신나는 프로토타이핑 Prototyping Lab

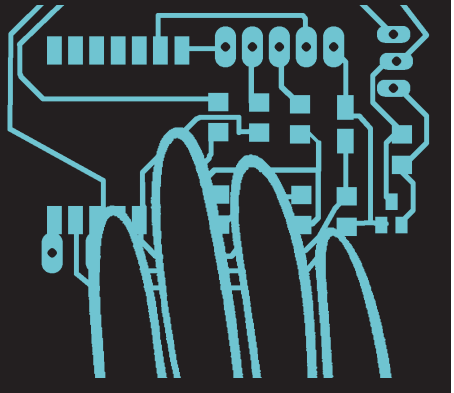
Prototyping Lab

by Shigeru Kobayashi

아두이노, 프로세싱, 액션스크립트3으로 실습하는 신나는 프로토타이핑

초판 1쇄 발행 2012년 3월 16일 **지은이** 고바야시 시게루 **옮긴이** 서효정 **펴낸이** 한기성 **펴낸곳** 인사이트 **편집** 김민희 **본문 디자인** 윤영준 **종이** 세종페이퍼 **표지 출력** 경운 프린테크 **본문 출력·인쇄** 현문인쇄 **제본** 자현제책 **등록번호** 제10-2313호 **등록일자** 2002년 2월 19일 **주소** 서울시 마포구 서교동 469-9 석우빌딩 3층 **전화** 02-322-5143 **팩스** 02-3143-5579 **블로그** http://blog.insightbook.co.kr **이메일** insight@insightbook.co.kr **ISBN** 978-89-6626-010-2 책값은 뒤표지에 있습니다. 잘못 만든 책은 바꾸어 드립니다. 이 책의 정오표는 http://insightbook.springnote.com/pages/8694362에서 확인할 수 있습니다. 이 책의 국립중앙도서관 출판시도서목록(CIP)은 e-CIP 홈페이지 (http://www.nl.go.kr/ecip)에서 이용할 수 있습니다. (CIP 제어번호: CIP2012000955)

신나는 프로토타이핑



고바야시 시게루 지음 | 서효정 옮김

인사이트
insight

리뷰 후기

책을 보기 전에도 아두이노가 피지컬 컴퓨팅에 사용된다는 것은 알았지만 어떻게 응용하는지가 궁금했었습니다. 「작품 소개」에 나오는 작품들을 보면서 사람의 상상력은 참 무궁무진하다는 사실을 깨달았습니다. 그리고 그러한 상상력을 아두이노로 펼칠 수 있게 여러 가지 예제가 구체적으로 설명된 부분이 참으로 좋았습니다. 아두이노는 이탈리아어로 좋은 친구Best Friend라는 뜻이라고 합니다. 이 책은 독자들이 좋은 친구(아두이노)에게 좀 더 다가갈 수 있는 방향을 제시하는 책이라고 생각합니다.

– 강성원, 자작매니아(http://cafe.naver.com/mademania) 매니저

펼치자마자 만나게 되는 작품들만으로도 흥미로운 이 책은, 아두이노를 활용한 각종 프로토타이핑 환경과 친해지기 좋은 실습으로 꽉 채워져 있습니다. 닐 거센펠드의 『팹 FAB(개인용 컴퓨터 PC의 시대를 지나 개인용 제작기 PF의 시대가 온다)』이 특정 시대를 예고하는 책이었다면, 『신나는 프로토타이핑』은 FAB(fabrication laboratory의 줄임말)에 빠지게 만드는 지름신 같은 책이라고 생각합니다. '신나는'이라는 표현이 책 제목에 잘 어울립니다. 앞으로 이 책을 본 독자들이 어떤 작품을 만들어 낼지 기대됩니다.

– 이동욱, http://nephilim.tistory.com

프로토타이핑은 목적한 바를 검증하는 가장 빠른 방법입니다. 이 책은 프로토타이핑 과정에 꼭 필요한 내용을 담고 있습니다. 레시피에서 소개하는 구체적인 사례들은 실제 구현을 통해서 그 이상의 가치를 창출합니다. 저자의 사상인 "만들면서 생각하고, 생각하면서 만든다"라는 말은, 과정 속의 실패 또한 성공의 이유라고 말하는 듯합니다. 이 책은 프로토타이핑을 하는 독자에게 입문서이자 활용서로써 많은 도움을 줄 것입니다.

– 이용우, 로보메카 기술이사

추천의 글

테크놀로지를 이용하는 방법을 가르치기란 쉽지 않습니다. 누군가에게 무엇을 가르칠 때는 인내, 명확함, 유머 감각이 필요합니다. 또 배우고자 하는 사람이 무엇을 배우고 싶어하는가를 이해하기 위해 귀를 귀울여야만 합니다.

고바야시 시게루는 이런 능력 이상을 갖고 있습니다. 제가 그를 존경하는 이유는 그가 우수한 교육자일 뿐만 아니라 매우 우수한 기술자이자 작가이기 때문입니다. 대부분의 사람들은 어떤 개념을 설명할 때 한 언어로 설명하는 것조차 힘들어 합니다. 그러나 고바야시 선생은 다양한 언어로, 그것도 쉽게, 개념을 설명합니다. 또한 복잡한 도구를 사용하기 용이하게 해주는 프로그램이나 프레임워크를 개발하기까지 합니다.

그는 다른 사람이 놓쳐버린 작은 부분을 알아차려, 알기 쉬운 방식으로 침착하게 오해를 바로잡습니다. 저는 그와 함께 일하면서 커뮤니케이션을 더욱 원활하게 하는 방법을 배웠습니다. 이 책이 출판되어 무척 설렙니다. 이 책은 독자가 전자공작, 프로그래밍, 인터랙션 디자인에 대한 능력을 향상시키도록 도와줄 것입니다.

톰 아이고

옮긴이의 글

제가 미디어 아트 작업을 시작했을 때는 웹캠webcam으로 관람객의 위치나 움직임을 인식하는 방법을 많이 사용했습니다. 하지만 작업을 발전시켜 나가는 과정에서, 작품이 지닌 의미를 보다 구체적으로 전달하기 위해 특정한 움직임을 디자인해야 하는 경우도 생겨났습니다. 또 조명 환경에 크게 영향을 받는 카메라로는 인식하기 어려운 다른 데이터를 사용해야 할 때도 있었습니다. 관람객으로부터 다양한 데이터를 얻기 위한 방법을 연구하던 중, sadi에 온 아두이노 개발자 중 한 명인 데이비드 꾸아르띠에예스David Cuartielles를 통해 아두이노를 처음으로 알게 되었습니다.

아두이노를 알기 전에도 다른 보드를 접해봤지만, 프로그래밍을 전공하지 않은 비전공자가 사용하기에는 어려운 점이 너무 많았습니다. 아두이노는 개발 환경을 제공하고, 더 나아가 오픈 커뮤니티를 형성하여 다양한 자료를 공개합니다. 따라서 처음 아두이노를 다루는 사람에게도 쉽게 다가갑니다. 아두이노 보드가 미디어 아트나 피지컬 컴퓨팅을 이용한 DIY 제작에서 표준이 되다시피 한 이유는 이런 특징 때문이라고 생각합니다.

이 책은 아두이노와 프로그래밍을 다루는 입문서이자, 필요할 때마다 해당 주제에 대해 찾아볼 수 있는 참고서입니다. 레시피 형식으로 구성했기 때문에 요리책(쿡북)처럼 사용할 수 있습니다. '우렁 된장찌개'를 만들고 싶을 때 요리책을 펼쳐 해당 레시피를 찾아보는 것처럼, '소리에 반응하는 작품'을 만들고 싶다면 역시 해당하는 레시피를 책에서 바로 찾아, 만들기를 쉽게 시작할 수 있습니다. 두 가지 이상의 레시피를 조합하여 새로운 '나만의 작품'을 만드는 일도 재미있는 시도가 될 듯합니다.

이 책의 저자인 고바야시 시게루 선생님은 게이너Gainer와 퍼널Funnel 같이 널리 알려진 프로토타이핑 툴을 개발했고, 미디어 아트와 디지털 디자인 분야에서 일본뿐만 아니라 국제적으로도 유명한 IAMAS에서 교편을 잡고 있습니다. 또한 Make 행사나 유스트림Ustream을 통한 강좌, 수퍼모노즈쿠리 강좌 등을 통해 많은 사람들에게 만들기의 가능성을 전하는 활동을 활발하게 펼치고 있습니다.

근래에 이러한 가능성을 아주 크게 느낀 계기가 된 사건이 있었습니다. 2011년 3월, 일본에는 사상 최대의 대지진과 원전 폭발사고, 뒤이은 방사능 누출사고가 있었습니다. 주목할 만한 부분은 바로 이 사고 직후 나타난 프로토타이핑에 대한 일본 사람들의 관심과 움직임이었습니다. 일본 정부가 발표하는 방사능 수치를

믿을 수 없게 된 일본 사람들이 직접 가이거 카운터Geiger Counter를 이용하여 방사선 측정 장치를 만들고, 집 앞의 방사능을 측정하였습니다. 그리고 그 정보를 패치베이를 이용해 네트워크에 올렸기 때문에 모든 사람이 데이터를 공유할 수 있었습니다. 심지어 정부에서는 발표하지 않은 수돗물의 방사능 수치까지 공유하는 지역도 있었습니다. 대량으로 제조하고 생산하는 방법으로는 이렇게 넓은 지역에 걸쳐 재빠르게 영향을 끼칠 수 없었을 것입니다. 앞으로 이런 개인 제작을 통해 세상이 얼마나 바뀌게 될지 예측하기 어려울 정도입니다.

이런 만들기의 가능성을 한국에도 전하고 싶었습니다. 이 책을 옮기면서, 독자들이 보다 쉽게 프로토타이핑을 이해하고, 다양한 작품과 예제를 접하면서 실제로 직접 만들고 싶은 물건에 대한 생각을 구체화하기를 기대했습니다. 또한 **'만들면서 생각하고, 생각하면서 만들어 보는'** 기회를 누리기를 바랐습니다. 독자 여러분도 『신나는 프로토타이핑』을 통해 얻은 경험을 살려 자신만의 새로운 만들기에 계속 도전해 보세요.

마지막으로, 이 책을 번역할 기회를 주신 한기성 대표님께 먼저 감사의 인사를 드립니다. 또 이 책이 나오도록 꾸준히 독려해준 김민희 편집자, 처음 시작할 때 많이 격려해준 김승호 편집자, 여러 가지로 많은 도움을 준 플러그하우스의 석근국 님, 리뷰 과정을 통해 많은 조언을 주셨던 강성원 님, 김병목 님, 이동욱 님, 이용우 님, 본문을 디자인해주신 윤영준 님, 표지를 디자인해주신 오필민 님께도 감사의 말을 전하고 싶습니다. 그리고 이 책을 읽고 멋진 작품을 만들 독자들께도 미리 감사하다는 인사를 드립니다.

한국어판 독자에게

먼저 이 책을 구입해 주셔서 정말 고맙습니다. 『신나는 프로토타이핑』은 처음으로 제가 혼자 집필한 책이며, 처음으로 일본어가 아닌 언어로 번역된 책이기도 합니다. 한국은 태어나서 처음으로 방문한 아시아 국가입니다. 그리고 제가 몸담고 있는 IAMAS에도 한국 유학생이 많습니다. 이런 인연을 가진 나라에서 책을 출판하게 되어 매우 기쁩니다.

이 책은 세 부분으로 구성되어 있습니다. 첫 번째는 아두이노를 사용해 어떤 것을 만들 수 있는지를 보여주는 「작품 소개」입니다. 「작품 소개」에서는 단순히 최종 작품을 보여주는 것이 아니라 완성에 이르기까지의 제작 과정을 소개합니다. 그 다음은 「시작하기」입니다. 여기서는 처음으로 전자회로를 배우고, 아두이노를 다루는 방법을 단계별로 설명합니다. 마지막은 「쿡북」입니다. 오라일리의 쿡북 시리즈를 참고해서 요리책과 같이 각 레시피별로 재료를 정리하고, 전자회로와 프로그래밍을 통해 실현하는 과정을 보여줍니다.

이렇게 구성한 데는 이유가 있습니다. '만들기'는 '무엇을 만들까what to make'와 '어떻게 만들까how to make' 두 가지로 구성되어 있다고 생각합니다. 이중 '어떻게 만들까'는 다양한 책이나 학교 수업에서 다룹니다. 하지만 '무엇을 만들까'에는 간단히 전하기에는 힘든 벽이 있다고 생각합니다. 전자회로나 프로그래밍 기술이 없는 독자라면 대부분 '작품 소개'와 그 뒤의 내용 사이에서 큰 수준 차이를 느낄지도 모릅니다. 완성도가 높은 작품일수록 마치 중간 과정 없이 처음부터 완벽한 모습으로 있었던 것처럼 보이기 때문입니다. 그러나 읽어보면 알겠지만, 각 작품의 작가는 자신의 손을 움직여서, 다양한 시행착오를 겪으면서 조금씩 아이디어를 발전시켰습니다. **만들면서 생각하고, 생각하면서 만드는** 과정을 통해 완성에 이른 것입니다.

처음부터 명확하고 완벽한 아이디어가 있다면 직접 만들지 않아도 전문적인 기술을 가진 사람에게 의뢰해서 자신이 원하는 것을 만들 수도 있습니다. 하지만 아이디어라는 것은 대부분 처음에는 애매하고 어설픕니다. 만들어가는 도중에 점점 다듬고 정리하여 완성해 가는 것이라고 생각합니다. 이 **'만들면서 생각하고, 생각하면서 만든다'**라는 생각이 이 책의 핵심입니다. 「시작하기」와 「쿡북」은 이를 위해 필요한 기술을 몸에 익히는 부분입니다. 반드시 실제로 따라해 보기 바랍니다.

마지막으로, 이 책의 번역을 서효정 씨에게 부탁할 수 있어서 매우 행복합니다. 훌륭한 미디어 아티스트이자 일본어뿐만 아니라 일본 문화를 매우 잘 이해하는 분이라고 생각합니다. 분량도 만만치 않은데다 전자회로와 프로그래밍까지 아우르는 분야의 책을 번역하는 일은 보통 힘든 일이 아니었으리라고 짐작합니다. 그러나 서효정 씨 같은 훌륭한 분이 담당해 주었으니 좋은 책으로 완성되리라고 믿어 의심치 않습니다.

이 책을 통해 많은 분들이 **'만들면서 생각하고, 생각하면서 만든다'**라는 프로토타이핑의 생각과 즐거움을 깨닫기를 진심으로 바랍니다.

2011년 7월 22일

고바야시 시게루

시작하면서

이 책은 넓은 의미에서의 '프로토타이핑'인, 새로운 생각하는 방법 또는 사고방식을 소개합니다. 또한 그것을 직접 실현하는 과정을 담은 실험실 같은 책입니다. 지금까지는 무언가 새로운 물건이나 체험을 만들 때, 머릿속에서 콘셉트를 세우고 여유 있게 다듬은 뒤에 형태로 만드는 방법이 주된 방법이었습니다. 그러나 프로토타이핑 중심 방법에서는 이론 단계에서 실제로 만들어 다뤄보고, 잘 되지 않으면 부숴서 또 새로 만들고 다듬어 나갑니다.

이렇게 하면 진정한 의미로 새로운 물건이나 체험을 만들 수 있고, 다른 사람과 공유하고 아이디어를 키워나가며 즐겁게 작업할 수 있습니다. 오픈소스 도구인 아두이노Arduino나 퍼널Funnel을 활용하면 프로토타이핑은 몇만 원으로 시작할 수 있습니다. 조금씩 시도하면서 나만의 실험실을 만들어 갑시다.

이 책의 독자

이 책의 대상 독자는 다음과 같습니다. 물론 여기서 예로 들지 않았어도 이 책을 한 장씩 넘겨보면서 흥미가 생겼다면 한번 천천히 읽어보세요.

- 점점 많이 알려지고 있는 아두이노를 어떻게 활용할 수 있는지 자세히 알고 싶은 분
- 『손에 잡히는 아두이노』 등의 입문서를 읽고 따라해 본 뒤, 좀 더 자세하게 알고 싶은 분
- 게이너Gainer[1]를 사용하거나 『+GAINER』를 읽어서 기본적인 부분은 알고 있지만 독립실행Stand Alone으로 작동시켜 보거나 무선통신을 해보고 싶은 분
- 소프트웨어는 다뤄본 경험이 조금 있고, 전자공학은 거의 해보지 않았지만 하드웨어도 직접 다뤄보고 싶은 분
- 디자인이나 아트 분야에서 제작 경험이 있어서 전자회로나 프로그래밍을 이용한 작품을 만들고 싶은 분
- 피지컬 컴퓨팅이나 프로토타이핑이라는 말을 들어봤지만 실제로 어떤 것인지 자세히 알고 싶은 분
- Make: Tokyo Meeting[2] 등에서 아두이노를 사용한 전시를 보거나 Make: Japan 블로그[3]에서 소개된 작품에 흥미가 있어서 직접 만들어 보고 싶은 분

1 (옮긴이) 게이너는 센서나 액추에이터를 PC에 연결하여 플래시, Max/MSP, 프로세싱과 같은 다양한 프로그래밍 환경에서 이용할 수 있도록 만든 보드입니다. http://gainer.cc

2 (옮긴이) 『Make』 잡지를 통해 과학, 전자공작, 예술 등의 전혀 다른 배경을 가진 사람들이 교류를 할 수 있는 장소를 제공한다는 의도로 기획된 이벤트. 『Make』 잡지에서 다루었던 프로젝트의 시연, 체험 코너, 워크숍, 프레젠테이션, 판매 코너 등으로 구성. 오라일리 재팬 주최. 도쿄 외에는 IAMAS가 있는 오오가키에서 열리고 있습니다.

3 (옮긴이) Make: Korea 블로그는 http://www.make.co.kr/입니다.

· 소프트웨어 혹은 하드웨어 엔지니어로 일하고 있지만 다른 영역의 기술도 익히고 싶거나 다른 영역에서도 사용되는 공통언어를 알고 싶은 분

» 이 책의 구성

이 책은 「작품 소개」 「시작하기」 그리고 「쿡북」까지 3개 부분으로 구성되어 있습니다. 「작품 소개」에서는 아두이노 등을 이용해 제작한 취미, 디자인, 아트 등 다양한 분야의 작품을 소개합니다. 단계별로 만드는 방법을 소개하지는 않지만, 완성작품 외에 제작 과정이나 프로토타입도 소개하므로 아이디어와 소재를 조합해 표현하는 과정을 살펴볼 수 있습니다.

「시작하기」는 이 책에서 다루는 내용을 소개하는 부분으로, 관련된 주제를 소개한 후 아두이노나 퍼널 라이브러리 설치를 실행합니다. 그 후 전자공작 경험이 거의 없는 사람을 위한 기초지식을 설명하고 실제로 아두이노나 퍼널(라이브러리)을 사용해 보는 부분까지 설명합니다.

「쿡북」은 이 책의 중심이 되는 부분입니다. '아두이노를 네트워크에 연결'하거나, '무선으로 통신'하는 등 하고 싶은 주제에 대해 재료나 회로, 프로그램 예제를 레시피로 제시합니다. 예제는 아두이노만 사용하거나, 프로세싱Processing과 함께, 액션스크립트3와 함께 사용하는 세 종류를 준비했습니다. 아두이노만 사용하는 예제는 기본적으로 PC에 연결하지 않고 작동할 수 있습니다. 나머지 두 종류는 PC에 연결해서 PC를 이용해 입력과 출력을 제어할 수 있습니다. 프로세싱은 무료로 이용할 수 있고, 실시간 그래픽 표현이 가능하다는 장점이 있는 간단한 프로그래밍 언어입니다. 액션스크립트3는 웹브라우저에서 작동하는 플래시 플레이어Flash player로 재생할 수 있는 다양한 미디어 콘텐츠를 만들기 위한 프로그래밍 언어입니다.

이 책의 샘플코드 다운로드, 지면 부족으로 미처 게재하지 못한 회로도나 관련 입문 킷 정보는 다음 페이지[4]를 참조하세요.

URL http://prototypinglab.com

4 (옮긴이) 국내 독자를 위해 샘플코드 등을 아래 페이지에 정리했으니 참고하세요.
URL http://insightbook.springnote.com/pages/8694362

주의 사항

이 책의 내용을 이용할 때, 독자의 안전은 스스로 책임집니다. 여기에는 적절한 기자재와 보호 도구를 사용하고 자신의 기술과 경험을 적절히 판단하는 일도 포함됩니다. 전동 공구, 전기 등 프로젝트에서 사용하는 요소는 주의깊게 취급하지 않거나 보호 도구를 사용하지 않으면 위험합니다. 해설에서 사용하는 사진, 일러스트는 순서를 알기 쉽게 하기 위해서, 안전에 필요한 준비나 보호 도구를 생략한 경우가 있습니다. 또한 이 책의 프로젝트는 어린이를 대상으로 만든 것이 아닙니다.

오라일리 재팬과 저자는 이 책을 따라하다 일어난 손해, 사고에 대해 책임을 지지 않습니다. 독자의 활동이 법률, 저작권을 침해하지 않는지 반드시 확인하세요.

인사이트 출판사와 옮긴이는 이 책의 내용을 따라하는 과정에서 생긴 손해, 부상, 비용에 대한 책임을 지지 않습니다. 독자 여러분의 안전에 항상 주의를 기울이기 바랍니다.

목차

10 움직임을 감지하려면

가속도 센서의 급격한 변화를 감지해 움직임을 파악할 수 있습니다.

레시피

레시피 9에서 소개한 것처럼 가속도 센서에서 나온 출력이 기울기에 따른 중력 가속도를 나타내므로 이를 이용해 기울기를 구할 수 있었습니다. 가속도에는 중력 가속도와 같은 정적(靜的) 가속도(물체의 질량에 따라 항상 걸려 있는 가속도)와 물체가 움직일 때 발생하는 동적(動的) 가속도가 있습니다. 여기에서는 센서를 재빠르게 흔들었을 때 발생하는 동적 가속도로 빠른 움직임을 감지하는 방법을 소개합니다.

재료
- 아두이노 보드: 1개
- 브레드보드: 1개
- 점프선: 적당량
- 가속도 센서: 1개 (3축 가속도 센서 모듈, 예: KXM52-1050)
- 저항기: 1개 (330Ω)
- LED: 1개

기울기를 구할 때, 동적 가속도는 불필요한 정보기 때문에 이동평균(running mean 혹은 moving average) 필터로 제거해주면 더욱 매끄러운 출력을 얻을 수 있습니다. 다음 그림은 센서를 천천히 기울였을 때와 센서를 흔들었을 때 각각 이동평균 필터를 걸기 전과 후를 비교한 것입니다.

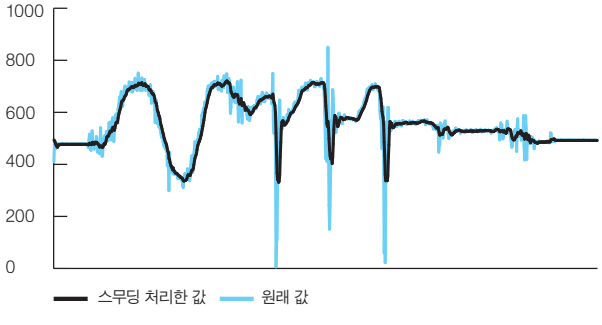


그림 10-1 스무딩 처리한 값과 원래 값 비교

천천히 센서를 기울였을 때는 두 값이 큰 차이를 보이지 않습니다. 미세한 손떨림 같은 움직임에 필터를 걸어서 그래프에서 보이는 것처럼 매끄럽게 만든 정

도일 뿐입니다. 이에 비해 센서를 재빠르게 흔든 경우에는 필터를 걸기 전과 후가 큰 차이를 보입니다. 이동평균 필터는 지금까지의 변화에서 다음 변화를 예측하는 필터이므로 예상 외의 변화가 일어났을 때는 더욱 큰 차이가 납니다. 바로 이 차이를 비교하여 움직임을 검출할 수 있습니다.

아두이노

레시피23에서 소개하는 Mean 필터를 이용해서 스무딩 처리한 값과 원래 값의 차이를 구해 그 값이 역치 이상이면 LED를 켭니다. 오디오의 수위계level meter처럼 작동하기 때문에 LED는 천천히 밝아지거나 어두워집니다. 깜깜한 방에서 손에 센서와 아두이노 보드를 가지고 움직이면서 장시간 노출로 촬영하면 재미있는 사진을 찍을 수 있습니다. 다만 너무 격하게 움직여 USB 케이블이 빠지지 않도록 주의하세요.

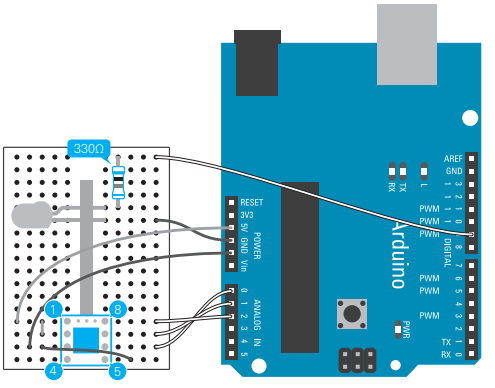


그림 10-2 가속도 센서와 LED를 연결한 배선도. 급격한 움직임을 감지하면 LED가 일정 시간 켜진다.[1]

예제 10-1 아두이노

```
// Mean 필터용 버퍼의 길이
const int bufferLength = 5;

// 센서가 흔들렸다고 판단할 수 있는 역치
const int threshold = 80;

// 가속도 센서의 x축에 연결한 아날로그 핀의 번호
const int sensorPin = 0;

// LED에 연결한 핀의 번호
const int ledPin = 9;

// Mean 필터용 버퍼
int buffer[bufferLength];
```

1 (옮긴이) 사용한 KXM52-1050의 경우 각 핀은 1-Vdd, 2-PSD, 3-GND, 4-Parity, 5-SelfTest, 6-OutX, 7-OutY, 8-OutZ으로 되어 있습니다. 다른 모델을 사용한다면 데이터시트에서 각 핀의 설명을 참조하여 배선합니다.

```
// 버퍼에 데이터를 쓰는 인덱스
int index = 0;

// intensity는 LED의 밝기를 저장할 변수
float intensity = 0;

void setup() {
  // ledPin을 출력으로 설정
  pinMode(ledPin, OUTPUT);

  // Mean 필터용 버퍼를 0~1023의 중간값으로 초기화
  for (int i = 0; i < bufferLength; i++) {
    buffer[i] = 511;
  }
}

void loop() {
  // 센서의 값을 읽어 들인다
  int raw = analogRead(sensorPin);

  // 스무딩 처리
  int smoothed = processSample(raw);

  // 센서의 값과 스무딩 처리한 값의 차이를 구한다
  int diff = abs(raw - smoothed);

  // 차이가 역치보다 크다면
  if (diff > threshold) {
    // LED의 밝기를 최대로 한다
    intensity = 255;
  }
  // 그렇지 않다면
  else {
    // LED의 밝기에 일정한 값을 곱해서 감소시킨다
    intensity = intensity * 0.9;
  }

  // LED의 밝기를 설정
  analogWrite(ledPin, round(intensity));

  // 5ms 동안 대기
  delay(5);
}

// 이동평균 필터(Running Mean)로 스무딩
int processSample(int raw) {
  // 읽어 들인 값을 버퍼에 쓴다
  buffer[index] = raw;
```

```
  // 쓴 다음에 쓴 위치를 새로 고침
  // 버퍼의 끝까지 오면 되풀이한다
  index = (index + 1) % bufferLength;

  // sum은 버퍼 값의 합계를 저장하기 위한 변수
  long sum = 0;

  // 버퍼 값의 합계를 계산
  for (int i = 0; i < bufferLength; i++) {
    sum += buffer[i];
  }

  // 평균을 계산하여 출력 결과로 보낸다
  return (int)(sum / bufferLength);
}
```

아두이노+PC

아두이노+PC도 기본적으로 같습니다. Mean 필터를 걸어준 값과 센서 값의 차이를 구해 그 값이 역치를 넘으면 LED의 밝기를 제어하는 오실레이터를 실행시켜 LED를 오디오의 수위계처럼 켭니다. (배선도는 그림 10-1과 같습니다)

예제 10-2 프로세싱

```
import processing.funnel.*;

// 버퍼의 길이
final int BUFFER_LENGTH = 8;

// 센서가 흔들렸다고 판단할 수 있는 역치
final float THRESHOLD = 0.1;

Arduino arduino;

// 가속도 센서의 x축에 연결한 아날로그 핀
Pin sensorPin;

// LED에 연결한 핀
Pin ledPin;

// 버퍼
float[] buffer = new float[BUFFER_LENGTH];

// 버퍼에 데이터를 기록할 인덱스
int index = 0;

// LED를 제어하기 위한 오실레이터
```

```
Osc osc;

// 그래프를 표시하기 위한 카운터
int count = 0;

void setup() {
  size(400, 200);

  // ledPin을 PWM으로 설정
  Configuration config = Arduino.FIRMATA;
  config.setDigitalPinMode(9, Arduino.PWM);
  arduino = new Arduino(this, config);

  // sensorPin과 ledPin을 초기화
  sensorPin = arduino.analogPin(0);
  ledPin = arduino.digitalPin(9);

  // 오실레이터를 초기화해서 이벤트 리스너를 설정
  // 인수는 Applet, 파형, 주파수, 반복 회수
  osc = new Osc(this, Osc.SAW, 5, 1);
  osc.addEventListener(Osc.UPDATE, "oscUpdated");

  // 배경을 전부 검게 칠한다
  background(0);
}

void draw() {
  // 이번에 그려지는 영역을 검게 칠한다
  stroke(0);
  line(count, 0, count, height - 1);

  // 그래프의 x축을 그린다
  stroke(100);
  line(0, 99, width - 1, 99);
  line(0, 199, width - 1, 199);

  // 센서의 값을 읽어 들인다
  float raw = sensorPin.value;

  // Mean 필터로 스무딩 처리한다
  float smoothed = processSample(raw);

  // 원래 값과의 차이를 구한다
  float diff = abs(raw - smoothed);

  // 원래 값을 위의 그래프에 그린다
  stroke(100);

  point(count, map(raw, 0, 1, 99, 0));
```

```
  // 스무딩한 값을 위의 그래프에 그린다
  stroke(255);
  point(count, map(smoothed, 0, 1, 99, 0));

  // 차이를 아래의 그래프에 그린다
  stroke(255);
  point(count, map(diff, 0, 1, 199, 100));

  // 만일 차이가 역치보다 크다면 LED를 깜박인다
  if (diff > THRESHOLD) {
    osc.reset();
    osc.start();
  }

  // 그래프 표시용 카운터를 새로 고침
  count = (count + 1) % width;
}

// Mean 필터
float processSample(float raw) {
  // 버퍼에 새로운 샘플을 쓰고 인덱스를 새로 고침
  buffer[index] = raw;
  index = (index + 1) % buffer.length;

  // 버퍼 값의 합계를 저장하기 위한 변수
  float sum = 0;

  // 버퍼 값의 합계를 구해 저장
  for (int i = 0; i < buffer.length; i++) {
    sum += buffer[i];
  }

  // 평균을 계산해 필터의 출력 결과로 보낸다
  return (sum / buffer.length);
}

// 오실레이터가 새로 고침되면 LED의 밝기를 새로 고침
void oscUpdated(Osc osc) {
  ledPin.value = osc.value;
}
```

액션스크립트3에서는 Pin 오브젝트가 가진 속성인 preFilterValue를 이용해서 이동평균 필터를 걸기 전과 건 후의 차이를 구하고 있습니다.

예제 10-3 액션스크립트3

```
package {
  import flash.display.Sprite;
  import flash.events.Event;
  import flash.utils.getTimer;
  import funnel.*;
  import funnel.ui.LED;

  [swf(backgroundColor = "cccccc")]

  public class DetectMovement extends Sprite {
    // 움직임이 있다고 판단할 수 있는 역치
    private const THRESHOLD:Number = 0.1;

    private var arduino:Arduino;

    // 센서의 입력과 스무딩 처리한 결과의 차이를 표시하는 스코프
    private var scopeForInputSignal:SignalScope;
    private var scopeForDiff:SignalScope;

    // 가속도 센서의 x축에 연결한 핀
    private var sensorPin:Pin;

    // LED
    private var led:LED;

    // 지난번 LED가 켜진 시각
    private var lastTrigger:int = 0;

    public function DetectMovement() {
      // LED에 연결한 핀의 모드를 PWM으로 설정하고 LED를 초기화
      var config:Configuration = Arduino.FIRMATA;
      config.setDigitalPinMode(9, PWM);
      arduino = new Arduino(config);
      led = new LED(arduino.digitalPin(9));

      // sensorPin에 이동평균 필터와 이벤트 리스너를 설정
      sensorPin = arduino.analogPin(0);
      sensorPin.addFilter(new Convolution(
                          Convolution.MOVING_AVERAGE));
      sensorPin.addEventListener(PinEvent.CHANGE, onChange);

      // 센서의 입력을 표시하는 스코프를 생성
      scopeForInputSignal = new SignalScope(10, 10, 200,
                                            "X axis");
      addChild(scopeForInputSignal);

      // 센서의 입력과 스무딩 처리한 결과의 차이를 표시하는 스코프를 생성
```

```
      scopeForDiff = new SignalScope(10, 70, 200,
                                     "Diff (raw - smoothed)",
                                     -1, 1);
      addChild(scopeForDiff);

      addEventListener(Event.ENTER_FRAME, onEnterFrame);
    }

    private function onEnterFrame(e:Event):void {
      // 센서의 입력을 표시
      scopeForInputSignal.update(sensorPin);

      // 센서의 입력과 스무딩 처리한 값의 차이를 표시
      var diff:Number = sensorPin.preFilterValue
                      - sensorPin.value;
      scopeForDiff.update(diff);
    }

    // 센서에 연결한 핀에 변화가 있다면 호출한다
    private function onChange(e:PinEvent):void {
      // 센서의 입력과 스무딩 처리한 값의 차이를 구한다
      var diff:Number = sensorPin.preFilterValue
                      - sensorPin.value;

      // 현재 시각을 읽어 들인다
      var now:int = getTimer();

      // 차이의 절대값이 역치보다 크고, 지난번 LED가 켜진 이후로 일정 시간이
         경과했다면
      if ((Math.abs(diff) > THRESHOLD)
          && ((now - lastTrigger) > 500)) {
        // LED를 SAW(톱날 모양)파로 깜박거림
        led.blink(300, 1, Osc.SAW);

        // 현재 시각을 lastTrigger에 저장
        lastTrigger = now;
      }
    }
  }
}
```

15 빛을 제어하려면

RGB LED를 이용해 다양한 빛을 제어할 수 있습니다

레시피

LED는 출력 부품 중에서도 가장 자주 사용되는 부품으로, '아두이노 따라하기'에서도 단색 LED를 제어하는 방법을 소개했습니다. 여기에서는 포탄형 또는 하이파워형 RGB LED를 제어하는 방법을 소개합니다.

포탄형 LED를 켜는 경우에는 저렴한 1/4W형 카본저항기로 충분합니다. 하이파워형 LED를 켜는 경우에는 1/2W형 이상의 금속피막 저항기, 산화금속피막 저항기, 시멘트 저항기 등을 사용합니다. 저항기 용량은 흐르는 전류의 배 이상을 기준으로 저항기를 선택합니다. 예를 들면, 순전압 3.3V의 LED에 20mA가 흐른다면, 전력은 3.3V × 0.02A = 0.06W가 되므로 그 배인 0.12W라도 1/4(0.25W)의 저항기로 충분히 대응할 수 있습니다. 소전력용 저항기에 많은 전류가 흐르면 저항기가 타버리는 경우도 있습니다.

더욱이 하이파워형을 사용하는 경우에는 방열판이 미리 부착되어 있는 것을 사용하거나, 방열판을 별도로 준비해서 사용하세요. 하이파워형은 발열이 꽤 심하기 때문에 방열하지 않은 상태로 전류를 흘리면 LED를 손상시킵니다.

재료 **[포탄형 RGB LED일 때]**

- 아두이노 보드: 1개
- 브레드보드: 1개
- 점프선: 적당량
- 포탄형 RGB LED: 1개
- 저항기: 3개 (330Ω 1개와 150Ω 2개, 1/4W형)
- 가변저항기: 3개 (10kΩ, B커브)

[하이파워형 LED일 때]

- 아두이노 보드: 1개
- 브레드보드: 1개
- 점프선: 적당량
- 하이파워형 RGB LED: 1개
- 트랜지스터 어레이: 1개 (도시바 TD62083APG 등)
- 저항기: 3개 (33Ω 1개와 15Ω 2개, 1/2W형 이상)
- 가변저항기: 3개 (10kΩ, B커브)

색	순전압	순전류 (최대)	전류제한저항
R	1.8	10 (25)	330
G	3.5	10 (25)	150
B	3.5	10 (25)	150

표 15-1 포탄형 RGB LED의 한 예 (OSTA-5131A)의 사양

색	순전압	순전류 (최대)	전류제한저항
R	2.0	100 (150)	33
G	3.5	100 (150)	15
B	3.5	100 (150)	15

표 15-2 하이파워형 RGB LED의 한 예 (EP204K-150G1R1B1-CA)의 사양

아두이노

RGB LED에는 애노드 측이 커먼 애노드인 것과 캐소드 측이 커먼 캐소드인 것 두 종류가 있습니다. 예를 들면 이번에 재료로 사용하고 있는 포탄형 RGB LED가 커먼 캐소드[1]지만 보통은 커먼 애노드가 일반적입니다.[2] 커먼 캐소드는 공통 단자를 GND 측에 연결해서 핀에서 소스 드라이브하기 때문에 analogWrite를 255로 했을 때 가장 밝아집니다. 이에 비해 커먼 애노드는 공통 단자를 전원의 플러스 측(5V)에 연결해서 핀에 싱크 드라이브하기 때문에 analogWrite를 0으로 했을 때 가장 밝아집니다. 이 예제는 커먼 캐소드를 사용한 예제지만, 드라이브 모드를 정의하고 있는 행을 변경하면 커먼 애노드를 사용할 때도 같은 결과를 얻을 수 있습니다.

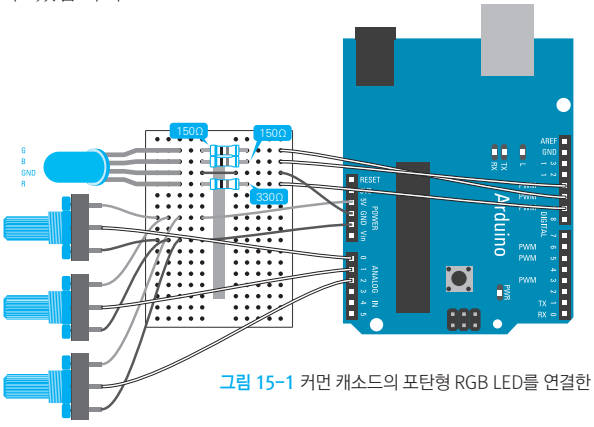


그림 15-1 커먼 캐소드의 포탄형 RGB LED를 연결한 배선도

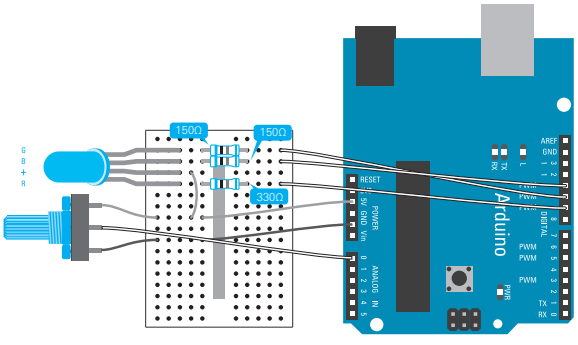


그림 15-2 커먼 애노드의 포탄형 RGB LED를 연결한 배선도(G와 B의 가변저항기는 생략). 커먼 애노드의 RGB LED를 사용하는 경우에는, RGB LED의 공통 핀을 +5V에 연결해서 스케치로 드라이브 모드를 정의하고 있는 부분을 변경한다.

1 영어의 스펠링은 cathode지만 머리글자로 생략 했을 경우, C만 쓰게 되면 콘덴서와 혼동하기 쉽기 때문에, 독일어의 스펠링인 kathode를 이용하는 경우도 있습니다.

2 이번에 사용하는 포탄형 LED의 다리는 다음과 같습니다.

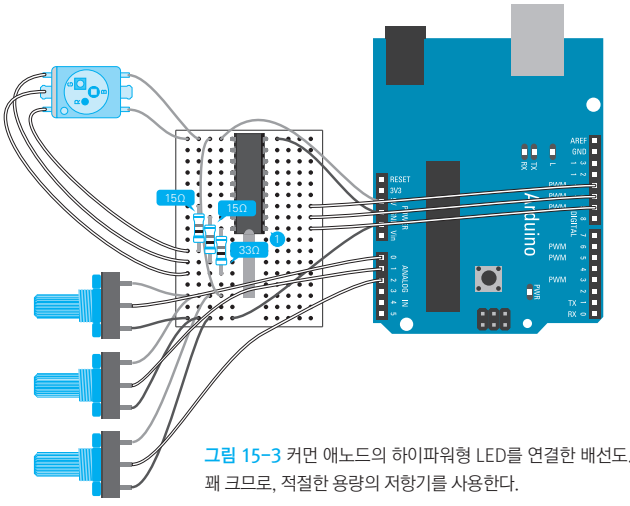


그림 15-3 커먼 애노드의 하이파워형 LED를 연결한 배선도. 하이파워 형은 발열량이 꽤 크므로, 적절한 용량의 저항기를 사용한다.

예제 15-1 아두이노

```
// LED의 드라이브 모드를 정하는 숫자
const int commonKathode = 0;
const int commonAnode = 1;

// R, G, B 각각의 LED에 연결한 핀의 번호
const int rLEDPin = 9;
const int gLEDPin = 10;
const int bLEDPin = 11;

// R, G, B 각각을 제어하는 가변저항기에 연결한 아날로그 핀의 번호
const int rPotPin = 0;
const int gPotPin = 1;
const int bPotPin = 2;

// LED의 드라이브 모드: 커먼 애노드의 경우에는 여기를 commonAnode로 변경한다
int driveMode = commonKathode;

void setup() {
  // R, G, B 각각의 LED에 연결한 핀의 모드를 출력으로 설정
  pinMode(rLEDPin, OUTPUT);
  pinMode(gLEDPin, OUTPUT);
  pinMode(bLEDPin, OUTPUT);
}

void loop() {
  // R, G, B 각각을 컨트롤하는 가변저항기의 값을 읽어 들인다
  int rPotValue = analogRead(rPotPin);
  int gPotValue = analogRead(gPotPin);
  int bPotValue = analogRead(bPotPin);
```

```
  // 드라이브 모드가 커먼 캐소드라면
  // 0~1023까지의 입력을 0~255로 변환
  if (driveMode == commonKathode) {
    analogWrite(rLEDPin, map(rPotValue, 0, 1023, 0, 255));
    analogWrite(gLEDPin, map(gPotValue, 0, 1023, 0, 255));
    analogWrite(bLEDPin, map(bPotValue, 0, 1023, 0, 255));
  }

  // 드라이브 모드가 커먼 애노드라면
  // 0~1023까지의 입력을 255~0으로 변환
  else if (driveMode == commonAnode) {
    analogWrite(rLEDPin, map(rPotValue, 0, 1023, 255, 0));
    analogWrite(gLEDPin, map(gPotValue, 0, 1023, 255, 0));
    analogWrite(bLEDPin, map(bPotValue, 0, 1023, 255, 0));
  }
}
```

아두이노 두에밀라노베 보드 위의 전압 조정기regulator의 출력이 800mA로 되어 있습니다. 이 때문에 이번에 사용하는 LED 정도는 이 전압 조정기를 통해 5V 핀으로 전력을 공급할 수 있습니다. 그러나 이보다 큰 전력이 필요한 경우라면 별도의 전원을 준비해서 공급해야 합니다.

아두이노+PC

아두이노+PC도 기본적으로는 아두이노만 사용할 때와 같습니다. R, G, B 각각에 연결한 핀의 모드를 PWM으로 설정하고 가변저항기의 값을 읽어 들여 출력 핀의 값으로 설정합니다. 배선도는 그림 15-1, 15-2 및 15-3과 같습니다.

예제 15-2 프로세싱

```
import processing.funnel.*;

// LED의 드라이브 모드를 정하는 숫자
final int COMMON_KATHODE = 0;
final int COMMON_ANODE = 1;

Arduino arduino;

// R, G, B 각각의 LED에 연결한 핀의 번호
Pin rLEDPin;
Pin gLEDPin;
Pin bLEDPin;
```

```
// R, G, B 각각을 컨트롤하는 가변저항기에 연결한 아날로그 핀
Pin rPotPin;
Pin gPotPin;
Pin bPotPin;

// LED의 드라이브 모드: 커먼 애노드의 경우에는 여기를 COMMON_ANODE로 변경한다
int driveMode = COMMON_KATHODE;

void setup() {
  size(200, 200);

  // 텍스트 표시용 폰트를 생성해서 설정
  PFont font = createFont("CourierNewPSMT", 18);
  textFont(font);

  // LED에 연결한 핀의 모드를 PWM으로 설정
  Configuration config = Arduino.FIRMATA;
  config.setDigitalPinMode(9, Arduino.PWM);
  config.setDigitalPinMode(10, Arduino.PWM);
  config.setDigitalPinMode(11, Arduino.PWM);
  arduino = new Arduino(this, config);

  // LED와 가변저항기에 연결한 핀을 나타내는 변수를 초기화
  rLEDPin = arduino.digitalPin(9);
  gLEDPin = arduino.digitalPin(10);
  bLEDPin = arduino.digitalPin(11);
  rPotPin = arduino.analogPin(0);
  gPotPin = arduino.analogPin(1);
  bPotPin = arduino.analogPin(2);
}

void draw() {
  // 커먼 캐소드라면 가변저항기의 값을 그대로 출력으로 설정
  if (driveMode == COMMON_KATHODE) {
    rLEDPin.value = rPotPin.value;
    gLEDPin.value = gPotPin.value;
    bLEDPin.value = bPotPin.value;
  }

  // 커먼 애노드라면 가변저항기의 값을 반전해서 출력으로 설정
  else {
    rLEDPin.value = 1 - rPotPin.value;
    gLEDPin.value = 1 - gPotPin.value;
    bLEDPin.value = 1 - bPotPin.value;
  }

  // 배경을 검게 칠하고 R, G, B 각각의 값을 표시
  background(0);
  text("R: " + rPotPin.value, 10, 20);
```

```
  text("G: " + gPotPin.value, 10, 40);
  text("B: " + bPotPin.value, 10, 60);
}
```

액션스크립트3은 funnel.ui 패키지에 준비되어 있는 RGBLED 클래스를 이용해서 제어합니다. 이렇게 하면 더욱 단순한 코드로 제어할 수 있습니다.

예제 15-3 액션스크립트3

```
package {
  import flash.display.Sprite;
  import flash.events.Event;
  import flash.text.TextField;
  import funnel.*;
  import funnel.ui.*;

  public class ControlRGBLED extends Sprite {
    // LED의 드라이브 모드: 커먼 애노드의 경우에는 여기를 변경한다
    private static const DRIVE_MODE:int
    = RGBLED.COMMON_KATHODE;

    // 아두이노
    private var arduino:Arduino;

    // RGBLED
    private var rgbLED:RGBLED;

    // R, G, B 각각을 컨트롤하는 가변저항기에 연결한 아날로그 핀
    private var rPotPin:Pin;
    private var gPotPin:Pin;
    private var bPotPin:Pin;

    // R, G, B 값을 표시하는 텍스트필드
    private var textField:TextField;

    public function ControlRGBLED() {
      // R, G, B 각 LED에 연결한 핀의 모드를 출력으로 설정
      var config:Configuration = Arduino.FIRMATA;
      config.setDigitalPinMode(9, PWM);
      config.setDigitalPinMode(10, PWM);
      config.setDigitalPinMode(11, PWM);
      arduino = new Arduino(config);

      // R, G, B를 컨트롤하는 가변저항기에 연결한 핀을 나타내는 변수를 초기화
      rPotPin = arduino.analogPin(0);
```

```
      gPotPin = arduino.analogPin(1);
      bPotPin = arduino.analogPin(2);

      // RGBLED의 인스턴스를 생성
      var rLEDPin:Pin = arduino.digitalPin(9);
      var gLEDPin:Pin = arduino.digitalPin(10);
      var bLEDPin:Pin = arduino.digitalPin(11);
      rgbLED = new RGBLED(rLEDPin, gLEDPin, bLEDPin,
                          DRIVE_MODE);

      // R, G, B의 각 가변저항기의 값을 표시하는 텍스트필드를 추가
      textField = new TextField();
      addChild(textField);

      // 프레임마다 발생하는 이벤트에 대해서 이벤트 리스너를 설정
      addEventListener(Event.ENTER_FRAME, onEnterFrame);
    }

    // 프레임마다 아래를 실행
    private function onEnterFrame(e:Event):void {
      // 가변저항기의 값으로 RGBLED의 색을 새로 고침
      rgbLED.setColor(rPotPin.value, gPotPin.value,
                      bPotPin.value);

      // 가변저항기의 값을 텍스트필드에 표시
      textField.text = "R:" + rPotPin.value.toFixed(2);
      textField.appendText(", G:" + gPotPin.value.toFixed(2));
      textField.appendText(", B:" + bPotPin.value.toFixed(2));
    }
  }
}
```

추가 설명

이 레시피에서 소개한 것처럼, RGB LED를 제어하기 위해서는 PWM을 이용할 수 있는 핀이 3개 필요합니다. 아두이노 두에밀라노베의 경우에는 PWM을 이용할 수 있는 핀이 6개 있기 때문에 보드 하나로 2개까지 제어할 수 있습니다. 이 이상의 RGB LED를 제어하기 위해서는 쿄리쓰 전자의 TLC5940 쉴드와 같이 여러 개의 LED를 제어하기 위해 만든 쉴드를 이용합니다.

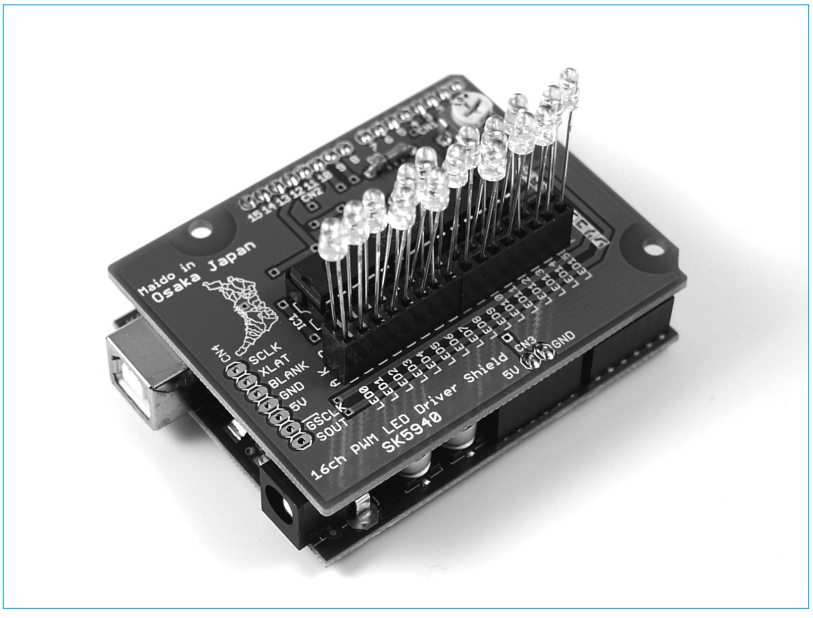


그림 15-4 코리쓰 전자의 TLC5940 쉴드. 16개의 LED 밝기를 4096 단계로 제어할 수 있다. 아두이노용 라이브러리도 있다 (URL http://code.google.com/p/tlc5940arduino)