Hanbit eBook

Realtime 86

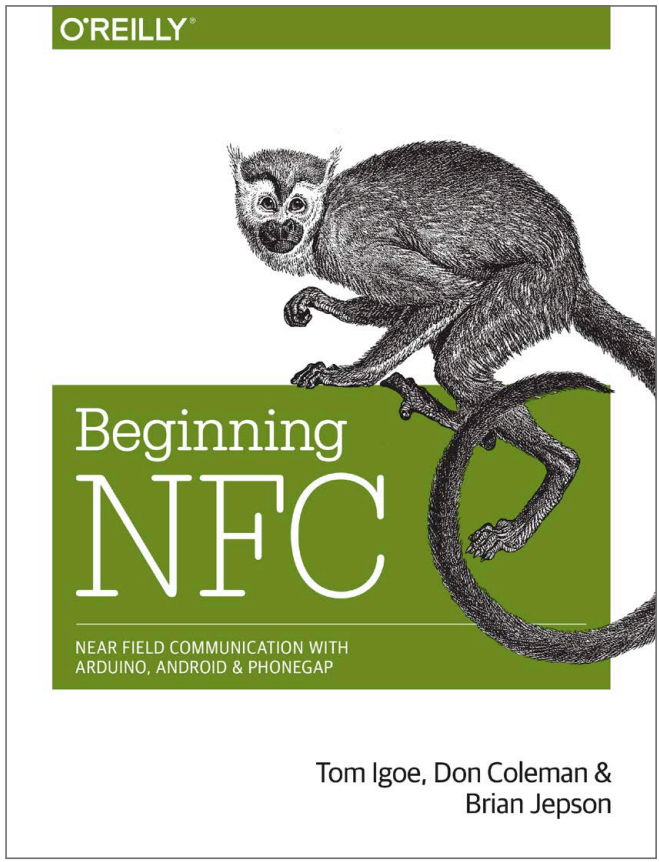
처음 시작하는 NFC 프로그래밍

아두이노, 라즈베리 파이에서 개발하기

Beginning NFC

톰 이고, 돈 콜먼, 브라이언 젭슨 지음 / 이희정 옮김

O'REILLY® 한빛미디어 Hanbit Media, Inc.



이 도서는 O'REILLY의
Beginning NFC의
번역서입니다.

처음 시작하는 NFC 프로그래밍 아두이노, 라즈베리 파이에서 개발하기

초판발행 2015년 1월 12일

지은이 톰 이고, 돈 콜먼, 브라이언 젭슨 / **옮긴이** 이희정 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-725-5 15000 / **정가** 11,000원

총괄 배용석 / **책임편집** 김창수 / **기획·편집** 김상민
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 최송실
영업 김형진, 김진불, 조유미 / **마케팅** 박상용

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

저자 소개

지은이_ 톰 이고

톰 이고는 뉴욕 대학교 티쉬 예술대Tisch School of the Arts에서 인터랙티브 텔레커뮤니케이션 과정의 피지컬 컴퓨팅과 네트워킹을 가르치고 있다. 다양한 교육과 연구를 통해 다양한 인간의 신체표현활동을 감지하고 이에 반응하는 디지털 기술을 찾고 있다. 또한, 『Making Things Talk』와 『Getting Started with RFID』의 저자며, Dan O'Sullivan과 함께 피지컬 컴퓨팅을 다룬 『Sensing and Controlling the Physical World with Computers』의 공동 저자다. MAKE 잡지의 후원자 중 한 명이고, 아두이노 오픈 소스 마이크로 컨트롤러 프로젝트의 공동 설립자기도 하다. 그는 언젠가 스발바르(노르웨이령), 남극 대륙을 방문하고 싶어한다.

지은이_ 돈 콜먼

돈 콜먼은 실제 기계는 물론 소프트웨어와 하드웨어까지 어우르는 숙련된 엔지니어로 기계공학의 발전과 함께했다. 그는 상당한 경험을 갖춘 폰갭 개발자로 폰갭의 시작부터 함께 해왔으며, 폰갭의 이점을 널리 알려왔다. 필라델피아 지역에 위치한 컨설팅 회사인 Chariot Solutions의 컨설팅 책임자로, 여러 팀을 이끌고 고객들과 함께 기존 기술을 진화시키고 발명하면서 미래를 위한 기반을 단단히 하고 있다.

지은이_ **브라이언 젭슨**

브라이언 젭슨은 MAKE 잡지의 편집자며 해커이자 Providence Geeks와 the Rhode Island Mini Maker Faire의 공동 책임자다. 또한, 로드아일랜드Rhode Island 주의 비영리 예술 센터인 AS220의 후원자기도 하다. 참고로 AS220은 로드아일랜드의 예술가들에게 검열과 심의가 필요없는 포럼을 제공하며, 갤러리, 공연, 제작 설비, 거주/작업 공간도 제공하고 있다.

역자 소개

옮긴이_ **이희정**

국민대학교 컴퓨터응용학과를 졸업하고, 한양대학교 대학원 컴퓨터공학과에서 석사를 취득했다. 모바일 관련 기업에서 7년간 근무했고, C와 C++ 기반 브라우저 개발에 참여했으며, 국내뿐만 아니라 동남아시아 여러 국가의 모바일 솔루션 상용화에도 참여했다. 또한, 『뿌리부터 이해하는 C 언어』(한빛미디어, 2014)의 저자며, 『자바 8 람다: 자바 개발자를 위한 함수형 프로그래밍』(한빛미디어, 2014)의 역자다. 현재 다양한 분야의 컴퓨터 관련 서적을 번역하며 안드로이드 앱을 개발하는 등 프리랜서로 활동 중이다.

- email: heejoung80@gmail.com
- homepage: http://hjscombooks.blogspot.com/

저자 서문

이 책의 시작은 2001년 3월에 브라이언이 톰에게 보낸 이메일이 발단이 되었다. 브라이언은 근거리 통신을 다룬 다른 책인 『Making Things Talk』의 2판을 작업 중이었으며, 이 책에 몇 줄을 추가하면 좋겠다고 생각했다. 이 책에는 이미 RFID(Radio Frequency IDentification)에 관한 내용이 포함되어 있었으므로 크게 힘들지 않을 것이라 생각했다. 하지만 2년 반 후 NFC에 대해 좀 더 많은 것을 알게 되었을 때 단순히 몇 줄로는 부족하다는 것을 깨달았다. 그래서 이 분야의 다양한 지식을 가진 돈 콜먼(폰갭을 위한 NFC 플러그인을 만들었다)과 함께 이 책을 집필하게 되었다.

NFC는 엄청난 잠재력이 있지만 프로그래머를 위한 책은 많지 않았다. 지금까지의 책에서는 NFC를 배우려면 납땜부터 준비해야 했다. 또한, 다양한 RFID 표준을 이해해야 하고 NFC 리더기에서 한 바이트씩 읽고 해석한 다음, 코드를 작성해야만 했다. 물론 이 모든 과정을 거치는 게 NFC를 완전히 이해하기에는 상당히 유용하긴 하다. 하지만 이 책에서는 후반부의 하드웨어 부분을 제외하면 전반적으로 NFC를 어떻게 더욱 다양한 방법으로 사용할 수 있는지에 초점을 맞추었다. NFC 툴 중에서는 돈 콜맨의 폰갭(PhoneGap) 라이브러리를 사용한다. 이 툴은 NFC 포럼에 있는 디자인처럼 NFC 데이터 교환을 디자인할 수 있다. 개발자는 교환할 메시지만을 고려하고 나머지 부분은 걱정할 필요가 없다.

이 책에서는 기기와 태그 또는 기계 간의 메시지를 읽고 쓰는 NDEF(NFC Data Exchange Format)의 기본 사항을 설명한다. 또한, 라즈베리 파이(Raspberry Pi), 비글본 블랙(BeagleBone Black)과 같은 임베디드 기기에서 동작하는 폰갭, 아두이노(Arduino), Node.js를 위한 예제 애플리케이션을 작성한다.

하지만 플랫폼마다 기술 수준이 다르기 때문에 NFC 포럼의 표준 명세에 기술된 모

든 플랫폼을 이 책에서 전부 알아볼 수는 없다. 이 책은 전체적인 지침을 알려주는 게 목적이며 책의 후반부에서는 현재 개발 상태와 앞으로 가능한 개선 방법에 대해 다룬다.

이 책이 프로그래머에게 NFC를 사용해 무엇을 할 수 있는지 알려줄 수 있기를 기대한다. 또한 전문 개발자에게도 NFC의 용도를 널리 전파하면서도 사용이 간단한 툴을 만드는 데 영감을 줄 수 있기 바란다.

역자 서문

일반 Windows 운영체제에서 구동되는 프로그램만을 개발했다면 NFC라는 용어가 생소할 수도 있고 어려울 것이라고 지레 겁먹을 수도 있다. 하지만 NFC는 이미 우리가 생각하는 것 이상으로 실생활에서 널리 사용되고 있으며 절대 복잡하고 어렵지 않다. 이 책이 NFC에 더욱 가깝게 다가가는 것을 도와줄 것이다.

NFC 태그는 상업적 목적으로 개발되었지만 개인적인 용도로도 충분히 가치가 있다. 예를 들어 NFC 태그로 개인 명함을 만들 수 있다. 명함 정보가 담긴 모바일 기기를 다른 기기에 가까이 가져가면 자동으로 명함 정보가 다른 기기에 옮겨지는 것이다. 흔한 종이 명함보다 훨씬 더 인상적이고 멋져 보일 것이다. 이 외에도 NFC 태그의 활용도는 무궁무진하며 계속 발전할 전망이다.

이 책은 NFC의 기본 개념을 익히는 것부터 실제 프로젝트를 개발하고 관련된 하드웨어를 조작하는 것까지 필요한 모든 내용을 담고 있다. 태그와 앱의 연동, 필립스 휴 조명 시스템 제어, 문 잠금 시스템 등 간단한 프로젝트부터 조금 복잡한 프로젝트까지 기술을 넓게 사용하는 데 큰 도움을 줄 것이다. 이 책을 통해 NFC의 기본을 익히고 더 나아가 다양한 NFC 개발에 영감을 얻기를 바란다.

이 책에 대하여

누구를 위한 책인가

전문 개발자가 아니어도 이 책을 읽을 수 있다. 프로그래밍에 대한 열정이 있고 딱딱하고 형식적이지 않은 방법으로 새로운 지식을 학습하려는 이들도 이 책을 읽을 수 있게 노력했다. 코딩을 전문적으로 배우지 않았어도 근거리 통신에 대한 실용적인 도입 방법과 안드로이드, 아두이노, 리눅스에서 어떻게 애플리케이션을 작성하는지를 학습할 수 있다.

물론 이 책의 독자는 약간의 프로그래밍 지식이 있다고 가정했다. 예제 대부분은 자바스크립트와 HTML로 작성되었고, 아두이노 프로젝트에서 잠시 C 언어를 접하게 된다. C 언어가 익숙하지 않더라도 자바스크립트나 자바에 익숙하다면 간단한 C 코드를 이해하는 것은 어렵지 않다. 후반부의 프로젝트에서는 기계공학 지식을 알고 있다면 매우 좋겠지만 모르더라도 크게 상관은 없다.

이 책의 구성

『처음 시작하는 NFC 프로그래밍: 안드로이드에서 개발하기』(1장 ~ 5장), 『처음 시작하는 NFC 프로그래밍: 아두이노, 라즈베리 파이에서 개발하기』(6장 ~ 8장)의 두 권으로 구성되어 있다.

1장에서는 NFC를 소개하고 NFC와 RFID를 비교해 본다. 간단히 말해서 NFC는 RFID의 확장이다. 근거리short-range의 RFID로 할 수 있는 대부분의 기능을 포함해 추가 기능도 사용할 수 있다. 중요한 용어와 NFC 시스템의 구조, 어떤 툴을 어디서 내려받는지를 알아본다.

2장에서는 폰갭과 폰갭의 NFC 플러그인을 소개한다. 안드로이드의 폰갭 애플리

케이션을 개발하기 위한 툴을 설치하고 간단한 애플리케이션을 구현해 본다. 2장 끝에서는 안드로이드 기기를 이용해 NFC 태그를 읽어본다.

3장에서는 NDEF(NFC Data Exchange Format)를 자세히 알아본다. NDEF의 구조를 익히고 NDEF 레코드의 각기 다른 타입을 사용하는 간단한 애플리케이션을 제작해 각각의 레코드 타입이 안드로이드와 어떻게 상호 작용하는지 알아본다.

4장에서는 안드로이드에서 어떻게 NDEF 메시지를 읽는지 알아본다. 다른 태그와 메시지를 걸러내는 방법, NFC 애플리케이션을 개발할 때 안드로이드 태그 처리 시스템이 어떻게 사용되는지를 익혀본다.

5장에서는 안드로이드에서 NFC 태그로 조정하는 사용자 인터페이스, 오디오 플레이, 웹 연결 컨트롤이 있는 NFC 애플리케이션을 만들어본다. NFC의 장점을 최대한 유지하면서 어떻게 애플리케이션의 디자인과 상호 동작하는지, 데이터를 설정하는지 설명한다.

6장에서는 또 다른 환경인 아두이노 마이크로 컨트롤러 개발 플랫폼을 사용한다. 아두이노 NDEF 라이브러리를 사용해 어떻게 NDEF 메시지를 읽고 쓰는지를 익혀본다. 또한, 아두이노와 Node.js를 사용한 애플리케이션을 만들어본다.

7장에서는 안드로이드에서 NFC의 피어-투-피어(peer-to-peer) 교환 방식을 소개한다. 피어-투-피어를 통해 레코드 타입이 수신하는 기기에 어떠한 영향을 주는지 알아본다. 추가로 블루투스(Bluetooth)와 와이파이(WiFi)와 같은 대용량 통신과의 연결을 시작하는 방법을 알아본다.

8장에서는 라즈베리 파이와 비글본 등 임베디드 리눅스 플랫폼에서의 NFC 개발을 대해 알아본다. 임베디드 리눅스에서 가능한 것과 Node.js 예제 애플리케이션을 알아본다. 이 장에 기술된 예제는 개선할 수 있는 사항들이 많으니 여러분이 직접 개선해 보길 바란다. 리눅스 명령어를 이미 알고 있다면 8장을 이해하는 데 많은 도움이 된다. 참고로 리눅스는 NFC를 사용할 수 있는 가장 흥미로운 플랫폼이며, 이를 잘 익히면 더욱 도움이 될 것이다.

이 책을 위해 필요한 것들

이 책의 연습 문제를 풀려면 몇 가지 하드웨어와 소프트웨어가 필요하다. 다행히도 모든 소프트웨어는 무료다. 필요한 하드웨어 중 NFC 기능이 탑재된 안드로이드 기기가 가장 비싸다. 다음 리스트는 필요한 품목을 나열한 것이다.

하드웨어

다음은 예제를 푸는 데 필요한 하드웨어다.

- NFC 기능이 탑재된 안드로이드 기기
- NFC와 호환되는 태그들(기기와 호환이 되는지를 확인한다. '『처음 시작하는 NFC 프로그래밍: 안드로이드에서 개발하기』 1.6 기기와 태그 종류 확인'에 기기별로 호환되는 태그 타입이 나열되어 있으니 참고하길 바란다.)

5장에서는 다음이 필요하다.

- 필립스 휴 라이팅 시스템Philips Hue lighting system
- 블루투스 음악 리시버(예를 들어, 벨킨Belkin 블루투스 음악 리시버 또는 홈스팟HomeSpot의

NFC 기능이 탑재된 블루투스 오디오 리시버 등이 있다.)

6장에서는,

- 아두이노 우노Arduino Uno 마이크로 컨트롤러(Arduino, Adafruit, Seeed Studio, RadioShack 사이트 등에서 구매할 수 있다.)
- NFC 실드NFC Shield(Adafruit의 PN532 NFC/RFID 컨트롤러 실드 또는 Seeed Studio의 NFC Shield나 NFC Shield v2.0)
- 에이다프루트 실드를 사용한다면 에이다프루트의 실드 스태킹 헤더shield stacking header도 준비해야 한다.
- 12V 또는 그 이하의 솔레노이드 잠금장치solenoid-driven door lock. 책에서는 아마존에서 구매한 Amico 0837L DC 12V 8W 오픈 프레임 타입의 솔레노이드 제품을 사용했지만, 다른 솔레노이드 타입을 사용해도 무방하다. 에이다프루트와 시드 스튜디오에서도 유사한 솔레노이드 잠금장치를 판매하고 있으니 실드를 구매할 때 함께 구매하는 것도 좋은 방법이다.
- TIP120 달링톤 트랜지스터Darlington transistor
- 12V, 1000mA 전력 공급기. 2.1mm ID, 5.5mm OD, 솔레노이드에 전력 공급을 위한 연결 어댑터center-positive connector
- 점퍼선Jumper wire 또는 22AWG 단심선solid-core wire
- 두 개의 LED(적색, 녹색). 전자 제품을 파는 상점에서 쉽게 구매할 수 있다. 참고로 에이다프루트에서도 판매 중이다.
- LED를 위한 두 개의 220Ω 저항기

8장에서는,

- 비글본 블랙 또는 리눅스 마이크로 컨트롤러에 탑재된 라즈베리 파이
- 보드를 위한 1A 또는 그 이상의 전력 공급기
- SLC3711 비 접촉식 USB 스마트카드 리더
- 필수품목은 아니지만, 있으면 유용한 품목들
  - 보드에 사용할 USB 와이파이 어댑터(예를 들어, 에이다프루트의 Miniature WiFi (802.11b/g/n) 모듈)
  - NFC 어댑터를 위한 USB A-to-A 확장선A-to-A extender
  - USB에서 TTL 시리얼로 연결하는 케이블 또는 콘솔console 케이블

앞서 나열한 하드웨어 품목을 다른 공급 업체에서 구매한다면 [표 0-1]을 참고하면 된다. 각 공급처와 품목 번호가 나열되어 있다.

[표 0-1] 앞으로 사용할 전자 부품들

항목	MakerShed	Jameco	Digikey	SparkFun
12V 1000mA DC 전력 공급기		170245		TOL-00298
아두이노 우노 rev3 마이크로 컨트롤러 모듈	MKSP11	2121105	1050-1017-ND	DEV-09950
초록LED	MKEE7	333227		COM-09592
빨강 LED	MKEE7	333973		COM-09590
파랑 LED		2006764		COM-00529
노랑 LED	MKEE7	34825		COM-09594
MIFARE RFID 태그	MKPX4			SEN-10128
NFC 실드	MKAD45			
TIP120 달링턴 트랜지스터		10001_ 10001_ 32993_-1	TIP120-ND	
3.3V USB/TTL 시리얼 디버그 케이블				DEV-09717

비글본 블랙	MKCCE3	2176149	BB-BBLK-000-ND	
라즈베리 파이 모델 B	MKRPI2			DEV-11546
항목	Adafruit	Farnell	Arduino	Seeed
12V 1000mA DC 전력 공급기	798	636363		
아두이노 우노 rev3 마이크로 컨트롤러 모듈	50	1848687	A000046	ARD132D2P
초록 LED		1334976		
빨강 LED		2062463		
파랑 LED		1020554		
노랑 LED		1939531		
MIFARE RFID 태그				
NFC 실드	789			SLD01097P
TIP120 달링턴 트랜지스터	976	9294210		
3.3V USB/TTL 시리얼 디버그 케이블	954			
비글본 블랙	1278	2291620		ARM00100P
라즈베리 파이 모델 B	998	43W5302		

소프트웨어

다음은 예제를 푸는 데 필요한 소프트웨어다.

- 안드로이드 소프트웨어 개발 킷(자세한 내용은 '『처음 시작하는 NFC 프로그래밍: 안드로이드에서 개발하기』 2.2.1 개발 환경 구축하기'를 참고하기 바란다.)
- 코도바Cordova CLI, 폰갭을 위한 툴들(자세한 내용은 '『처음 시작하는 NFC 프로그래밍: 안드로이드에서 개발하기』 2.2.1 개발 환경 구축하기'의 '폰갭에서 코도바 CLI 설치하기'를 참고하기 바란다.)
- Node.js와 Node 패키지 매니저npm, Node Package Manager
- 텍스트 편집기(다양한 플랫폼에서 사용할 수 있는 GUI 편집기인 Sublime Text 2를 추천

한다. 물론, 일반적인 텍스트 편집기를 사용해도 무방하다.)

5장에서는 다음 항목이 필요하다.

- Zepto jQuery 라이브러리(Zepto.js에서 이용할 수 있다.)

6장에서는,

- 아두이노 IDE
- Seeed-Studio PN532 라이브러리
- 아두이노 NDEF 라이브러리(GitHub에서 이용할 수 있다.)
- 마이클 마골리스Michael Margolis의 아두이노 Time 라이브러리

당장 이 모든 항목을 설치하는 것은 아니니 걱정할 필요는 없다. 앞으로 각 소프트웨어가 필요할 때마다 설치 방법에 대해 자세히 알려줄 것이다.

다른 유용한 NFC 앱들

다음 항목은 앞으로 이 책에서 전반적으로 유용한 내용이다.

- 'NFC TagInfo by NXP'에서는 NFC 또는 마이페어Mifare 태그를 읽을 수 있으며, NDEF 레코드를 시험해볼 수 있다.
- 'NFC TagWriter by NXP'도 역시 같은 기능을 제공한다. 또한, 태그를 쓰고 복원할 수 있어 매우 유용하다.
- NFC Research Lab의 'NFC TagInfo'에서는 태그의 정보를 확인할 수 있다. NXP보다 더욱 고급 정보를 제공하며 태그의 메모리 덤프를 볼 수 있다. 이는 애플리케이션의 문제점을 해결하는 데 매우 유용하다.

3장에서는 다음 항목이 필요하다.

- TagStand의 Trigger
- TagStand의 NFC Writer
- 삼성의 TecTiles(미국과 캐나다에서만 사용할 수 있다.)
- vvakame의 App Lancher NFC Tag Writer
- 포스퀘어Foursquare 계정, 안드로이드 포스퀘어 앱

참고사항

이 책에서 사용하는 예제 코드는 https://github.com/tigoe/beginningnfc에서 내려받을 수 있습니다.

이 아이콘은 제안이나 팁을 의미한다.

이 아이콘은 일반적인 주석을 의미한다.

이 아이콘은 경고나 주의를 의미한다.

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 보다 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. 무료로 업데이트되는, 전자책 전용 서비스입니다.

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여 DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

6 | 아두이노와 NFC 소개

『처음 시작하는 NFC 프로그래밍 : 안드로이드에서 개발하기』 5장에서 NFC로 실생활에서 사용하는 기기를 연결해 기기를 초기화하고 제어할 수 있음을 알아봤다. 앞에서는 특정 하드웨어에 의존해 프로그램을 작성했지만, 반드시 그럴 필요는 없다. 예를 들어, 앞서 만들었던 무드 세터Mood Setter 애플리케이션이 조명/음악 컨트롤뿐만 아니라 창문의 블라인드를 올리고 내리는 기능, 문을 잠그는 기능, 더 나아가 불꽃놀이 기능 등을 추가할 수 있다면 더욱 만족스러울 것이다. 이러한 기능은 사용자 정의 컨트롤러, 몇 가지 컴포넌트, NFC를 사용하면 충분히 가능하다. 이제 우리는 스마트폰과 태블릿이 NFC를 사용하는 유일한 기기가 아니라는 것을 알게 되었다. 실제로 슈퍼마켓에만 가도 마이크로컨트롤러로 사용하는 NFC 모듈이 있다. 이번 장에서는 아두이노를 기반으로 동작하는 작은 규모의 플랫폼에 대해서도 알아보겠다.

6.1 디지털과 생활의 만남 : 아두이노

전자 기기를 이용해 새로운 형태의 물리적 인터페이스를 만드는 데 관심이 있다면 흔히 마이크로컨트롤러microcontroller부터 시작한다. 마이크로컨트롤러는 일상생활에서 사용하는 모든 기기에 들어있는 매우 간단하지만 프로그래밍을 할 수 있는 작은 컴퓨터다. 마이크로컨트롤러의 디지털 입력 핀과 출력 핀은 센서의 전기적 신호를 읽거나 생성할 수 있으며 모터, 조명, 스피커 등을 조작할 수도 있다. 아두이노 마이크로컨트롤러 플랫폼에는 개발 소프트웨어에 포함된 간단한 통합 개발 툴과 하드웨어에 미리 탑재된 다양한 마이크로컨트롤러 인터페이스가 있어 새로운 프로젝트 개발을 쉽게 시작할 수 있다.

흔히 사용하는 아두이노 프로젝트는 여러 가지 기능을 포함하고 있다. 사용자 정의

형태의 악기, 원격에서 애완동물의 먹이를 주는 기능, 환경 센서 프로젝트, 여러 가지 다른 형태로 동작하는 게임 컨트롤러 등이 포함되어 있다.

참고로 아두이노 소프트웨어와 하드웨어는 모두 오픈 소스다. IDE(통합 개발툴)를 위한 소스 코드, 펌웨어 라이브러리를 위한 코드, 전자 기판 회로 설계도도 아두이노 웹사이트에서 쉽게 수 있다.

또한, 아두이노는 범용성을 지닌 마이크로컨트롤러다. 실제로 입/출력 핀은 특정 애플리케이션에 종속적이지 않다. 즉, 핸드폰에 들어가는 CPU라고 생각해도 된다. 물론 아무리 프로그래밍을 할 수 있더라도 무선 라디오, 디스플레이 컨트롤러, 오디오 컨트롤러가 없다면 아무 의미가 없다(앞서 예시에서는 NFC 컨트롤러는 포함하지 않았다). 그러므로 아두이노를 더욱 유용하게 사용하려면 특정 형태의 컴포넌트에 연결해서 사용해야 한다. 통신 라디오, 자동차 컨트롤러, 센서 등이 좋은 예다. 아두이노 플랫폼은 이렇게 서로 다른 컴포넌트와 작업할 수 있도록 다양한 프로그램을 제공한다. 이를 실드shield라고 부른다. 이번 장에서는 가장 많이 사용할 아두이노 모델인 아두이노 우노를 NFC 라디오 실드에 연결하는 법을 배워보자.

마이크로컨트롤러가 전자 기기를 개발하는 데 자주 소개되지만, 이 책에서는 이를 많이 다루지는 않는다. 이 책의 프로젝트에서는 아두이노나 마이크로컨트롤러로 조종할 수 있는 미리 만들어진 전자 기기 모듈을 사용한다. 즉, 새로운 설계나 회로를 만들 필요는 없다. 기기에 필요한 기능을 실드와 미리 설치한 모듈에서 가져와 사용하므로 아두이노 환경에서 NFC 사용법을 익히는 데 더욱 집중할 수 있다.

NOTE_ 마이크로 컨트롤러와 운영체제

아두이노의 심장인 프로세서는 실생활에서 쉽게 접할 수 있는 커피메이커, 알람시계, 난방 기구 등에서 동작하는 저전력 컨트롤러다. 또한, 작동을 감지하는 센서를 이용하는 전등도 마이크로 컨트롤러가 있다. 이러한 아두이노 프로세서를 벌크[01] 제품으로 산다면 1달러 남짓한 금액에 구입할 수 있다. 실제로 이러한 전자 제품의 가격은 마이크로 컨트롤러로 정해지는 것이 아니라 전자회로, 다양한 입출력 단자, 제품 디자인, 쉽게 사용이 가능한 사용자 인터페이스 등으로 정해진다.

이러한 컨트롤러는 일반적으로 하나의 기능을 수행하는 애플리케이션에서 사용되며, 사용하는 전력이 매우 제한적이므로 운영체제의 특별한 부하 없이 동작하는 것이 보통이다. 이 책에서 사용하는 아두이노는 한 번에 구동할 수 있는 프로그램을 하나로 제한했으며 마이크로 컨트롤러의 전원이 꺼질 때까지 해당 프로그램을 구동한다. 최근까지의 마이크로 컨트롤러 프로그래밍은 운영체제가 없는 게 일반적이었다. 마이크로 컨트롤러 개발자는 원하는 기능을 잘 수행하는 프로그램을 간단하게 작성할 수 있는 스킬(기술)이 필요하다.

프로세서 사용 비용이 낮아지면 이 프로세서를 사용하는 운영체제의 비용도 같이 낮아진다. 이는 대부분 마이크로 컨트롤러에 같이 적용되는 부분이다. 라즈베리 파이의 개발 환경은 제한된 버전의 리눅스 운영체제에서도 충분히 사용할 수 있어서 많은 개발자가 취미 삼아 쉽게 접근할 수 있다. 몇 년 후에는 50달러 이하로도 이 모든 내장형 리눅스 개발 환경을 구축할 수 있을 것이다. 8장에서는 라즈베리 파이나 비글본 블랙 같은 내장형 시스템에서 동작하는 NFC에 대해 자세히 알아보겠다.

하지만 여기서 알아야 할 것은 이렇게 비용을 줄이면 줄일수록 운영체제에 가해지는 부담은 더 커진다는 것이다. 실생활에서 사용하는 센서나 작동 기기들은 기존에 사용하던 운영체제보다 훨씬 더 빠르게 응답을 할 수 있는 프로세서가 필요하다. 실시간 운영 체제RTOS는 멀티스레딩 같은 일반적인 운영체제가 가지는 중요 기능을 포함하며 비 운영체제 시스템의 빠르고 정확한 응답성도 가지고 있다. 하지만 파일 시스템이나 메모리 관리 기능은 상대적으로 많이 부족하다. 또한, 아두이

01 역자주_ 벌크란, 포장이 되어 있는 완제품이 아니라, 납품할 용도로 대량으로 생산된 제품을 뜻하기도 한다. 중고, 역수제품을 이렇게 부르는 경우도 있다.

노처럼 간단한 시스템도 없다. 그러므로 내장 운영체제와 하나의 프로그램만 구동하는 방법을 모두 잘 이해하면 상당한 도움이 될 것이다. 이번 장에서는 운영체제에서 NFC를 사용하는 방법과 비 운영체제를 이용하는 방법도 알아보겠다.

6.2 NFC 하드웨어의 중심

NFC 통신의 중심에는 무선 컨트롤러가 있다. 지금까지 사용한 태그처럼 수동적인 대상을 읽을 때는 컨트롤러가 개시자가 된다. 컨트롤러는 『처음 시작하는 NFC 프로그래밍 : 안드로이드에서 개발하기』 1장에서 설명했던 것처럼 태그를 읽는 무선 신호을 생성한 후 다시 받게 되는 신호을 기다린다. 개시자가 대상 자체인 경우 신호로 들어오는 무선 신호과 응답을 읽는다. 반면에 컨트롤러는 모뎀(단말기)처럼 동작한다. 컨트롤러는 전송과 무선 신호 처리를 담당하고 이렇게 읽은 신호을 중앙 프로세서를 이용해 통신 데이터로 변환한다.

현재 시장에서 가장 많이 쓰이는 NFC 컨트롤러 중 하나는 NXP PN532이다. 여러 가지 다른 NFC 모듈과 PN53x 계열의 컨트롤러는 시중에 있는 대부분의 NFC 가능 기기에서 쉽게 찾아볼 수 있다. 또한, 다음 5가지 다른 컨트롤러와 호환되는 모듈도 있다. 먼저 PN532, 에이다프루트Adafruit의 v1.3에서 파생된 PN532 NFC/RFID 컨트롤러, 또 다른 에이다프루트 파생 버전이면서 아두이노를 위해 만들어진 PN532 NFC/RFID 컨트롤러 실드, Seeed 스튜디오의 NFC 실드와 NFC 실드 v2.0 등이 있다. 이 모듈들은 이번 장에서 사용할 NDEF 라이브러리와도 완벽하게 호환된다.

안드로이드 기기의 NFC 컨트롤러는 CPU의 I/O 핀과 연결되어 있고 곧 살펴보게 될 내용과 비슷한 구조의 프로토콜을 사용해 통신한다. 폰갭의 NFC 플러그인은 아두이노 라이브러리와 같은 일을 수행하는데, 이에 대해서도 곧 살펴보겠다. 이를

이용하면 안드로이드 기기의 중앙 프로세서를 이용해 기기의 NFC 칩을 제어할 수 있다.

6.3 아두이노 개발 환경

아두이노 개발을 시작하려면 다음 기기가 필요하다.

- 아두이노, 에이다프루트, Seeed 스튜디오 등에서 구할 수 있는 아두이노 우노 마이크로컨트롤러
- 아두이노 IDE

이번 장의 모든 라이브러리와 예제는 모두 우노에서 테스트했다. 하지만 Due나 Leonardo에서는 테스트하지 않았다. 이들은 Atmega328 프로세서를 사용하는 아두이노에서 파생된 모든 기기에서 잘 동작한다. 다른 비주류 프로세서에서도 잘 동작하리라 생각하지만 100% 확신할 수는 없다.

아두이노 IDE를 내려받은 후 설치하고 실행해 보자. 다음과 같은 코드를 담고 있는 편집기의 창에 나타날 것이다.

```
void setup() {
  // 프로그램 실행 중 한 번만 수행되는 초기화 코드를 이곳에 작성한다.
}
void loop() {
  // 반복적으로 수행되는 핵심 코드는 이곳이 작성한다.
}
```

아두이노 프로그래밍 프레임워크와 아두이노를 위한 코드는 모두 C++로 작성한다. 너무 걱정할 필요는 없다. 프레임워크를 이용하면 C++의 복잡한 부분들을 간소화해 더욱 쉽고 편안하게 코드를 작성할 수 있다. 모든 아두이노 프로그램에는

필수로 setup()과 loop()라는 중요한 두 개의 함수를 가진다. 먼저 setup() 함수는 프로세서의 전원이 들어왔을 때나 리셋되었을 때 딱 한 번만 수행된다. loop() 함수는 setup() 함수 다음에 수행되며 전원이 꺼지거나 리셋될 때까지 반복적으로 수행된다. 콜백도 없고 멀티스레딩도 없는 단순한 하나의 반복 구문이다. 물론 추가 작업을 위한 함수를 loop() 안에서 마음대로 호출할 수 있다.

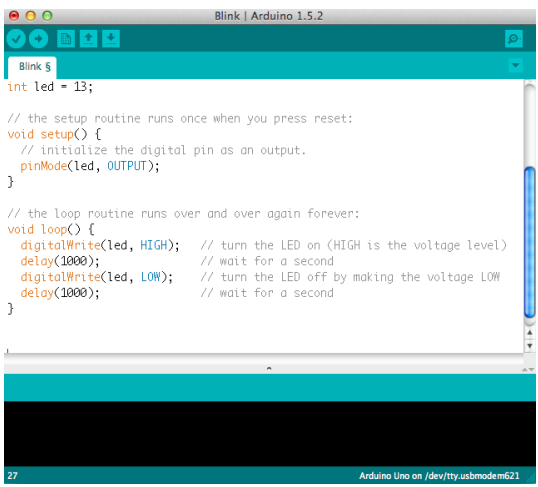
이제 아두이노의 첫 번째 프로그램을 작성해 보자.

```
const int led = 13; // 사용하는 LED 핀 번호 변수 선언

void setup() {
  pinMode(led, OUTPUT); // 13 핀을 출력으로 초기화
}

void loop() {
  digitalWrite(led, HIGH); // LED 켜기(HIGH는 전압의 세기, 즉 고전압을
의미한다)
  delay(1000); // 1초 대기
  digitalWrite(led, LOW); // LED 끄기(LOW는 낮은 전압을 의미한다)
  delay(1000); // 1초 대기
}
```

코드 작성을 마친 후 메뉴의 'Tools'을 열어 'Board Menu'로 이동한 후 '아두이노 우노'를 선택한다. Tools의 Serial Port 메뉴(최신 버전 아두이노에서는 Tools → Port)에서 사용할 수 있는 직렬 포트를 볼 수 있다. 지금 단계에서는 '아두이노' 이름의 포트는 존재하지 않는다. 일단 아두이노 우노를 USB에 연결한다. Windows 사용자라면 드라이버를 설치해야 한다. IDE를 내려받은 위치에서 드라이버를 찾는다. 연결을 마친 후 다시 리스트를 확인해 보자. 이제 아두이노 우노 이름을 지닌 새로운 포트를 리스트에서 찾을 수 있을 것이다. OS X의 경우에는 두 가지 포트가 있다. 하나는 /dev/cu.usbmodem-xxx이고, 다른 하나는 /dev/tty.

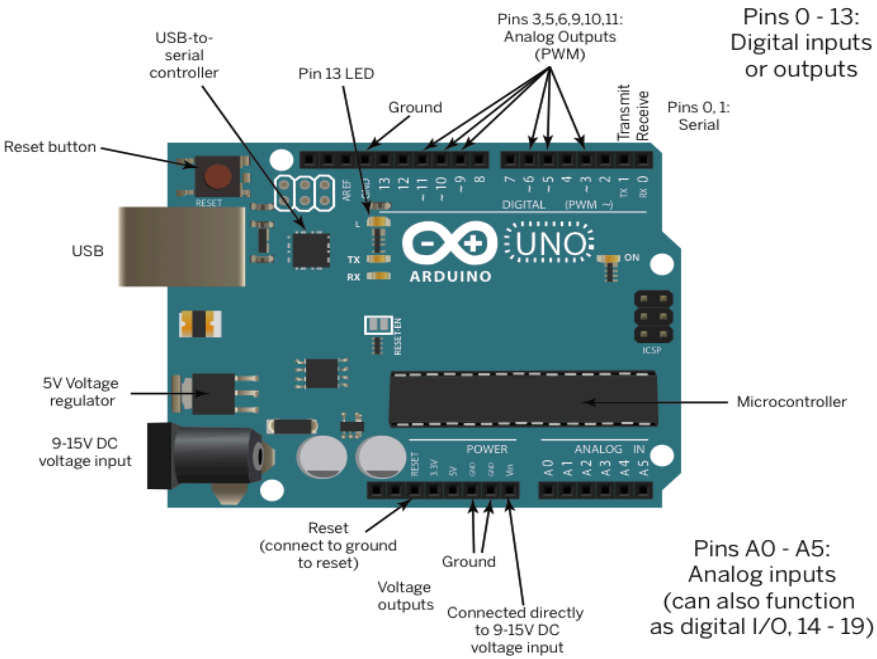
usbmodem-xxx이다. 참고로 xxx는 보드의 직렬 번호다. 이들은 결국 기능상 같은 포트이므로 어느 곳에 연결하든 상관없이 잘 동작한다. 예를 들면 OS X 10.8을 구동 중인 2011-era 맥북 에어는 /dev/tty.usbmodem621 또는 /dev/tty.usbmodem421 포트를 가진다. Windows 7 또는 8에서의 포트는 COM3이다. 리눅스에서는 일반적으로 /dev/ttyACM0이다. 포트를 선택한 후 툴바의 'Upload' 버튼을 누르면 된다. [그림 6-1]을 참고해서 따라 해 보자.

[그림 6-1] 아두이노 IDE



[그림 6-1]에서 툴바의 버튼도 확인할 수 있다. 왼쪽부터 Verify(코드를 컴파일), Upload(컴파일한 후 업로드), New, Open, Save 버튼이다. 가장 오른쪽 버튼은 직렬 포트 모니터 기능이다. 선택한 보드의 종류와 직렬 포트의 정보는 전체 창의 오른쪽 아래에 있다. IDE는 코드를 컴파일할 뿐만 아니라 해당 보드에 업로드도 담당한다. 업로드하면 보드는 자동으로 재시작한다. 이때 [그림 6-2]에서 볼 수 있는 것처럼 아두이노 로고 오른쪽의 LED가 깜빡인다. 실제로 이 LED는 컨트롤러 I/O핀 중 13번과 연결되어 있다. 이러한 이유로 앞의 코드에서 13번을 사용한 것이다.

[그림 6-2] 아두이노 우노(이미지는 Maker Media에 저작권이 있고, 톰 이고[Tom Igoe]가 작성한 『Making Things Talk』 2판에서 발췌했다. 그린 사람은 조디 컬킨[Jody Culkin]이다)



모든 마이크로컨트롤러와 마찬가지로 아두이노도 I/O 핀으로 들어오는 전기 신호를 읽고 분석하면서 동작한다. `digitalRead()` 명령은 주어진 핀에서 1이나 0을 읽어 온다. 이는 전기 신호의 강도에 따라 HIGH나 LOW라고도 불린다. 우노의 경우는 5V를 사용하므로 HIGH나 1 값은 5V를 의미하고, 반대로 LOW나 0 값은 0V를 의미한다. 한편 몇몇 I/O 핀들은 아날로그로 동작하기도 한다. A0 ~ A5의 아날로그 핀들은 0 ~ 5V까지의 아날로그 전기 신호를 읽는다. 0.05V의 정밀도와 최대 1,023단계를 가진다. 디지털 핀 3, 5, 6, 9, 10, 11의 경우에는 256단계의 정밀도를 바탕으로 다수의 핀에 빠르게 다른 신호를 주어서 끊임없이 변하는 아날로그 출력 신호를 구현할 수 있다.

앞의 코드에서 아두이노는 디지털 핀 13을 사용해 출력 핀을 초기화하는데, 13번 핀을 이용해 전기 신호를 주고받겠다는 것을 뜻한다. 반복 구문에서 프로세서는 `digitalWrite(ledPin, HIGH)` 함수를 사용해 5V의 전기 신호를 HIGH에 설정하고 `digialWrite(ledPin, LOW)` 함수를 호출해 LOW에 0V를 설정했다. 그 중간에 있는 코드 `delay(1000)`로 인해서 1초 동안 아무 일도 하지 않는다. 실제로 이 프로그램은 아두이노 프로세서에서 구동 중인 유일한 프로그램이므로 `delay()` 함수는 프로세서를 아무 일도 하지 않게 만든다. 나쁘지는 않지만 추천되지도 않는 방법이기에 자주 이용하지는 않는다.

IDE는 이러한 스케치 프로젝트(초기 프로젝트를 뜻한다)를 Documents 디렉터리 밑의 Arduino 디렉터리에 생성한다. 편의상 이번 프로그램을 Blink라 이름 지었다. 앞으로 다양한 스케치 디렉터리를 접하게 될 것이다.

핵심적인 아두이노 명령어들을 자세히 살펴보려면 아두이노 언어 레퍼런스 페이지를 참고하자. 또한, 플랫폼을 이용하는 방법을 보여주는 많은 양의 예제 코드가 Learning 부분에 있다. 더욱 간편한 방법으로는 IDE에서 File 메뉴의 'Examples'을 열면 다양한 예제 코드를 확인할 수 있다.

6.3.1 직렬 통신

기본적인 I/O 기능을 넘어서 아두이노는 USB로 비동기 직렬 통신을 할 수 있고 PC532 칩을 사용하는 I2C, SPI 프로토콜과도 통신할 수 있다. 또한, 사용하는 컴퓨터로 메시지를 주고받는 용도로 직렬 통신과 USB를 함께 이용하는 경우도 많다. 스케치라고 불리는 개발 프로그램을 우노로 올릴 때도 직렬 통신을 사용한다. 다음 스케치는 USB와 직렬로 연결된 컴퓨터에서 입력을 받으면서 13번 핀을 바탕으로 LED 기능을 제어하는 방법을 보여주는 코드다. 전체 소스 코드는 GitHub에서 찾을 수 있다.

```
const int led = 13; // 사용하는 LED 핀 번호 변수 선언

void setup() {
  Serial.begin(9600); // 9600 bps 속도의 직렬 통신 연결
  pinMode(led, OUTPUT); // LED 핀을 출력으로 설정
}

void loop() {
  if (Serial.available()) {
    char input = Serial.read(); // 직렬 포트로부터 바이트 읽기
    if (input == 'H' || input == 'h') { // 'h'나 'H'인 경우
      digitalWrite(led, HIGH); // LED 켜기
      Serial.println(input); // 사용자의 입력을 화면에 출력
    }
    else if (input == 'L' || input == 'l') // 'l'이나 'L'인 경우
      digitalWrite(led, LOW); // LED 끄기
      Serial.println(input); // 사용자의 입력을 화면에 출력
    }
  }
}
```

이 스케치를 업로드하고 툴바의 오른쪽에 있는 직렬 포트 모니터Serial Port Monitor를 선택하면 직렬 모니터 새 창이 열린다. 'H'나 'h'를 입력한 후 'send'나 엔터 키를 누르면 13번 핀과 연결된 LED에 불이 들어온다. 'L'이나 'l'을 입력하면 불이 꺼진다.

지금까지 살펴본 디지털 입출력, 아날로그 입출력, 직렬 통신을 이해했다면 아두이노의 기본 동작을 완전히 이해한 것이다. 실제로 모든 일은 입력 핀으로부터 전기 신호 읽기, 출력 핀으로 전기 신호 전송, 직렬 프로토콜을 이용한 디지털 통신으로 이루어진다. 직렬 프로토콜은 특정 I/O 핀들에 더 강하고 길게 지속적인 전기 신호를 보내는 것이라고 이해하면 된다.

아두이노 개발 프로젝트를 더욱 자세히 알고 싶다면 마시모 벤치Massimo Banzi의 『Getting Started with Arduino』나 톰 이고Tom Igoe의 『Making Things Talk』나 마이클 마고리스Michael Margolis의 『Arduino Kochbuch』과 같은 아두이노 책을 참고하기 바란다. 최근에는 웹에서 다양한 프로젝트, 예제 코드, 라이브러리, 추가 모듈을 쉽게 구할 수 있다.

NOTE_ SPI, I2C, UART?

앞서 언급한 실드의 중심인 PN532 NFC 칩은 마이크로 컨트롤러가 제어하는 세 가지 다른 통신 인터페이스가 있다. 이 칩은 UARTUniversal Asynchronous Receiver-Transmitter를 이용하는 비동기 직렬 통신을 사용해 데이터를 주고받을 수 있다. 또한, 동기 직렬 통신의 두 가지 형태인 SPISerial-Peripheral Interface)와 I2CInter-Integrated Circuit communication를 사용할 수도 있다. 비동기 직렬 통신은 두 개의 컴퓨터가 서로 각자의 내부 클록을 가지고 독립적으로 동작하는 환경에서 흔하게 사용된다. 이와는 반대로 SPI와 I2C는 하나의 마스터 컴퓨터와 다른 기기들이 있고 통신 방향을 마스터가 결정하는 방법을 사용한다. 비동기 직렬 연결은 언제나 일대일 통신이다. 이 경우에도 SPI와 I2C는 다르게 동작하는데 버스 프로토콜이므로 같은 통신 연결을 공유하는 개별 주소가 있는 기기들을 다수 연결할 수 있다. 프로그래밍할 수 있는 대부분의 마이크로 컨트롤러는 이 세 가지 인터페이스를 제공한다. 이 인터페이스를 더 알아볼 수도 있겠지만 이 책의 범주에서 넘어서므로 자세히 설명하지는 않겠다.

안타깝게도 아두이노에서 사용하는 모든 모듈과 실드가 이 세 가지 인터페이스를 지원하지는 않는다. 아두이노 보드는 이 세 가지의 인터페이스를 제공하지만 에이다프루트의 실드는 SPI와 I2C만 지원하고 Seeed Studio는 오직 SPI만을 지원한다.

이 책을 쓰는 시점에서는 사용할 수 있는 PN532 컨트롤러를 위한 아두이노 라이브러리가 몇몇 존재한다. 이들 중 몇몇은 SPI 인터페이스를 사용하고 다른 것들은 I2C 인터페이스를 사용한다. 이 책에서 사용한 NDEF 라이브러리는 Seeed Studio의 PN532 라이브러리의 래퍼인데 SPI와 I2C의 기능들을 지원한다. 이번 장에서는 알맞은 버전의 PN532 라이브러리와 NDEF 라이브러리를 포함하는 방법도 살펴볼 것이다.

6.3.2 아두이노 라이브러리 설치하기

아두이노 환경은 다른 일반적인 프로그래밍 플랫폼처럼 확장이 가능한 라이브러리들로 구성된다. NFC를 아두이노와 함께 사용하려면 몇 가지 라이브러리가 필요하다. 돈의 GitHub 레퍼지토리에서 NDEF 라이브러리를 내려받는다. 이것을 아두이노 라이브러리로 설치하려면 스케치 메뉴의 'Import Library…'를 선택한 후 하위 메뉴의 'Add Library…'를 클릭한다. 압축을 풀지 않은 ZIP 파일을 선택해 라이브러리를 설치할 수도 있고 원하는 디렉터리에서 'All Files'를 선택해 설치할 수도 있다. IDE는 양쪽 경우 모두를 잘 구별해 설치한다. 참고로 각 라이브러리 디렉터리에는 'Examples'이라 불리는 하위 디렉터리가 있다. 라이브러리를 성공적으로 설치했다면 파일 메뉴에 있는 'Examples'에서 이 예제들의 이름을 확인할 수 있다.

또한, 사용 중인 실드를 위한 어댑터 라이브러리도 필요하다. PN532 칩에 사용할 수 있는 많은 라이브러리가 있지만 다음 예제에서는 Seeed 스튜디오의 라이브러리를 사용한다. 이 라이브러리는 Seeed 실드 뿐만 아니라 에이다프루트 실드에서도 잘 동작한다. 이는 Seeed 스튜디오의 GitHub 레퍼지토리에서 받을 수 있다. 참고로 Seeed 스튜디오의 NFC 라이브러리는 세 가지 다른 컴포넌트로 구성된다. PN532 폴더, PN532_I2C 폴더, PN532_SPI 폴더가 각각의 컴포넌트다. 이들을 아두이노 1.5 이상의 버전에서 사용하려면 각 라이브러리를 시스템에 하나씩 불러들여야 한다. 라이브러리들을 성공적으로 불러들인 후 코드를 컴파일한다. 이때 사용하지 않는 통신 프로토콜을 위한 드라이버를 추가하지 않으므로 메모리 사용량을 줄일 수 있다.

다른 라이브러리와 라이브러리 네임스페이스의 충돌

에이다프루트의 NFCShield I2C도 PN532와 함께 사용할 수 있는 매우 훌륭한 라이브러리로 GitHub에서 간단하게 내려받을 수 있다. 에이다프루트 라이브러리는

오직 I2C 인터페이스와 동작하므로 에이다프루트 실드만이 사용할 수 있다. 하지만 매우 안정적으로 동작하고 Seeed 라이브러리에 비해서 적은 양의 메모리를 사용하게 한다. 또한, 에이다프루트 라이브러리는 아두이노 NDEF 라이브러리와 자동으로 동작하지는 않는다. 하지만 약간 수정해주면 동작하게 만들 수 있다.

저수준 NFC 라이브러리를 사용했다면 아마도 여전히 오래된 버전의 라이브러리들을 가지고 있을 것이다. 예제 코드를 이 환경에서 컴파일하면 에러가 발생한다. 이 경우에는 먼저 IDE를 종료하고 PN532와 관련된 모든 라이브러리를 아두이노 라이브러리 디렉터리에서 삭제하고 다시 IDE를 실행한다. 그런 후 아두이노 스케치 디렉터리에 있는 새로운 라이브러리를 사용한다.

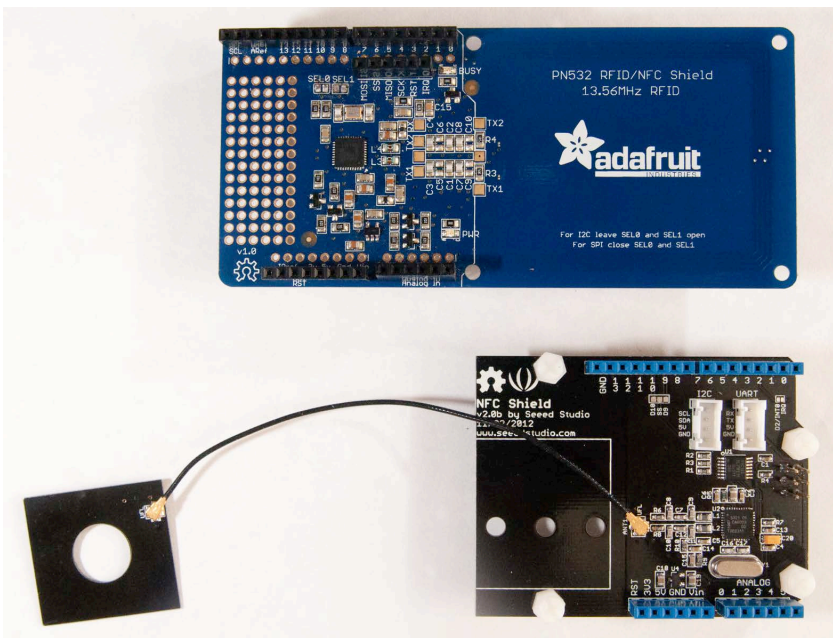
다음 예제들을 실행하려면 다음의 하드웨어와 소프트웨어가 필요하다.

- 아두이노 우노 마이크로컨트롤러
- 아두이노 IDE
- NFC 실드(에이다프루트의 PN532 NFC/RFID 컨트롤러, Seeed 스튜디오의 NFC 실드, NFC 실드 v2.0 등도 사용할 수 있다.)
- 에이다프루트 실드를 사용 중이라면 에이다프루트의 실드 스태킹 헤더들도 필요할 것이다. 이처럼 이 실드를 제대로 사용하려면 납땜 등의 몇 가지 추가 작업들이 필요하지만 충분히 가치 있는 일이다.
- 몇 가지 NFC 태그(특별히 이번 장 예제들은, 마이페어 클래식이 사용하는 아두이노 라이브러리와 가장 잘 동작할 것이다. 하지만 이는 몇몇 안드로이드 기기와는 동작하지 않는다. '1.6 기기와 태그의 종류 확인'을 참고하여 사용 가능한 기기를 잘 선택하도록 한다)
- 아두이노 IDE 1.5.3 이상의 버전
- 아두이노 NDEF 라이브러리(돈 콜맨의 GitHub 레퍼지토리에서도 구할 수 있다.)

- Seeed 스튜디오의 PN532 라이브러리

에이다프루트 실드는 핀 헤더와 결합하지 않은 형태로 출시되므로 직접 납땜을 해야 한다. 깔끔하게 납땜하기가 약간 어려울 수도 있으니 튜토리얼에 있는 사진을 참고하면서 차분히 해보기 바란다. 납땜을 마친 후 실드를 아두이노 우노에 연결하면 모든 준비가 완료된 것이다. 실제로 연결하는 방법은 매우 간단하므로 미리 겁먹을 필요는 없다.

[그림 6-3] 그림의 상단 사진은 아두이노를 위한 에이다프루트 NFC 실드고, 하단 사진은 아두이노 v2.0을 위한 Seeed 스튜디오의 NFC 실드다.

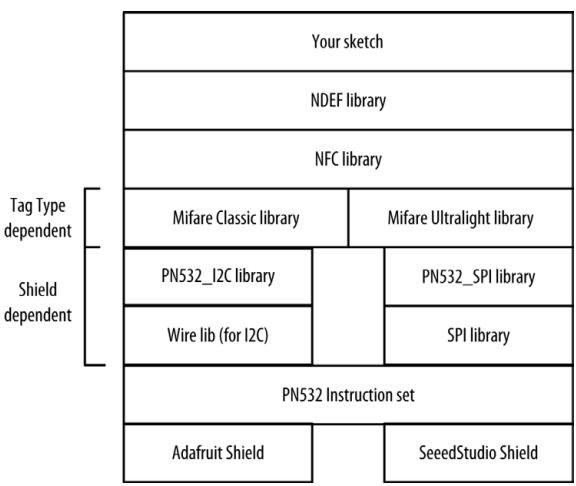


6.4 아두이노 NDEF 라이브러리

현재 사용 중인 라이브러리를 스택이라 생각해도 된다. PN532 어댑터 라이브러리

는 실드의 PN532 칩과 통신하는 저수준의 명령어를 제공하는데, 하나는 SPI고 다른 하나는 I2C다. 이 명령어들은 명령어 자체를 태그에서 추출한 데이터 문자열 바이트 형태로 보낸다. 이 바이트 데이터를 확인하려면 바이트 배열을 분석할 수 있는 라이브러리가 필요하다. 『처음 시작하는 NFC 프로그래밍 : 안드로이드에서 개발하기』 1장에서 살펴봤던 마이페어 클래식, 마이페어 울트라라이트 등이 그 예다. 이 태그 라이브러리 위에는 태그와 전송하는 바이트 스트림의 차이점을 제거할 수 있는 NFC 라이브러리가 있다. NFC 위에는 태그 타입 라이브러리에서 전달된 바이트를 읽어 연속된 NDEF 메시지나 레코드를 알아내는 NDEF 라이브러리가 있다. [그림 6-4]는 이러한 스택 단계와 라이브러리의 관계를 보여준다.

[그림 6-4] 아두이노 NFC 라이브러리 스택



NDEF 라이브러리를 사용하지 않는다면 태그에 전송되는 바이트 데이터를 하나씩 단계적으로 분석해야 한다. 이를 사용하면 저수준에서 이루어지는 일을 간단히 무시할 수 있고, 이전에 해왔던 것처럼 NDEF 메시지를 읽고 쓰는 데 더욱 많은 시간을 투자할 수 있다. 즉 저수준 작업을 할 필요가 없다면 라이브러리를 사용하자.

아직 태그 타입 헬퍼 라이브러리 작성이 끝나지 않았다. 지금까지지 알아본 NDEF 라이브러리 레퍼지토리의 태그 라이브러리들은 마이페어 클래식 태그를 읽고 쓰거나 태그 타입 2(마이페어 울트라라이트)를 읽을 수 있다. 반면에 다른 라이브러리들은 그렇지 않으므로, 이 중 하나를 사용하거나 고유한 태그 작성 라이브러리를 직접 만들어야 한다.

아두이노를 위한 NDEF 라이브러리는 몇몇 객체와 앞서 살펴본 친숙한 개념들을 포함하고 있다.

NfcAdapter
: 이 객체는 저수준 어댑터 라이브러리의 인스턴스로 태그를 찾아내는 일을 한다.

NfcTag
: 이 객체는 태그의 UID, 사용된 기술 및 크기 등의 메타데이터 정보가 들어있다.

NdefMessage
: NDEF 서식을 갖춘 태그를 읽을 때 NdefMessage도 얻게 된다. 이 객체는 바이트를 NDEF 서식으로 인코딩/디코딩하는 일을 담당한다.

NdefRecord
: 언어정보, TNF, 타입 등, 모든 레코드의 메타데이터와 페이로드 전송을 담당한다.

이 라이브러리는 메시지를 읽고 쓰게 하는 다양한 헬퍼 함수들이 있다. 물론 이 모든 함수를 이용하지는 않지만 대부분 함수는 이미 알고 있는 NDEF의 내용과 비슷한 이름을 사용하므로 기능을 직관적으로 알 수 있다. [표 6-1]은 이 함수들을 클래스별로 정리한 것이다.

[표 6-1] 아두이노를 위한 NDEF 라이브러리 함수 목록

NfcAdapter 함수	NfcTag 함수	NdefMessage 함수	NdefRecord 함수
begin(void)	getUidLength()	getEncodedSize()	getEncodedSize()
tagPresent()	getUid()	encode()	getTypeLength()
read()	getUidString()	addRecord()	getPayloadLength()
write()	getTagType()	addMimeMediaRecord()	getIdLength()
	hasNdefMessage()	addTextRecord()	getTnf()
	getNdefMessage()	addUriRecord()	getPayload()
		addEmptyRecord()	getType()
		getRecordCount()	getId()
		getRecord()	setTnf()
			setType()
			setPayload()
			setId()

6.4.1 아두이노에서 NDEF 읽기

NDEF 라이브러리에 포함된 ReadTagExtended 예제는 라이브러리의 다양한 함수들을 올바르게 사용하는 방법을 보여주고 코드의 흐름도 잘 설명한다. 예제를 확인하려면 File 메뉴로 간 다음 'Examples', 그 후 'NDEF', 마지막으로 'ReadTagExtended'를 선택한다. 이 프로젝트를 아두이노에 올린 후 이미 만든 태그를 몇 가지를 읽어보자.

사용 중인 실드 기종에 따라 스케치의 시작 부분을 수정해야 할 수도 있다. 만약 Seeed 스튜디오 실드와 같은 SPI 기반 실드를 사용 중이라면 다음과 같은 코드로 시작한다.

```
#include <SPI.h>
#include <PN532_SPI.h>
#include <PN532.h>
#include <NfcAdapter.h>
```

```
PN532_SPI pn532spi(SPI, 10);
NfcAdapter nfc = NfcAdapter(pn532spi);
```

에이다프루트 실드와 같은 I2C 기반의 실드를 사용 중이라면 다음 코드와 같이 시작한다.

```
#include <Wire.h>
#include <PN532_I2C.h>
#include <PN532.h>
#include <NfcAdapter.h>

PN532_I2C pn532_i2c(Wire);
NfcAdapter nfc = NfcAdapter(pn532_i2c);
```

이후의 코드는 어떤 실드 종류와 관계없이 똑같다. SPI 기반의 실드를 사용 중이라면 앞의 코드처럼 시작 부분을 반드시 바꿔야 한다는 점을 꼭 기억하자.

다음 코드의 구현 부분은 기존 코드와 상당히 유사하다. 가장 처음으로 `nfc.tagPresent()`를 이용해 태그가 있는지 확인한다. 아두이노는 이벤트 핸들러를 가지고 있지 않으므로 이 작업을 위해서는 계속 태그의 여부를 확인해야 한다. 태그의 존재를 확인했다면 `NfcTag tag = nfc.read()` 구문을 통해 데이터를 읽고 태그 객체를 생성한다. 다음으로 `tag.getNdefMessage()` 구문으로 수신한 메시지를 확인한다. 메시지가 있는 경우 메시지 안의 레코드를 차례대로 확인하면서 TNF, 타입, 페이로드, ID 같은 메시지 속성을 꺼내온다.

```
#include <Wire.h>
#include <PN532_I2C.h>
#include <PN532.h>
#include <NfcAdapter.h>
```

```
PN532_I2C pn532_i2c(Wire);
NfcAdapter nfc = NfcAdapter(pn532_i2c);

void setup() {
   Serial.begin(9600);
   Serial.println("NDEF Reader");
   nfc.begin();
}

void loop() {
   Serial.println("\nScan a NFC tag\n");

   if (nfc.tagPresent()) {
     NfcTag tag = nfc.read();
     Serial.println(tag.getTagType());
     Serial.print("UID: ");
     Serial.println(tag.getUidString());

     if (tag.hasNdefMessage()) { // 모든 태그가 메시지를 가지고 있는 것은
아니다
       NdefMessage message = tag.getNdefMessage();
       Serial.print("\nThis NFC Tag contains an NDEF Message with ");
       Serial.print(message.getRecordCount());
       Serial.print("NDEF Record");
       if (message.getRecordCount() != 1) {
         // 한 개 이상의 레코드가 있다면, 복수형 표현을 추가한다.
         Serial.print("s");
       }

       // 모든 레코드를 확인하면서, 각 레코드의 정보를 출력한다.
       int recordCount = message.getRecordCount();
      for (int i = 0; i < recordCount; i++)
      {
         Serial.print("\nNDEF Record ");
         Serial.println(i+1);
```

```
        NdefRecord record = message.getRecord(i);

        Serial.print(" TNF: ");
        Serial.println(record.getTnf());
        Serial.print(" Type: ");
        Serial.println(record.getType()); // will be "" for TNF_EMPTY

      // TNF와 Type을 바탕으로 어떠한 방식으로 페이로드를 처리할 지
결정한다
      // 페이로드를 처리하기 위한 일반적인 방법은 없으며,
      // byte[]로 분석해야 한다
      int payloadLength = record.getPayloadLength();
      byte payload[payloadLength];
      record.getPayload(payload);

      // 데이터를 문자열로 변환한다
      String payloadAsString = "";
      for (int c = 0; c < payloadLength; c++) {
        payloadAsString += (char)payload[c];
      }
      Serial.print(" Payload (as String): ");
      Serial.println(payloadAsString);

      // id가 포함되어 있지 않다면, 빈 문자열("")을 반환한다
      String uid = record.getId();
      if (uid != "") {
        Serial.print(" ID: ");Serial.println(uid);
      }
    }
  }
 }
 delay(3000);
}
```

이 코드를 기기에 올린 후 IDE 창의 상단 오른쪽 구성에 있는 직렬 포트 모니터 아이콘을 클릭한다. 이전에 사용하던 태그를 사용해 이 프로그램을 실행하면 다음과 같은 결과가 나타난다.

```
NDEF Reader
Found chip PN532
Firmware ver. 1.6

Scan a NFC tag

Mifare Classic
UID: A4 AF 77 5D

This NFC Tag contains an NDEF Message with 2 NDEF Records
NDEF Record 1
  TNF: 2
  Type: text/hue
  Payload (as String): {"lights":{"1":{"name":"Living room","state":
    {"on":true,"bri":223,
    "hue":47293,"sat":0}}

NDEF Record 2
  TNF: 1
  Type: U
  Payload (as String): /sdcard/myMusic/Forever.mp3
```

tagPresent()와 delay()는 블로킹 함수다. 즉 함수가 일을 마칠 때까지 프로그램을 중단한다. tagPresent()는 100ms, delay()는 설정한 값만큼 중단된다. 그러므로 꼭 필요한 경우에만 이러한 함수를 사용해야 한다. 프로그램이 중단된 경우에도 처리해야 할 이벤트가 있는데 기기의 버튼에서 발생한 물리적 입력이나 직렬 통신으로 들어오는 데이터가 이러한 경우다. 이러한 이벤트 처리에 우선순위를 부여해 문제를 해결할 수 있다. 곧 보게 될 애플리케이션 스케치 코드에서 관련된 코드를 참고하자.

6.4.2 아두이노에서 NDEF 작성하기

레코드를 NDEF 메시지로 작성하거나 보내는 방법은 간단하다. WriteTagMultipleRecords는 이를 위한 좋은 방법을 보여준다. 'File → Examples → NDEF → WriteTagMultipleRecords'를 선택한다. 아두이노에 해당 코드를 올린 후 태그를 작성하고 폰에서 읽어보자.

가장 먼저 할 일은 NdefMessage 객체를 생성하는 것이다. 그 후 addTextRecord(), addUriRecord(), addMimeMedia() 등의 헬퍼 함수로 레코드를 추가한다. 메시지를 만든 다음에는 NfcAdapter의 write() 함수를 사용해 메시지를 전송한다. 이 함수는 전송의 성공 여부를 반환한다.

```
#include <Wire.h>
#include <PN532_I2C.h>
#include <PN532.h>
#include <NfcAdapter.h>

PN532_I2C pn532_i2c(Wire);
NfcAdapter nfc = NfcAdapter(pn532_i2c);

void setup() {
    Serial.begin(9600);
    Serial.println("NDEF Writer");
    nfc.begin();
}

void loop() {
    Serial.println("\nPlace a formatted Mifare Classic NFC tag on the
reader.");
    if (nfc.tagPresent()) {
        NdefMessage message = NdefMessage();
        message.addTextRecord("Hello, Arduino!");
```

```
            message.addUriRecord("http://arduino.cc");
            message.addTextRecord("Goodbye, Arduino!");
            boolean success = nfc.write(message);
            if (success) {
                Serial.println("Success. Try reading this tag with your
    phone.");
            } else {
                Serial.println("Write failed");
            }
        }
        delay(3000);
    }
```

지금까지 코드를 잘 따라왔다면 NDEF 라이브러리의 기본적인 읽기와 쓰기 기능을 어느 정도 익혔을 것이다. 이제 이를 사용해 새로운 애플리케이션을 만들 준비가 된 것이다.

6.5 NFC 애플리케이션을 위한 마이크로컨트롤러 : 호텔 카드키

지난 10년 안에 호텔에 숙박한 경험이 있다면 RFID 기반의 방 카드키를 사용해 봤을 것이다. 카드키는 NFC를 쉽게 적용할 수 있는 임베디드 NFC 예제이므로 간단한 NFC 기반 호텔 방 카드키 애플리케이션을 만들어보자. 이번 애플리케이션에서는 아두이노를 위한 NFC 카드 읽기/쓰기 스케치를 작성한다. 읽기 스케치의 경우 추가적인 하드웨어가 필요하다. 원한다면 실제 문에 부착된 전자 잠금장치를 이용할 수도 있다. 실생활에서 사용할 수 없는 예제로 학습한다면 마이크로컨트롤러를 공부하는 데 재미를 찾기 힘들 것이다.

앞으로 작성할 애플리케이션의 초반부는 아두이노용 태그 작성기다. 또한, 접수 데스크 안쪽의 컴퓨터에 연결된 작은 상자도 만들어야 한다. 이 상자는 고객의 이름,