AR.Drone IoT 애플리케이션 개발에서의 기술적 이슈

한승호, 강충훈, 최지예, 김문권*, 김수동
숭실대학교 컴퓨터학과
e-mail : { gks8030, kangch2410, cjy2426, mkdmkk, sdkim777}@gmail.com

Technical Challenges in Developing AR.Drone IoT Application

Seung Ho Han, Chung Hoon Kang, Ji Ye Choi, Moon Kwon Kim*, and Soo Dong Kim
Dept. of Computer Science, Soongsil University

요 약

비행용 IoT 디바이스 중 하나인 Drone 은 사람이 수행하기 힘든 작업 등에 활용되면서 다양한 분야에서 주목 받고 있다. 현재 상용화가 많이 된 AR.Drone 은 기본 비행 관련 기능뿐만 아니라 GPS 센서를 이용한 정밀 비행 및 복귀, 카메라 센서 이용한 사물 추적(Object Tracking) 등의 여러 목적으로 활발히 개발이 이뤄지고 있다. 그러나 AR.Drone IoT 애플리케이션 개발은 전통적인 소프트웨어 개발과는 다르게 다양한 기술적 이슈가 발생한다. 본 논문에서는 AR.Drone IoT 애플리케이션 개발 시 발생하는 개발 및 실행 관련 기술적 이슈를 설명하고, 각 이슈의 필요성과 한계점, 발생하는 문제점들을 제시한다.

1. 서론

IoT 디바이스란 센서와 액츄에이터(Actuator)를 가지며 무선 네트워크로 연결된 하드웨어 장비를 의미한다[1]. 이러한 IoT 디바이스 중, Drone 은 카메라, GPS 등의 센서와 프로펠러 등의 액츄에이터를 가진 대표적인 비행용 IoT 디바이스이다. Drone 은 사람이 수행하기에는 비효율적인 작업에 투입되어 안전과 효율을 증대시킴으로써, 다양한 분야에서 주목 받고 있다. 예를 들어, Drone 은 방사능 오염지역, 전쟁지역 등에서의 위험이 높은 작업이나 야간 정찰 등에 투입된다. 이처럼 Drone 에 대한 관심이 높아 지면서 Drone 을 직접 컨트롤 할 수 있는 Drone IoT 애플리케이션 개발에 대한 관심 또한 높아지고 있다.

Drone IoT 애플리케이션 개발은 소프트웨어, 하드웨어, 통신, 환경과 관련된 다양한 기술적 이슈가 존재하여 전통적인 소프트웨어 개발에 비해 많은 노력과 비용이 요구된다. 현재까지의 연구는 Drone IoT 애플리케이션 개발 및 운영 관련 이슈 도출 및 이슈에 대한 효과적인 솔루션 연구가 부족하다. Drone IoT 애플리케이션 개발의 노력과 비용을 줄이기 위해서는 이슈들에 대한 파악, 체계적인 정리와 해결책 제시가 필요하다.

본 논문에서는 대표적인 상용 Drone 장비인 AR. Drone 2.0 을 기준으로 하여 AR.Drone IoT 애플리케이션 개발 과정에서 나타날 수 있는 이슈에 대한 필요성과 한계점, 발생하는 문제점을 정리하여 이슈를 도출한다.

2. 관련연구

Bart 의 연구는 실외에서의 자율적 Drone 군집 비행을 위한 고수준 (High-Level) API 와 저수준 (Low-Level) API 를 제공하는 Paparazzi 플랫폼을 제안한다[2]. 이 플랫폼은 사용자가 Drone IoT 장비를 고수준 API 를 통해 전문적인 지식 없이도 쉽게 제어할 수 있고, 저수준 API 를 통해 정교하게 제어할 수 있도록 한다. 이 연구는 API 추상화 정도에 대한 소프트웨어적 이슈를 고려하고 있으나, 하드웨어적, 환경적 등의 이슈에 대한 고려가 부족하며, 실외에서의 Drone 제어에 한정적이다.

Bartak 의 연구에서는 GPS 센서와 Drone 자동 제어를 위한 Proportional-Integrative-Derivative (PID) 알고리즘을 이용하여 목표 지점으로 이동하고, 카메라를 이용하여 GPS 의 오차를 보정하는 기법을 제안한다[3]. 이 연구는 GPS 오차에 대한 이슈를 고려하지만, 바람 등의 환경적 요인으로 인한 오차의 고려가 부족하며, Wi-Fi 범위의 근거리 원격 제어 방식을 사용하기 때문에 원거리에서의 Drone 원격 제어가 어렵다.

[4][5]연구에서는 Drone 의 실내 군집비행을 위한 제어 기법을 제안한다. 이 연구는 모션캡쳐(Motion Capture) 시스템을 이용하여 Drone 의 실내 위치를 파악하고, 이해하기 쉬운 정성적인 제어 인자를 통해 Drone 을 제어한다. 이 연구는 실내에 모션캡쳐 시스템이 설치되어 있어야 하기 때문에, 일반적인 실내 및 실외 환경에서의 적용이 어렵다.

이와 같이 기존 연구들은 Drone IoT 애플리케이션 개발 및 운영에서 발생할 수 있는 특정 이슈에 대한 해결책을 제시하지만 다양한 관점에서의 이슈를 고려한 효과적인 해결책 제시가 부족하다.

3. 개발 관련 기술적 이슈

본 장에서는 AR.Drone IoT 애플리케이션 개발 시

* Corresponding author

발생 할 수 있는 기술적 이슈를 소개한다. 3.1 절에서는 API 선정에 관한 이슈를, 3.2 절에서는 AR.Drone 통신에 관한 이슈를 설명한다. 또한 각 이슈의 필요성, 한계점, 발생하는 문제점에 대하여 설명한다.

3.1. 애플리케이션 요구사항을 고려한 API 선정

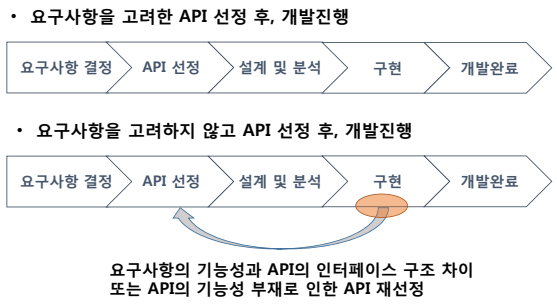
AR.Drone 제어를 위한 공식 API 는 C 언어로 제공되고, Java, Android, Node.js 등에서 사용 가능한 서드파티(Third Party) API 또한 존재한다. 서드파티 API 는 다양한 플랫폼 및 언어로 애플리케이션 개발 가능성을 제공하지만, 다음과 같은 API 선정에 따른 개발 효율성 영향력의 문제점을 갖는다.

- 동일한 기능성에 대한 인터페이스 구조의 차이
- 여러 API 종류 간의 제공되는 기능성의 부분적인 차이점
- 디바이스 제어 관점의 API 다양성

이러한 문제에 대한 고려 없이 개발을 진행할 경우, 개발 노력과 비용이 늘어 날 수 있다. 따라서 AR.Drone IoT 애플리케이션 개발에 있어 애플리케이션 요구사항에 최적인 API 선정이 중요하다.

API 선정을 위해 개발자는 애플리케이션의 요구사항에 대한 기능성 분석 및 개발 환경을 고려하여 모두 만족시켜야 한다. 이에 따라, API 선정은 개발 초기 개발자의 API 학습에 대한 많은 노력을 요구하며, API 선택폭을 좁히는 한계점을 갖는다.

다음 (그림 1)은 요구사항에 따라 API 선정 후, 애플리케이션 개발 과정을 비교한 그림이다.



(그림 1) 요구사항 고려에 따른 API 선정과 개발과정 비교

(그림 1)과 같이, 요구사항을 고려하지 않고 API 를 선정한 후, 과제를 진행하는 경우 여러 문제가 발생한다. 만약 개발 해야 되는 기능성이 사용중인 API 와 인터페이스 구조의 차이가 있다면, 과제 진행 중 전체 설계 및 아키텍처를 수정 하거나, API 를 교체해야 되는 문제가 생긴다.

개발 해야 되는 기능성 자체를 API 가 제공하지 않는다면, API 를 교체할 뿐 아니라 개발 환경 자체를 바꿔야 되는 심각한 문제를 발생시킨다. 예를 들어, GPS 센서를 이용한 기능성의 경우, Node.js 용 API 는 GPS 센서에 대한 기능성을 제공하지만, Android 용 API 는 제공하지 않는다. 만약 초기 API 선정에 대한 고려 없이, Android 용 API 를 선택하여 개발을 진행하면, 개발 중간 GPS 센서에 대한 API 가 제공되지 않는 것을 발견하게 되고, Node.js 용 API 교체뿐 아니라 개발 환경까지 바뀌게 된다.

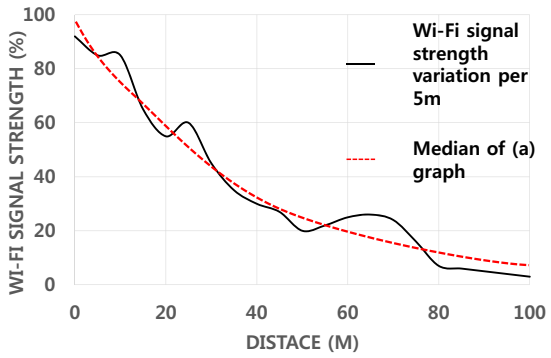
디바이스 제어 관점의 API 는 고수준 API 와 저수준 API 로 나뉘며 수준에 따라 제공되는 기능성에 차이가 있다. 고수준 API 는 사용자가 전문적인 지식 없이도 AR.Drone 을 쉽게 제어할 수 있도록 이해하기 쉬운 인터페이스를 제공하지만 정교한 제어에 있어 한계를 갖는다. 반면, 저수준 API 는 AR.Drone 을 정교하게 제어할 수 있지만 AR.Drone 제어를 위한 통신 패킷 구조 등의 전문적인 지식을 요구한다. 이에 따라, API 의 수준을 고려하지 않을 경우, API 의 수준에 따라 개발 중간에서 필요한 기능성이 API 에 없거나 애플리케이션 요구사항에 맞지 않는 API 를 선택하는 문제가 있다.

3.2. 통신 거리에 따른 데이터 교환 효율성 영향

AR.Drone 은 Wi-Fi 를 이용하여 애플리케이션과 통신을 제공하며, 양방향 통신을 함으로써 AR.Drone 에 대한 원격제어를 제공한다. 그러나 Wi-Fi 를 통한 원격제어는 Wi-Fi 신호의 강도에 따라 데이터 효율성이 변하게 되며, AR.Drone 제어에 있어 안정성이 달라진다, 이에 따라, AR.Drone 으로부터 안정적인 Wi-Fi 신호 전달이 필요하며, 신뢰성 있는 AR.Drone 제어를 위해 효율적인 데이터 교환이 필요하다.

AR.Drone 의 Wi-Fi 통신은 통신 거리에 따라 신호의 세기가 달라지며, AR.Drone 과의 효율적인 데이터 교환이 어려워 안정적인 AR.Drone 제어가 힘들다. 또한, 디바이스가 자체 이동성을 가지고 있으므로 신호 강도의 변화가 커질 수 있으며, 디바이스의 이동에 따라 신호 범위가 변할 수 있다. 따라서, 자체 이동성을 가진 AR.Drone 은 Wi-Fi 의 신호 강도나 범위가 계속 변하게 되어 효율적인 데이터 교환에 있어서 한계가 있다.

(그림 2)은 거리에 따른 AR.Drone 의 Wi-Fi 신호 세기를 나타내는 그래프이다.

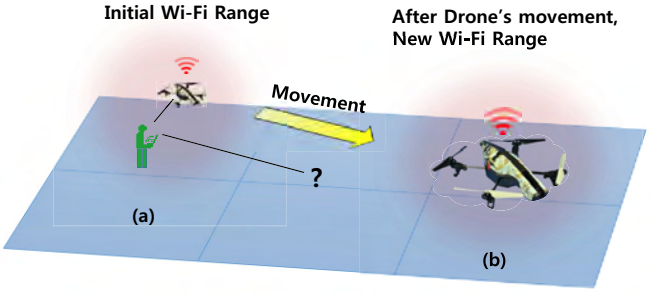


(그림 2) AR.Drone 의 거리 별 Wi-Fi 신호 강도

그래프에서 실선의 경우, 통신거리 5m 증가 당 Wi-Fi 신호 세기 변화를 나타내며, 점선의 경우, 실선 그래프의 중간 값에 대한 그래프이다. (그림 2)의 그래프와 같이, AR.Drone 원격제어에서 통신거리가 증가할수록 Wi-Fi 신호 세기가 약해진다. 이에 따라, 데이터 교환의 지연 문제나 데이터 유실문제가 발생 할

수 있다. 예를 들어, AR.Drone 의 실시간 영상 처리 기능은 통신 거리가 멀어져 신호의 세기가 약해지면 데이터 교환이 불안정해지고, 실시간 영상이 멈추거나 끊기는 등의 문제가 발생한다.

(그림 3)는 AR.Drone 이동에 의한 Wi-Fi 신호 범위의 변화를 보여주는 그래프이다.



(그림 3) AR.Drone 의 이동에 의한 Wi-Fi 신호 범위 변화

(a)의 경우, 초기 AR.Drone 의 Wi-Fi 신호 범위 안에서 AR.Drone 과 정상적인 데이터 교환을 하는 모습이고, (b)의 경우, AR.Drone 이 이동하여 Wi-Fi 신호 범위가 변화되고 통신 범위를 벗어나 데이터 교환이 불가능한 모습이다. (b)와 같은 경우, AR.Drone 의 원격 제어가 불가능하며, 제어를 벗어난 AR.Drone 의 충돌이나 추락 등의 위험 문제가 발생한다.

4. 실행 관련 기술적 이슈

본 장에서는 AR.Drone 실행 시 나타나는 기술적 이슈를 체계적으로 설명한다. 4.1 절에서는 비행환경에 관련된 이슈를, 4.2 절에서는 충돌 및 추락 이슈를, 4.3 절에서는 배터리 잔여량 관련 이슈를 제시한다.

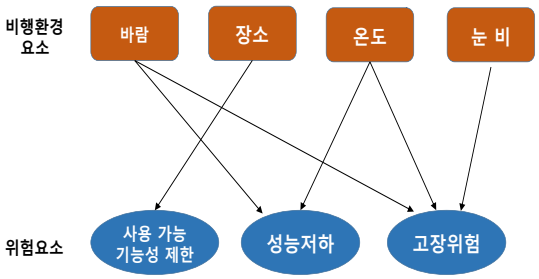
4.1. 비행 환경에 대한 높은 민감성

AR.Drone 은 비행용 IoT 장비이기 때문에 실행 시 비행환경에 큰 영향을 받는다. 비행 환경 중 큰 영향을 줄 수 있는 요소들과 각각의 영향은 다음과 같다.

- 바람 요소 → 제어 어려움
- 온도 요소 → 성능 저하
- 눈, 비 요소 → 손상 가능성
- 장소 요소 → 정교한 비행의 제한

이러한 비행환경 요소에 의한 영향을 극복해야 AR.Drone 의 손상 및 장애를 막을 수 있고, 정교한 비행이 가능하다. 그러나, 이러한 비행환경 요소로 인한 영향은 해결하기 어렵다. 배터리는 비행용 덮개로 감싸져 있고, 별도의 쿨러(Cooler)가 없기 때문에 발생하는 열을 감소시킬 수 없다. 또한, 눈이나 비가 올 경우, 외부에 노출되어 있는 센서 및 액츄에이터는 손상을 입을 수 밖에 없다. 바람이 불 경우에는 정교한 제어가 불가능하기 때문에 어려움이 있다. AR.Drone 의 센서 및 액츄에이터는 외부로 노출되어 있기 때문에, 눈 비를 막을 수 없다. GPS 센서는 장애물이 적은 실외에서만 가용하기 때문에, 이외의 지역에서는 정교한 위치추적이 불가능하다.

(그림 4)는 주요 비행환경요소에 따른 위험요소를 분석한 결과이다.



(그림 4) 비행환경요소에 따른 위험요소

이러한 한계 때문에, 바람이 부는 상황을 감지 못할 경우, 제어의 정확성이 떨어지며, 충돌 위험성이 증가한다. 고온의 환경에서 장시간 비행시 배터리 과열로 인해 성능저하 및 배터리 폭발의 위험이 커진다. 눈과 비를 막을 수 없기 때문에 고장의 위험성이 커진다. 또한, 정교한 위치추적은 일부 장소에서만 가능하며, 이외의 지역에서 비행을 하게 될 경우에는 자율비행이 어렵다.

4.2. 충돌 및 추락 상황 관리의 어려움

AR.Drone 은 비행체이기 때문에, 충돌 시 기체의 흔들림 및 손상이 발생할 수 있으며 충격이 강할 경우 기체가 중심을 잃어 추락할 수 있다. 이에 따라, 충돌 및 추락 시 충격을 완화하거나, 이를 사전에 효과적으로 예방할 수 있는 장치나 알고리즘이 필요하다.

그러나, AR.Drone 의 충돌 및 추락에 대한 대비책은 다소 미흡하다. 충돌에 대비한 스티로폼으로 만들어진 커버(Cover)가 있지만, 기체의 모든 부분을 보호하지 못하고, 벗겨질 위험이 있기 때문에 효과적이지 못하다. 또한, 충돌에 대비해 API 수준에서 비상착륙(Emergency Landing)기능을 제공하지만, 비상착륙 또한 기체에게 추락만큼의 충격을 주기 때문에 효과적인 해결책으로 볼 수 없다.

이에 따라, 충돌 및 추락에 의해 프로펠러(Propeller), 기어, 중심축 등의 부품이 손상될 수 있다. 부품이 손상되면 비행 중 기체의 흔들림 및 오작동의 위험성이 증가한다. 또한, AR.Drone 은 추락 시 비행관련 설정 값을 유지하고 있기 때문에, 추락 후 즉시 실행하면 급 발진 같은 오작동을 일으킬 수 있다.

4.3. 배터리 잔여량에 따른 민감도

AR.Drone 은 배터리를 동력으로 하는 무선 디바이스이다. AR.Drone 의 배터리 동력은 모터의 파워에 직접적인 영향을 끼치므로 배터리를 사용한 다른 디바이스들 보다 배터리의 잔여량에 더 의존적일 수 밖에 없다. 따라서, 배터리 잔여량이 충분하지 못한 상태에서 비행을 하면 다음과 같은 문제가 발생한다.

- 배터리 잔여 량 감소에 따른 비행 속력 및 균형 유지 기능이 저하되는 문제
- 배터리 잔여 량이 기준치 미달 시, 비행 관련 일부 기능을 실행 하지 못 하거나 멈추는 문제

이에 따라, AR.Drone 의 정교하고 완전한 비행을 위해서는 배터리 잔여량에 따른 민감도를 고려해야 할

필요가 있다.

AR.Drone 의 최대 비행 가능 시간은 1500mAh 배터리를 기준으로 18 분이나, 실제 AR.Drone 이 정상적인 비행을 수행 할 수 있는 시간은 12 분정도 이다. 비행 가능 시간을 이보다 더 늘릴 수 없는 이유는 배터리 크기가 증가 함에 따라 전체 무게가 증가 하게 되고, 이에 따라 비행 성능이 저하 되는 한계가 존재하기 때문이다.

배터리의 잔여량이 낮은 상태에서 비행을 할 경우 모터의 동력이 저하 되어 속력이 감소하고, 기기의 균형성이 역시 저하 되면서 정교한 비행이 어려워진다. 예를 들어, AR.Drone 이 앞으로 움직이는 기능을 수행 할 때 배터리 잔여량이 70% 이상인 상태에 비해 30% 이하인 상태에서는 속력이 저하되어 실제 이동한 거리가 더 작으며, 균형에 따른 오차가 더 발생한다.

(그림 5)은 AR.Drone 의 배터리 잔여량에 따른 비행 관련 기능의 수행 여부를 나타낸다.

기능 / Batt. (%)	이륙	플립	조율	움직임	착륙
30~	O	O	O	O	O
20~30	O	X	X	O	O
5~20	X	X	X	O	O
~5	X	X	X	X	O

(그림 5) AR.Drone 의 배터리 잔여량에 따른 비행 관련 기능의 수행 여부

(그림 5)에서 보이는 바와 같이, AR.Drone 은 배터리 잔여량에 따라 이륙, 플립, 조율과 같은 특정 기능을 수행 하지 못 한다. 또한, 배터리 잔여량이 5% 미만일 경우, AR.Drone 은 모든 비행 기능을 멈추고 착륙 한다. 이는 사용자가 요구 하는 기능을 수행 하지 못하고, 비행 중 배터리가 방전 될 경우 의도한 비행을 완전히 완료 하지 못하는 문제가 발생한다. 예를 들어, 사용자가 조율 기능을 실행 하려 하였을 때 배터리가 30% 미만인 경우 조율 기능이 작동하지 않으므로 내부 센서에 대한 정확성이 떨어질 위험이 있고, AR.Drone 이 복귀 할 때 배터리가 충분하지 않다면 원 위치로 돌아오기 전에 비행을 멈추고 착륙하게 된다.

5. 결론

AR.Drone IoT 애플리케이션개발은 순수소프트웨어 개발에 비해 새로운 기술적 이슈가 존재하며, 이를 고려하지 않고 개발할 경우 많은 노력과 비용이 요구된다. 그러나 기존 연구는 AR.Drone IoT 애플리케이션 개발 시 나타나는 기술적 이슈에 대한 정리와 솔루션 연구가 부족하다. 따라서 본 논문에서는 AR.Drone IoT 애플리케이션 개발 시 나타나는 기술적 이슈를 개발관련 이슈와 실행 관련 이슈로 분류하여 체계적으로 제시하고, 각 이슈에 대해서 필요성, 한계, 문제점을 제시하였다. 이에 따라, AR.Drone IoT 애플리케이션 개발 시에 제시된 기술적 이슈를 고려할 경우 개발 시 노력과 비용을 효과적으로 줄이고, 실행 시 안정성과 제어의 정교함을 높일 수 있다.

Acknowledgement

이 논문은 2012 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2012R1A1B3004130).

참고문헌

[1] Dieter Uckelmann, Mark Harrison, and Florian Michahelles, “Architecting the Internet of Things”, *Springer*, pp.4, April 2011.

[2] Bart Remes, Dino Hensen, Freek van Tienen, Christophe De Wagter, “Paparazzi: how to make a swarm of Parrot AR Drones fly autonomously based on GPS,” *In Proceedings of International Micro Air Vehicle Conference and Flight Competition (IMAV2013)*, September 2013

[3] Roman Bartak, Andrej Hrasko, and David Obdrzalek, "A Controller for Autonomous Landing of AR.Drone," *In Proceedings of Control and Decision Conference, IEEE*, pp.329-334, May - June 2014.

[4] 조동현, 문성태, 류동영, "실내 군집비행을 위한 AR.Drone의 제어기 개발," *한국항공우주연구원, 항공우주 기술*, Vol. 13, No.1, pp.153-165, July 2014.

[5] 문성태, 조동현 외 "실내 환경에서의 AR.Drone 군집 비행 시스템 개발," *한국항공우주연구원, 항공우주 기술*, Vol. 13, No.1, pp.153-165, July 2014.