



무인비행체 비행제어 Open Source 소프트웨어에 대한 정적분석 및 개선방안

장정훈¹, 강유선², 이지현³

Static Analysis and Improvement Opportunities for Open Source of UAV Flight Control Software

Jeong-hoon Jang¹, Yu-sun Kang² and Ji-hyun Lee³
MOASOFT Corp.

ABSTRACT

In this paper, We analyze and present improvements to problems in software quality through Static Analysis for Open Source, which is widely used as the Flight Controller software for small unmanned aerial vehicle drones. MISRA coding rules, which are widely applied based on software quality, have been selected. Static analysis tools were used by LDRA tools certified international tools used in all industries, including automobiles, railways, nuclear power and healthcare, as well as aviation. We have identified some safety-threatening problems across the quality of the software, such as structure of open source modules, analysis of usage data, compliance with coding rules, and quality indicators (complexity and testability), and have presented improvements.

초 록

소형 무인비행체 드론의 비행제어기(Flight Controller) 소프트웨어로 널리 사용되고 있는 오픈소스(Open Source)에 대한 정적분석(Static Analysis)을 통해 소프트웨어 품질의 문제점을 분석하고 개선 방안을 제시한다. 소프트웨어 품질 기준으로는 국제적으로 널리 적용되고 있는 MISRA 코딩 규칙을 선정하였으며, 정적분석 도구는 국제 도구인증(Tool Certification)을 받아 항공분야 뿐만 아니라 안전성(Safety)이 요구되는 자동차, 철도, 원자력, 의료 등 모든 산업에서 활용되고 있는 LDRA Tool을 사용하였다. 오픈소스 모듈의 구조, 사용 데이터 분석, 코딩 규칙 준수, 품질 지표(복잡도 및 시험성) 등 소프트웨어의 품질 전반에서 안전성을 위협하는 문제점을 발견하였으며, 이에 대한 개선 방안을 제시하였다.

Key Words : UAV(무인비행체), Flight Control Software(비행제어 소프트웨어), Static Analysis(정적분석), Code Standards(코딩 규칙), MISRA(자동차 산업 소프트웨어 신뢰성협회), LDRA(리버풀 데이터 리서치 협회)

I. 서 론

최근 항공기, 자동차, 철도, 선박, 원자력 등 산업 전반에서 소프트웨어의 역할이 확대됨에 따라 소프

트웨어 품질(Quality)을 향상시키기 위한 소프트웨어 검증(Verification)이 주요 이슈로 떠오르고 있다. 소프트웨어 검증은 검토(Review)와 시험(Test)으로 수행되며, 구현된 소프트웨어를 시험하기 전에 소스코

[†] Received : January 11, 2021 Revised : March 5, 2021 Accepted : March 29, 2021

¹ Director, ² Head Researcher, ³ Senior Researcher

³ Corresponding author, E-mail : jhlee@moasoftware.co.kr, ORCID 0000-0001-8165-2657

© 2021 The Korean Society for Aeronautical and Space Sciences

드에 대한 검토를 선행함으로써 소프트웨어의 잠재적 결함을 조기에 발견함과 아울러 소프트웨어 시험비용에 대한 절감 효과도 있다.

소형 무인비행체인 드론(Drone)의 활용 분야가 넓어지면서 사회·경제적 관심이 높아지고 있다. 드론은 각 산업에서 요구하는 수행임무(Mission)에 따라 활용성이 점점 더 높아지고 있는데, 철도 시설물 점검, 하천 조사, 산불 대응, 다중이용시설 사고예방, 우편 배송뿐만 아니라 최근 세계적으로 유행하고 있는 코로나-19 방역임무에도 활용되고 있다. 무인비행체 관련 기술에 대한 활용 분야로 최근에 정보 및 대기업 주도로 무인 항공기 기반의 도심 항공 모빌리티(UAM, Urban Air Mobility) 산업에 대한 미래 대응 전략을 수립하면서 제도, 기술, 인프라 등 많은 과제를 안고 있는 실정이다.

이러한 무인비행기 시대의 핵심 기술로는 프로펠러 구동용 고성능 모터, 고효율 고용량 전기배터리, 기체 무게를 줄일 수 있는 초경량 소재 등 물리·화학적 하드웨어 기술과 무인비행체의 비행제어기에 탑재되는 소프트웨어 기술로 구분된다. 비행제어 소프트웨어는 Pixhawk 개발팀에서 개발한 PX4 오픈소스가 전 세계적으로 가장 많이 사용되고 있다[1].

본 연구에서는 오픈소스로 제공되는 비행제어용 PX4 소프트웨어 모듈에 대하여 정적분석을 수행한 소스코드 분석 결과와 개선 방안을 제시하고자 한다.

기존에는 드론 운용에 사용되는 오픈소스는 대부분 HILS(Hardware In-the-Loop)를 이용하여 기능적 검증만 수행하여 사용하고 있다. 소프트웨어가 내포하고 있는 잠재적 오류에 대한 검증은 대부분 수행되지 않고 있으며, 입력 변수 기반 코드 검증, 소프트웨어 구조 또는 문장 기반 검증 등 제한적인 부분에서 수행되고 있다[2,3].

국제 도구인증을 받은 테스트 자동화 도구인 LDRA Tool을 활용하여 정적분석을 수행하여 소스코드의 모듈별 품질 수준과 국제적으로 널리 적용하고 있는 MISRA 코딩규칙에 대한 위반사항(Violation)을 검출한다[4-6]. 테스트 자동화 도구인 LDRA Tool은 다양한 산업분야에서 적용하고 있는 코딩 규칙(Coding Rule)을 지원하고 있으며, MISRA 코딩규칙에 대하여 매핑된 LDRA Rule을 적용하여 코딩규칙 위반 사항을 검출하는 기능을 지원한다[7].

소스코드의 모듈별 정적분석 결과와 코딩규칙의 심각도에 따른 분류 기준에 따라 심층적으로 분석한 결과로부터 소프트웨어 안전성에 대한 문제점을 인지하고 몇몇 사례에 대하여는 개선 방안을 제시한다.

II. 본 론

2.1 PX4 소프트웨어 개요

Table 1. Properties for PX4 Source Code

Properties	Description
Name of Source Code	Pixhawk 4 firmware-master v1.10.1
Location of Source Code	/firmware-master/src/modules
Location of Header Files	/firmware-master/src/include
Language	C++
# of modules	37
# of files	195
# of procedures	2,578

2.1.1 PX4 소스코드 현황

본 연구의 대상인 소형 무인비행체 비행제어 소프트웨어 오픈소스는 Pixhawk 개발팀이 제공하는 Pixhawk4(PX4)이다. PX4 소스코드 중에서도 /firmware-master/src/modules/ 폴더의 하위의 37개 폴더에 있는 C++ 언어로 구현된 소스코드(*.cpp, *.hpp, *.h) 파일을 정적분석 대상으로 한다(Table 1). 정적분석에 필요한 추가적인 전역 데이터(Global Data) 및 함수를 정의하고 있는 헤더 파일은 /firmware-master/src/include 폴더에 있다.

2.1.2 PX4 모듈 구성

PX4 소스코드의 37개 모듈(Module) 폴더(/firmware-master/src/modules)를 품질 수준 분석을 위하여 Table 2와 같이 무인이동체 비행제어 소프트웨어 기능적 특성에 따라 9개 기능으로 재분류하였다.

Table 2. Function Identification for PX4 Modules

Function Id.	Function	Remark	PX4 Module
F1	Sensor Input	Temperature Sensor and Camera	camera_feedback sensors temperature_compensation
F2	Aircraft State Estimation	Navigation System	ekf2 local_position_estimator sih simulator
F3	Attitude Control	Rotary Wing/ Fixed Wing, Hovering	airship_att_control attitude_estimator_q fw_att_control mc_att_control mc_hover_thrust_estimator uuv_att_control vtol_att_control

F4	Position Control	Latitude, Longitude, Altitude	fw_pos_control_l1 mc_pos_control navigator
F5	Flight Control	Firmware, Event Management	airspeed_selector commander dataman events load_mon logger mc_rate_control muorb uORB
F6	Landing Control	Landing Mark	land_detector landing_target_estimator rover_pos_control
F7	Communication Control	MAVLink	mavlink microrts_bridge
F8	Battery State Management	State of Battery Charge	battery_status esc_battery
F9	HW I/O	Relay Output	px4iofirmware rc_update replay vmount

PX4 소스코드를 9개 기능에 포함되는 모듈에 대하여 소스코드 품질 수준과 코딩규칙 위반율을 산출하여 각 모듈의 SW 안전성에 대한 심각성 수준을 분석한다.

2.2 소프트웨어 정적분석

2.2.1 정적분석 환경

2.2.1.1 정적분석 도구

PX4 소프트웨어에 대하여 정적분석은 테스트 자동화 도구인 LDRA Testbed v.9.8.4를 활용하였다. LDRA 도구는 도구인증을 받은 도구이며, 항공분야뿐만 아니라 안전성이 요구되는 자동차, 철도, 원자력, 의료 등 모든 산업에서 활용되고 있다.

LDRA Testbed 도구의 주요 지원 기능은 다음과 같다.

- 국제 표준 코딩규칙 및 사용자 정의 코딩규칙 지원
- 소프트웨어 품질 메트릭 기준설정, 산출 및 보고서 생성
- 코딩 규칙 위반사항 검출 및 보고서 생성
- 데이터 흐름 및 분석 보고서 생성
- 정적분석 통합보고서 생성

2.2.1.2 정적분석 기준

DO-178C(항공 SW), IEC 61508-3(전기전자 SW), ISO 26262-6(자동차 SW), IEC 62279(철도 SW), IEC 60880(원자력 SW), IEC 62304(의료 SW) 등 모든 국제 산업규격에서는 개발되는 소프트웨어 소스코드 대하여 코딩규칙을 준수하도록 요구하고 있다(Table 3)[8-13].

Table 3. Requirements for Coding Rules by Industry Standards

Industry	Standards	Requirements for coding rules
Aviation	DO-178C	Annex A. Table A-5-4. Source Code conforms to standards
Electric Electronic	IEC 61508-3	Annex B. Table B.1 design and coding standards
Road vehicles	ISO 26262-6	Section 8. Table 8. Design principles for software unit design and implementation
Railways	IEC 62279	Annex A. Table A.12 - Coding Standards
Nuclear power	IEC 60880	Annex B. B5.d Coding rules
Medical	IEC 62304	Annex B. B.5.5 Software unit implementation and verification

본 연구에서 적용한 코딩 규칙은 국제적으로 가장 널리 적용하고 있는 코딩 표준(Coding Standards)인 MISRA-C++:2008이며, LDRA 도구에서는 228개 MISRA Rule을 397개의 LDRA Rule로 매핑하여 코딩규칙 위반사항을 좀더 상세하게 검출해준다.

LDRA Rule은 Table 4와 같이 9개의 카테고리(Category)로 분류된다.

Table 4. LDRA Rule Classification

Classification of Rule	Name of Rule Classification	Description	No. of Rules
R1	Allocation	Rules relating to resource allocation and use	21
R2	Complexity	Rules relating to code complexity	25
R3	Dependability	Rules relating to code robustness	123
R4	Fault	Rules relating to fault elimination	247
R5	Maintainability	Rules relating to code maintainability	160
R6	Portability	Rules relating to code portability issues	185
R7	Style	Rules relating to source code style	39
R8	Testability	Rules relating to code testability	20
R9	Vulnerabilities	Rules relating to potential vulnerabilities	97

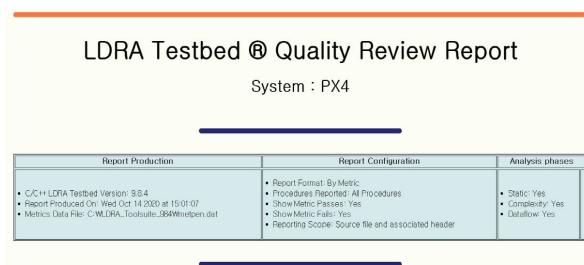


Fig. 1. LDRA Quality Review Report

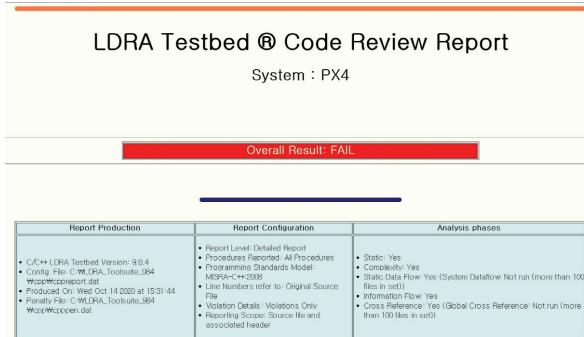


Fig. 2. LDRA Code Review Report

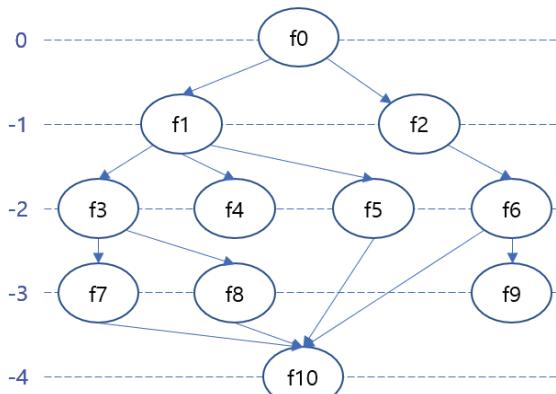


Fig. 3. Call Graph

2.2.2 PX4 소스코드 정적분석 결과

LDRA 도구를 사용하여 산출되는 PX4 소스코드에 대한 정적분석 결과 중에서 본 연구에서 분석하는 대상은 크게 Quality Review Report(Fig. 1), Code Review Report(Fig. 2), Call Graph(Fig. 3)의 3가지 형태이다.

2.2.2.1 Quality Review 결과

Quality Review Report에는 복잡도(Cyclomatic Complexity), 주석율(=주석문(Total Comments)/ 실행라인 수(Executable Lines)), 반복문 깊이(Depth of Loop Nesting), 모듈 제어도(Pan In/Pan Out) 등 60여개의 품질 지표(Quality Metric)가 산출되어 있다. 전체적인 소스코드의 품질 수준을 나타내기 위하여 명확성(Clarity)¹, 유지보수성(Maintainability)², 시험가능성

Table 5. Summary of Quality Review Report

Quality Metrics	Measured Values	Less than 70 of modules (70: LDRA Tool default value)
All Metrics: Calculation by aggregating indicators such as Clarity, Maintainability, and Testability	88	21.6% (8 / 37)
Clarity: Calculation from 14 relevant indicators	83	40.5% (15 / 37)
Maintainability: Calculation from 15 relevant indicators	91	51.3% (19 / 37)
Testability: Calculation from 11 relevant indicators	88	56.7% (21 / 37)

(Testability)³ 등을 산출하는 품질지표를 각각 14개, 15개, 11개로 정의하고, 각각의 품질지표의 기본 설정값에 대한 충족 여부를 측정하여 품질수준을 측정하며, 전체적인 소스코드의 품질 지표는 총괄 지표(All Metric)로 산출한다.

PX4 소스코드에 대한 품질지표 측정결과를 요약하면 Table 5와 같다.

전체 소스코드에 대한 총괄지표는 88, 명확성 83, 유지보수성 91, 시험가능성 88로 측정되었으며, LDRA Tool에서 기본 설정값 기준인 70을 충족하였다. 다만 모듈 단위의 품질 지표에서는 70 미만으로 측정된 모듈의 비율이 총괄지표에 대하여는 21.6%, 명확성은 40.5%, 유지보수성은 51.3%, 시험가능성은 56.7%로 나타났다.

1) Clarity: Executable ref. Lines, Total Comments, Comments in Headers, Comments in Declarations, Comments in Executable Code, Blank Lines, Total Comments/Exe. Lines, Declaration Comments/Exe. Lines, Code Comments/Exe. Lines, Average Length of Basic Blocks, Unique Operands, Total LCSAJs, Depth of Loop Nesting, Expansion Factor

2) Maintainability: Knots, Cyclomatic Complexity, Executable reformatted Lines, Number of Basic Blocks, Total Operands, Number of Loops, Procedure Exit Points, Number of Procedures, Total LCSAJs, Unreachable LCSAJs, Maximum LCSAJ Density, Unreachable Lines, Unreachable Branches, File Fan in, Fan Out

3) Testability: Essential Knots, Essential Cyclomatic Complexity, Knots, Cyclomatic Complexity, Vocabulary, Number of Procedures, Total LCSAJs, Unreachable LCSAJs, Maximum LCSAJ Density, Unreachable Lines, Unreachable Branches

Table 6. Quality Metrics for PX4 Functions

Function Id.	Function	Quality Metrics:			
		A	C	M	T
F1	Sensor Input	91	87	95	90
F2	Aircraft State Estimation	91	89	95	87
F3	Attitude Control	94	90	98	93
F4	Position Control	89	84	92	88
F5	Flight Control	86	79	88	85
F6	Landing Control	93	86	100	93
F7	Communication Control	80	75	80	83
F8	Battery State Management	91	83	100	93
F9	HW I/O	90	84	90	89

PX4의 9개 기능별로 품질지표 측정 결과는 Table 6과 같다.

총괄지표의 경우 PX4의 9개 기능 중에서 통신제어(F7)가 가장 낮은 80으로 측정되었으며, 직접적으로는 명확성이 75로 측정되었기 때문이며, 명확성이 낮게 측정된 이유는 소스코드 내에 Comment 문장이 부족한데서 그 원인을 찾을 수 있다.

PX4의 9개 기능 중에서 가장 핵심적인 기능인 비행제어(F5)에 대한 품질지표 측정 결과는 Table 7과 같다.

품질 지표 70 미만의 소스코드 모듈 또는 File은 소프트웨어를 복잡하게 구현되어서 포함된 잠재적 결함밀도가 상대적으로 높으므로 SW 시험 또는 유지보수 시 시험자 또는 개발자의 업무부담이 발생할 가능성이 높다고 볼 수 있다.

Table 7. Quality Metrics for Flight Control Function (F5)

Quality Metrics	Measured Values	Less than 70 of files
All Metric	86	1.6% (1 / 62)
Clarity	79	11.3% (7 / 62)
Maintainability	88	17.7% (11 / 62)
Testability	85	19.4% (12 / 62)

전체 품질지표 중에서 가장 중요한 품질지표는 복잡도이며, LDRA Tool에서 기본 설정값이 10으로 설정되어 있다. Table 8과 같이 복잡도에 대하여 PX4 전체 소스코드의 각 모듈을 구성하고 있는 2,578개 프로시저(Procedure) 중에서 233개(9%)가 기본 설정값 기준인 10을 초과하였다.

복잡도 10을 초과하는 233개의 프로시저에 대한 F1 ~ F9의 분포는 Table 9와 같으며, 배터리 상태 관리(F8) 모듈은 최대 복잡도가 3으로 복잡도 기준 설정값 10을 충족하였다.

소스코드의 사용하는 변수(매개변수, 전역변수, 지역변수), 분기문 등의 처리로직 및 함수 호출관계 등을 검토를 수행하여 단순화할 수 있다면 복잡도 10 이하의 프로시저로 분리하여야 한다.

Table 8. Distribution of Complexity for Procedures

Complexity Range	Frequency	Percent	Accumulated Percent
			Accumulated Percent
1	1,261	48.9%	48.9%
2~5	848	32.9%	81.8%
6~10	236	9.2%	90.1%
11~20	140	5.4%	96.4%
21~50	77	3.0%	99.4%
51 or more	16	0.6%	100%
Total	2,578	100%	

Table 9. Distribution of PX4 Functions with Higher Complexity (>10)

Function Id.	Function	No. of Functions with Higher Complexity (>10)			
		11~20	21~50	51 or more	Sum
F1	Sensor Input	13	7	0	20
F2	Aircraft State Estimation	6	2	2	10
F3	Attitude Control	11	7	1	19
F4	Position Control	28	11	4	43
F5	Flight Control	38	28	6	72
F6	Landing Control	4	2	0	6
F7	Communication Control	31	14	1	46
F8	Battery State Management	0	0	0	0
F9	HW I/O	9	6	1	16
Total		140	77	16	233

2.2.2.2 Code Review 결과

LDRA 도구에서는 228개 MISRA Rule을 397개의 룰로 매핑하여 위반사항을 검사한다. PX4 소스코드 전체에 대한 LDRA Rule 검사 결과를 요약하면 Table 10과 같다.

PX4의 9개 기능별로 코딩규칙 위반율은 Table 11과 같다.

PX4 소스코드 전체에 대한 LDRA에서 분류한 코딩규칙별 위반율은 Table 12와 같다.

LDRA Rule 분류 중에서 SW의 안전성과 가장 밀접한 관련이 있는 것은 R2, R4, R8, R9이다. PX4 소스코드는 R2, R7, R8, R9에서 코딩규칙 위반율이 60% 이상으로 나타났으며, 이러한 분석 결과로 볼 때 코딩규칙을 위반한 프로시저에 대하여는 SW 안전성을 충분히 검증해야 한다.

PX4 소스코드가 위반한 LDRA Rule 중에서 매우 심각한 코딩규칙은 Table 13과 같다.

Table 10. Summary of Code Review Report

Rule Category	checked	violated	%
# of Required Rules	310	177	75%
# of Required (Checking) Rules	1	1	100%
# of Advisory Rules	31	18	58%
# of Document Rules	55	20	36%
Total	397	216	54%

Table 11. Violation Rate for PX4 Functions

Function Id.	Function	Rate of Violated Coding Rule (=Number of Violation Rules/Number of Rules)
F1	Sensor Input	25.4%(=101/397)
F2	Aircraft State Estimation	24.7%(=98/397)
F3	Attitude Control	19.4%(=77/397)
F4	Position Control	24.7%(=98/397)
F5	Flight Control	39.3%(=156/397)
F6	Landing Control	14.1%(=56/397)
F7	Communication Control	32.7%(=130/397)
F8	Battery State Management	9.3%(=37/397)
F9	HW I/O	24.2%(=96/397)

Table 12. Coding Rule Violation Rate by LDRA Rule Classification

Classification of Rule	Name of Rule Classification	Rate of Violated Coding Rule (=Number of Violated Rules/Number of Rules)
R1	Allocation	33.3%(=7/21)
R2	Complexity	60.0%(=15/25)
R3	Dependability	55.3%(=68/123)
R4	Fault	57.1%(=141/247)
R5	Maintainability	55.0%(=88/160)
R6	Portability	48.2%(=41/85)
R7	Style	66.7%(=26/39)
R8	Testability	60.0%(=12/20)
R9	Vulnerabilities	66.0%(=64/97)

Table 13. Major of Violated Coding Rules

LDRA Rule Code	Rule Description	Classification of Rule	MISRA-C: 2008 Code
47 S	Array bound exceeded	R3	5-0-16
56 S	Equality comparison of floating point.	R4	6-2-2
85 S	Incomplete initialisation of enumerator.	R6	8-5-3
53 D	Recursion in procedure calls found.	R9	8-5-1
69 D	Procedure contains UR data flow anomalies.	R9	8-5-1
6 D	Attempt to use uninitialised pointer.	R2	7-5-4
45 D	Pointer not checked for null before use.	R3	0-3-1
1 J	Unreachable Code found.	R4	0-1-1
67 X	Identifier is typographically ambiguous.	R5	2-10-1

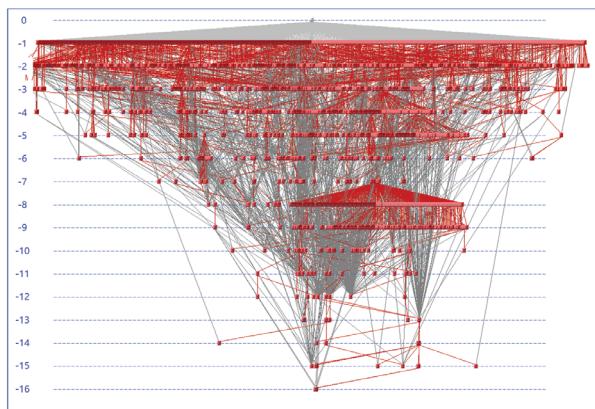


Fig. 4. Overall Call Graph for PX4

2.2.2.3 Call Graph

PX4 전체 소스코드의 2,578개의 단위 프로시저 간의 호출 관계를 시각적으로 보여주는 Call Graph는 Fig. 4와 같으며, 최상위 단계(Level 0)의 프로시저에서 다음 하위 단계(Level -1)의 프로시저를 호출하고 계속해서 최하위 단계(Level -16)의 프로시저까지 호출하게 되는 총 16 단계의 호출깊이(call-depth)를 보여준다.

PX4의 9개 모듈별로 호출깊이는 Table 14와 같다. 소프트웨어의 안전성을 확보하기 위하여 소프트웨어 구조 설계 시 모듈화(Modularization) 또는 분할(Partitioning) 기법을 적용하여 모듈 또는 함수 간의 독립성 유지 및 호출 최소화를 하게 되는데, PX4 소스코드의 경우에는 비행제어(F4)와 통신제어(F7) 모듈에 대하여 모듈화 및 분할 기법을 적용하여 구조 재설계(Re-engineering)를 통해 함수 간 호출깊이를 10 단계 미만으로 줄일 필요성이 있다.

Table 14. Call-Depth for PX4 Functions

Function Id.	Function	Call-Depth
F1	Sensor Input	7
F2	Aircraft State Estimation	8
F3	Attitude Control	5
F4	Position Control	8
F5	Flight Control	16
F6	Landing Control	5
F7	Communication Control	14
F8	Battery State Management	4
F9	HW I/O	10

2.2.3 PX4 소스코드 정적분석 결과 요약

PX4 소스코드에 대한 정적분석 결과는 Table 15와 같이 요약된다.

소스코드 품질 평가에서는 전체 소스코드에 대한 명확성, 유지보수성, 시험가능성은 기준값 70을 충족하였으나 모듈 단위의 시험가능성에서는 약 60%가 미달하였다. 또한 전체 소스코드에 대한 Call Graph에서는 호출깊이가 16으로 나타났으며, 소스코드 모듈 구성을 단순화하여 10 미만으로 낮출 필요가 있다.

Code Review 결과에서는 전체 소스코드에 대하여 54%의 코딩규칙을 위반하였으며, 매우 심각한 코딩 규칙도 다수 위반하고 있다.

복잡도에 대하여는 PX4 전체 소스코드의 각 모듈을 구성하고 있는 2,578개 프로시저 중에서 233개(9%)가 기본 설정값 기준인 10을 초과하였다.

이러한 PX4 소스코드에 대한 정적분석 결과로 근거로 다음과 같은 개선 방안을 제시한다.

- 품질지표 측정 결과를 반영하여 소스코드 품질 수준 개선
- 복잡도 및 호출깊이를 줄이기 위한 소프트웨어 모듈 구성의 단순화
- 중요 코딩규칙 적용에 대한 검토 후 적용 여부 판단
- 중요 코딩규칙을 위반한 관련 모듈에 대한 충분한 테스트 수행

Table 15. Summary of Static Analysis for PX4 Source Code

Static Analysis	Findings	
Quality Review(1)	Less than 70 of modules	
	All Metric	21.6%
	Clarity	40.5%
	Maintainability	51.3%
	Testability	56.7%
Quality Review(2)	No. of Procedures with Higher Complexity (>10)	233 (among 2578)
	Rule Violation Rate	54% (=216/397)
	Rule categories with violation rate 60% above	R2, R7, R8, R9
Code Review	Critical Violated Rules	47 S, 56 S, 85 S, 53 D, 69 D, 6 D, 45 D, 1 J, 67 X
	Call Graph	Maximum Call-Depth
		16

III. 결 론

본 연구에서는 무인비행체 비행제어기에 탑재되는 오픈소스에 대하여 테스트 자동화 도구 LDRA를 사용하여 정적분석을 수행하고 그 결과를 분석하여 각각의 문제점에 대하여 개선 방안을 제시하였다.

무인비행체 개발과 관련된 많은 전문업체 및 소프트웨어 개발자는 비행제어 오픈소스에 대한 안전성에 대한 심각성을 인지하여야 하며, 좀더 명확한 요구사항 기반의 테스트를 통해 문제점을 발견하고 개선하여야 한다.

또한 본 연구에서 다룬 PX4의 비행제어 관련 모듈 이외에도 탑재컴퓨터인 PX4 운용을 위한 OFP(Operational Flight Program)에 대한 정적분석 및 동적시험도 향후에 꼭 연구되어야 한다.

PX4 소스코드에 대한 동적시험을 효율적으로 수행하기 위해서는 자동화 테스팅 도구의 활용이 절대적으로 필요하며, 자동화 테스팅 도구와 하드웨어 타겟을 연동하여 적합한 시험환경을 구축하여야 하며, 충분한 시험 수행을 위하여 구조적 커버리지(문장 커버리지, 분기 커버리지, 수정조건/결정 커버리지)를 달성하여야 한다.

후 기

본 연구는 국토교통부/국토교통과학기술진흥원의 지원으로 수행되었음(과제번호: 21DPIW-C153651-03, 과제명: 공공혁신조달 무인이동체 통합기술관리 및 시험평가체계 개발).

References

- 1) The open standards for drone hardware (<https://pixhawk.org>)
- 2) Kim, T. G. Kim, C. H. Rhee, J. H. Fan Fei,

Zhan Tu, Gregory Walkup, Xiangyu Zhang, Xinyan Deng and Dongyan Xu, "RVFuzzer: Finding Input Validation Bugs in Robotic Vehicles Through Control-Guided Testing," 28th USENIX Security Symposium, August 14-16, 2019, pp. 425~442.

3) ALIAS ROBOTICS Robot Cybersecurity, *The Cybersecurity Status of PX4*, pp. 11~18.

4) SGS-TUV Saar GmbH, Certificate NO FS/71/220/15/0105 for LDRA tool suite (CERT. REPORT NO. K1C20003), 2015.

5) LDRA Ltd., *User Guide for LDRA tool suite Version 9.5*, 2016, pp. 341~343.

6) MISRA, *MISRA-C++:2008, Guidelines for the use of the C++ language in critical systems*, 2008, pp. 17~173.

7) LDRA Ltd., *MISRA-C++:2008 Standards Model Compliance for C++*, 2020.

8) RTCA Inc, *DO-178C, Software Considerations in Airborne Systems and Equipment Certification*, 2011, p. 100.

9) IEC, *IEC 61508-3, Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements*, 2nd Ed., 2010, p. 57.

10) ISO, *ISO 26262-6, Road Vehicles-Functional Safety-Product development at the software level*, 2011, p. 26.

11) IEC, *IEC 62279, Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems*, 2nd Ed., 2015, p. 75.

12) IEC, *IEC 60880, Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions*, 2nd Ed., 2006, p. 161.

13) IEC, *IEC 62304, Medical device software - Software life cycle processes*, 1st Ed., 2006, pp. 90~91.