

학사학위 청구논문

IEEE 802.15.4 기반의 Wireless USB to Serial Converter 구현

(IEEE 802.15.4 based Wireless USB to Serial)

2022년 11월 25일

승실대학교 IT대학

AI융합학부

김 준 철

학사학위 청구논문

IEEE 802.15.4 기반의 Wireless USB to Serial Converter 구현

(IEEE 802.15.4 based Wireless USB to Serial)

지도교수 : 노 동 건

이 논문을 학사학위 논문으로 제출함

2022년 11월 25일

승실대학교 IT대학

AI융합학부

김 준 철

김준철의 학사학위 논문을 인준함

심사위원장 신동화 (인)

심사위원 노동건 (인)

2022년 11월 21일

승실대학교 IT대학

감사의 글

논문이 완성되도록 물심양면으로 지원해주신 Awair의 주희수, 한창민, 이원구, 허재영, 윤덕현, Adnan Hussain, Kevin Cho님을 비롯한 동료 분들, 숭실대학교의 노동건 교수님과 이주희 조교님, 친구 이승우, 그 외 모든 분께 깊은 감사를 전합니다.

목 차

목차	i
표 및 그림 목차	ii
국문초록	iii
I. 서론	1
II. USB to Serial 관련 기술 소개	3
II-1. UART 기초	3
II-2. USB CDC-ACM 요약	3
III. IEEE 802.15.4 프로토콜 소개	5
III-1. IEEE 802.15.4 개요	5
III-2. IEEE 802.15.4 프로토콜 명세 요약.....	5
IV. Wireless USB to Serial Converter 구현	7
IV-1. 시스템 구조 설계	7
IV-2. CC2531EMK 하드웨어 수정	8
IV-3. Contiki-OS와 6LoWPAN 통신 구현	10
V. Wireless USB to Serial 동작 예시	12
V-1. USB Dongle 간 무선 통신	12
V-2. UART 데이터 무선 송수신	13
VI. 결론 및 향후 과제	14
참고문헌	15

표 및 그림 목차

표 1. CC2531 IO 커넥터 핀아웃	9
표 2. CC2531 Peripheral I/O 핀 매핑	9
그림 1. OSI 계층 별 IoT 프로토콜 예시	1
그림 2. Arduino의 UART 핀 연결 예시	3
그림 3. USB 장치 클래스와 적용사례	4
그림 4. Raspberry Pi와 USB to Serial 케이블	4
그림 5. IEEE 802.15.4 계층 구성도	5
그림 6. Wireless USB to Serial 시스템 구조	7
그림 7. CC2531 USB Dongle 하드웨어	8
그림 8. CC Debugger가 연결된 CC2531EMK	8
그림 9. CC2531 디버그 커넥터 핀아웃	9
그림 10. UART 케이블 구성 예시	9
그림 11. UART 0 (Alternative 2) 결선 예시	9
그림 12. 다중 Dongle 무선 송수신 테스트	12
그림 13. Serial Monitor에 출력된 무선 수신 데이터	12
그림 14. 송신측 구성 예시 (보조 배터리 사용)	13
그림 15. 수신측 구성 예시	13
그림 16. UART 데이터 무선 수신 결과	13

IEEE 802.15.4 기반의 Wireless USB to Serial Converter 구현

AI융합학부 김 준 철

지도교수 노동건

IoT 및 임베디드 장치의 개발 과정에서 자주 애용되는 USB to Serial Converter는 장치에 주는 부담이 거의 없이 문제 분석을 가능하게 한다. 하지만 물리적으로 분산되어 배치된 다수의 장치로 이루어진 네트워크를 분석해야하는 경우, 유선 연결을 필요로하는 USB to Serial은 적용하기에 어려움이 있다. 이러한 USB to Serial Converter의 유선 연결부를 무선 통신으로 대체하여 IoT 및 임베디드 장치의 기존 네트워크 대역폭에 영향을 주지 않으면서도 여전히 편의성이 높은 방법을 구현할 수 있다.

본 논문에서는 기존 유선 USB to Serial Converter의 접근성과 확장성을 향상시키기 위하여 IEEE 802.15.4 기반의 무선 네트워크 기술이 접목된 하드웨어 및 소프트웨어를 구현한다. TI사의 저전력 SoC솔루션인 CC2531 USB Dongle을 하드웨어로 활용하고, 그 위에서 IoT를 위한 운영체제 Contiki-OS를 구동한다. Contiki-OS와 함께 제공되는 IEEE 802.15.4 기반의 저전력 무선 사설 네트워크 프로토콜 6LoWPAN을 활용하여, 한 쪽의 CC2531EMK USB Dongle로 무선 네트워크를 구성한다. 한 쪽의 Dongle은 UART로 연결된 임베디드 장치로부터 받은 데이터를 무선 네트워크로 송신하고, 다른 한 쪽의 Dongle은 무선으로 수신받은 데이터를 USB로 연결된 PC에 제공하는 방식으로 USB to Serial 기능을 수행한다.

I. 서 론

최근 들어 산업계에 IoT 기술 개발과 관련한 변화의 바람이 불고 있다. 일례로, 스마트홈 연결 표준인 Matter 1.0 버전이 얼마 전 발표되었다. Matter는 OSI 모형에서 응용계층(Application Layer)에 해당하는 프로토콜로, IP를 지원하는 어떤 하위 계층 프로토콜이라도 연동할 수 있게 되어 있다. 다음 그림은 Matter를 통해 지원되는 프로토콜 예시를 계층 별로 나타낸 것이다.

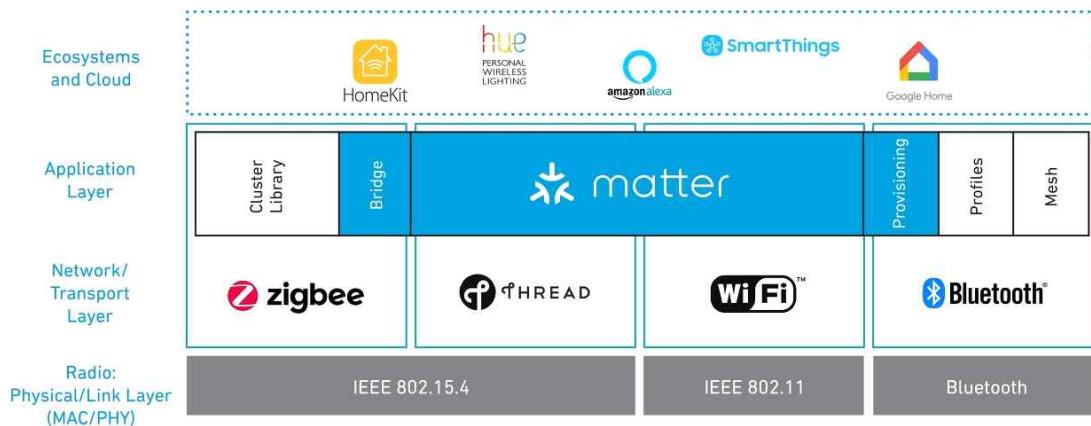


그림 1 : OSI 계층 별 IoT 프로토콜 예시

위 그림에서 알 수 있듯이, 시장에는 다양한 IoT 프로토콜이 공존하고 있다. Matter와 같은 높은 호환성의 연결 표준이 등장하면서 앞으로도 더 많은 종류의 IoT 프로토콜이 시장에서 상생할 가능성이 대두되었다. 다시 말해 이러한 통신 기능을 구현하는 임베디드 장치의 개발이 다가올 산업 발전 과정에서 더 주요하게 다뤄질 것을 의미하기도 한다.

현재 많은 IoT 제품이 현장에서 겪게 되는 어려움은 다수의 장치로 이루어진 네트워크상의 문제에 대응하는 것이다. IoT 제품의 특성상 범용적인 제어보다는 특정 기능만을 수행하기 위해 만들어지기에, 그것의 제한된 자원과 입출력 방식만으로는 문제를 분석하기가 쉽지 않다. 그 때문에 일반적인 응용 프로그램과는 조금 다른 접근법을 사용한다. 한 가지 예시로는 그 네트워크상에 상태 분석을 돋기 위한 추가적인 메시지를 송신하는 방법이 있다. 그러나 만약 해당 임베디드 장치에서 사용하는 프로토콜이 Zigbee, Thread, Bluetooth 등의 낮은 대역폭을 가진 통신 방식이라면, 그런 기능을 구현하는 것이 근본적으로 불가능하거나 통신이 불안정한 상황을 마주할 수 있다.

반면, 장치에 주는 부담이 거의 없는 좋은 대안으로 USB to Serial이 있다. IoT 및 임베디드 장치는 개발 단계에서의 문제 분석 및 수명 주기 동안의 관리를 수월하게 하기 위해 Serial Logging 기능을 탑재하는 것이 일반적이다. 그러나 USB to Serial은 유선 연결 방식이기에 각 장치가 물리적으로 멀리 떨어진 위치에 설치된 상황의 통신 문제를 분석하기 어렵다는 단점이 있다. 모든 장소마다 USB to Serial Converter를 연결한 PC를 하나씩 둔다거나, 충분한 길이의 전선을 사용하여 하나의 PC에 연결하는 등의 비효율적인 방식을 적용해야 할 것이다.

본 논문에서는 이러한 문제를 극복하기 위해 USB to Serial 방식의 유선 연결부를 무선 통신으로 대체하는 기술을 구현한다. Wireless USB to Serial Converter를 구현하기 위해 IEEE 802.15.4 전용 SoC인 Texas Instrument사의 CC2531EMK USB Dongle을 사용한다. 이 시스템은 두 가지 역할을 지닌 Dongle로 구성된다. 하나의 Dongle은 임베디드 장치와 직접 Serial 통신을 하며 무선 네트워크에 데이터를 송신하고, 다른 Dongle은 PC와 직접 USB로 통신하며 무선 네트워크로부터 데이터를 수신한다. CC2531이 제공하는 H/W 인터페이스 USART와 USB를 통해 USB to Serial 기능을 구현하고, 무선 통신부는 Contik-OS가 제공하는 6LoWPAN 라이브러리를 통해 802.15.4 네트워크를 구성한다. 이렇게 만들어진 Wireless USB to Serial Converter를 IoT 및 임베디드 장치 개발을 보조하는 도구로써 사용하고자 한다. 기존 USB to Serial이 가진 유선 연결의 물리적인 제약을 해결함으로써 제품의 개발 시간 단축, 유지보수의 용이성, 모니터링의 편의성 등의 효과가 예상된다.

서론에 이어 본 논문은 2장에서는 USB to Serial과 관련된 하드웨어 인터페이스 기술들을 간략히 살펴보고, 3장에서는 6LoWPAN 등의 표준 IEEE 802.15.4 구현체에 대해 알아본다. 4장에서는 시스템의 구현 방법을 다루고, 5장에서는 동작 예시를 보인다. 마지막으로 6장에서는 본 논문의 결론과 향후 과제를 다룬다.

II. USB to Serial 관련 기술 소개

II-1. UART 기초

UART는 Universal Asynchronous Receiver/Transmitter의 약자로, 비동기 Serial 통신을 위한 컴퓨터 하드웨어 장치이다. 초기에는 전기적 신호 제어 방식에 따라 RS-232, RS-485 등으로 구분되어 많은 컴퓨터와 주변장치의 통신에 사용되었지만, 오늘날에는 주로 임베디드 장치와 같은 저사양의 시스템에서 프로세서 간 TTL 레벨의 신호를 주고받는 용도로 쓰인다. 다음 그림은 임베디드 장치 개발에 사용되는 Arduino 보드의 UART 핀 연결이다.

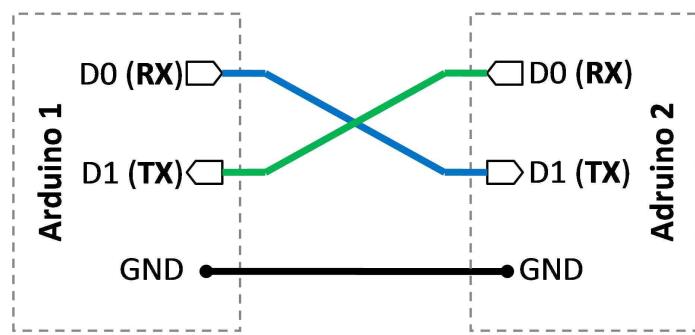


그림 2. Arduino의 UART 핀 연결 예시

위에서 볼 수 있듯 UART는 핵심적으로 RX, TX, GND 세 핀으로 동작한다. 통신하는 상대에게 전원을 함께 공급해야 하는 경우에는 VCC 핀을 추가로 연결하기도 한다. 여기서 중요한 점은 서로 간에 TX와 RX를 교차한다는 점이다. 위의 그림에서, Arduino 1이 데이터 신호를 TX로 전송하면 Arduino 2는 RX로 수신한다, 반대 방향의 송수신도 마찬가지로 Arduino 2의 TX 신호는 Arduino 1의 RX로 들어간다. GND는 이 과정에서 전기적 신호 레벨이 올바르게 인식될 수 있도록 기준 전압을 맞춰주는 역할을 한다.

II-2. USB CDC-ACM 요약

USB는 컴퓨터와 주변기기를 연결하는 입출력 표준 프로토콜이다. 본 논문에서는 USB 중에서도 USB to Serial과 밀접한 관련이 있는 CDC-ACM만을 다룬다. 다음 그림은 USB에 정의된 장치 클래스와 적용사례를 보여준다.

Host		Peripheral	
Target Product (example)	USB F/W	USB F/W	Target Product (example)
	Mass Storage Class Driver (HMSC)	Mass Storage Class Driver (PMSC)	USB Thumb Drive
	Communication Class Driver (HCDC)	Communication Class Driver (PCDC)	RS232C-USB Conversion
	HID Device Class Driver (HHID)	HID Device Class Driver (PHID)	Mouse, Keyboard
	USB Basic Firmware (USB-BASIC-F/W)	USB Basic Firmware (USB-BASIC-F/W)	Specialized product (Vendor original communication)

■ USB Device Class Firmware ■ USB Basic Firmware

그림 3. USB 장치 클래스와 적용사례

일반적으로 사용하는 PC가 USB Host에 속하고, 모든 클래스를 지원한다. 반면, USB Peripheral은 특정 클래스만을 구현한다. 여기서 주목할 것은 CDC (Communication Device Class)이다. RS232 to USB Conversion이 그 예시인 것에서 알 수 있듯, 이 클래스는 UART to Serial에 사용된다. 앞서 언급한 ACM(Abstract Control Model)은 CDC의 서브 클래스로, PC가 장치를 가상 직렬 포트(Virtual COM Port)로 인식하게 한다. 이 같은 CDC-ACM 클래스와 UART 간의 Bridge를 구성한 것이 바로 USB to Serial Converter이다.[1]



그림 4. Raspberry Pi와 USB to Serial 케이블

위의 USB 케이블 끝 4개의 핀이 UART의 RX, TX, GND, VCC이다. Raspberry Pi가 USB to Serial 케이블과 UART로 연결되어 입출력한 데이터는 USB CDC-ACM을 통해 Host PC에서 모니터링할 수 있게 되는 것이다.

III. IEEE 802.15.4 프로토콜 소개

III-1. IEEE 802.15.4 개요

IEEE 802.15.4 프로토콜은 CPU 및 메모리 성능 제약이 있는 저전력 IoT 디바이스에 적합한 무선 통신 기술이다. 해당 표준은 OSI 참조 모델의 최하단 두 계층인 물리(PHY) 계층과 데이터 링크(MAC) 계층에 걸쳐 정의되어 있다. 같은 계층에서 정의된 표준인 Wi-Fi의 경우 높은 전력 소모를 요구하기 때문에 저사양의 임베디드 디바이스에 적합하지 않다. 그로 인해 Zigbee, 6LoWPAN, Thread와 같이 현재 주목받는 대부분의 IoT 무선 통신 프로토콜들은 IEEE 802.15.4의 상위 계층으로서 정의되어 있다. 특히 6LoWPAN은 WPAN(Wireless Personal Area Network) 상에서의 IPv6 호환성을 제공하며, Zigbee와 달리 공개된 표준으로서 최근 IoT 업계의 많은 주목을 받고 있다.[2]

III-2. IEEE 802.15.4 프로토콜 명세 요약

IEEE 802.15.4의 PHY 계층은 통신 매체인 무선 트랜시버에 대한 인터페이스를 정의하며, 상위 계층인 MAC 부계층에게 Channel Assessment, 비트 변복조, 패킷 동기화와 같은 서비스를 제공하며 저수준 종단 간 무선 데이터 통신을 담당한다. IEEE 802.15.4의 MAC 부계층은 상위 계층에게 공유 채널 간 접근 제어, 디바이스의 Association/Dissociation 메커니즘, 비콘 생성, GTS(Guaranteed Time Slot) 등의 서비스를 제공한다. IEEE 802.15.4의 MAC 부계층은 IEEE 802.15.4 표준 바깥에 있는 LLC(Logical Link Control) 부계층과 함께 데이터 링크 계층을 구성하는 것이 일반적이다.

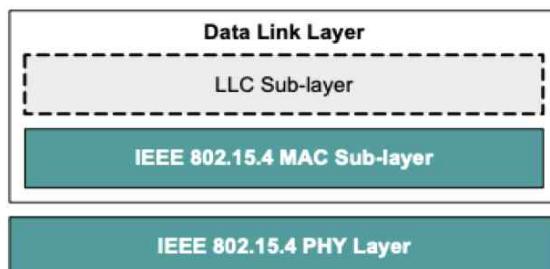


그림 5. IEEE 802.15.4 계층 구성도

IEEE 802.15.4는 무면허 주파수 대역에서 운용되도록 정의되어 있다. 각 국가마다 무면허 주파수 대역은 상이하지만, 네트워크를 구성하고자 하는 지역에서 868, 915, 2400MHz의 세 가지 중 최소 하나를 지원하여야 한다. IEEE 802.15.4는 제한된 대역폭의 저속 무선 통신으로, 250kbps의 전송 속도를 지원한다.

IEEE 802.15.4 기반으로 네트워크를 구축할 때 단일 PAN(Personal Area Network) 네트워크 내에 반드시 단 하나의 PAN 코디네이터가 노드가 존재하여야 한다. PAN 코디네이터는 자신의 속한 네트워크의 PAN ID 할당 및 해당 네트워크에 가입(Join)하려는 디바이스의 요청 핸들링, 최적의 주파수 탐색, 노드 간의 메시지 릴레이(hopping) 등의 핵심 기능을 담당한다.

IEEE 802.15.4 기반 PAN 네트워크 내의 디바이스들은 FFD(Full Function Device) 또는 RFD(Reduced Function Device) 중 하나의 타입을 갖는다. FFD는 IEEE 802.15.4 MAC 서비스를 모두 제공하는 디바이스로서써 코디네이터로 기능할 수 있다. RFD는 IEEE 802.15.4 MAC 서비스의 제한된 서브셋을 제공하는 디바이스로써 컴퓨팅 성능이 제한된 디바이스에 적합하며, 코디네이터로 기능할 수 없다. 이러한 디바이스 노드들의 네트워크 구성은 위해서는 스타형, 트리형, 메시형 토폴로지를 일반적으로 사용한다. 현재 IoT 업계에서는 확장과 제거에 유연하며 통신 에러에 구조적으로 강건한 메시형 토폴로지가 주목받고 있다.[3]

IV. Wireless USB to Serial의 구현 및 동작

IV-1. 시스템 구조 설계

IEEE 802.15.4 기반의 Wireless USB to Serial에 사용되는 통신 기술들의 역할과 관계는 계층적 구조로 나타낼 수 있다. 먼저 2.4 GHz 주파수 대역의 물리 계층 위에서 IEEE 802.15.4의 MAC (Media Access Control)이 링크 계층 요소로 동작한다. 6LoWPAN은 앞서 언급된 물리와 링크 계층 요소와 긴밀히 협력하여 저전력 IPv6, 즉 네트워크 계층을 구성한다. 그렇게 만들어진 6LoWPAN 네트워크 위에서 실질적인 응용 기능을 위한 요소들이 정의된다. UART와 USB로부터 받은 텍스트로 된 실시간 데이터를 송수신할 것이기에, 단순하게 UDP를 전송 계층으로 하는 소켓을 만들어 상호 간의 연결을 구성한다. 여기서, Serial 통신의 특성에 맞게 TX와 RX를 위한 소켓을 각각 만들고, 원격지의 포트는 교차로 연결한다. 즉, TX 소켓으로 전송한 데이터는 상대의 RX 소켓에 도달하게 한다. 응용 계층에서 구현에서는 UART 또는 USB에서 받은 데이터를 TX 소켓으로 전송한다, RX 소켓에서 받은 데이터를 UART 또는 USB로 전송한다. 그럼으로 나타낸 시스템 구성은 다음과 같다.

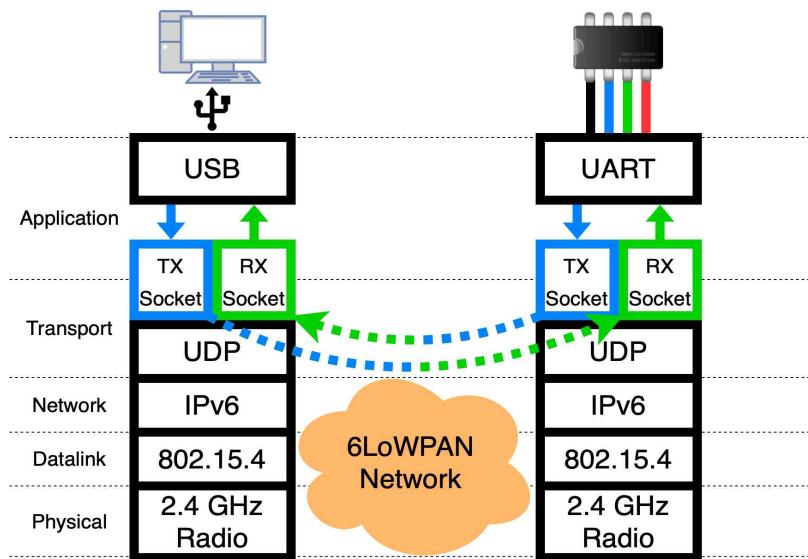


그림 6. Wireless USB to Serial 시스템 구조

그림에서 각각의 스택은 Dongle을 의미하고, 컴퓨터 기호는 Dongle에 USB로 연결하는 PC를, IC 칩은 Dongle에 UART로 연결하는 IoT 장치를 의미한다.

IV-2. CC2531EMK 하드웨어 수정

CC2531EMK^[4] 또는 CC2531 USB Dongle은 Texas Instrument 사의 저전력 802.15.4 애플리케이션 전용 SoC 평가 키트이다. H/W 구성요소로 USB 인터페이스, 디버그용 포트, 8개의 IO 핀, 각각 2개의 Button 및 LED가 제공된다.

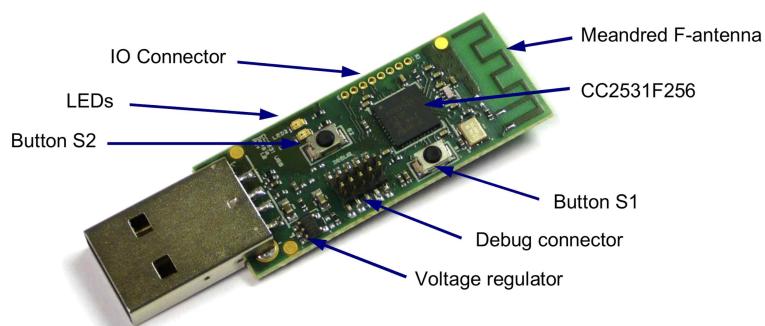


그림 7. CC2531 USB Dongle 하드웨어

개발 환경으로 Intel 8051 기반 MCU용 컴파일러 SDCC(Small Device C Compiler)로 펌웨어를 빌드하고^[5], TI 칩 전용 디버거인 CC Debugger로 펌웨어를 업로드한다.^[6]



그림 8. CC Debugger가 연결된 CC2531EMK

Wireless USB to Serial Converter 구현을 위해서 위와 같은 구성을 사용했다. 하지만, CC2531EMK에는 개발 과정에 필요한 UART 입출력 구성이 기본적으로 제공되지 않는다. UART 사용을 위해서는 펌웨어에서 특정 레지스터를 설정하여 IO 핀을 할당 해야하고, 핀헤더 등을 IO 핀에 납땜하여 점퍼선을 쉽게 연결할 수 있어야한다. 따라서 약간의 하드웨어 변경도 필요하다. 다음 문단에서는 그것을 위해 알아야하는 기본적인 정보들과 구성 예시를 다룬다.

UART를 사용하기 위해서는 먼저 CC2531EMK의 IO Connector의 핀 정보가 필요하다. IO 핀 번호는 안테나에 가까운 쪽이 1번이고, USB에 가까운 쪽이 8번이다.[7]

IO Connector	CC2531	Periphery/Function	P0								P1							
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	P0.2	USART 0 SPI Alt. 2			C	SS	MO	MI										
2	P0.3											M0	MI	C	SS			
3	P0.4	USART 0 UART Alt. 2			RT	CT	TX	RX					TX	RX	RT	CT		
4	P0.5																	
5	P1.7	USART 1 SPI Alt. 2			MI	M0	C	SS				MI	M0	C	SS			
6	P1.6																	
7	P1.5	USART 1 UART Alt. 2			RX	TX	RT	CT					RX	TX	RT	CT		
8	P1.4																	

표 1. CC2531 IO

커넥터 핀아웃

표 2. CC2531 Peripheral I/O 핀 매핑

표 2는 CC2531 User Guide[8]에 제공한 Peripheral I/O 핀 설정 표이다. 이 표를 참조하여 펌웨어에서 레지스터를 어떻게 설정해야 원하는 IO 핀에 UART가 할당되는지 알아낼 수 있다. 예를 들어, USART 0의 기본 설정을 쓰는 경우 RX가 P0.2이고, TX가 P0.3이다. 표 1을 참조하면, IO 1번 핀이 RX, IO 2번 핀이 TX임을 알 수 있다. 이는 후술하겠지만, Contiki-OS에 정의된 컴파일 옵션으로 쉽게 변경할 수 있다.

이렇게 정해진 UART 핀과 핀헤더를 연결해야 한다. IO Connector에 1.27 mm 피치 규격의 헤더를 납땜하면 된다. 본 논문에서는 핀헤더 대신에, 이미 실장된 디버그 커넥터의 비어있는 핀들을 활용해서 UART 핀들을 전용 USB-C 케이블에 연결했다.

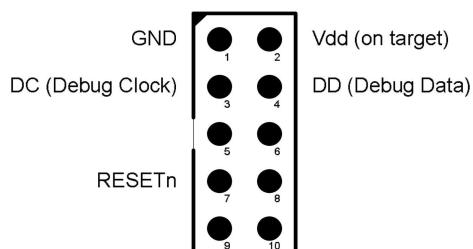


그림 9. CC2531 디버그 커넥터 핀아웃



그림 10. UART 케이블 구성 예시

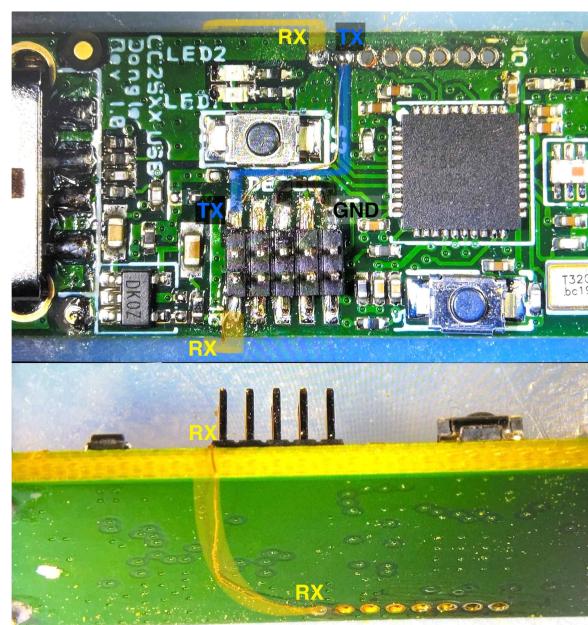


그림 11. UART 0 (Alternative 2) 결선 예시

IV-3. Contiki-OS와 6LoWPAN 통신 구현

Contiki-OS는 메모리 제약 시스템과 저전력 IoT 장치에 초점을 맞춘 운영 체제이다.[9] Contiki-OS가 지원하는 H/W 플랫폼 중에는 앞서 다룬 CC2531 USB Dongle[9] 있다. Contiki-OS의 공식 Github 레포지토리에 포함된 코드 중에는 cc2531-usb-demo가 있는데, 이는 CC2531 USB Dongle의 USB CDC-ACM을 활용한 Serial 입출력을 검증해 볼 수 있는 예제이다[10]. 또 다른 예제 cc2530dk/udp-ipv6로는 IPv6 UDP 서버와 클라이언트를 검증해 볼 수 있다[11]. 본 논문에서 구현하는 Wireless USB to Serial 기능의 펌웨어는 기본적으로 이 두 예제를 결합하고 몇 가지를 보완하여 완성되었다. 다음 문단부터는 두 예제를 실행하고 결합하는 과정에 대해 다루도록 하겠다.

Contiki-OS 예제의 실행을 위해서는 먼저 해당 레포지토리 Wiki의 8051 Requirements[12]문서를 참고하여 전용 컴파일러 SDCC(Small Device C Compiler)를 빌드해야한다. Contiki-OS만을 위한 특수한 사전 작업들이 요구되기 때문에 컴파일러에 몇 가지 수정도 겸해야 한다. 가장 대표적인 예로, CC2531은 223바이트의 스택 밖에 제공되지 않기에, 개발 도중에 문제를 마주할 경우 8051 Even More Stack[13]문서를 참고하여 컴파일러와 펌웨어 코드에 최적화를 적용해야 할 수도 있다. 컴파일러의 빌드 이후엔, 명령행에서 해당 예제 폴더로 이동한 뒤에 make를 수행하여 예제 펌웨어의 바이너리 파일을 빌드해야한다. 같은 폴더에 '.hex' 확장자의 파일이 생성되면 성공적으로 빌드가 된 것이다. 빌드한 펌웨어는 cc-tool[14]과 같은 소프트웨어를 사용하여 CC Debugger로 업로드한다.

앞의 과정을 수행한 뒤 cc2531-usb-demo 예제 폴더로 이동하여 펌웨어를 빌드하고, 결과로 나온 hex파일을 CC2531에 업로드한다. 성공적으로 진행되면 USB Dongle을 연결한 PC에서 가상 직렬 포트 장치가 인식되고, Serial Monitor 프로그램을 통해 1초 주기로 ‘cc2531 USB Dongle Out: 0x01’와 같은 출력이 나오는 것을 확인할 수 있다. 예제 폴더의 ‘cc2531-usb-demo.c’ 파일에 작성된 코드로 알 수 있듯, 키보드로 입력하는 문자열 또한 ‘cc2531 USB Dongle In : abcd’와 같은 출력으로 확인할 수 있다. 이러한 입력은 serial-line[15]이라 는 라이브러리 프로세스를 통해 처리되는데, ‘contiki-main.c[16]’의 초기화 단

계에서 ‘io_arch_set_input’ 함수에 콜백 함수 ‘serial_line_input_byte’를 등록하여 달성된다.

cc2530dk/udp-ipv6도 동일한 방식으로 빌드한다. 결과물로 ‘server.hex’와 ‘client.hex’가 나오는데, 두 개의 CC2531EMK를 준비하여 각각 업로드 한다. 이번엔 USB 가상 직렬 포트로 인식되지 않는 것을 볼 수 있는데, 이는 본래 USB 인터페이스가 없는 CC2530EMK^[17]를 위해 작성된 예제이기 때문이다. 앞선 예제에서는 ‘DEFINES+=MODELS_CONF_CC2531_USB_STICK=1’이라는 설정을 Makefile에 추가했기에 USB로 인식되는 펌웨어가 빌드된 것이다. 하지만, 본 예제에서 그 방식을 쓰는 것은 Stack Overflow를 초래하여 기기가 반복적으로 재부팅되는 현상을 유발하기에 권장하지 않는다. 대신 앞서 4-2에서 준비한 UART 사용을 권장한다. 별도의 USB to Serial 케이블 등을 준비하여 CC2531EMK와 연결한다. 본 논문의 경우 UART 0에 Alternative 2 설정을 전제로 했기에 ‘DEFINES+=UART0_ALTERNATIVE_2’를 udp-ipv6 예제의 Makefile에 추가하여 빌드했다. 이를 통해 UART 출력을 통해, server 와 client가 서로 통신하지 못함을 알 수 있었다. 이는 client측이 실제 server 가 사용하는 ipv6주소가 아니라 하드 코딩된 값을 사용하고 있기에 생긴 문제라 판단되었고, 멀티캐스트 주소를 사용하여 통신에 성공할 수 있었다. 해결 과정에서 ‘tcpip.c’, ‘uip6’, ‘sicslowpan.c’를 비롯한 코드 상단의 ‘DEBUG’ 정의 값을 ‘DEBUG_NONE’에서 ‘DEBUG_PRINT’ 대체하여 통신 과정을 UART로 출력하고 분석하는 것이 도움 되었다.

이제 남은 것은 두 예제를 결합하는 과정이다. server 예제의 server_conn 을 RX 소켓으로, client 예제의 l_conn을 TX 소켓으로 활용한다. TX는 멀티 캐스트 주소로 생성하고, TX와 RX 포트를 서로 연결한다. tcpip_event와 serial_line_event_message 두 가지 이벤트를 처리하도록 한다. RX로 수신한 패킷은 usb-demo 예제의 Serial 출력으로 전달되고, Serial 입력은 TX로 전송된다. ‘uart0_init()’과 ‘uart0_set_input(serial_line_input_byte)’를 호출하여 USB가 활성화된 상태에서도 UART 입출력이 동작하게 한다. 이외에도 안정화를 위해 필요한 설정들이 있지만, 본 논문에서는 다루지 않는다. 위의 과정들을 Github 개인 레포지토리^[18]에 취합하여 추후 개발 이력 참조를 용이하게 했다.

V. Wireless USB to Serial 동작 예시

V-1. USB Dongle 간 무선 통신

4장에서 설명한 일련의 과정을 통해 만들어진 USB Dongle을 테스트하기 위해서 먼저 UART 입출력이 아닌 USB 입출력만으로 테스트를 수행하였다. USB Dongle 16개를 동시에 PC에 연결하고 Serial Monitor 프로그램 창을 각각 열어 한 창씩 커서를 옮겨가며 키보드로 문자열을 입력했다.



그림 12. 다중 Dongle 무선 송수신 테스트

테스트 결과 한 창에서 입력한 내용과 동일한 내용이 다른 모든 창에 출력되는 것을 볼 수 있었다. 이는 키보드 입력 데이터가 다른 동글들에게 무선으로 멀티캐스트 전송되었고, 무선으로 데이터를 수신한 동글들은 받은 내용을 그대로 UART와 USB CDC-ACM 출력으로 내보냈다는 의미이다.

The image displays four separate windows of the Awair Debugger software, each representing a different USB port (45x19, 47x19, 47x19, and 47x19). Each window shows a series of text messages being transmitted and received over a serial connection. The messages include Korean text like "Hello world!" and "Hello this is me". The software interface includes tabs for selecting different ports and buttons for sending and receiving data.

```
Awair Debugger 4 - 45x19 - 115200.B.N.1
11-09 05:26:27.838 }f4e6{
11-09 05:26:28.838 가 나
11-09 05:26:35.948 다 라 마 바 사
11-09 05:26:44.012 가 나 다 라 마 바 사

11-09 05:26:58.446 }f9cb{
11-09 05:26:58.446 안녕하세요

11-09 05:26:56.833 }f4e6{
11-09 05:26:56.833 반갑습니다

11-09 05:27:12.674 }fe9d{
11-09 05:27:12.674 This is Low Power Wireless USB Demo

11-09 05:27:26.832 }ee76{
11-09 05:27:26.832 Hell
11-09 05:27:34.408 Hello wold!

Awair Debugger 4 - 47x19 - 115200.B.N.1
11-09 05:26:03.289 }fa8e{
11-09 05:26:03.336 hello this is me

11-09 05:26:27.828 }f4e6{
11-09 05:26:27.838 가 나
11-09 05:26:35.948 다 라 마 바 사
11-09 05:26:44.012 가 나 다 라 마 바 사

11-09 05:26:58.405 }f9cb{
11-09 05:26:58.462 안녕하세요

11-09 05:26:56.833 }f4e6{
11-09 05:26:56.833 반갑습니다

11-09 05:27:26.833 }ee76{
11-09 05:27:26.833 Hell
11-09 05:27:34.408 Hello wold!

Awair Debugger 4 - 47x19 - 115200.B.N.1
11-09 05:26:27.838 }f4e6{
11-09 05:26:27.838 가 나
11-09 05:26:35.948 다 라 마 바 사
11-09 05:26:44.012 가 나 다 라 마 바 사

11-09 05:26:58.462 }f9cb{
11-09 05:26:58.462 안녕하세요

11-09 05:26:56.833 }f4e6{
11-09 05:26:56.833 반갑습니다

11-09 05:27:12.673 }fe9d{
11-09 05:27:12.727 This is Low Power Wireless USB Demo

11-09 05:27:26.833 }ee76{
11-09 05:27:26.833 Hell
11-09 05:27:34.408 Hello wold!

Awair Debugger 4 - 47x19 - 115200.B.N.1
11-09 05:26:27.837 }f4e6{
11-09 05:26:27.837 가 나
11-09 05:26:35.941 다 라 마 바 사
11-09 05:26:44.013 가 나 다 라 마 바 사

11-09 05:26:58.462 }f9cb{
11-09 05:26:58.462 안녕하세요

11-09 05:26:56.833 }f4e6{
11-09 05:26:56.833 반갑습니다

11-09 05:27:12.726 }ee76{
11-09 05:27:12.726 This is Low Power Wireless USB Demo

11-09 05:27:26.834 }ee76{
11-09 05:27:26.834 Hell
11-09 05:27:34.407 Hello wold!
```

그림 13. Serial Monitor에 출력된 무선 수신 데이터

V-2. UART 데이터 무선 송수신

5-1의 실험으로 USB CDC-ACM 입출력을 무선 송수신 데이터로 전환할 수 있음이 확인되었으므로, 이번에는 실질적인 USB to Serial 기능을 확인한다. 다음 그림은 테스트에 사용된 장치 구성 예시를 나타낸다.



그림 14. 송신측 구성 예시 (보조 배터리 사용)



그림 15. 수신측 구성 예시

송신측은 Dongle과 임베디드 장치를 서로 UART 통신이 가능한 케이블로 연결했고, 수신측은 Dongle을 PC에 직접 연결했다. 송신측은 임베디드 장치의 전원이 켜져 있어야 하기에, 케이블에 VCC 핀을 추가로 연결하고 보조 배터리 등으로 Dongle과 임베디드 장치에 전원을 공급해주는 방법을 사용할 수 있다. 실험 과정에서 발견된 사실은 USB가 활성화된 경우 알 수 없는 이유로 UART 입력 인터럽트가 동작하지 않는다는 것이다. USB기능을 펌웨어에서 비활성화 하여 빌드하고 다시 테스트를 수행하였다.

```
13:42:48.036 0006 00:00:01.250 Dev initied  
13:42:48.149 0000  
13:43:51.042 [Disconnected]  
13:43:54.315 [Connected]  
  
13:44:36.072 }fe9d{  
13:45:15.376 0002 00:00:00.007 Storage Manager ini
```

그림 16. UART 데이터 무선 수신 결과

위 그림처럼 UART 데이터를 수신할 수 있음을 확인하였다. 하지만 길이가 긴 입력에서는 USB만을 사용할 때에 비해 불안정했다. 이는 패킷 길이의 제한과 메모리 제약 때문으로, 추가적인 최적화가 필요한 영역으로 생각된다.

VI. 결론 및 향후 과제

본 논문에서는 USB to Serial 기능을 무선으로 수행할 수 있는 Dongle형태의 하드웨어를 구현하는 방법을 서술하고, 몇 가지 동작 예시들을 보였다. CC2531EMK와 같은 저가형 IEEE 802.15.4 전용 하드웨어로 보편적인 사용 사례에 부합하는 무선 USB to Serial 장치를 구현할 수 있음을 보였고, 물리적으로 더 넓은 연결 범위와 다중 연결 가능성 등 기존 유선 방식의 USB to Serial에 대해 많은 비교 우위를 갖는 기술임을 확인할 수 있었다. 본 논문에서 구현한 Wireless USB to Serial Converter가 실제 IoT 제품의 개발 과정에 적용 가능한 도구로 발전할 수 있기를 기대한다.

향후 해결해야 하는 과제는, 저사양의 SoC에서도 대량의 UART 데이터를 안정적으로 무선 전송하는 방법, 멀티캐스트를 배제하면서 손쉬운 1:1 연결을 가능하게 하는 방법, 임베디드 장치와 연결되는 측의 효과적인 전원 공급 방법 등이 있다.

참고문헌

- [1] <https://electronics.stackexchange.com/questions/427128>. Stack Exchange, n.d. Web. 2022.
- [2] https://ettrends.etri.re.kr/ettrends/108/0905001311/22-6_024_032.pdf. ETRI, Dec 2007.
- [3] <https://koreascience.kr/article/JAKO200722941497058.pdf>. 한국통신학회, Jun 2007.
- [4] <https://www.ti.com/tool/CC2531EMK>. Texas Instruments, n.d. Web. 2022.
- [5] <https://github.com/contiki-os/contiki/wiki/8051-Based-Platforms>. Github, n.d. Web. 2022.
- [6] https://www.zigbee2mqtt.io/guide/adapters/flashing/flashing_the_cc2531.html. Zigbee2MQTT, nd Web. 2022
- [7] <https://www.ti.com/lit/ug/swru221a/swru221a.pdf#page=10>. Texas Instruments, n.d.
- [8] <https://www.ti.com/lit/ug/swru191f/swru191f.pdf#page=81>. Texas Instruments, n.d.
- [9] <https://github.com/contiki-os/contiki>. Github, n.d. Web. 2022.
- [10] <https://github.com/contiki-os/contiki/tree/master/examples/cc2530dk/cc2531-usb-demo>. Github, nd Web. 2022
- [11] <https://github.com/contiki-os/contiki/tree/master/examples/cc2530dk/udp-ipv6>. Github, nd Web. 2022
- [12] <https://github.com/contiki-os/contiki/wiki/8051-Requirements>. Github, n.d. Web. 2022.
- [13] <https://github.com/contiki-os/contiki/wiki/8051-Even-More-Stack>. Github, n.d. Web. 2022.
- [14] <https://github.com/dashesy/cc-tool>. Github, n.d. Web. 2022.
- [15] <https://github.com/contiki-os/contiki/blob/master/core/dev/serial-linec>. Github, nd Web. 2022
- [16] <https://github.com/contiki-os/contiki/blob/master/platform/cc2530dk/contiki-main/L187>. Github, nd Web. 2022
- [17] <https://www.ti.com/tool/CC2530EMK>. Texas Instruments, n.d. Web. 2022.
- [18] <https://github.com/fetiu/lowusb>. Github, n.d. Web. 2022.