리눅스기반 무인항공기 제어 시스템 설계

김명현*, 문승빈*, 홍성경**
*세종대학교 컴퓨터공학과
**세종대학교 항공우주공학과
e-mail:sbmoon@sejong.ac.kr

Controller Design for Unmanned Aerial Vehicle Employing Linux OS

Myoung-Hyun Kim*, Seungbin Moon*, and Sung-Kyung Hong**
*Dept. of Computer Engineering, Sejong University
**Dept. of Aerospace Engineering, Sejong University

요 약

본 논문에서는 PC104 모듈을 탑재하여 만들어진 무인 항공 제어 시스템에 관한 내용을 기술한다. 임베디드 리눅스를 사용하여 제작된 항공기 제어 시스템은 디바이스 드라이버와 제어 애플리케이션으로 구성되어 있다. PC104 모듈에는 비행에 필요한 외부 장치들이 연결되는데, 연결된 장치에서 측정한 데이터를 처리하여 애플리케이션으로 전달해 주는 역할을 하는 디바이스 드라이버를 설명하고, 디바이스 드라이버에서 전달받은 데이터를 기반으로 구현한 애플리케이션에 대한 내용을 설명한다. 또한 향후 시스템 운용에 시뮬레이션 기능 구현의 필요성과 RTOS 적용 가능성을 제시해 본다.

1. 서론

리눅스 OS(Operating System)는 현재 다양한 CPU에 포팅되어 사용되고 있으며, 최근에는 임베디드 시스템 등 여러 시스템 분야에서 응용되고 있다.[1] 임베디스 시스템이란 미리 정해진 특정한 기능들을 수행하기 위하여 컴퓨터 하드웨어와 소프트웨어가 결합된 시스템을 말한다. 임베디드 시스템은 메모리와 CPU 파워 등 제한적인 환경에서 원하는 기능을 최적으로 구현해야 하기 때문에 프로그래밍 기술뿐만 아니라 하드웨어에 대한 폭넓은 지식이 요구된다.

무인항공기 제어 시스템은 비행체 자동화 비행 기술에 의해 외부 운용자의 도움 없이 주어진 경로를 제어 시스템에서 해석하여 비행 가능하도록 개발되고 있다.[2][3] 무인항공기 제어 시스템은 메인 제어기에 외부 장치들(센서, GPS 등)을 연결하여 데이터를 획득하고, 획득한 데이터를 분석하여 최종적으로 항공기 날개를 제어하게 된다.

본 논문에서는 무인항공기 제어 시스템을 임베디드 시스템에 접목시켜 무인 비행체 자동화 비행기술을 바탕으로 임베디드 리눅스[4][5]를 PC104 모듈에 탑재하여 무인 항공이 가능한 제어 시스템[6]을 설계하였다.

본 논문의 구성은 다음과 같다. 2 절에서는 시스템 하드웨어 구성과 각 기능을 설명한다. 3 절에서는 제어 시스템이 운용되는 OS 에 대한 설명과 메인 제어기와 연결되는 외부 장치들에서 데이터를 획득하기 위해 제작된 Device Driver를 설명한다. 4 절에서는 제어 Application의 데이터 처리 과정과 비행 제어를 설명한다. 5 절에서는 현재 제어 시스템 구현 상황을 설명하고, RTAI(Real-Time Application Interface)[7] 구현 가능성과 항공기 시뮬레이션을 통해 제어 시스템의 효율적인 관리에 관하여 기술한다.

2. Hardware Platform

무인항공기를 제어하는 메인 제어기는 3개의

PC104 모듈로 구성되어 있다. 최하층에는 PCM-3350 CPU Board, 중간층에 UMIO-100 Multi-IO Board, 최상층에 PCM-3291 GPS Board로 이루어져 있다.

PCM-3350 모듈의 하드웨어 구성은 다음과 같다. CPU는 Embedded Low power NS Geode GX1-300MHz processor를 사용하고, BIOS는 AWARD 256KB flash 메모리가 내장되어 있다. 시스템 메모리는 128MB SDRAM을 사용하였다. OS가 적재될 저장 공간으로는 256MB 의 비휘발성 Flash 메모리가 보조기억장치로 사용되었다.

PC104 모듈에는 RS-232 통신이 가능한 4개의 포트가 있다. 2개는 외부 Serial Connector로 연결되어 있고, 나머지 2개는 버스라인을 통하여 연결된다. 2개의 Serial connector 는 COM1, COM2 포트로 사용되고, 버스 라인을 통하여 연결되는 포트는 COM3, COM4로 사용된다. COM1 포트에는 UHF Modem이 연결되고, COM2 포트에는 Inertial & Gyro System 장치가 연결된다.

UMIO-100 모듈에는 3개의 외부 장치가 연결되어 있다. PWM input signal로 통신하는 Remote Control Receiver, PWM output signal로 제어하는 Servo Motor, A/D input signal로 통신하는 SensorPack이 있다. PCM-3291 모듈은 RS-232 통신으로 GPS Antenna가 GPS 신호를 수신하면 PCM-3291 모듈이 PCM-3350 모듈로 COM3 포트를 통하여 데이터를 송신해준다.

탑재되는 PC104 모듈의 모습은 그림 1 에서 살펴볼 수 있다. 탑재되는 PC104 모듈들이 외부 버스

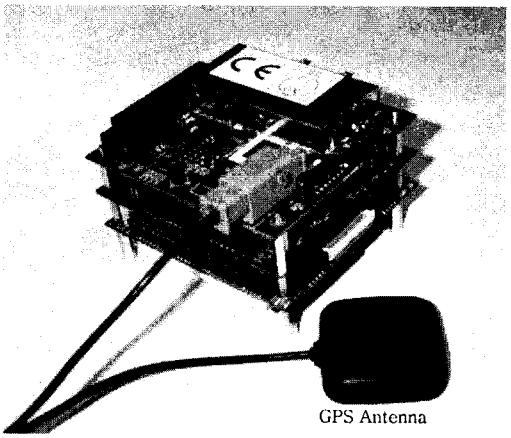

그림 1. 조립된 PC104 제어기 모습

라인을 통해 통신이 가능하도록 조립해 놓은 모습이다. 하단부터 PCM-3350 모듈, UMIO-100 모듈, PCM-3291 모듈의 모습을 볼 수 있다. 그림 2 는 항공기에 탑재될 PC104 모듈들과 각 모듈에 연결될 외부 장치들을 도식화해서 보여주고 있다

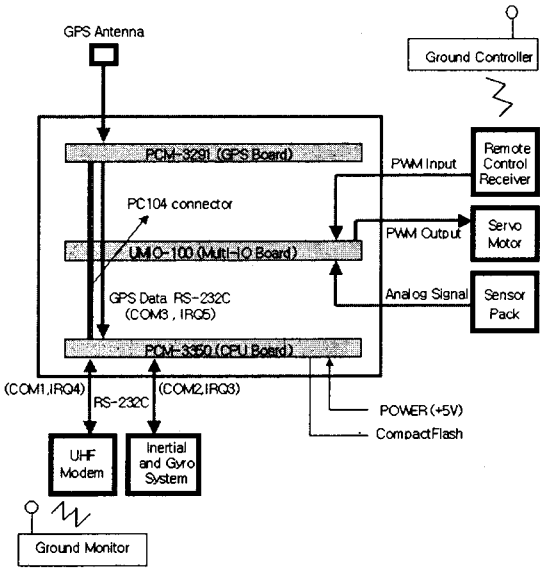

그림 2. PC104 제어기 구성 및 외부장치

3. 제어기용 OS 및 Device Driver의 설계

무인항공기용 제어기에서 사용된 운영체제는 리눅스로서 PCM-3350의 256MB의 Flash 메모리에 이식되었다. 다양한 프로세서가 있지만 무인항공기는 x86계열의 프로세서가 사용되어 Kernel patch 없이 이식되었다. 일반 Desktop 처럼 리눅스 설치가 불가능 하므로 Booting에 필요한 Boot loader를 Flash 메모리에 이식시켰다.[8] 그다음 Kernel 컴파일 후 생성된 zImage를 boot loader의 명령어를 통해서 RAM 영역에 전송하였다. 압축된 Kernel 이미지를 zImage라 하고 그 크기는 약 900KB 정도의 크기이다. Kernel 이미지가 적재된 후 File System을 전송하여 Flash 메모리에 항공기를 운용할 수 있는 하나의 완전한 리눅스 OS 가 구축되었다.

Linux Device Driver[9] 는 크게 Character Device Driver, Block Device Driver, Network Device Driver 로 분류할 수 있다. 무인 항공기에서는 연결되는 외부 장비들은 스트림 기반의 데이터를 송수신 하므로 Character Device Driver 를 사용하였다. 리눅스에서는 이미 많은 Device Driver가 존재하고, UMIO-100 module에 연결되는 외부 장치

이외에 COM1, COM2, COM3 에 연결되는 외부 장치들은 리눅스에서 제공하는 디바이스로 연결할 수 있다.

항공기를 제어하는 Application에서 효율적인 데이터 처리를 위한 인터럽트 핸들러 구현이 요구되었다. 그러나 리눅스에서는 User Application 영역에서 인터럽트 핸들러 구현을 지원하지 않는다. 그래서 장치에 적합한 인터럽트 처리를 위해서 Device Driver를 제작하였다. 또한 UMIO-100 module에 연결되는 장치들은 리눅스에서 Device Driver에서 제공하지 않기 때문에 사용이 가능하도록 Device Driver를 제작하였다. 무인항공기에 사용되는 Device Driver는 4가지로 제작되었다. UHF Modem과 통신하는 Modem Device Driver, Inertial & Gyro System과 통신하는 Gyro Device Driver, GPS 데이터를 수신하는 GPS Device Driver, UMIO-100 에 연결되는 장치들과 통신하는 Multiboard Device Driver가 있다.

UHF Modem, Inertial & Gyro System, GPS Receiver 장치에 적합하게 제작된 Device Driver들은 인터럽트를 발생시켜 데이터를 수신하고, 수신된 데이터를 User 영역으로 전달해주게 된다. 인터럽트 발생시 1 Byte를 커널 메모리에 복사하는데 UHF Modem Device가 0.8ms, Inertial & Gyro System Device가 0.2ms, GPS Receiver Device가 1.7ms의 시간이 소요된다. ISR(Interrupt Service Routine)에서는 외부 장치로부터 데이터를 복사하기 위한 일 이외에 다른 작업은 하지 않는다.[9]

UMIO-100 module을 사용하기 위해서 PWM input/output, A/D input, D/A output, RS232 통신을 Device Driver가 처리한다. 데이터를 처리하기 위해서 보드내의 해당 레지스터를 선택하여 각 채널에 값을 읽거나 값을 써주었다. 채널을 선택하여 데이터 값을 읽어나 쓸 때 2 Byte 의 값을 써주어야 되는데, Read/Write 레지스터의 크기가 1 Byte 이므로 Low Byte, High Byte를 조합하여 데이터를 추출할 수 있었다.

유닉스 계열의 특징 중 하나는 장치도 하나의 파일로서 다룰 수 있다는 점이다. Device Driver는 이것을 가능하게 해주는 교량적 역할을 한다. 무인항공기에 사용하기 위해 제작된 Device Driver도 파일 인터페이스를 통하여 User Application이 장치로부터 데이터를 획득할 수 있게 했다. 일반적으로 파일을 제어하는 함수는 open, close, read, write, ioctl 등이 있다. 무인항공기에 적재된 Device Driver들은 ioctl 함수를 사용하여 제어할 수 있도록 제작되었다. 그림 3 은 제어기 S/W 구조도를 보여주고 있다.

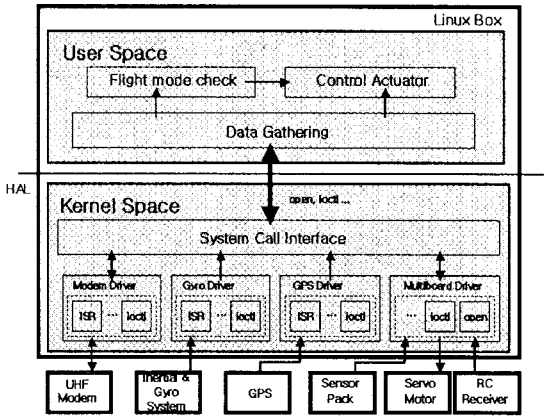

그림 3 제어기 S/W 구조도

4. 무인항공기 제어 Application의 설계

무인항공기 Application은 Device Driver에서 전달 받은 데이터를 사용하여 지상국에서 입력한 비행 모드에 해당하는 임무를 수행한다. 그리고 현재 자신의 상태와 획득한 데이터를 지상 모니터링 컴퓨터에 송신하여 현재 항공기의 상태를 보여준다.

제어기가 동작되고 Application이 실행되면 항공기는 Device Driver 에서 전달해 주는 데이터를 읽어오는데, 그 데이터는 다음과 같다. Inertial & Gyro System 데이터, GPS 데이터, PWM input 데이터, A/D input 데이터, Modem 데이터가 있다.

Inertial & Gyro System 에서는 진동 센서, 가속도계, 자력계 데이터를 보내준다. Application 은 항공기 자세의 균형을 유지할 수 있도록 데이터를 읽어온다. GPS 에서는 고도, 경도, 위도 등의 데이터를 보내준다. 항공기의 위치를 알수 있도록 GPS 에서 보내주는 데이터를 읽어온다. PWM input 은 무선 조종기에서 보내주는 데이터를 읽어온다. 무선 조종기로는 직접적으로 항공기의 제어 장치와 연결할 수 없다. 무선 조종기와 항공기 날개를 제어하는 모터 또는 그 밖에 다른 제어 장치들은 PC104 모듈에서 데이터 이동 후 제어가 가능하기 때문이다. A/D input은 공기압력, 풍속, 전원 잔류량, 연료량을 아날로그 신호로 읽어온다. Modem은 지상국과 항공기와 통신을 위한 교량적 역할을 한다. 항공기는

현재 항공기의 상태를 지상국으로 송신하고, 지상국은 항공기에서 송신된 데이터를 분석하여 항공기의 상태를 알 수 있다. 또 지상국은 항공기에 명령을 송신하여 항공기가 부여된 특정 임무를 수행하도록 한다.

무인항공기 Application은 항공기가 올바른 자세를 유지하는데 가장 초점이 맞추어져 있다. 올바른 자세를 유지하기 위하여 Application은 Device Driver에서 읽어온 데이터를 수학적인 계산을 통하여 Application에서 사용 가능하도록 만들어 주고, 현재 항공기가 수행하고 있는 임무를 지속적으로 확인해야 한다. 항공기가 현재 수행하고 있는 임무는 Device Driver 에서 읽어 온 데이터와 비교함으로서 확인할 수 있다. 항공기에 발생하는 결함에 대비하여 획득된 데이터는 파일 형태로 저장하게 된다. 이것은 곧 비행 기록이 된다. 저장된 비행 기록은 추후 제어기 성능 향상을 위한 자료가 된다.

5. 결론 및 향후 연구과제

항공기 제어 Application은 크게 2개의 부분으로 분리할 수 있다. 하나는 커널 영역에서 동작하는 Device Driver가 있고, 다른 하나는 User 영역에서 동작하는 제어 Application이 있다. 현재 Modem Device Driver를 제외한 Device Driver들은 Application에서 사용 가능하도록 제작되어 있고, User 영역에서 동작하는 제어 프로그램은 각각의 Device Driver에서 읽어온 데이터를 비행에 필요한 데이터로 처리해주는 루틴만 제작되어 있다.

항공기 제어 Application은 외부 장치에서 읽어온 데이터를 기반으로 현재 항공기의 상태를 확인하여 운용자가 요구하는 비행이 가능하도록 처리하는 시스템이다. 제어 시스템은 어떻게 더 정밀하게 시스템을 제어할 것인가에 초점이 맞춰져야 한다. 데이터 처리 응답성 향상을 일례로 들 수 있다. 이를 위해서는 RTOS(Real-Time Operating System) 적용이 필요하다.

일반적인 운영체제는 ms 단위의 정확성으로 제어를 하지만, 통신기기나 정밀 제어의 경우 수십 μs 단위로 정확하게 시간을 측정해야 하는 경우가 많다. 또한 인터럽트가 발생하였을 때 해당 Processor나 Task 가 수십 마이크로 초 이내로 동작해야 하는 경우에는 일반적인 범용 운영체제로는 이러한 조건을 만족할 수가 없다. 무인항공기는 복잡성을 최소화하여 인터럽트 및 Task 의 실시간성을 보장하여 RTOS 의 특징을 만족시켜야 할 필요성이 있다. RTOS 의 특징은 대표적으로 짧은 시간에 외부 신호에 응답성하는 빠른 속도와 수행하는 Task 의 완료시간에 명확함에 있다. 리눅스 커널 patch 만으로 RTOS 사용 가능한 RTAI 를 향후 적용할 것을 목표로 한다. RTAI 는 리눅스 커널을 기본으로 선점형 시스템을 지원해주는 기능을 제공한다.

무인항공기는 지상에서 운용되는 시스템처럼 항공기 자체를 시각적으로 살펴보는 것은 불가능하다. 항공기가 비행을 수행하면서 시스템 운영자가 항공기에서 알려주는 데이터 값만으로 항공기의 상태를 확인하게 된다. 향후 항공기의 상태를 시각적으로 직접 모니터링 가능할 수 있도록 GUI 구현도 목표로 한다.

참고문헌

[1] 노영욱, 변정용, 이정배, "임베디드 리눅스 개발 도구 기술동향", 한국정보처리학회지, 제9권 제1호, pp35-42, 1월 2002

[2] 홍성경, 김태연, 탁민재, "스트랩다운 AHRS를 이용한 무인항공기(RPV) 자동조종장치의 실시간 실물 모의시험", 한국자동제어학술회의논문집, pp135-140, 10월 1992

[3] 박춘배, 최가영 "무인항공기 탑재전자장치" 제어자동화시스템공학회지, 제7권 제5호, pp28-36, 2001

[4] 박윤미, 성영락, 이철훈, "임베디드 리눅스 시스템 설계 및 구현", 정보과학회 춘계학술대회, 제30권 제1호, 4월 2003

[5] C. Hollabaugh, *Embedded Linux Hardware, Software, and Interfacing*, Addison-Wesley, 2002

[6] 윤석준, "무인항공기 비행제어시스템", 제어자동화시스템공학회지, 제5권 제6호, pp26-32, 2001

[7] DIAPM RTAI, *RTAI Programming Guide 1.0*, Lineo, 2000

[8] 최용식, 이상락, 신승호, "Real-Time Linux에서 소규모 파일 관리를 위한 파일 시스템의 설계", 한국정보처리학회 추계학술대회논문집, 제9권 제2호, 11월 2002

[9] L. Rubini, J. Corbet, *Linux Device Driver*, O'REILLY, 2001