PX4 기반 무인비행체의 GPS 시간동기화 오차 최소화에 관한 연구

심후엽, 김황남

고려대학교 전기전자공학과

{hooyp, hnkim}@korea.ac.kr

A Study on the Minimization of GPS Time Synchronization Errors of Unmanned Aircraft Based on PX4

Hooyeop Shim, Hwangnam Kim

School of Electrical Engineering, Korea University

요 약

본 논문은 PX4 펌웨어 기반의 무인비행체들이 GPS 모듈로부터 수신한 시간정보를 이용하여 비행 컨트롤러의 시간을 동기화 할 때 발생하는 오차를 측정하고 이를 최소화하기 위한 방법을 제안한다. GPS 모듈이 GPS, GLONASS와 같은 GNSS 시스템으로부터 수신하는 시간정보는 수신 환경에 따라 수십~수백 나노초의 오차가 발생하지만 시간정보가 각 무인비행체의 비행 컨트롤러에 전달되고 내부 운영체제(RTOS)의 시간이 업데이트 될 때까지 전송 지연, 오실레이터의 부정확성 등 다양한 요인이 작용하게 된다. 결국 무인비행체에 GPS 모듈을 장착하더라도 수~수십 밀리초의 시간동기화 오차가 발생하게 된다. 본 논문에서는 GPS 모듈과 비행 컨트롤러의 각종 파라미터를 일치화하고 PX4 펌웨어 코드 수정, RTOS(NuttX OS) 설정 변경 등을 통해 무인비행체 간 발생하는 평균 시간동기화 오차를 약 42.9% 감소시킬 수 있었다.

Ⅰ. 서 론

최근 산업자동화, 자율주행, IoT 등의 기술이 발전함에 따라 실시간, 저지연 통신에 대한 수요가 급증했고 deterministic한 패킷 기반 네트워크를 구축하기 위해 Time Sensitive Network(TSN) 관련 표준이 제정됐으며 관련 기술이 발전해오고 있다[1].

무선 네트워크 분야에서도 이러한 TSN 개념을 적용한 Wireless TSN을 구축하고자하는 움직임이 있어왔으며 기존 TSN과 유사한 수준의 정밀도를 가지도록 시간동기화가 이루어져야만 유무선 TSN 링크 및 노드가 혼재되어 있는 전체 네트워크를 효과적으로 통합할 수 있고 중앙 컨트롤러에 의해 TSN의 핵심 기능이 작동하게 되며 초저지연, 초정밀 네트워크를 구축할 수 있다[2].

다수의 무인비행체가 군집을 이루고 무선 네트워크의 노드로서 동작할 때, 자세 제어, 군집대형 변경, 임무 전달 및 수행 등의 제어 패킷이 최대한 손실되지 않고 모든 노드에 전달되어야 하므로 이 경우에도 노드 간 시간이 매우 정밀하게 동기화되어야 한다. GPS 모듈이 부착된 무인비행체의 경우, 가장 효율적인 시간동기화 방법은 WiFi나 Zigbee 등의 다른 통신수단을 이용하지 않고 GPS 모듈이 수신하는 위성의 시간을 기준으로 Absolute 시간동기화를 수행하는 것이다.

그러나 GPS의 시간정보가 각 무인비행체의 비행 컨트롤러에 전달되고 내부 운영체제(RTOS)의 시간이 업데이트 될 때까지 전송 지연, 오실레이터의 부정확성 등 다양한 요인이 작용하게 되어 무인비행체에 GPS 모듈을 장착하더라도 수~수십 밀리초의 시간동기화 오차가 발생하게 된다. 본 논문에서는 GPS 모듈과 비행 컨트롤러의 각종 파라미터를 일치화하고 PX4 펌웨어 코드 수정 및 비행 컨트롤러의 운영체제의 설정을 변경함으로써 무인비행체 간 발생하는 시간동기화 오차를 최소화하는 방안을 제안하고 실제 드론을 이용한 테스트베드를 구축하여 시간동기화 오차를 측정하며 성능검증을 수행한다.

Ⅱ. 기본 구성

이 장에서는 GPS 모듈을 주로 사용하는 무인이동체인 드론이 GPS 위성으로부터 수신한 시간정보를 기준으로 동기화를 수행했을 때 드론 간에 발생하는 시간동기화 오차를 최소화하기 위한 방안을 세부적으로 기술한다. 본 논문에서 제안하는 방안을 실제 드론에 적용하였으며 해당 드론을 구성하는 여러 H/W 및 S/W에 대한 세부정보는 표 1과 같다.

표 1. 드론의 H/W 및 S/W에 대한 세부정보

Component	Description
Drone Model	DJI F550
Flight Controller(MCU)	Pixhawk1 (Cortex M4, STM32F427
Firmware Version	PX4 1.9.0
GPS Module	Here+(U-blox M8P)

Ⅲ. GPS 모듈 및 비행 컨트롤러 파라미터 설정

위성시간의 매초가 시작될 때 GPS 모듈은 3번 핀에서 TIMEPULSE 신호를 발생시키며 그와 동시에 추정된 위치정보를 계산하고 시리얼 포트를 통해 Navigation 메시지를 비행 컨트롤러로 전송한다[3]. 따라서 모든 드론은 위성시간의 매초가 시작될 때를 기준으로 GPS 모듈로부터 시간, 위치정보가 포함된 메시지를 수신할 수 있으며 이를 위해 GPS 모듈의 파라미터 중 UBX-CFG-TP5의 설정을 표 2와 같이 변경해야 한다.

표 2. UBX-CFG-TP5 파라미터 설정

Classification	Name	Value
Parameters	freqPeriod	1000000(us)
	pulseLenRatio	100000(us)
Flags	active	1
	isFreq	0
	isLength	0
	lockGnssFreq	1
	alignToTow	1
	polarity	1

비행 컨트롤러 파라미터 중 SER_GPS1_BAUD 설정이 최초에는 Auto로 되어 있는데 GPS 모듈의 UBX-CFG-PRT 파라미터의 Baudrate와 함께 서로 115200으로 일치화한다면 서로 같은 수준의 delay가 발생하므로 시간 동기화에 미치는 영향을 최소화 할 수 있다.

Ⅳ. PX4 펌웨어 코드 분석 및 수정

PX4 펌웨어에서는 termios.h의 read 함수가 비교적 많은 컴퓨팅 자원을 필요로 하기 때문에 시스템의 성능저하를 방지하고자 MinimumBytes를 32로 설정하고 읽을 수 있는 데이터의 크기가 이보다 작으면 Baudrate을 기반으로 식 (1)과 같이 Sleeptime을 계산하고 그만큼 대기한다[6].

$$MinimumBytes \times 10^6 / (Baudrate / 10) \quad (1)$$

GPS 모듈과 비행 컨트롤러의 파라미터를 설정할 때 115200의 속도로 Baudrate을 일치시켰으므로 Sleeptime은 2777us로 계산된다. 그러나 펌웨어 소스코드 컴파일 시 초기 설정한 NuttX OS의 1 Tick의 간격이 1000us로 되어 있기 때문에 펌웨어에서 마이크로 초 단위까지 정밀하게 카운팅하며 대기할 수 없다. 그러므로 식 (1)을 이용하여 계산된 Sleeptime을 사용하지 않고 마이크로 초 단위의 수를 버린 임의의 상수인 5000(us)으로 수정하였다.

Ⅴ. NuttX OS 코드 분석 및 수정

NuttX OS에서는 MCU가 제공하는 SYSTICK 기능을 이용하여 1 Tick에 설정된 시간이 지나면 Exception이 발생시키고 Elaspsed Time을 계산하는데 사용되는 uint32_t 타입의 g_system_timer 변수 값을 1씩 증가시킨다. 1 Tick의 간격은 CONFIG_USEC_PER_TICK의 값에 따라 적용되는데 최초 설정값은 1000us(1ms)로 되어있다. 해당 설정은 Firmware/boards/px4/fmu-v2/nuttx-config/nsh/defconfig 파일에 저장되어 있고 수정 가능하다. 초기 설정 값인 1000us로는 시간 오차가 마이크로 초 단위로 발생할 수가 없기 때문에 CONFIG_USEC_PER_TICK의 값을 50, 100, 200, 500, 1000 등 다양하게 변경해가며 소스코드를 컴파일 했다.

또한, Cortex M4의 경우 Nested Vectored Interrupt Controller(NVIC) 기능을 제공하여 240개의 서로 다른 인터럽트와 Exception을 효율적으로 관리할 수 있고 각 인터럽트에 0 ~ 255 사이의 값을 우선순위로 부여할 수 있다[4]. SYSTICK Exception은 System Handler에 의해 처리되기 때문에 System Handler Priority Register에 우선순위 값이 저장되며 Systick Exception의 우선순위는 System Handler Priority Register3에 저장된다. -1, -2, -3 값을 우선순위를 가진 Exception은 해당 값을 변경할 수 없으나 이 외에 모든 Exception의 우선순위는 변경할 수가 있다.

NuttX OS가 부팅되며 SYSTICK의 우선순위를 0x80으로 초기화하는데 Firmware/build/px4_fmu-v2_default/NuttX/nuttx/arch/arm/src/stm32/stm32_irq.c에 정의된 up_prioritize_irq(15, 0x00) 함수를 호출하여 PRI_15 필드에 전부 0을 할당함으로써 SYSTICK의 우선순위를 설정가능한 범위 내에서 가장 높게 설정하였다.

Ⅵ. 실험 결과

본 실험에서 노트북에 두 대의 Pixhawk를 USB 케이블로 연결하며 노트북에서는 두 개의 스레드를 생성하고 각 스레드는 시리얼 포트를 통해 mavlink 프로토콜로 OS의 Timestamp값이 포함된 PING 메시지를 PX4 펌웨어로 전송한다. 펌웨어는 메시지 수신 시간과 PING 메시지에 포함된 Timestamp값을 배열에 저장하며 1001번째 메시지 수신 시 실험을 종료하고 배열에 있는 데이터를 SD카드에 기록한다.

두 대의 드론의 GPS 시간 동기화 오차를 측정한 결과는 아래 표 3과 같다. GPS 모듈과 비행 컨트롤러의 파라미터를 설정하고 펌웨어 코드를 수정한 경우가 'STEP Ⅰ'에 해당되고 두 대의 Pixhawk 사이 시간 오차는 평균 6.78ms로 측정되었다. 이에 더해 NuttX OS의 코드를 수정하여 SYSTICK의 우선순위를 조정하고 1 Tick의 시간 간격을 다르게 설정한 경우가 표 4의 'STEP Ⅱ'에 해당되며 가장 작은 평균 시간 오차는 3.87ms로 측정되었다. 인터럽트는 설정값 변경 전 보다 20배 더 많이 발생했지만 우선순위가 높게 설정되었기 때문에 더 정밀하게 1 Tick을 카운팅 할 수 있었으며 그 결과, 약 42.9% 정도로 평균 시간 오차가 감소했다는 것을 알 수 있다.

표 3. 드론 두 대의 GPS 시간 동기화 오차 측정 결과

	STEP Ⅰ	STEP Ⅱ			
CONFIG_USEC_PER_TICK	1000	50	100	200	500
Mean Time Error	6.78ms	3.87ms	4.00ms	7.61ms	8.28ms

Ⅶ. 결론

본 논문에서 GPS 모듈과 비행 컨트롤러의 각종 파라미터를 일치화하고 PX4 펌웨어 코드 수정, RTOS (NuttX OS) 설정 변경 등을 통해 무인비행체 간 발생하는 평균 시간동기화 오차를 약 42.9% 감소시킬 수 있었다. 그러나 동시에 Pixhawk1 MCU의 최대 클럭은 168Mhz 밖에 되지 않고 High Speed External(HSE) 오실레이터의 정밀도 또한 -500ppm에서 500 ppm[5]으로 부정확하기 때문에 GPS 시간을 기준으로 비행 컨트롤러의 시간을 동기화 했음에도 밀리 초 단위의 오차를 보인다. 이와 같은 비행 컨트롤러 H/W의 한계를 뛰어넘어 더욱 정밀한 시간동기화를 달성하기 위해, 비교적 성능이 뛰어난 SBC에 직접 GPS의 Pulse를 전달함으로써 시간동기화 오차를 최소화하는 연구를 진행할 계획이다.

ACKNOWLEDGMENT

본 연구는 방위사업청과 국방과학연구소가 지원하는 군집형 무인 CPS 특화연구실 사업의 일환으로 수행되었습니다.(UD190029ED)

참 고 문 헌

[1] Mildner, Alexander. "Time Sensitive Networking for Wireless Networks-A State of the Art Analysis." Network 33 (2019).

[2] Mahmood, Aneeq, et al. "Clock synchronization over IEEE 802.11—A survey of methodologies and protocols." IEEE Transactions on Industrial Informatics 13.2 (2016): 907-922.

[3] u-blox8 / u-blox M8 Receiver description Including protocol specification(2020), u-blox

[4] Cortex M4 Devices Generic User Guide(2010), ARM

[5] STM32F427xx STM32F429xx DataSheet(2018), STMicroelectronics

[6] https://github.com/PX4/Firmware