의 FP(Function Priority)는 기능 간 우선순위이며, IP(Item Priority)는 한 기능 안에서의 PBI 간 우선순위, T(Item name)는 PBI에 대한 최종 우선순위 값, MIP는 이전까지 구한 T(Item name) 중 최댓값이다.

$$FP = 1, T(ItemName) = IP$$
$$FP > 1, T(ItemName) = MIP + IP \quad (1)$$

네 번째 단계에서는 중복 기능성을 갖는 PBI를 병합하고 모든 그룹에 대해 위와 같은 방법을 반복한다. 결합된 PBI는 PBI들 중 우선순위가 가장 높았던 PBI의 위치로 배정한다. 그리고 첫 번째 단계에서 식별한 모든 그룹에 대해 반복적으로 방법을 수행하여 최종적인 제품 백로그를 만든다. 이때, 항목 선정은 각 그룹에서 구한 PBI 간의 최종 우선순위를 고려하여 선정한다.

Fig. 3은 세 번째와 네 번째 단계가 적용되는 과정을 보여준다. T(PBI A1)는 FP가 1이고, IP가 1이므로 값은 1이 된다. PBI C3의 경우, T(PBI C3)값은 FP가 2이고 MIP가 2이며, IP가 2이므로 4가 된다. 이와 같이 반복하여 모든 PBI에 대해 수행하면 Fig. 3의 왼쪽과 같은 그림이 나온다. 또

한 PBI A2와 PIB B2처럼 중복되는 PBI의 경우 PBI A2B2 처럼 하나로 합친다.

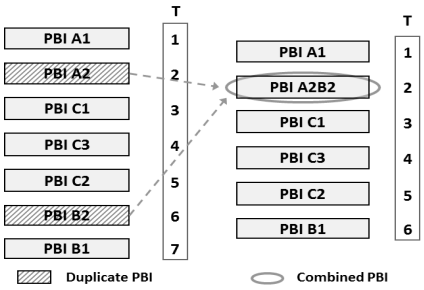


Fig. 3. Combining Duplicated PBIs

위와 같은 단계를 수행하면 IoT 소프트웨어 개발에서 PBI 선정 시 각 기능 간의 의존도를 고려한 모듈화가 가능하고, 이에 따라 프로젝트 태스크 작성시 발생하는 어려움을 해결할 수 있으며, 정확한 PBI를 선정할 수 있다.

3.4 IoT 프로젝트의 복잡도로 인한 계획의 어려움

이슈 : 계획 단계에서 팀원의 역량을 파악해 적절한 태스크를 분배하는 것은 개발 가능성 유무를 판단할 수 있게 해주고, 개발이 지연되지 않도록 도와준다. 그러나 IoT 애플리케이션 개발 진행 시 일반 소프트웨어 개발과는 다르게 다음과 같은 원인들로 인해 계획 단계가 복잡해진다.

- **새로운 패러다임의 지식 부재 :** IoT 같은 새로운 패러다임에 대한 프로젝트 진행은 개발자의 지식이 부족하여 습득하는 데 시간을 소모하여 실제 가용할 수 있는 시간보다 비용이 커진다. 이로 인해 개발 기간이 지연되고 추가적인 비용이나 고객의 만족도가 낮아지게 된다.
- **높은 기술 의존도에 따른 객관적 역량 측정의 어려움 :** IoT 기술에 대한 경험의 유무는 실제 역량을 측정하는 데 영향을 주어 정확한 측정에 어려움이 따른다. 이로 인해, 실제 역량보다 지나친 태스크로 개발 지연 및 잘못된 계획이 될 수 있다.

솔루션 : Agile에서의 태스크는 개인이 자발적으로 분배, 할당한다. 그러나 IoT 애플리케이션 개발은 다양한 분야에 대한 전문적 기술을 요구하기 때문에 개인 역량에 대한 객관적인 수치를 측정하여 할당하는 것이 효과적이다. 즉, 팀원들은 개인이 자신 있는 태스크를 할당하지만 역량에 따라서 할당방법을 조정할 필요가 있어 다음과 같은 3단계를 통해 효율적인 개인 태스크 할당 방법을 제시한다.

먼저 프로젝트 특성에 맞춰 개인이 역량을 측정한다. 예를 들어, IoT 애플리케이션 개발은 IoT 디바이스, 네트워킹, 소프트웨어 등 필요로 하는 기술이 다양하므로 이에 따른 태스크에 대한 개인 역량 측정이 필요하다. Table 2는 프로젝트 특성에 따른 개인 역량 측정 템플릿이다.

개인 역량을 구하는 방법은 첫 번째 항목의 개발 프로젝트에서 사용할 기술에 대한 항목과 해당하는 태스크를 나열하고, 두 번째 항목에서 자신에게 상대적인 점수를 0~5 사이에서 점수 측정을 한다. 세 번째 항목에서는 프로젝트에서 중요도에 따라 가중치를 적용하여 점수와 가중치를 곱해서 나온 점수들을 모두 더하여 개인 역량(IR, Inner Resource) 점수를 측정한다.

Table 2. Example of Measuring Personal Technical Capabilities for IoT Application Development

Item to Measure Capability for IoT Project evelopment		Score	Weights	Total
IoT Device	Motion Controlling	2	X0.4	0.8
	Handling Device Using Sensor	1		0.4
Network	Data Transmission	0	X0.3	0
	Connecting Device	1		0.3
Programming Language	UI Design	5	X0.3	1.5
	Displaying Device State	4		1.2
	Data Acquisition Algorithm	5		1.5
Total				5.7

두 번째 단계에서는 개인이 측정한 역량에 따라 태스크를 할당한다. Table 3은 측정된 팀원들의 역량과 자신의 총 점수에서 각 태스크별 점수를 나눈 백분율을 보여준다. 이는 팀원들이 자신의 역량 중 얼마나 많은 부분을 각 태스크 수행에 활용할 수 있는지를 보여준다. 여기서 태스크 이름 옆에는 각 태스트의 복잡도를 작성하며, 이는 기능함수 등을 활용하여 측정한다. 예를 들어, A는 자신의 IR 점수가 5.7이며, 이 중 Task #1을 수행하는 데 자신의 역량의 23%에 해당하는 1.2을 사용함을 의미한다.

Table 3. An Example of Assigning Tasks

Name(Total) / Task Name	A (5.7)	B (7.0)	C (9.0)
Task #1 (12)	1.2 (23%)	1.6 (23%)	1.6 (17%)
Task #2 (14)	0.4 (7%)	1.2 (17%)	2.0 (22%)
Task #3 (10)	0 (0%)	0.9 (12%)	0.6 (7%)
…	…	…	…

모든 팀원이 각자 작성한 것을 기반으로, 다음과 같은 규칙을 준수하여 태스크를 분배한다. 이때 복잡도가 높은 순서대로 먼저 태스크를 할당한다.

- **Rule 1 :** 각 태스크에 분배된 팀원 중 백분율이 가장 높은 팀원에게 할당한다.
- **Rule 2 :** 한 태스크에 백분율이 같은 팀원이 여러 명 있으면, 점수가 더 높은 팀원에게 할당한다.

먼저, 가장 복잡도가 높은 Task #2는 Rule 1에 따라 C에

게 할당한다. 그리고 두 번째로 복잡도가 높은 Task #1은 A와 B의 백분율 점수가 동일하므로, Rule 2에 따라 점수가 더 높은 C에게 할당을 하게 된다.

두 번째 단계까지 진행하여 태스크를 할당하면, 역량이 높은 팀원에게 태스크가 집중되는 문제가 발생한다. 그러므로 세 번째 단계에서는 팀원의 가용 스케줄을 고려하여 태스크 할당 결과를 조정한다. 팀원에게 집중된 태스크 중 복잡도가 낮은 태스크를 개인 역량 및 가용 스케줄을 고려하여 적당한 팀원에게 분배한다.

제시된 솔루션 절차에 따라, 개인의 역량과 태스크의 복잡도에 대한 객관적인 측정이 가능하여 역량에 따른 태스크 할당이 필요할 때 적절한 할당과 이에 따른 개발 가능성 판단을 도와준다. 또한, 역량이 부족한 태스크로 개발이 지연되는 문제를 해결할 수 있어 스케줄 조정에 용이할 수 있다.

4. 개발 관련 이슈 및 솔루션

4.1 요구사항을 충족하는 디바이스 API 미흡 및 부재

이슈 : IoT 애플리케이션 개발 시 요구사항에 따른 기능성은 벤더에서 제공되는 API를 사용함으로써 구현해야 한다. 그러나 고객의 요구사항을 충족시킬 수 있는 직접적인 API가 존재하지 않거나, API가 대부분의 기능을 제공하지만 일부 기능이 부족할 경우가 발생한다. 그리고 제공되는 API가 모든 기능성은 제공하지만 결과값의 정확도가 떨어지는 경우도 있다. 이런 원인들로 인하여 프로젝트의 산출물 품질이 저하되고, 최악의 경우 고객의 요구사항을 만족시키지 못하는 문제가 발생한다.

솔루션 : 앞서 제시된 원인들로 인하여 생긴 문제는 다음과 같은 방법으로 해결할 수 있다.

- **다른 디바이스 API 활용 :** API 미흡 및 부재의 경우, API A와 API B에서 필요한 기능성을 뽑아 요구사항에 필요한 새로운 기능성을 만들거나, 미흡한 부분을 다른 디바이스의 API C가 제공해주는 경우 이를 결합하여 해결할 수 있다. 이때 서로 다른 API를 통합해야 하므로 입력값이나 결과값의 데이터 타입과 같은 API 구조를 일치시켜야 하고, 기능성들이 동일한 자원에 접근할 경우 기능 간의 동기화 작업이 필요하다.
- **알고리즘 구현 :** 다른 대체 API가 없을 경우, 개발자는 기존 데이터를 이용한 새로운 알고리즘을 구현하여 필요한 기능성을 만들 수 있다. 이를 위해서 개발자에게는 사용하는 디바이스와 API 구조에 대해 높은 이해도가 요구된다. 예를 들어, 고객 요구사항이 Sphero Ball이 이동한 거리를 계산하여 처음 위치로 돌아오는 기능일 경우, 기존 API만으로는 정확한 이동거리를 파악하기 어렵다면 각속도 공식에 따른 알고리즘을 추가 구현하여 고객 요구사항을 만족시킬 수 있다.

이와 같은 방법으로 해결이 힘든 경우, 상황에 따라서는 두 가지 해결책을 함께 사용할 수 있다. 또한 디바이스의 기술적 한계 때문에 제시한 솔루션이 적용될 수 없는 경우 소프트웨어적으로 요구사항을 충족시키기 어려우며, 이러한 경우 그루밍 단계에서 클라이언트와 협상을 통해 요구사항을 구현 가능한 범위로 다시 재정의하는 방법이 있다.

4.2 IoT 디바이스 자체 이동성으로 인한 연결성 불안정

이슈 : IoT 디바이스는 블루투스, 와이파이, 지그비, RFID (Radio Frequency Indentification)와 같은 다양한 종류의 무선 네트워크로 연결된다. 사용자가 IoT 디바이스를 안정적으로 제어하기 위해서는 신뢰성 있고 효율적인 데이터 교환이 필요하지만, 무선 네트워크를 사용하기 때문에 거리에 따라 신호 강도가 비례할 수 있다. 또한, IoT 디바이스는 자체 이동성을 가지기 때문에 신호 강도가 빈번하게 변화하며 신뢰성 있는 데이터 교환이 어려워진다. 즉, 데이터가 지연되거나 손실될 수 있으며, 네트워크 연결이 끊기는 문제도 발생한다. 이에 따라, 디바이스의 손상이나 사고 위험 등의 문제가 발생할 수 있다.

솔루션 : IoT 디바이스의 자체 이동성으로 인한 데이터 지연 및 손실, 끊김 문제를 해결하기 위해서는 디바이스의 신호 강도를 확인하여 네트워크 끊김 문제를 사전 예방하거나 신뢰성 있는 데이터 전달을 위한 방법을 적용해야 한다. 이를 위해서 IoT 애플리케이션을 설계하기 위한 가이드라인을 제시한다.

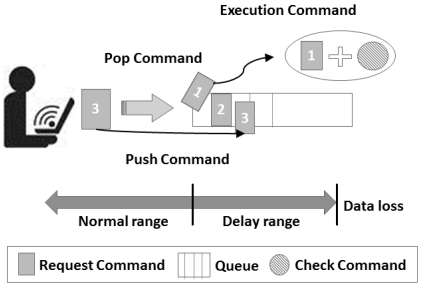


Fig. 4. Execution Commands and the Ranges of Timeout

IoT 애플리케이션은 데이터 지연 및 손실을 해결할 수 있도록 설계되어야 한다. 즉, 명령한 커맨드에 대한 재전송 및 백업이 가능해야 한다. IoT 애플리케이션에서 별도의 큐를 관리하여 명령한 기능에 대한 커맨드를 큐에 넣고, 순차적으로 꺼내 디바이스에게 전달해야 하며 요청 커맨드와 체크 커맨드를 합쳐서 실행 커맨드를 생성하여 디바이스에게 전달한다. 동시에, Fig. 4와 같이 응답 시간을 체크하면서 타임아웃 범위에 따라 계속 진행, 네트워크 신호 강도에 대한 모니터링, 디바이스 정지 중 하나를 수행한다.

응답 시간이 정상 범위일 경우, 큐에서 이전 요청 커맨드를 삭제하며, 다음 요청 커맨드를 계속 진행한다. 응답 시간이 지연 범위일 경우에는 애플리케이션상에서 네트워크 신호 강도 모니터링 기능을 수행시킨다. 이후, 지속적으로 지연 현상이 발생하거나 지연 시간이 길어질 경우 사용자의 판단에 따라 네트워크 연결 상태 불안정으로 판단하여 디바이스를 정지시킨다. 마지막으로 응답이 오지 않아 Timeout 범위를 초과할 경우, 데이터 손실로 판단하여 큐에서 이전

커맨드를 다시 수행시킨다.

제시된 솔루션 절차를 따르면, IoT 디바이스의 자체 이동성으로 인한 연결성 불안정으로 발생하는 데이터 지연 및 손실 문제에 대해 신뢰성 있는 데이터 전달을 보장할 수 있다. 또한, 네트워크 끊김 문제는 지연 발생 시 지속적인 신호 강도 모니터링과 응답시간을 측정하여 사전에 디바이스를 정지시켜 예방할 수 있도록 한다.

4.3 정교하지 않은 디바이스 제어

이슈 : IoT 디바이스는 자체 이동성을 가지고 있어 스스로 특정 위치로 이동할 수 있지만, 다음과 같은 원인으로 정교한 제어가 어렵다.

- **배터리 잔여량에 의존적인 디바이스 제어 :** 일부 디바이스는 배터리 잔여량이 동력장치에 영향을 주어 정교한 제어가 어려워진다. 예를 들면, AR.Drone은 잔여량이 30~20%일 때 플립 기능과 조율 기능 수행이 어렵고, 20~5%일 때는 이륙, 플립 조율 기능이 어렵다.
- **실행 환경에 의존적인 디바이스 제어 :** IoT 디바이스는 실행 환경을 파악하고, 환경에 맞도록 동작해야 한다. 그러나 대부분의 IoT 디바이스는 자체적으로 주위 환경을 파악할 수 있는 센서가 미흡하다. 따라서 자체적으로 동작할 때, 실행 환경에 따라 제어의 정교함이 달라질 수 있다[11]. 예를 들어, 지상 이동형 디바이스의 경우 실행하는 표면 상태의 기울기에 따라서 실행 결과가 달라질 수 있어 예상치 못한 결과가 발생할 수 있다. 또한, 관성이 발생할 가능성이 높아 속도를 제어하는 데 제한이 있어 방향 전환 시 사용자가 원하는 위치에 도달하지 못할 가능성이 크다.

솔루션 : 디바이스가 사용자의 의도대로 이동하기 위해서 다음과 같은 해결책을 제시한다. 배터리 잔여량에 대한 디바이스 의존도는 크게 두 가지 해결책이 있다. 첫 번째, 애플리케이션이 배터리 잔여량에 따라 사용자가 알 수 있도록 경고 메시지를 보내주는 수동적인 해결 방법이다. 두 번째는 배터리 잔여량에 상관없이 제한되는 기능을 해결하기 위해 잔여량에 따른 기능 수행도를 통계적으로 파악하여 기존 수행도에 맞출 수 있도록 계산하여 명령을 하는 방법이다. 예를 들면, 배터리가 30% 이하일 때 수행하는 기능이 기존 수행 능력의 80%라고 가정하면, 100%로 맞추기 위해 125%의 수행 기능을 명령해야 한다.

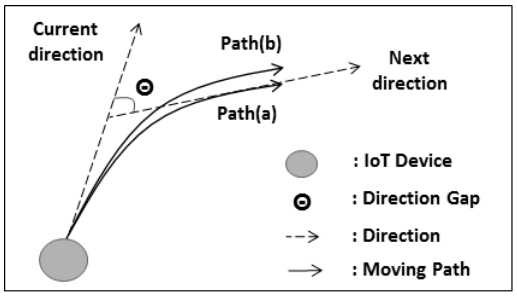


Fig. 5. Adjusting Movement by Considering Inertia

실행 환경에 의존적인 디바이스 제어는 상황에 따라 해결 방법이 다르다. 실행 환경에 맞는 센싱 기능을 추가하거나 주위 환경에 따라 자체적으로 제어를 할 수 있는 알고리즘을 적용한다. 예를 들면, Fig. 5과 같이 관성에 의해 디바이스 제어의 정교함이 영향을 받을 경우 속도 제어를 통해 해결할 수 있다. Path(a)는 운용자가 원하는 최적의 방향 전환의 경로이며, Path(b)는 관성의 영향을 받은 방향 전환의 경로이다. 두 경로의 차이를 줄이기 위해서, 방향 전환 전 IoT 디바이스가 가지고 있는 속도를 관성의 영향이 최소화되는 최적의 속도까지 줄여준다. 최적 속도는 디바이스 종류 및 Θ값에 따라 달라지며, 실험을 통해서 얻을 수 있다.

4.4 디바이스 제어권 제약 및 상실

이슈 : IoT 디바이스에 대한 제어권 유지는 사용자의 만족도와 애플리케이션의 기능성과 직결되므로 반드시 고려해야 할 기능이다. 그러나 애플리케이션 실행 중 다음과 같은 원인들로 디바이스 제어권의 상실 가능성이 발생한다.

- **네트워크 범위의 한계 :** 디바이스 자체에서 제공하는 네트워크는 사용하는 통신 방식에 따라 범위가 달라질 수 있다. 그러나 무선 통신으로 제어되는 IoT 디바이스는 통신 범위에 한계가 있어 제공되는 범위를 벗어나면 사용자는 디바이스 제어권을 상실하여 디바이스의 손상이나 사고 등의 위험 문제가 발생한다.
- **가시성을 벗어난 디바이스 제어의 한계 :** 환경이나 지형에 따라 디바이스가 사용자의 가시성을 벗어나면 제어에 한계가 있다. 예를 들어, 장애물이 많은 환경의 경우 사용자가 디바이스 제어 중 장애물에 의해 보이지 않는 지역에서 이동하는 명령을 하게 되면, 디바이스의 충돌이 발생할 수 있고 사용자가 의도한 이동이 어려워 기능성이 하락한다.

솔루션 : IoT 디바이스의 제어권을 유지할 수 있도록 다음의 솔루션을 고려하여 IoT 애플리케이션을 설계한다.

네트워크 범위의 한계가 있는 디바이스의 경우 이동 중 거리를 측정하며 사용 네트워크의 범위를 벗어나지 않도록 제어하는 알고리즘을 적용한다. 즉, 디바이스의 네트워크 범위가 100m라 가정했을 때, 사용자가 제어하는 노드와 디바이스의 거리 간격을 100m가 넘지 않도록 보이지 않는 경계선을 만든다. 디바이스는 측정된 거리를 기반으로 운용 중 경계선을 넘어가는 명령을 수행하지 않거나 사용자의 의도에 제한적인 이동을 수행한다. 네트워크 범위의 한계를 구할 수 없는 경우, 미리 네트워크 신호를 측정하며 제한된 범위를 지정해야 한다.

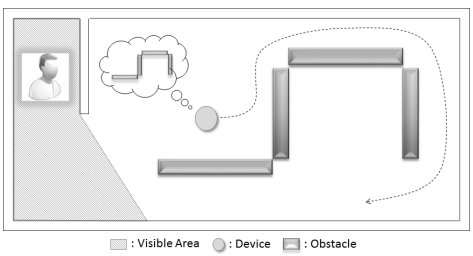


Fig. 6. Map-based IoT Device Movement

디바이스가 사용자의 가시성을 벗어나는 경우 애플리케이션 혹은 디바이스 자체를 판단하기 어려워 어느 정도 사용자의 판단이 개입되어야 한다. 그리고 미리 센싱 되어있는 맵 혹은 위치 정보를 이용하여 가시성을 벗어나도 자체적으로 이동할 수 있도록 개발한다. 즉, 가시성을 벗어나 제어권을 상실한다 해도 위치를 매핑하면서 디바이스의 충돌이나 손상의 위험을 줄일 수 있다[12]. 예를 들면, 디바이스는 Fig. 6과 같이 장애물이 많은 환경에서 사용자의 가시성을 벗어나면 사전에 파악되어있는 위치를 토대로 자체적인 이동이 가능해진다.

제시된 솔루션을 적용하면, 애플리케이션은 네트워크의 범위를 자체적으로 벗어나지 않도록 제어해 사용자가 예상치 못하게 네트워크에서 벗어나 제어권을 상실하게 되는 문제를 해결해준다. 또한, 가시성을 벗어난 지역에서 위치 정보를 이용할 수 없는 경우는 제한적이지만, 사전에 파악한 위치 정보 혹은 맵을 이용하여 자체적인 이동을 가능하게 해주면 디바이스의 안정성을 더해주며 사용자가 위치할 수 없는 위험한 지역에서도 유용하게 사용될 수 있다.

4.5 테스트 환경 구축 및 테스트 수행의 어려움

이슈 : IoT 디바이스 제어는 디바이스가 직접 기능을 수행하고 동작하기 때문에 개발에서 구현한 내용과는 다르게 동작할 수 있다. 따라서 개발 과정 중 테스트 단계가 매우 중요하며, 정확한 디바이스 동작 확인을 위해 테스트 구현과 병행되어야 한다. 그러나 IoT 디바이스 테스트는 다음과 같은 원인으로 인하여 어려움이 있다.

- **IoT 디바이스에 따른 테스트 환경 제약 :** 실외 환경에서 수행하는 테스트는 비, 바람, 기온 등과 같은 다양한 요인들의 영향으로 제약이 따른다[10].
- **테스트 중 발생한 오류의 원인 파악의 어려움 :** IoT 애플리케이션을 활용하여 디바이스와 통신하는 경우, 테스트 중 발생하는 오류의 원인은 IoT 디바이스 한계, IoT 디바이스 자체의 문제점, 애플리케이션 구현의 문제 등으로 다양하다. 따라서 문제점을 정확하게 파악하는 데 어려움이 있다.
- **디바이스 문제 발생 예측의 어려움 :** 테스트 수행 중 예상치 못한 외부충격으로 인한 디바이스 손상, 애플리케이션을 이용한 제어가 불가능한 상황 등이 발생 가능하다. 디바이스의 구동에 문제점이 발생하는 경우 모든 애플리케이션 개발 일정에 영향을 주고, 프로젝트 수행에 추가적인 비용이 발생한다.

솔루션 : 테스트 작업수행 시 발생 가능한 문제점들을 해결하기 위한 두 가지 솔루션을 제시한다. 먼저, IoT 디바이스에 따른 테스트 환경 제약성을 줄이기 위해, 테스트 환경을 고려하여 유연하게 테스트 수행 일정을 계획한다.

테스트 수행 일정 계획 시 다음과 같은 고려사항을 고려하여 일정을 계획해야 한다. 먼저, 선행되어야 하는 태스크의 우선순위를 정해야 하기 때문에 변경하는 태스크들의 선행도를 고려해야 한다. 두 번째로, 태스크 변경 시 수행시간을 고려해 전체 스프린트에 영향을 주지 않아야 한다. 마지막으로, 유동적으로 계획을 해야 전체 프로젝트 일정 관리가 편하기 때문에 계획은 한 스프린트로 한정한다. 이러한 고려사항을 바탕으로 다음의 2단계 방법을 진행하여 개발 전 테스트 환경에 맞는 테스트 수행 일정을 계획한다.

첫 번째 단계에서는 디바이스 특성을 고려한 테스트 환경을 선정해야 한다.실제 구동 환경과 유사한 환경에서 이루어진 테스트가 아니라면 신뢰성을 보장할 수 없다. 디바이스의 테스트 환경은 특성에 따라 실내/실외, 이동성 유/무, 지상/공중, 기후요인의 영향 여부, 통신거리 등으로 다양하며, 이러한 환경을 갖춘 테스트 장소를 사전에 파악해야 한다.

두 번째 단계에서는 계획된 태스크와 테스트 수행 시의 환경을 고려한 유동적인 테스트 일정을 계획한다. Fig. 7의 왼쪽 표는 계획된 태스크와 테스트 시 요구되는 환경이다. 오른쪽은 각 태스크의 테스트 수행 시 예상되는 테스트 환경이다. 태스크 2는 실외에서 테스트를 수행해야 하지만, 비가 올 것으로 예상된다. 이는 각 태스크의 수행시간을 고려하여 태스크 3, 4와 일정을 바꿀 수 있다. 단, 태스크 교환 시, 제시한 고려사항들을 적용한 상태에서 진행되어야 한다.

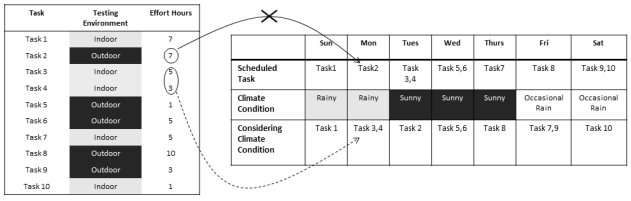


Fig. 7. An Example of Modifying a Test Schedule

소프트웨어 개발에서 테스트 시, 테스트하려는 기능과 의존도를 갖는 기능이 구현되지 않았거나, 계획에 따른 테스트 수행이 어려운 경우 스텁을 이용한다. 그중, IoT 애플리케이션 개발에서의 테스트는 IoT 애플리케이션의 특성상, 기능 간 의존도나 선행관계가 높기 때문에 전자의 경우보다는 후자의 상황이 많이 발생하며, 이 경우 스텁을 Controller나 Data Manager같이 나눠 사용하는 것이 효과적이다. 그러나 IoT 애플리케이션은 테스트의 결과가 디바이스의 움직임으로 나타나기 때문에 단순히 스텁을 이용한 테스트 수행 결과는 높은 신뢰도를 얻기 어렵다. 이에 따라 기능 구현 시, 해당 기능의 수치화된 값을 미리 측정해놓고, 스텁을 이용한 테스트 수행 시에 테스트 결과값과 비교하여 스텁 테스트의 신뢰도를 높일 수 있다.

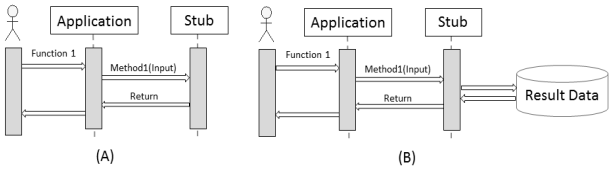


Fig. 8. Figuring Out Causes for Fault Occurrence with Stubs

Fig. 8의 (A)는 일반적인 스텁 테스트 수행 그림이고, (B)

는 미리 측정된 수치를 이용하는 스텁 테스트 수행 그림이다. (A)에 비해 (B)의 경우, 스텁 단계에서 미리 저장된 수치와 비교하여 그 결과를 개발자에게 제공한다. 이에 따라 스텁 테스트의 결과가 측정해놓은 수치와 비슷한 경우, 결과에 대한 신뢰도가 높다고 판단할 수 있다. 만약 테스트 시 문제가 발생하여, 문제의 원인을 찾을 수 없는 경우에도 스텁 테스트값과 측정해놓은 수치를 비교하여, 값의 차이가 날 경우 소프트웨어적 문제, 값의 차이가 적을 경우 하드웨어적 문제로 판단할 수 있다.

5. 사례연구 도메인

5.1 사례연구 도메인

제안한 솔루션의 실효성을 검증하고 이슈 해결에 효과가 있음을 증명하기 위해, 사례연구를 수행하였다.

1) 프로젝트 팀의 특징

개발 팀은 3명으로 구성되어 있으며, Agile 방법론을 적용해 3주씩 두 번의 스프린트를 진행하였다. Agile 관리를 위해 Ice Scrum 툴을 사용하였으며 요구사항 분석, 태스크 선정, 설계, 구현, 테스트 단계를 거쳐 안드로이드용 IoT 디바이스 컨트롤 애플리케이션을 개발했다. 참가자는 모두 IoT 개념이 생소한 상태였고 Agile 방법론에 대한 이론만 숙지한 상태로 개발에 참여했다.

2) 고객 요구사항

IoT 디바이스 컨트롤 애플리케이션을 실행 후 어떤 디바이스를 제어할 것인지 선택하면 Fig. 9의 GUI처럼 각 디바이스 모드가 보여진다. 공통적으로 적용되는 기능인 자유이동 모드는 디바이스가 랜덤으로 생성되는 명령이나 방향에 따라 스스로 이동하며, 리턴 모드는 자유 이동이 끝나면 스스로 시작 지점으로 돌아오는 기능이다. 매뉴얼 모드는 사용자가 Fig. 9와 같이 화면에 나타난 버튼을 통해 직접 컨트롤한다. 각 버튼은 디바이스에 따라 다르게 나타나며 해당 기능을 수행할 수 있다.

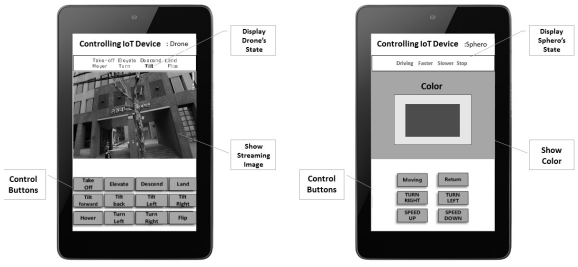


Fig. 9. GUI for IoT application's requirement

3) 사용된 IoT 디바이스

IoT 애플리케이션으로 컨트롤 할 수 있는 디바이스 선정은 운행 성격이 다른 비행용 디바이스인 AR Drone과 지상용 디바이스인 Sphero Ball로 하였다.

5.2 관찰된 기술적 이슈

본 절에서는 프로젝트 수행 시 실제로 직면했던 이슈들을 나열한다. 프로젝트 계획 단계에서 모든 팀원이 IoT 및 Agile 관련 지식이 부족하여, 프로세스를 적용하기 어려웠다. 이에 따라 팀원들은 프로젝트 태스크를 도출하고, 프로젝트 계획을 수립하는 데 많은 시간을 소비하였다. 이 중 가장 심각도가 높은 이슈는 "태스크 분배 실패"였다.

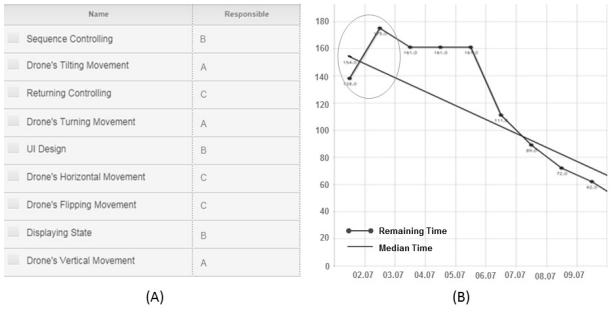


Fig. 10. Assigning Tasks without consideration Task Complexities and Burn down Chart

연구에 참가한 구성원은 모두 IoT 디바이스를 제어해본 경험이 없고 팀원들 간 역량을 제대로 파악하지 못한 상태였다. 이로 인해 PBI를 선정하고 태스크를 나누는 과정에서 Fig. 10과 같은 이슈가 발생하였다.

Fig. 10의 (A)는 AR.Drone 부분에 대한 태스크 분배를 나타내고 (B)는 첫 스프린트 기간 동안의 Burn down chart를 나타낸 모습이다. 그림과 같이, (A)에서 임의로 태스크를 분배함에 따라 개발 초기에 오히려 Burn down chart가 상승하는 모습을 보여주고 있다. 즉, 개인의 역량 고려가 없이 스프린트를 진행함에 따라 개발 초기에 자신이 부족한 부분에 대한 관련 지식 습득 등에 시간을 소모하게 되고, 이로 인하여 각 태스크에 지연이 생기고, 전체 스프린트 계획에 차질이 생기게 되었다.

프로젝트 개발 단계에서는 AR Drone과 Sphero Ball의 API의 미흡으로 요구사항대로 디바이스를 정교하게 제어하는 데 한계가 있었고, IoT 디바이스가 이동함에 따라 연결성 불안정 이슈와 제어권 상실 문제에 직면하였다. 이 중 프로젝트 성공에 부정적인 영향을 미쳤던 이슈는 다음과 같다.

- **정교하지 않은 디바이스 제어 :** 정교하지 않은 Sphero Ball 컨트롤은 실제 구동 시 관성으로 발생한 공회전으로 인한 오차와 관성에 따른 오차를 보정하기 위한 API도 존재하지 않았기 때문에 발생하였다. Sphero Ball API에서 제공되는 이동거리 측정은 디바이스 모터의 회전량을 활용한다. Sphero Ball의 모터 회전량은 관성에 의해 공회전이 생길 수 있기 때문에 이동거리 값에 오차가 발생할 위험이 높았고 리턴 모드를 수행할 때 시작지점과 많은 오차가 발생했다.
- **테스트 수행의 어려움 :** AR.Drone 컨트롤의 경우, 고객 요구사항에는 실외 비행을 마치고 시작 위치로 돌아오는 기능이 포함되어 있었다. 이 기능성을 구현하기

위해서는 GPS를 이용해야만 하고, GPS를 사용하려면 넓은 실외 공간이 필요하였다. 그러나 실외 테스트를 하는 과정에서 비, 바람, 장애물과 같은 환경적 제약사항 때문에 어려움을 겪었다. 구현한 코드 결과물을 확인하기 위해서는 실제 테스트를 통하여 확인해야 하지만, 이러한 제약사항 때문에 테스트를 통한 확인이 늦어지면서 구현도 지체되었다. 또한, 테스트 과정에서 AR.Drone의 중심축이 안정적이지 않고 회전각도의 변화가 차이 나는 경우를 발견하였다. 그러나 디바이스의 자체적 결함인지 애플리케이션 구현상의 문제인지 확인할 수 없었다.

5.3 적용한 솔루션

본 절에서는 AR.Drone과 Sphero Ball 컨트롤 애플리케이션 개발 프로젝트에서 관찰된 기술적 이슈 중 심각도가 높았던 3가지 이슈들에 대한 솔루션을 설명한다.

첫째, 각 태스크를 팀원에게 분배하는 과정에서 발생한 이슈는 3.4절의 솔루션을 적용하여 해결했다. Fig. 11은 태스크 점수와 팀원의 개인 역량 (IR)을 측정하여 계산된 값을 보여준다. 먼저, 개인의 역량이나 특정 기술 전문성 평가를 위해 디자인, 네트워크, 컨트롤링 부분으로 나누어 3명의 팀원 개개인이 역량을 측정한다. 그리고 Rule 1에 따라 가장 높은 백분율(%)을 갖는 팀원이 해당 태스크를 가져간다. 백분율이 같으면 Rule 2에 따라 역량이 높은 팀원이 가져가게 된다. Fig. 11의 Returning with GPS 태스크가 예에 해당된다. 결론적으로 팀원은 전체 스프린트 기간 동안 비교적 균등한 속도로 할당된 태스크를 수행하였다.

	A (10)	B (6)	C (7.7)
Button Handling (5)	0.8 (8%)	0.8 (13%)	0.2 (3%)
Page Rendering (9)	0.2 (2%)	0.8 (13%)	0.4 (5%)
UI Design (3)	0.1 (1%)	1 (17%)	0.2 (3%)
Random Generating (5)	2 (20%)	0.5 (8%)	0.5 (7%)
Manual Piloting (15)	2.5 (25%)	0.3 (5%)	1 (13%)
Returning with GPS (18)	2 (20%)	1.2 (20%)	1 (13%)
Auto Piloting (20)	1.5 (15%)	0.5 (8%)	0.5 (7%)
Socket Networking (8)	0.3 (3%)	0.6 (10%)	1.5 (20%)
Video Handling (12)	0.3 (3%)	0.3 (5%)	1.2 (16%)
Receiving & Playing Video (5)	0.3 (3%)	0.3 (5%)	1.2 (16%)

Fig. 11. A Reesult of Figuring out PBIs by considering Duplicated Functional Items and their Priorities

Fig. 12의 (A)는 위의 과정에 따른 실제 태스크 할당 그림이고, (B)는 해당 스프린트 기간 동안의 Burn down chart 그림이다. 관찰된 이슈와 비교하여 (B)의 chart를 보면, 상승하는 구간이 없고 전체적으로 내려가는 모습을 보이고 있다. 즉, 개인의 역량을 고려한 태스크 분배를 함으로써, 불필요한 지식 습득 등의 시간을 줄이고, 전체적인 팀원 간의 태스크 수행 속도를 맞춤으로써 계획된 스프린트 기간 동안 낭비되는 시간 없이 프로젝트를 진행할 수 있었다.

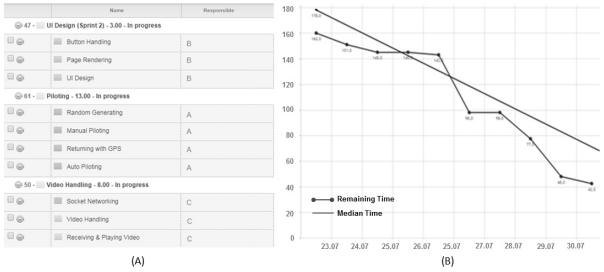


Fig. 12. A Result of Assigning Tasks by considering Task Complexities and Burn down Chart

둘째, Sphero Ball의 커브기능 구현 시 발생한 이슈들을 해결하기 위해, Fig. 5와 같은 커브 알고리즘 설계 방법을 제시한다. 먼저, 공회전을 발생시키지 않는 속도로 최소한의 감속을 한다. Sphero Ball의 경우 방향 전환으로 인한 공회전을 발생시키지 않는 속도는 0.3m/s 이하이므로, 그 이상의 속도로 구동 중인 경우 속력을 최적의 속도인 0.3m/s로 감속시켰다. 그 다음, 급격한 방향 전환은 공회전의 주요 원인이므로 디바이스 방향 전환 시 작은 각의 회전을 여러 번 수행한다. 전환하고자 하는 각을 15도 이하로 나누어, 작은 각의 방향 회전 명령을 여러 번 전달하도록 구현하였다. 그리고 오차가 줄었는지 확인하기 위해 임의의 방향으로 커브를 10번씩 한 후 시작 지점으로 돌아오는 기능을 수행하여 시작 지점과 돌아온 지점의 오차를 적용 전과 적용 후로 비교한다. Fig. 13은 이를 비교한 결과이다.

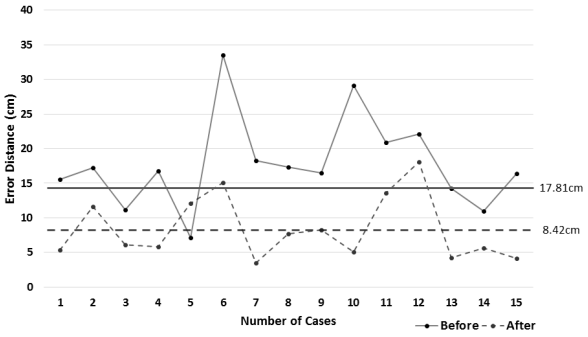


Fig. 13. Comparison of Error Distance to Return to Departure Point

제시한 알고리즘 설계를 활용하여 리턴 모드를 수행한 결과, 커브를 여러 번 거친 후 시작지점으로 돌아왔을 때 발생하는 오차는 적용 전 시작 지점으로부터 17.81cm이고, 적용 후 8.42cm이었다. 즉, 커브를 수행 시 공회전이 줄어들어 오차를 절반으로 줄일 수 있었다. 이동거리 오차 발생을 방지할 수 있다.

마지막으로, 태스크 수행 어려움의 문제를 해결하기 위해, Fig. 14와 같이 태스크 목록을 테스트 환경에 따라 수정하였다. 외부 테스트가 많이 있는 Week 2의 태스크 중 일부를 Week 1의 태스크와 교환하여 개발 시 문제가 발생하지

않도록 하였다. 그러나 단순히 환경만을 고려하여 변경하면 태스크 사이의 의존도나 선행 관계로 인하여 문제가 발생할 수 있으므로 변경 시 앞서 언급된 것들을 고려하여 먼저 개발되어야 하는 태스크는 남기고 변경했다. Fig. 14에서 T2는 비행 시 디바이스의 입력값을 랜덤으로 생성해주는 태스크이다.

이에 따라, 테스트 상황만 고려해보았을 때 랜덤값을 콘솔로도 확인이 가능하므로 왼쪽 그림에서 확인한 날씨를 고려해 Week 2로 옮겼다. 또한 T4, 5, 6은 AR.Rrone의 카메라 센서를 이용한 기능으로서 비행 기능을 수행하는 T7, 8, 9로 변경했다.

Week 1	Sun (27)	Mon (28)	Tues (29)	Wed (30)	Thurs(31)	Fri (8/1)	Sat (2)
Scheduled	T1	T1	T2	T3	T4	T5	T6
Changed Schedule	**T1**	**T1**	**T3**	**T7**	**T8**	**T9**	**T9**
Week 2	Sun (3)	Mon (4)	Tues (5)	Wed (6)	Thurs (7)	Fri (8)	Sat (9)
Scheduled	~~T6~~	~~T7~~	~~T8~~	~~T9~~	~~T9~~		
Changed Schedule	**T2**	**T4**	**T5**	**T6**	**T6**		

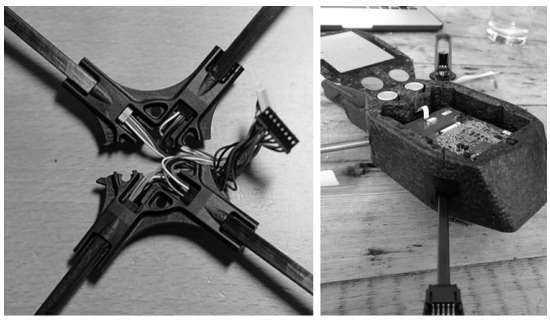
	Task	Testing Environment	Effort Hours
Week 1	T1. Drone Controller	Outdoor	7
	T3. Generate Sequence	Indoor	3
	T7. Auto Piloting	Outdoor	6
	T8. Manual Piloting	Outdoor	6
	T9. Returning Algorithm	Outdoor	10
Week 2	T2. Random Generating	Indoor	4
	T4. Socket Networking	Indoor	4
	T5. Video Handling	Indoor	5
	T6. Displaying Streaming	Indoor	7

Fig. 14. Modified Test Schedules with Test Environments

이와 같이 개발 초기에 계획한 태스크와 스케줄을 비교하여, 테스트가 어렵다고 판단되면 태스크 간의 의존도나 선행도 등을 고려하여 목록을 변경함으로써 개발 기간에 지연을 방지할 수 있었다.

테스트 단계에 AR.Drone 소프트웨어 개발 중, 디바이스 자체 결함이나 소프트웨어상의 논리적 오류로 인한 문제 원인 파악이 어려운 경우 스텁을 활용하여 해결하였다. 디바이스 제어에 관한 각 기능을 구현 후, 그에 대한 신뢰도 있는 수치를 측정하고, 스텁 테스트 시 테스트값과 수치값을 비교하였다. 이에 따라 높은 신뢰도로 다양한 문제의 원인을 분석하고 빨리 해결책을 찾을 수 있었다. 특히 AR.Drone의 경우, 비행용 디바이스이기 때문에 날개 손상이나 모터 손상, 중심축 부러짐, 보드 손상 등의 하드웨어적 결함이 발생할 수 있는데, 이러한 문제의 원인은 단순 테스트만을 통해서 유추하기는 어렵고 많은 시간이 소요된다.

Fig. 15의 (A)는 개발 당시 AR.Drone의 중심축(Central Cross)이 부러진 모습이고, (B)는 부러진 중심축을 교체하는 모습이다. 그림과 같이, 중심축은 AR.Drone의 내부에 들어가 있기 때문에 부러져도 외관상 문제가 보이지 않지만, 테스트 시 미세한 오차가 발생하기 때문에 개발자는 정확한 원인을 찾기가 힘들다. 이런 경우 스텁을 이용하여 미리 저장된 수치값과 비교를 통해 하드웨어적 결함인지, 소프트웨어상의 논리적 문제인지를 파악할 수 있었고, 이에 따라 빠르게 해결책을 찾아 스프린트 진행에 영향을 최소화하며 개발을 진행할 수 있었다.



(A) (B)

Fig. 15. Broken AR.Drone Central Cross and Repair

본 프로젝트를 시작했을 당시에는 IoT 프로젝트 가이드라인이 부족하여, 모든 팀원들이 프로젝트를 수행하는 데 많은 어려움이 있었다. 프로젝트 계획 단계를 위한 솔루션 중 프로젝트 태스크 선정과 태스크 분배 가이드라인은 팀원들의 역량 및 스프린트 수행 기간을 고려하여 실행 가능한 태스크 선정 및 분배를 가능하게 하였다. 그리고 해당 가이드라인에 따라 프로젝트를 수행한 결과, 프로젝트 지연이라는 심각한 문제에 직면하지 않게 되었다. 프로젝트 개발 단계를 위한 솔루션 역시 본 프로젝트에 매우 효과적으로 적용되었다. 팀원 모두 IoT 디바이스를 다루어본 경험이 없어 다양한 문제에 직면하였지만, 제안된 솔루션을 적용하면서 보다 빠른 시간 내에 해당 문제를 해결할 수 있었다.

6. 결 론

IoT 애플리케이션은 전형적인 애플리케이션에 비해 개발 복잡도가 상당히 높고 기존의 프로젝트 수행 가이드라인을 적용하기 어려워, 프로젝트 진행 시 다양한 문제가 발생할 수 있다. 또한, IoT는 새로운 패러다임으로 처음 접하는 개발자에게는 다소 생소하여 개발 진행에 어려움을 겪을 수 있고 이로 인해 비용이 증가할 가능성이 있다. 그러나 현재까지 디바이스의 종류나 특성, 네트워크 등 IoT 환경을 고려한 개발 지침이 부족하다.

본 논문에서는 애플리케이션 전체 개발 단계인 준비 및 계획 단계와 설계 및 개발, 운영 단계에서 발생할 수 있는 기술적 이슈에 대한 중요성과 원인을 분석하였다. 그리고 해당 이슈에 대한 솔루션을 제시하여 개발자가 개발 도중 해당 이슈에 직면하는 경우 제시된 솔루션을 기반으로 문제 해결에 도움을 준다. 마지막으로, 실제 IoT 애플리케이션 개발 진행 시 제시된 솔루션을 사례연구를 통해 검증하여 연구의 실효성을 검증한다.

제시된 솔루션을 적용하면, IoT 애플리케이션 개발 초보자가 직면할 수 있는 시행착오로 인해 발생하는 개발 지연 및 비용 증가를 예방할 수 있고 직면한 이슈를 효과적으로 해결할 수 있다. 즉, 효율적인 애플리케이션 개발이 가능하고 이슈를 효과적으로 해결하여 전체 품질 향상을 기대할 수 있다.

References

[1] S. Haller, S. Karnouskos, and C. Schroth, “The Internet of Things in an Enterprise Context,” *FUTURE INTERNET - FIS 2008*, Vol.5468, pp.14-28, 2009.

[2] International Telecommunication Union, ITU Internet Reports 2005, The Internet of Things, Nov., 2005, Available : http://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf(downloaded Sep., 26, 2014).

[3] R. Pressman, B. Maxim, *Software Engineering : A Practitioner's Approach*, 8th Edition, McGraw-Hill Science/Engineering/Math, Jan., 2014.

[4] D. Miorandi, S. Sicari, F.D. Pellegrini, and I. Chlamtac, “Internet of Things: Vision, Applications, and Research Challenges,” *Ad Hoc Networks,* Vol.10, No.7, pp.1497-1516, Sep., 2012.

[5] Y.K. Chen, “Challenges and Opportunities of Internet of Things,” *In Proceedings of the 17th Asia and South Pacitic Design Automation Conference(ASP-DAC 2012)*, pp.383-388, Jan., 2012.

[6] M.A. Chaqfeh, N. Mohamed, “Challenges in Middleware Solutions for the Internet of Things,” *In Proceedings of 2012 International Conference on Collaboration Technologies and System(CTS 2012)*, pp.21-26, May, 2012.

[7] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, “Future Internet: The Internet of Things Architecture, Possible Applications, and Key Challenges,” *In Proceedings of the 2012 10th International Conference on Frontiers of Information Technology(FIT 2012)*, pp.257-260, Dec., 2012.

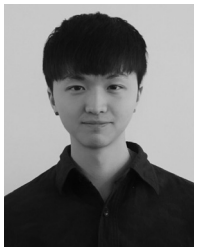
[8] H. J. La, S. D. Kim, “Unconventional Issues and Solutions in Developing IoT Applications,” *KIPS Tr. Comp. and Comm. Sys.*, Vol.3, No.10, pp.337-350, Mar., 2014.

[9] K. S. Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process*, 1st Edition Addison-Wesley, Aug., 2012.

[10] A. Gluhak, S. Krco, M. Nati, D. Pfistere, N. Mitton, and T. Razafindralambo, “A Survey on Facilities for Experimental Internet of Things Research,” *IEEE Communication Magazines,* Vol.49, No.11, pp.58-67, Nov., 2011.

[11] M. H. Kim, M. H. Lee, D. W. Lee, and H. M. Whang, “Study of Acceleration Control on effectiveness for Curve Driving of Torque Driving Apertures,” *In Proceedings of 2011 Information and Control Symposium*, pp.69-73, Apr., 2011.

[12] B. M. Albaker, N. A. Rahim, “Unmanned Aircraft Collision Detection and Resolution: Concept and Survey,” *In Proceedings of the 5th IEEE Conference on Industrial Electronics and Applications(ICIEA 2010)*, pp.248-253, June, 2010.



신 동 하

e-mail : shindh159@gmail.com

현 재 숭실대학교 컴퓨터학부 학사과정

관심분야 : 사물 인터넷 컴퓨팅(Internet of Things Computing), 컨텍스트 인지 컴퓨팅(Context-Aware Computing)

한 승 호

e-mail : gks8030@gmail.com

현 재 숭실대학교 컴퓨터학부 학사과정

관심분야 : 사물 인터넷 컴퓨팅(Internet of Things Computing), 컨텍스트 인지 컴퓨팅(Context-Aware Computing)

라 현 정

e-mail : hjla80@gmail.com

2003년 경희대학교 전자정보학부(학사)

2006년 숭실대학교 컴퓨터학과(공학석사)

2011년 숭실대학교 컴퓨터학과(공학박사)

2011년~2013년 숭실대학교 모바일서비스 소프트웨어공학센터 연구교수

현 재 (주)스마티랩 대표

관심분야 : 소프트웨어 아키텍처(Software Architecture), 모바일 클라우드 컴퓨팅(Mobile Cloud Computing), 사물 인터넷 컴퓨팅(Internet of Things Computing)

김 수 동

e-mail : sdkim777@gmail.com

1984년 Northeast Missouri State University 전산학(학사)

1988년~1991년 The University of Iowa 전산학(석사/박사)

1991년~1993년 한국통신 연구개발단 선임연구원

1994년~1995년 현대전자 소프트웨어연구소 책임연구원

현 재 숭실대학교 컴퓨터학부 교수

관심분야 : 객체지향 모델링(Object-Oriented Modeling), 소프트웨어 아키텍처(Software Architecture), 컨텍스트 인지 서비스(Context-Aware Service), 사물 인터넷 컴퓨팅(Internet of Things Computing)