임베디드 시스템 개발을 위한 가상 ECU 기반 테스트 환경 설계 및 실제 액추에이터 연동 실험

이하림 [1], 김형래 [2], 조정훈 [3]
[1] 경북대학교 전자전기공학부 석사과정
[2] 경북대학교 전자전기공학부 박사과정
[3] 경북대학교 전자전기공학부 교수

hw05165@knu.ac.kr, hrsin95@ knu.ac.kr, jcho@ knu.ac.kr

Virtual ECU Based Test Environment for Embedded Systems: Design and Real-World Actuator Integration

Ha-Rim Lee[1], Hyeong-Rae Kim[2], Jeong-Hun Cho[3]
[1]Dept. of Electronic and Electrical Engineering, Kyung-Pook National University
[2]Dept. of Electronic and Electrical Engineering, Kyung-Pook National University
[3]Dept. of Electronic and Electrical Engineering, Kyung-Pook National University

요 약

본 논문은 Renode 가상화 환경을 활용하여 임베디드 제어 소프트웨어를 설계하고 실제 액추에이터와 연동한 하드웨어-소프트웨어 통합 시뮬레이션 구조를 제안합니다. 실험을 통해 가상 제어기의 성능은 실제 제어기와 유사한 수준임을 확인하였으며, USB-CAN 통신에서 발생하는 지연과 같은 몇 가지 한계점도 함께 분석하였습니다. 본 연구는 가상 환경 기반 임베디드 시스템 개발 접근법의 가능성을 제시하고, 그리고 향후 실시간성 향상을 위한 개선 방향을 제안합니다.

1. 서론

임베디드 시스템 개발 과정에서 가상화 기술은 중요한 역할을 합니다. 특히 더 빠른 연산 속도와 높은 안정성이 요구되는 분야의 경우 ECU(Electronic Control Unit)의 비용은 증가하게 됩니다. 이러한 상황에서 가상화를 활용한 개발 방식은 단순한 비용 절감을 넘어, 개발 과정의 유연성과 반복 가능성을 높여 개발 효율성까지 향상시키는 이점을 제공합니다.

하지만 가상화 환경에서는 모터와 같은 실제 액추에이터의 동작을 모델링하는 것에 한계가 있습니다. 본 논문에서는 Renode 가상화 환경을 활용하여 소프트웨어 중심의 제어 시스템을 설계하고, 이를 실제 액추에이터와 연동하여 검증 가능한 테스트 환경을 설계하여 가상화 환경의 한계를 해결하도록 하였습니다. 특히 제어기 소프트웨어를 가상화하여 사전 개발하고, 실제 하드웨어 액추에이터와 통신하는 구조를 구현함으로써, 시제품 제작 이전 단계에서도 시스템 동작을 실험적으로 확인할 수 있음을 보였습니다.

실험 결과, 가상 제어기의 제어 성능은 실제 하드웨어와 유사한 수준을 보였으나, USB-CAN 인터페이스를 통한 통신 과정에서 통신 지연 등의 중간 통신 장비의 한계를 확인하였습니다. 이러한 지연은 특히 빠른 주기를 가진 메시지 처리 및 실시간 제어 응답성을 요구하는 시스템에서는 고려되어야 할 요소입니다.

결과적으로 가상화 기반 임베디드 소프트웨어 개발과 테스트, 그리고 하드웨어-소프트웨어 통합 시뮬레이션(co-simulation) 구조의 가능성을 제시합니다. 또한 실험을 통해 확인된 한계는 향후 보다 정밀한 실시간 장비 도입 및 타이밍 보정 기법의 적용을 통해 개선 가능성을 보입니다.

2. 배경지식

I. Renode

Renode[1]는 Antmicro 에서 개발한 오픈소스 하드웨어 에뮬레이터로, 실제 임베디드 시스템 환경을 소프트웨어적으로 가상화하여 재현할 수 있도록 지원하는 프레임워크입니다. 다양한 종류의 마이크로컨트롤러(MCU), 주변 장치(Peripherals), 센서, 버스 등을 구성하여 복잡한 SoC(System-on-Chip) 구조를 소프트웨어 상에서 시뮬레이션할 수 있습니다. 특히 Renode 는 네트워크 및 I/O 인터페이스를 가상화하고, 다중 노드 시뮬레이션과 하드웨어 수준의 이벤트 추적 및 디버

깅 기능을 지원하므로, 실제 하드웨어 없이도 개발 초기 단계에서 시스템의 동작을 검증하고 분석할 수 있는 장점이 있습니다. 본 논문에서는 Renode 를 활용하여 가상 ECU 환경을 구성하고, 실제 액추에이터와 연결하여 동작을 재현하는 하드웨어-소프트웨어 통합 시뮬레이션(co-simulation) 환경을 구축하였습니다. 이를 통해 실물 ECU 없이 제어 알고리즘의 검증 및 개발 효율성을 향상시킬 수 있는 방법을 제시합니다.

II. uCAN 3.0 하드웨어

시스템베이스(Sysbas)의 uCAN 3.0[2] 하드웨어는 USB 인터페이스를 통해 CAN(Controller Area Network) 버스에 연결할 수 있도록 설계된 USB to CAN 컨버터입니다. 이 장치는 복잡한 설정 없이도 CAN 네트워크에 쉽게 접속하여 다양한 장치의 제어, 모니터링, 데이터 통신을 수행할 수 있도록 지원합니다. 본 논문에서는 시스템베이스에서 제공하는 샘플 코드를 활용하여 Renode 의 CAN 프레임을 실물 CAN 버스와 연결될 수 있도록 인터페이스를 구현하였습니다.

3. 구현

기존 Renode 에뮬레이터는 CAN 통신을 시뮬레이션하기 위해 Linux 기반의 환경에서 사용 가능한 SocketCAN[3] 인터페이스를 채택하고 있습니다. 하지만 이 논문에서는 실제 하드웨어와의 CAN 통신을 사용하기 때문에, Windows 환경에서 Renode 와 uCAN 3.0 하드웨어 간의 연결을 위한 인터페이스를 별도로 구현하였습니다.

Renode 내부에서 사용되는 CAN 메시지 포맷과 uCAN 3.0 에서 사용되는 CAN 메시지 포맷이 다르기 때문에 서로 간의 변환과정이 필요했습니다. Renode CAN 포맷에는 CAN 메시지 ID, Data Payload, Extend Format 여부, CAN FD 여부, Bit rate switch 여부의 정보를 포함하는 객체로 구성되어 있습니다. 반면, uCAN 하드웨어는 ASCII 기반의 문자열 형식으로 CAN 메시지를 주고받습니다. 따라서 Renode 에서 생성된 CAN 메시지를 uCAN 에서 요구하는 ASCII 기반 프레임으로 변환해야 합니다. 변환된 메시지는 Windows 소켓을 통해 uCAN 의 Socket Server 로 송신되며, uCAN 소프트웨어는 이를 수신한 뒤 USB 를 통해 하드웨어에 전달, 실제 CAN 프레임으로 다시 변환되어 물리적인 CAN Bus 에 전송됩니다. 반대의 경우도 동일한 절차를 따릅니다. 물리적인 CAN Bus 에서 수신된 메시지는 uCAN 하드웨어에 의해 ASCII 포맷으로 변환되고, 이후 소켓 통신을 통해 Renode 로 전달됩니다. Renode 에서는 이를 내부 포맷으로 재변환하여 시뮬레이션에 활용합니다. 추가로 송수신된 모든 CAN 메시지를 uCAN 소프트웨어 측에서 로깅할 수 있도록 구현함으로써, 시뮬레이션 결과에 대한 분석이 가능하도록 하였습니다.

4. 실험 환경 구축

Renode 상의 가상 ECU 가 실제 하드웨어 기반의 ECU 와 통합 시뮬레이션(co-simulation) 환경에서 정상적으로 동작 가능한지를 검증하기 위해 두 가지 실험 시나리오를 구성하였습니다. 각 실험에서는 두 개의 ECU 가 사용되며, 하나는 실물 엔코더 모터와 연결되어 이를 구동하는 '동작 ECU', 다른 하나는 PID 제어 알고리즘을 수행하는 '제어 ECU'로 구성됩니다. 이 두 ECU 는 CAN 통신을 통해 제어 신호와 피드백 신호(모터 속도)를 상호 교환합니다.

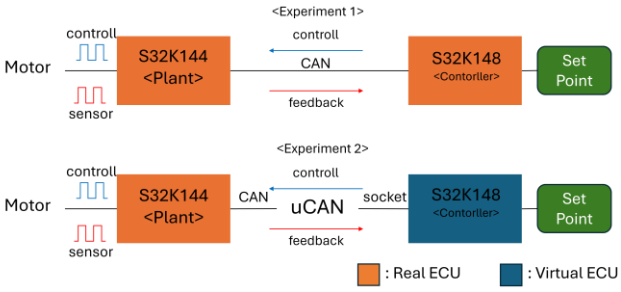
두 실험의 주요 차이는 제어 ECU 가 실제 하드웨어인지, 또는 Renode 상에서 구동되는 가상 ECU 인지를 기준으로 합니다. 이를 통해 가상 ECU 기반 제어의 유효성과 정확성을 비교 평가할 수 있도록 하였습니다.

동작 ECU 로는 S32K144 보드를, 제어 ECU 로는 S32K148 보드를 사용하였습니다. Renode 가상화 환경에서는 각 보드의 스펙에 맞게 주변장치들에 대한 레지스터들에 대한 동작과 정보들을 미리 구현할 필요가 있습니다[4]. S32K144 보드에서는 모터를 구동하기 위해 PWM 신호를 사용하며, S32K144, S32K148 둘 모두 CAN 모듈을 사용합니다.

PID 제어 시나리오는 총 10 초 동안 진행되며, 목표 속도는 2~4 초 구간 및 6~8 초 구간에 걸쳐 설정되며 나머지 구간에서는 0 으로 설정하였습니다. 제어 ECU 는 5ms 주기로 PID 연산을 수행하고, 그 결과를 50ms 간격의 CAN 메시지로 동작 ECU 에 전송합니다. 동작 ECU 는 실물 모터의 엔코더 센서를 통해 속도를 측정하고, 이를 역시 50ms 주기로 CAN 메시지를 통해 제어 ECU 로 피드백합니다.

각 실험은 아래의 두 가지 구성으로 진행:

- 실험 1: 제어 ECU 와 동작 ECU 모두 실제 하드웨어로 구성
- 실험 2: 제어 ECU 는 Renode 상의 가상 환경에서 구동, 동작 ECU 는 실물 하드웨어 사용

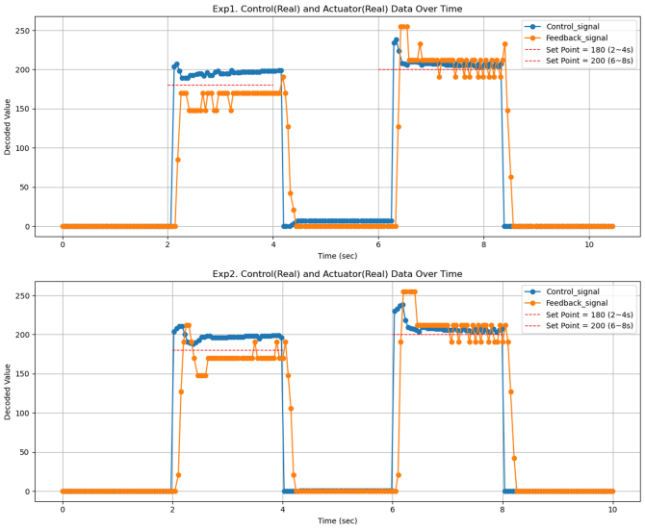


(그림 1) 실험 환경 예시.

5. 실험 결과 분석

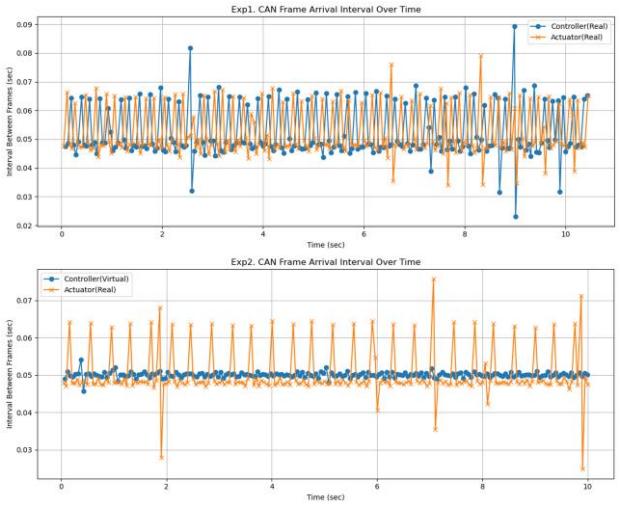
그림 2 는 실제 제어 ECU 와 Renode 가상 제어 ECU 가 각각 제어 신호를 생성하고, 이를 실물 액추에이터에 연결된 동작 ECU 에게 전달하여 피드백 신호를 수신한 결과를 나타냅니다. 이를 통해 가상 제어

ECU 와 실제 제어 ECU 의 동작이 유사하게 나타남을 확인할 수 있으며, 이는 Renode 기반 가상 환경과 실물 시스템 간의 CAN 통신이 정상적으로 수행되고 있음을 알 수 있습니다. 단, 실험에서는 다음과 같은 한계점도 관찰되었습니다.



(그림 2) 제어기와 액추에이터 간 신호값 변화 비교.

그림 2 의 Exp.1 결과에서는 목표치가 설정된 2~4 초, 6~8 초 구간이 전체적으로 오른쪽으로 편향되어 있는 것이 확인됩니다. 이는 uCAN USB-CAN 인터페이스 장비가 50ms 이하 주기의 메시지 처리 시 지연이 발생하고, 해당 지연이 로그 수집 과정에서 누적되었기 때문입니다. 그럼에도 불구하고 두 실험에서 기록된 각 CAN 프레임의 총 수가 두개의 ECU 가 50ms 주기의 메시지를 10 초간 송수신할 때인 400 개임을 바탕으로, 메시지 유실은 발생하지 않았음을 알 수 있습니다.



(그림 3) CAN 프레임 도착 간격 비교.

그림 3 은 시간에 따른 CAN 프레임 도착 간격을 나타냅니다. Exp.1 에서의 두 MCU 와 Exp.2 에서의 동작 MCU(Actuator)에서, CAN 프레임 간격이 진동하는 양상이 확인됩니다. 이는 uCAN 장비가 생성하는 로그의 타임스탬프 정확도가 낮거나, 내부 처리 로직에 따라 USB 전송 지연이 발생했기 때문으로 해석할 수 있습니다. 반면, Exp.2 의 가상 제어 MCU(Controller) CAN 프레임 로그는 매우 일정한 간격을 보이는 듯하지만, 이는 실제 CAN 버스 상의 전송 시점을 기록한 것이 아니라 Windows 소켓 수신 시점을 기준으로 한 것이며, 실제 물리적 송신 시에는 변환 및 전송 지연이 존재했을 가능성이 큽니다.

Index	Sender	CAN ID(hex)	Payload 1st value	Payload 2nd value
1	Controller	002	0xCC	
2	Actuator	003	0x00	(expected = 0xCC)
3	Controller	002	0xD0	
4	Actuator	003	0x15	0xCC
5	Controller	002	0xD3	
6	Actuator	003	0x7F	0xD0
7	Controller	002	0xD3	
8	Actuator	003	0xBF	0xD3

(표 1) Exp2. 두 ECU 간 송수신된 CAN 프레임 예시.

표 1 은 가상 제어 MCU 와 실물 동작 MCU 간의 실제 CAN 프레임 송수신 내용을 일부입니다. 실험 2 에서 예상되는 동작으로는 제어 ECU(Controller)가 보낸 메시지의 첫번째 바이트의 페이로드는 제어 값이며, 이를 수신한 동작 ECU(Actuator)가 해당 값으로 모터를 제어하고, 자신이 측정한 피드백 값과 응답 메시지의 두 번째 페이로드에 수신한 제어 값을 그대로 송신합니다. 여기서 확인할 수 있는 바와 같이, 제어 ECU 가 보낸 메시지의 첫 번째 페이로드 값이, 다음 주기에 동작 ECU 응답 메시지의 두 번째 페이로드 필드에 포함되어 전달되고 있습니다. 이는 동작 ECU 가 한 주기 뒤에 제어 신호를 반영하고 있음을 나타내며, 실시간성이 중요한 제어 시스템에서는 주기 지연에 민감합니다.[5]

이러한 몇 가지의 한계로 인해 두 실험에서 오차를 만들게 됩니다. 만약 더 민감한 시스템의 경우에 적용하기에는 부적합해 보이나 향후 더 정밀한 USB-CAN 인터페이스 장비를 활용하거나 지연을 미리 예측하여 보상하는 기법들을 활용하여[6] 이러한 한계점들을 해결할 수 있을 것입니다.

6. 결론

본 연구에서는 하드웨어와 소프트웨어를 통합한 시뮬레이션 구조를 제시합니다. 이 구조에서는 임베디드 제어 소프트웨어가 Renode 기반의 가상화 환경에서 시뮬레이션되며, 실제 액추에이터를 제어할 수 있

도록 USB-CAN 인터페이스 장비를 이용하였습니다. PID 제어 모델을 가상 ECU 에서 직접 실행하여 실제 ECU 와의 CAN 네트워크를 통해 상호 통신을 가능케 하여, 프로토타입이 준비되기 전에도 제어 알고리즘의 사전 검증이 가능한 실험 환경을 조성했습니다.

이 실험의 결과는 가상 ECU 가 실물 ECU 와 유사하게 작동한다는 것을 확인시켜주었으며, 실제 시스템을 개발하는 데 있어 가상 환경 기반 개발 개념의 가능성을 보여줍니다.

하지만 동시에 실험 결과에서 볼 수 있듯이, USB-CAN 인터페이스 장치(uCAN 3.0)를 통해 메시지를 전송 및 수신하는 과정에서 통신 지연이 관찰되었습니다. 이런 지연은 제어 지연을 초래하여 제어 응답의 정확성에 영향을 미칠 수 있으며, 고속 제어나 정확한 타이밍 동기화가 필요한 시스템에서는 주요 제한 사항이 될 수 있습니다.

이러한 제한은 보다 정확한 장비를 사용하거나 시간 보정 기술를 활용함으로써 해결할 수 있습니다. 본 연구의 결과는 Renode 를 사용한 가상화 기반 제어 시스템 개발의 실질적 유용성을 실험적으로 확인합니다. 이는 임베디드 시스템의 초기 개발에서 중요한 영역인 설계 검증, 실험 반복, 자원 절약에서 효율적일 대안이 될 수 있습니다.

ACKNOWLEDGMENT

이 논문은 2025 년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임 (RS-2024-00415938, 2024 년 산업혁신인재성장 지원사업).

참고문헌

[1] Renode.io, 2025. https://renode.io/ (accessed Apr. 18, 2025).

[2] Sysbas.com, 2025. https://www.sysbas.com/portfolio-item/ucan-2/ (accessed Apr. 18, 2025).

[3] “SocketCAN - Controller Area Network — The Linux Kernel documentation,” docs.kernel.org. https://docs.kernel.org/networking/can.html

[4] H. Kim, J. Kwak, and J. Cho, "AUTOSAR-Compatible Level-4 Virtual ECU for the Verification of the Target Binary for Cloud-Native Development," Electronics, vol. 13, no. 18, p. 3704, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/18/3704.

[5] T. Ersal, M. Brudnak, A. Salvi, Y. Kim, J. B. Siegel, and J. L. Stein, "An Iterative Learning Control Approach to Improving Fidelity in Internet-Distributed Hardware-in-the-Loop Simulation," Journal of Dynamic Systems, Measurement, and Control, vol. 136, no. 6, 2014, doi: 10.1115/1.4027868.

[6] P. Baumann, O. Kotte, Lars Mikelsons, and D. Schramm, “Enhancing the Coupling of Real-Virtual Prototypes: A Method for Latency Compensation,” Electronics, vol. 13, no. 6, pp. 1077–1077, Mar. 2024, doi: https://doi.org/10.3390/electronics13061077.