리눅스 기반 무인항공기 제어 애플리케이션 개발

Development of Flight Control Application for Unmanned Aerial Vehicle Employing Linux OS

김 명 현, 문 승 빈*, 홍 성 경
(Myoung-Hyun Kim, Seungbin Moon, and Sung Kyung Hong)

Abstract : This paper describes UAV (Unmanned Aerial Vehicle) control system which employs PC104 modules. It is controlled by application program based on Linux OS. This application consists of both Linux device driver in kernel-space and user application in user-space. In order to get data required in the unmanned flight, external devices are connected to PC104 modules. We explain how Linux device drivers deal with data transmitted by external devices and we account for how the user application controls UAV on the basis of data processed in the device driver as well. Furthermore we look into the role of GCS (Ground Control Station) which is to monitor the state of UAV.

Keywords : UAV (Unmanned Aerial Vehicle), FCS (Flight Control System), multi-tasking software

I. 서론

무인항공기 제어 시스템은 항공기에 소형 전자 장비를 탑재하여 이러한 장치로부터 획득한 데이터를 분석하여 항공기에 부여된 임무를 수행하도록 개발되고 있다[1-7]. 최근에는 정찰, 감시, 전투, 통신중계, 원격탐사 등의 다양한 분야로 민간과 군용으로 개발되고 소형화[8], 저가형[9,10]으로 개발되어 더 많은 분야로 그 사용이 증가하는 추세에 있다. 대부분의 무인항공기는 군용으로 개발되어 정찰용으로 사용되고 있고, 공격용으로도 그 사용범위를 넓혀가고 있다[11]. 실례로 미공군은 '프레디터'라는 대형 무인정찰기를 코소보, 아프카니스탄, 이라크 등에서 사용했으며, 최근에는 미사일을 장착하여 테러리스트들을 공격하는데 사용하기도 했다. 또한 미해군이 아프카니스탄과 이라크에서 사용한 '실버 폭스'라고 불리는 모형비행기 같은 프로펠러 무인항공기는 길이 1.5m, 무게 10kg 정도의 크기로 정찰용으로 전술적으로 활용함을 보여준다.

본 논문에서 RC(Remote Control) 비행체에 전자 장비를 탑재하여 리눅스 OS(Operating System)를 기반으로 구축한 무인항공기 제어 시스템에 대해 설명한다. 제어 시스템은 비행체 자세 제어를 위해 외부 장치들로부터 여러 종류의 데이터를 획득하여 무인 비행이 가능하도록 데이터를 변환한 후, 자신에게 부여된 임무와 변환된 데이터를 비교하여 비행체의 자세를 제어하면서 임무를 수행하게 된다. OS를 사용하는 제어 시스템이라는 특징을 고려하여, 소프트웨어적인 측면에서 항공기 제어 흐름을 살펴보고, 리눅스 OS에서 구현한 제어기의 디바이스 드라이버와 제어 애플리케이션이 연계되어 제어 시스템이 동작하는 방법을 설명한다.

* 책임저자(Corresponding Author)
논문접수 : 2004. 10. 21., 채택확정 : 2005. 8. 9.
김명현, 문승빈 : 세종대학교 컴퓨터공학과
(myjimi@yahoo.co.kr/sbmoon@sejong.ac.kr)
홍성경 : 세종대학교 항공우주공학과(skhong@sejong.ac.kr)
※ 본 논문은 학술진흥재단 중점연구소 지원과제(D20002)에서 지원하여 연구하였음.

제어 애플리케이션은 멀티태스킹을 지원하는 OS라는 특성을 이용하여 다중 테스크로 구성되어 있다. 다중 task로 이루어진 애플리케이션이 데이터 처리율을 높여줌으로서 빠른 응답성을 보여줌을 살펴볼 것이다.

II장에서는 제어기를 구성하는 하드웨어에 대해 살펴본다. III장에서는 항공기 운용의 기본이 되는 리눅스 OS에 대한 구성과 탑재 방법을 설명한다. IV장에서는 항공기에 탑재되는 전자 장비들을 추상화시킨 디바이스 드라이버의 구성과 기능을 살펴본다. V장에서는 항공기 애플리케이션의 중심이 되는 제어 애플리케이션의 task 구성과 디바이스 드라이버에서 전달된 데이터를 해석하여 항공기를 제어하는 방법[12-14]을 설명하고, 애플리케이션이 디바이스에서 발생하는 에러처리 방법에 대하여 설명한다. VI장에서는 지상통제 시스템의 구성과 역할에 대하여 설명한다. VII장에서는 본 논문에서 구현한 제어 시스템의 문제점과 해결방안을 제시해 본다.

II. Hardware 구성

본 논문에서 언급한 무인항공기용 제어기는 3개의 모듈로 구성되어 있다. 3가지 모듈은 다음과 같다. 제어 시스템에서 소형 컴퓨터로 사용되는 CPU 모듈, 다중의 입출력 장치를 제어하는 Multi-IO(Input-Output) 모듈, 그리고 GPS 데이터를 수신하는 GPS 모듈이다. CPU 모듈과 Multi-IO 모듈에는 외부적으로 비행에 필요한 전자 장비들이 연결된다. 연결되는 전자 장비를 모듈별로 살펴보면 다음과 같다.

CPU 모듈에는 외부 커넥터로 연결되어 RS-232 직렬 통신을 하는 UHF(UltraHigh Frequency) 모뎀과 AHRS(Attitude-Heading Reference System)가 연결된다. Multi-IO 모듈에는 원격 조종기의 신호를 받아 PWM 신호를 출력하는 원격조종 수신기, 항공기 조종면에 연결되어 직접적으로 자세제어 역할을 수행하는 서보 모타, 그리고 아날로그 신호로 고도, 대기압력, 그리고 대기속도의 데이터를 전달하는 sensor pack이 연결된다. GPS 모듈은 GPS 수신기로 내부 버스 라

표 1. 제어기 하드웨어 구성요소와 사용.

Table 1. Flight control system hardware platform and components.

◆ CPU 모듈	
CPU	Embedded Low power NS Geode GX1 300MHz
BIOS	AWARD 256KB Flash Memory
System Memory	128MB SDRAM
I/O Interface	RS-232 Serial 2 Ports
외부연결장치	UHF 모뎀, AHRS
UHF 모뎀	RF(Radio Frequency) 통신 GCS와 UAV 데이터 송수신
AHRS	3축 각도, 3축 각속도의 데이터 출력
◆ Multi-IO 모듈	
외부연결장치	원격조종 수신기, 서보 모터, SensorPack
원격조종 수신기	5채널, 원격 조종기의 신호 수신
서보 모타	4채널, 항공기 자세제어
SensorPack	고도, 대기압력, 대기속도 데이터 출력
◆ GPS 모듈	
GPS 수신기	NMEA(National Marine Electronics Association) 방식의 GPS 데이터 송신

인을 통해 CPU 모듈과 연결되어 있다. GPS 수신기도 RS-232 직렬 통신을 사용하여 데이터를 전달해준다.

표 1은 제어기 하드웨어 구성 요소와 사양을 나타낸다. 그림 1은 탑재되는 PC104 모듈들이 외부 커넥터를 통해 통신 가능하도록 조립해 놓은 모습이다. 하단부터 CPU 모듈, Multi-IO 모듈, GPS 모듈의 모습을 볼 수 있다. 그림 2는 항공기에 탑재되는 PC104 모듈들과 각 모듈에 연결되는 외부 장치들을 보여주고, 항공기 운용에 필요한 전체 시스템 구조도를 보여주고 있다. FCS(Flight Control System)는 무인항공기를 제어하는 메인 모듈들과 항공기 제어를 위해 연결된 외부 장치들을 보여주고 있다. GCS(Ground Control Station)는 지상에서 항공기의 실시간적인 상태를 확인하고 항공기에 임무를 부여하는 역할을 위하여 UHF 모뎀을 통한 무선 통신으로 항공기와 데이터를 송수신 한다. 무선 조종기는 시스템 오류가 발생하여 무인비행 수행이 불가능할 경우 수동으로 항공기를 제어하게 된다.

III. 제어기 OS

무인항공기 제어기는 리눅스 OS를 탑재하여 항공기 제어 프로그램이 운용되도록 구축하였다. 탑재된 리눅스 OS는 커널버전 2.4.20으로 시스템이 구성되어 있다. 제어기에 사용된 리눅스는 CPU 모듈의 연결되는 256MB의 비휘발성 메모리인 CompactFlash에 적재하였다. 제어기용 프로세서는 저 전력용 x86계열의 프로세서가 사용되어 커널 패치 없이 OS를 설치하였다. 부트로더는 LILO(Linux Loader)를 사용하였고, LILO 설정파일 내에 디스크 바이오스 번지를 수정하여 부팅 가능하도록 만들었다. 데스크탑에서 사용되는 컴파일러, 라이브러리를 시스템 운용에 필요한 최소한의 사양만 선택하여 Flash에 탑재하였다. 많은 임베디드 장치에서는

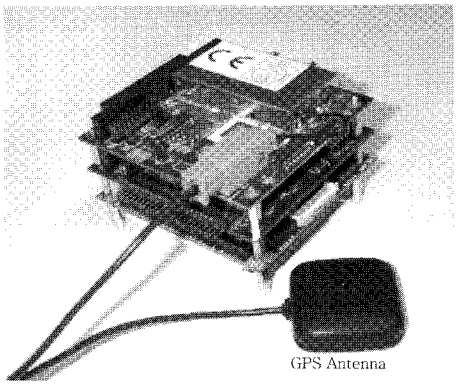


그림 1. 조립된 PC104 제어기 모습.

Fig. 1. PC104 flight control system.

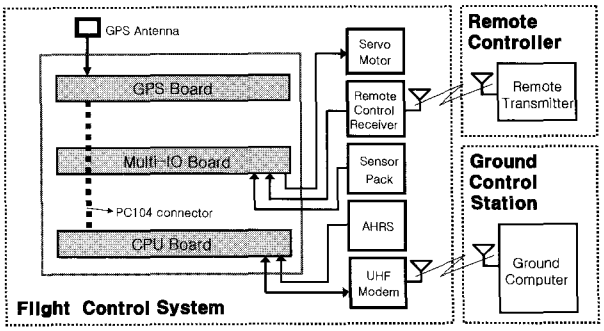


그림 2. 제어기 시스템 구성 및 외부장치.

Fig. 2. Flight control system architecture and external electric device.

작은 기억장치의 용량으로 크로스 컴파일 방식을 취하는데, 제어기 OS가 탑재되는 CompactFlash는 기억장치의 용량이 충분하여 컴파일러도 탑재할 수 있었다. 제어기 자체적으로 애플리케이션 코드의 수정, 바이너리 파일 생성이 가능하여 유지, 보수에 편리성을 보장하였다. 항공기 운용에 불필요한 서비스들은 제거하였고, 시스템 운용에 필요한 기본적인 프로세스만 생성하도록 설정하여 하나의 완전한 리눅스 OS를 구축하였다.

IV. 디바이스 드라이버

제어기에는 외부 장치를 제어하기 위한 4개의 디바이스 드라이버가 제작되어 있다. modem device driver, AHRS device driver, GPS device driver, multiboard device driver가 있다. 리눅스 디바이스 드라이버는 크게 character device driver, block device driver, network device driver 3가지로 분류된다. 제어기용으로 제작된 디바이스 드라이버는 스트림 기반의 데이터 통신이 가능한 character device driver[15]로 제작하였다. modem device driver, AHRS device driver, GPS device driver는 각각 RS-232 직렬 통신을 하고, 각각 COM1, COM2, COM3 포트를 제어하게 된다.

리눅스에는 이미 많은 디바이스 드라이버가 존재하고, Multi-IO 모듈에 연결되는 외부 장치를 제어하기 위한 디바이스 드라이버를 제외하고는 COM1, COM2, COM3 에 연

결되는 외부 장치들은 리눅스에서 제공하는 디바이스 드라이버로 연결할 수 있다. 그러나 무인 비행에 최적화된 디바이스의 제어와 COM 포트들의 제어를 위해서는 디바이스 드라이버 제작이 필요하였다. 왜냐하면 리눅스는 사용자 영역 애플리케이션에서는 인터럽트 핸들러 구현을 지원하지 않고, COM 포트로 연결되는 외부 장치들은 지정된 시간마다 인터럽트를 발생하여 연결되는 장치로 데이터를 전달해 주기 때문에 올바른 데이터를 사용하기 위해서는 디바이스 드라이버 제작이 필요하였다. 또한 Multi-IO 모듈과 연결되는 장치들도 리눅스에서 사용이 가능하도록 지원해주는 디바이스 드라이버가 존재하지 않기 때문에, Multi-IO 모듈과 연결되는 장치들을 통합적으로 제어하고 애플리케이션에서 이용 가능하도록 디바이스 드라이버를 제작하였다. 제작된 디바이스 드라이버들은 모듈 적재 명령을 사용하여 커널에 적재되고, 커널에 적재된 후 제작된 애플리케이션에서 system call 명령을 받을 때 해당하는 루틴을 수행하게 된다. system call 명령과는 무관하게 각각의 디바이스가 커널에 적재되면 인터럽트 핸들러가 구현된 COM 포트 디바이스 드라이버들은 ISR(Interrupt Service Routine)을 처리하게 된다. 인터럽트 발생시 COM 포트 디바이스 드라이버들은 1 바이트의 데이터를 커널 메모리에 복사하게 된다. 이때 소요되는 시간은 modem device driver가 0.8ms, AHRS device driver가 0.2ms, GPS device driver가 1.7ms이 걸린다. 각각의 디바이스 드라이버가 1 바이트의 데이터를 커널 메모리에 복사하는데 소요되는 시간이 다른 이유는 물리적인 전자 장비의 인터럽트 발생 시간이 다르기 때문이다. 각 디바이스 드라이버의 특징을 살펴보면 다음과 같다.

첫째, modem device driver는 GCS(Ground Control Station)와 RF 통신을 위한 매개체로 바이트 단위의 데이터를 전송하고 GCS에서 전달된 데이터를 커널 메모리에 저장하는 역할을 한다. 둘째, AHRS device driver는 AHRS 장치에서 자세 제어에 필요한 항공기 자세각과 각속도의 데이터를 전달받아 커널 메모리에 저장한다. 셋째, GPS device driver는 GPS에서 송신하는 항법 데이터를 전달받아 커널 메모리에 저장하는 역할을 한다. 마지막으로 multiboard device driver는 아날로그 신호를 출력하는 SensorPack, PWM 신호를 출력하는 원격조종 수신기의 장치의 지정된 채널에서 데이터를 읽어오고, PWM 신호를 출력하여 4개의 서보 모타를 제어하는 역할을 한다. 각각의 디바이스 드라이버들은 비행에 필요한 데이터를 하나의 프레임으로 저장하여 제어 애플리케이션에서 데이터를 요청하면 커널 메모리에 저장된 데이터를 User 영역으로 전달해주게 된다. 일반적으로 디바이스를 제어하는 함수는 open, close, read, write, ioctl 등이 있다. 제어기용으로 제작된 디바이스 드라이버들은 open, close, ioctl 함수를 사용하여 제어할 수 있도록 제작되었다. 제어기 소프트웨어의 구성도는 그림 3에서 살펴볼 수 있다. 최하단의 물리적인 외부 장치들을 제어하는 4개의 디바이스 드라이버가 커널 영역에서 동작하고, 사용자 영역에서 다중 task로 이루어진 애플리케이션이 system call interface를 사용하여 디바이스 드라이버와 데이터를 교환하는 것을 볼 수 있다.

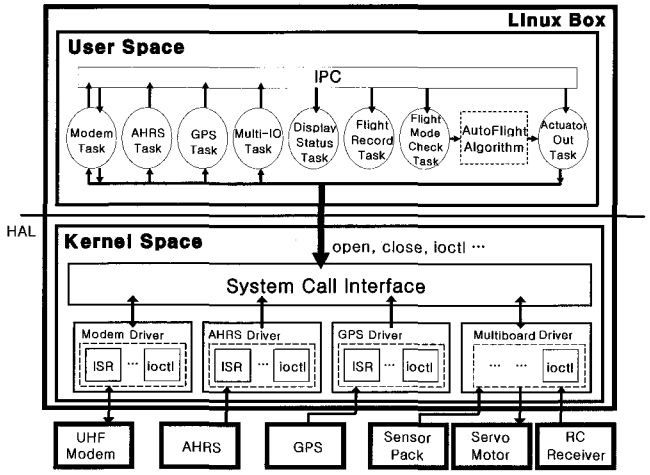


그림 3. 제어기 소프트웨어 구성도.

Fig. 3. Flight control system software architecture.

V. 제어 애플리케이션

제작된 디바이스 드라이버를 기반으로 항공기 제어 애플리케이션을 작성하였다. 하나의 프로세스로 제작된 애플리케이션은 선행 처리를 마친 후에 다음 처리 루틴으로 넘어감으로서 저조한 데이터 처리 및 낮은 시스템 응답성을 보일 수 있고, 다중 프로세스로 제작된 애플리케이션은 본 논문에서 사용한 프로세서로는 제어 시스템에서 원하는 성능이 나오지 않을 것으로 기대되어 본 논문에서는 프로세스 내의 다중 쓰레드로 애플리케이션을 제작하여 데이터 처리율을 높여주고, 빠른 응답성을 보여주는 애플리케이션을 제작하였다. 본 논문에서는 편의상 쓰레드를 task로 표시하였다. 애플리케이션은 총 8개의 task로 구성되어 항공기를 제어하고 자신에게 부여된 임무를 수행하게 된다. task를 분류하면 다음과 같다. modem task, AHRS task, GPS task, Multi-IO task, display status task, flight mode check task, flight record task, actuatorOut task로 이루어져 있으며 task 구성도는 그림 3에서 살펴볼 수 있다. 각 task는 SCHED_RR(Round-Robin)[16] 방식으로 스케줄링 된다.

Modem, AHRS, GPS task는 타이머 함수를 등록하여 task가 정해진 루틴을 지정된 시간마다 처리하지 못하였을 경우 다시 task의 진입 부분으로 돌아와서 수행이 가능하도록 수행 주기를 지정해 주었다[17]. ISR이 등록된 디바이스를 사용하는 task는 각각의 장치의 자체적인 문제, 커넥터 연결 문제 또는 공급되는 전원 부족 등의 이유로 비행 중에 예측하기 어려운 문제가 발생할 수 있다. 이때 물리적인 장치에서 데이터를 전달해주지 못하면 인터럽트는 발생되지 않고, 제작된 디바이스 드라이버는 ISR에서 처리된 데이터를 사용할 수 없으므로, blocking 상태가 되어 system call 호출시 task 자체가 blocking 되어 무인 비행에 필요한 데이터를 전달받지 못하고, 전체 시스템에 심각한 영향을 미칠 수 있다. 그래서 modem task, AHRS task, GPS task에 수행주기를 지정하였다. modem task는 수행주기가 평균 254.7ms로 worst-case로는 수행주기가 500.3ms로 측정되었다. AHRS task는 평균 수행주기가 34ms로 worst-case로는 수행주기가

표 2. Task 수행주기 비교.
Table 2. Task execution time comparison.

	평균	worst-case	수행주기
Modem task	254.7 ms	500.3 ms	501 ms
AHRS task	34 ms	37 ms	38 ms
GPS task	988 ms	1068 ms	1070 ms

User Application Start
- Read data from the devices
- Confirm assigned task and execute auto-flight control algorithm
- Send the command to servo motors to maintain the posture and navigation
- Transmit flight data to GCS
- Record flight status

End

그림 4. 제어 애플리케이션 알고리즘.
Fig. 4. Control application algorithm.

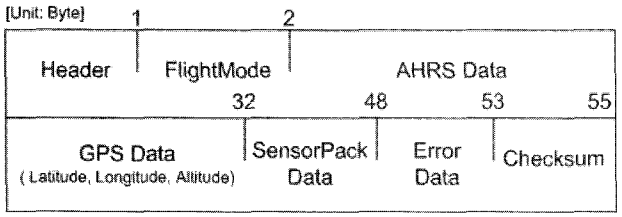


그림 5. 모뎀 프로토콜 - UAV에서 송신하는 데이터.
Fig. 5. Modem protocol - data transmitted from UAV.

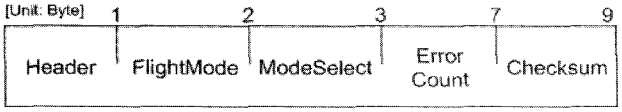


그림 6. 모뎀 프로토콜 - GCS에서 송신하는 데이터.
Fig. 6. Modem protocol - data transmitted from GCS.

37ms로 측정되었다. GPS task는 평균 수행주기가 988ms로 worst-case로는 수행주기가 1050ms가 측정되었다. task의 수행주기는 표 2에서 살펴볼 수 있다. 각 task가 수행함에 있어서 데이터 획득에 실패하면 다른 task의 수행과 항공기의 전반적인 제어에 실패로 이어질 수 있다는 이유로 응답성을 높여주기 위해서 modem task, AHRS task, 그리고 GPS task는 각각 501ms, 38ms, 1070ms로 worst-case 수행주기보다 조금 더 보장된 시간으로 수행주기를 지정하였다.

8개의 task가 이루는 애플리케이션이 처리되는 알고리즘은 다음과 같다. 4개의 task 즉, modem task, AHRS task, GPS task, Multi-IO task가 비행에 필요한 데이터를 디바이스로부터 전달받아 그 값을 분석한다. 그리고 flight mode check task는 현재 비행 상태를 판단하고, 주어진 임무가 있다면 그 임무를 수행하게 된다. 항공기에 주어지는 임무는 GCS에서 전달받게 된다. 최종적으로 자세제어가 요구되고, 항로 변경이 요구된다면 ActuatorOut task에서 그 역할을 수행하게 된다. 그리고 현재 비행 기록을 파일 형태로 저장한다. 애플리케이션의 알고리즘을 나타내면 그림 4와 같다.

각 task의 특징을 살펴보면 다음과 같다. 첫째, modem task는 modem device driver에서 전달된 데이터를 기반으로 GCS 시스템에서 전달된 명령을 확인하고, GCS 시스템에 현재 항공기의 상태를 알 수 있는 데이터를 실시간으로 송신해준다. GCS와 통신을 위한 프로토콜을 정의하였다. 항공기와 GCS 간에 통신 프로토콜 그림 4와 그림 5에서 살펴볼 수 있다. GCS로 송신하는 데이터의 크기는 총 55 바이트로 자세 제어를 위한 AHRS 장치의 데이터와 항법을 위한 GPS 데이터, 항공기의 연료상태, 축전지의 상태 그리고 CRC checksum을 포함하여 송신하도록 만들었다. GCS에서 무인항공기로 송신하는 데이터 크기는 총 9 바이트로 비행시 추가적인 임무를 부여하거나 변경하기 위하여 데이터를 송신한다. 둘째, AHRS task는 AHRS device driver에서 바이트 단위로 전달된 데이터를 조합하여 자세각과 각속도의 데이터로 변경하여 항공기 자세 제어에 기반이 되는 데이터를 획득하게 된다. 셋째, GPS task는 GPS device driver에서 전달된 데이터를 사용하여 항법에 필요한 데이터를 추출하고 변환하는 역할을 한다. 이때 추가적으로 필요한 연산은 UTM(Universal Transverse Mercator) 투영법을 사용하여 좌표변환을 수행한다. 넷째, Multi-IO task는 multiboard device driver에서 전달된 데이터를 분석하여 항법과 자세제어를 위한 추가적 역할을 한다. 그 역할은 다음과 같다. 첫째로 시스템에 예측하지 못한 심각한 상태가 발생하여 제어가 불가능한 경우 원격조종 수신기에서 수동조종을 위한 데이터를 획득하여 최종적으로 서보 모타를 제어하는데 필요한 데이터를 처리하는 역할을 한다. 서보 모터는 ActuatorOut task에서 제어하게 된다. 둘째로 제어기에 연결되는 SensorPack에서 고도, 대기압력, 그리고 대기속도의 값들을 확인하는 역할을 한다. 다섯째, display status task는 실제 비행에서는 이용되지 않는 task이고, 비행 전에 지상에서 제어 시스템의 이상 유무를 자체적으로 모니터링 하여 항공기 시스템 동작을 점검한다. 여섯째, flight mode check task는 현재 비행이 무인 비행 상태인지 remote controller를 사용한 수동 조종 상태인지 판단하여, 무인 비행일 경우 자신에게 부여된 무인 비행 알고리즘을 수행하기 위한 task이다. 무인 비행도 그 상태가 분류되어 항공기에 부여된 임무에 따라 다른 알고리즘을 수행하게 된다. 비행 상태는 GCS에서 전달되는 명령에 따라 바뀔 수 있다. 마지막으로 flight record task는 비행 기록을 파일 형태로 저장하여 추후 제어기의 성능 향상을 위한 자료로 쓰인다. 저장되는 비행 데이터는 AHRS 데이터와 GPS 데이터이다. 항공기의 black box의 역할과 같다. 위에서 언급한 8개의 멀티 task가 애플리케이션에서 실행되면서 항공기의 자세를 제어하고, 항공기에 부여된 임무를 수행하도록 되어있다. 항공기라는 특성상 시각적으로 그 상태를 확인할 수 없기 때문에 GCS에서 항공기의 상태를 지속적으로 확인하여 시스템의 이상 유무를 판단하게 된다.

```
Start Modem and AHRS Task Error Process Algorithm
  • Error detection
  • Is Task Blocking?
    • Yes? Set a status bit to Failure
      • Continuous error?
        • Yes? FCS reloads the device driver
    • No? Set a status bit to Success
End
```

그림 7. Modem task와 AHRS task 에러 처리 알고리즘.
Fig. 7. Error process algorithm of modem task and AHRS task.

```
Start GPS Error Process Algorithm
  • Error detection
  • Is Task Blocking?
    • Yes? Set a status bit to Failure
      • Continuous error?
        • Yes? FCS reloads the device driver
    • No? Set a status bit to Success
      • Is GPS time data correct?
        • True? Set a status bit to True
          • Is SatNo data correct?
            • True? Set a status bit to Success
            • False? Set a status bit to Failure
        • False? Set a status bit to False
End
```

그림 8. GPS task 에러 처리 알고리즘.
Fig. 8. Error process algorithm of GPS task.

애플리케이션에는 에러 처리 기능이 작성되어 있다. 에러 처리 기능이 필요한 task는 직렬 통신 디바이스 드라이버에서 데이터를 획득하는 3개의 task와 무인 비행 제어 알고리즘 처리를 하는 task이다. task 자체의 응답성을 보장해 주기 위해서 task 수행 주기를 지정해 주었지만, task가 비행에 필요한 데이터를 정확하게 처리하고 지정된 루틴이 완전하게 수행되었는지 판단할 수 없기 때문에 에러 검출 기능을 추가적으로 작성하였다. 각각의 task는 에러 발생 여부를 task가 Blocking 상태에 빠졌는지 여부에 따라 에러 상태를 판단하게 되고, Blocking 상태에 빠지지는 않았지만 올바른 데이터를 디바이스에서 받아 왔는지 판단하여 에러를 검출하게 된다.

Modem task는 GCS로부터 카운터 값을 전달받아 카운터 값이 증가되지 않으면 에러로 판단하고 modem 데이터가 신뢰성을 보장하지 못하는 것으로 판단한다. task의 blocking 상태가 지속되면 커널에서 modem 디바이스 드라이버를 제거 후 다시 적재하여 task가 modem 장치에서 데이터를 받아올 수 있도록 만들어주었다. AHRS task도 modem task에서 에러를 검출하는 방식으로 에러를 검출한다. task의 blocking 여부를 판단하여 task가 blocking 상태라면 커널에서 AHRS 디바이스 드라이버를 다시 적재하고 AHRS 장치에서 데이터를 받아올 수 있도록 만들어준다. AHRS 장치에 문제가 발생하면 무인 비행 수행이 어렵기 때문에 AHRS 디바이스 드라이버의 에러가 해결되지 않으면 즉각적으로 GCS에서는 무인 비행을 취소하고 수동 조종으로 항공기를 제어해야 된다. GPS task에서 에러 검출은 GPS 디바이스 드라이버의 blocking 상태와 blocking 상태는 아니지만 신뢰성이 낮은 데이터를 확인하는 에러 검출 방법이 작성되어 있다. GPS task는 task가 blocking 여부에 따라 에러를 판단하고, blocking 상태가 아니라면 현재 GPS에서 전달되는 데이터가 정확한 데이터인지를 판단하게 된다. 데이터 판단은 GPS 위성수와 시간 값으로 확인하게 된다. GPS 데이터를 사용하기 위해서는 최소 4개 이상의 위성이 확인되어야 하며, GPS에서 전달되는 데이터에는 UTC (Universal Time Coordinated) 시간 값이 전달되어 시간이 증가되지 않으면 신뢰성이 낮은 데이터로 판단하게 된다. 항공기의 제어를 위해 연결된 전자 장비에서 획득한 데이터를 기반으로 자동 항법과 자세 제어의 역할을 하는 무인 알고리즘은 알고리즘이 해당 루틴을 완전하게 수행하였는지 판단하여 지정된 임무를 수행하지 못하면 에러 여부를 GCS에 알려주고 다른 무인 비행 제어 알고리즘을 수행하게 된다. 그림 7에서는 Modem task와 AHRS task 에러 처리 알고리즘을 살펴볼 수 있다. 그림 8에서는 GPS task 에러 처리 알고리즘을 살펴볼 수 있다.

VI. 지상통제 시스템

GCS는 항공기의 상태를 실시간으로 모니터링 해주는 역할을 한다. 현재 제작된 GCS는 텍스트 기반 사용자 인터페이스로 이루어져 있다. GPS 데이터의 위도, 경도, 고도 데이터와 AHRS 장치의 3축의 각도 그리고 축전지의 잔류량, 항공기 속도, 연료량을 데이터 값으로 확인할 수 있다. 추가적으로 modem, AHRS, 그리고 GPS의 동작 상태를 확인하기 위해 에러 발생 여부를 보여준다.

지상 통제소에서는 항공기로 명령을 전달하여 임무를 부여하거나 명령 변경 등의 이유로 항공기로 데이터를 전달할 수 있다. 현재 비행 모드를 지정하여 미리 입력된 비행 임무를 수행 할 수 있도록 되어 있다. GCS에서 비행 모드를 지정하면 항공기는 각 임무에 해당하는 역할을 수행하게 된다. 만약 항공기가 디바이스의 오류를 전달받으면, 오류복구 모드를 수행하여 디바이스의 오작동을 수정하고, 원하는 임무를 수행할 수 있도록 만들어준다. 그러나 오류가 복구되지 못하면 이를 다시 지상 통제소에 통지하고, 지상 통제소에서는 자동 비행이 불가능한 경우에는 수동으로 항공기를 제어하여 비행체를 회수하게 된다.

제작된 GCS는 2개의 멀티태스크로 구성되어 항공기에서 전달된 데이터를 화면에 보여주는 태스크와 모뎀 통신을 위한 태스크로 구성되어 있다. 송수신 데이터들은 데이터 값의 크기를 최소화하기 위하여 원시 데이터 수준으로 데이터를 송수신 하게 된다. 연산이 필요한 부분은 추가적인 연산 부분을 작성하여 GCS에서 처리하도록 만들어 주었다.

VII. 결론

본 논문에서는 RC(Remote Control) 비행체에 전자 장비를 탑재하여 리눅스 OS를 기반으로 구축한 무인항공기 제어 시스템에 대해 설명하였다. 제어 시스템의 운용에 필요

한 전자 장비를 제어하는 디바이스 드라이버를 개발하여, 애플리케이션이 디바이스 드라이버와 연계되어 제어 시스템이 데이터를 처리 방법을 설명하였다. 애플리케이션은 8개의 task로 이루어져 있어 각 task의 특징을 설명하고 그 역할을 살펴보았다. 애플리케이션은 실시간성을 보장해주고 응답성을 높여주기 위하여 시리얼 디바이스를 사용하는 task의 수행주기를 정하고, 에러 처리 기능을 추가하여 task 수행의 응답성을 보장하였다. 공중에서 운용되는 항공기의 특성상 육안으로 비행체의 상태를 확인하는 것은 불가능하기 때문에 항공기를 실시간적으로 감시하고 제어하는 지상 통제소도 제작하였다.

제어 애플리케이션은 고 정밀도, 빠른 응답성 등의 특징을 가진 장치들을 사용하면 보다 응답성이 좋은 애플리케이션을 구현할 수 있다. 그러나 비용적인 측면에서 하드웨어적인 변경은 효과적이지 못하다. 따라서 소프트웨어적으로 해결 방안을 찾아야 한다. 본 논문에서 구현한 제어 애플리케이션은 다음과 같은 문제가 해결된다면 더욱 향상된 성능을 보일 것으로 기대된다.

FCS가 GCS와 데이터 교환을 위하여 modem device driver가 인터럽트를 처리하여 FCS에서 데이터를 송수신함을 이미 설명했다. 그런데 GCS에서 데이터의 수신율이 떨어진다. FCS에서 송신율을 높여서 GCS에서의 수신율을 개선시킨다면 본 논문에서 구현한 제어 애플리케이션은 더욱 향상된 성능을 보여줄 것으로 기대된다.

참고문헌

[1] E. Charles, Jr. Hall, "A real-time linux system for autonomous navigation and flight attitude control of an uninhabited aerial vehicle," *Digital Avoinics Systems Conference,* 20th, vol 1, pp. 1A1/1~1A1/9, Oct., 2001.

[2] J. S. Dittrich, E. N. Johnson, "Multi-sensor navigation system for an autonomous helicopter," *Digital Avoinics Systems Conference,* 21st, vol. 2, pp. 8C1/11~8C1/19, 2002.

[3] T. J. Cord, S. Newbern, "Unmanned air vehicles: new challenges in design," Aerospace Conference, *IEEE Proceedings,* vol. 6, pp. 2699-2704, March., 2001.

[4] H. Shim, D, H. J. Kim, S. Sastry, "Control system design for rotorcraft-based unmanned aerial vehicles using time-domain system idenfication," *Proceedings of the 2000 IEEE Int. Con. on Control Applications,* pp. 808-813, Sept., 2000.

[5] T. J. Koo, B. Sinopoli, Sangiovanni-Vincentelli, A, "A formal approach to reactive system design: unmanned aerial vehicle flight management system design example," *Proceedings of the 1999 IEEE Int. Symp. on Computer Aided Control System Design,* pp. 522-527, Aug., 1999.

[6] M. Gordon, S. Kondor, E. Corban, D. Schrage, "Rotorcraft Aerial Robot-Challenges and Solutions," *Digital Avionics Systems,* pp. 298-305, Oct., 1993

[7] Y. Hui, C. Zhiping, X. shanjia, W. Shisong, "An unmanned air vehicle(UAV) GPS location & navigation system," ICMMT, pp. 472-475, 1998.

[8] H. Wu, D. Sun, Z. Zhou, "Micro air vehicle: configuration, analysis, fabrication, and test," *Mechatronics,* pp. 108-117, March., 2004.

[9] A. D. Kahn, J. C. Kellogg, "Low complexity, low-cost, altitude/heading hold flight control system," *Aerospace and Electronic System Magazine,* IEEE, vol. 18, pp. 14-18, April., 2003.

[10] 유혁, 이장호, 김재은, 안이기, "저가형 무인 항공기의 자동비행시스템 개발," 제어·자동화·시스템공학논문지, 제10권 제2호, pp. 131-138, 2월 2004.

[11] 박선규, "한국적 무인항공기 개발전략에 관한 연구," 국방대 국방관리대학원 석사학위논문, 2003.

[12] 홍성경, 김태연, 탁민제, "스트랩다운 AHRS를 이용한 무인항공기(RPV) 자동조종장치의 실시간 실물 모의시험," 한국자동제어학술회의논문집, pp. 135-140, 10월 1992.

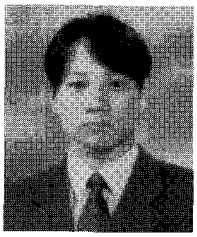
[13] H. Lamela, M. A. Ferreras, A. J. Varo, "Sensor and navigation system integration for autonomous unmanned aerial vehicle applications," Industrial Electronics Society, *IECON '99 Proceedings, The 25th Annual Conference of the IEEE,* vol. 2, pp. 535-540, 29 Nov.~3 Dec., 1999.

[14] C. D. Ozimina, S. K. Tayman, H. E. Chaplin, "Flight control system design for a small unmanned aircraft," *American Control Conference,* vol. 5, pp. 2964-2969, June., 1995.

[15] L. Rubini, J. Corbat, *Linux Device Driver,* O'Reilly, pp. 54-96, 2001.

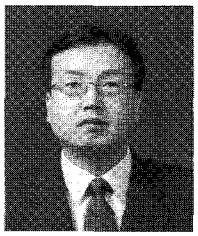
[16] D. R. Butenhof, *Programming with POSIX Threads,* Addison-Wesley, pp. 174-175, 2003.

[17] B. O. Gallmeister, POSIX. 4: *Programming for the Real World,* O'Reilly & Associates, Inc, pp. 41-83, 1995.



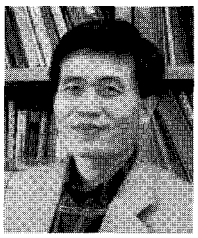
김 명 현

2003년 세종대 컴퓨터공학과 졸업. 2005년 동 대학원(석사). 관심분야는 임베디드 시스템, RTOS.



홍 성 경

1987년 연세대학교 기계공학과 졸업. 1989년 동 대학원(석사). 1998년 Texas A&M Univ. 기계공학(박사). 1989년~2000년 국방과학 연구소 선임 연구원. 현재 세종대학교 기계항공우주공학부 조교수. 관심분야는 지능제어, 강건제어 및 센서응용.



문 승 빈

1985년 명지대 전기공학과 졸업. 1988년 Univ of Michigan 석사. 1993년 Purdue University 박사. 1999년~현재 세종대학교 컴퓨터공학과 부교수. 관심분야는 로보틱스, RTOS, 컴퓨터비전.