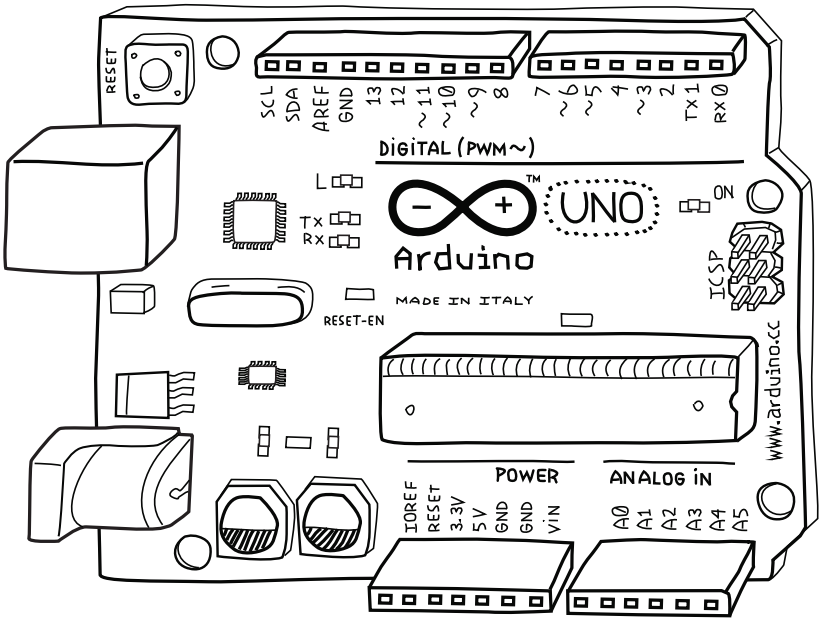
Make: Getting Started with Arduino

Since 2009 UPDATED! Bestseller

3rd Edition



The Open Source Electronics Prototyping Platform

Massimo Banzi co-founder of Arduino

& Michael Shiloh

THIRD EDITION

Getting Started with Arduino

Massimo Banzi and Michael Shiloh

Getting Started with Arduino

by Massimo Banzi and Michael Shiloh

Copyright © 2015 Massimo Banzi and Michael Shiloh. All rights reserved.

Printed in the United States of America.

Published by Maker Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Maker Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://www.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Brian Jepson
Production Editor: Nicole Shelby
Copyeditor: Kim Cofer
Proofreader: Sharon Wilkey
Indexer: WordCo Indexing Services
Interior Designer: David Futato
Cover Designer: Brian Jepson
Illustrator: Judy Aime' Castro

December 2014: Third Edition

Revision History for the Third Edition

2014-12-09: First Release
2014-12-19: Second Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781449363338* for release details.

Make:, Maker Shed, and Maker Faire are registered trademarks of Maker Media, Inc. The Maker Media logo is a trademark of Maker Media, Inc. *Getting Started with Arduino* and related trade dress are trademarks of Maker Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Maker Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

978-1-449-36333-8

[LSI]

Contents

Preface

The third edition of *Getting Started with Arduino* adds two new chapters: Chapter 8 is an ambitious project, which illustrates a more complex circuit and program. This chapter also talks about project design, testing, and construction, and makes use of schematic diagrams, which were (and still are) described in Appendix D.

The second chapter, Chapter 6, introduces the Arduino Leonardo. The Leonardo is a different sort of Arduino, because the USB controller is implemented in software, and not in a separate chip as had been the case prior to the Leonardo. This allows the USB behaviour of the board to be modified.

Apart from these new chapters, other updates have taken place:

The Third Edition is written for version 1.0.5 of the IDE. In anticipation of the imminent release of version 1.5, differences between 1.0.5 and 1.5 have been noted.

Numerous suggestions from students and readers have been incorporated.

In keeping with the spirit of the original text, British spelling is used throughout.

—Michael

Preface to the Second Edition

A few years ago I was given a very interesting challenge: teach designers the bare minimum in electronics so that they could build interactive prototypes of the objects they were designing.

I started following a subconscious instinct to teach electronics the same way I was taught in school. Later on I realised that it simply wasn't working as well as I would like, and started to remember sitting in a class, bored like hell, listening to all that

theory being thrown at me without any practical application for it.

In reality, when I was in school I already knew electronics in a very empirical way: very little theory, but a lot of hands-on experience.

I started thinking about the process by which I really learned electronics:

- I took apart any electronic device I could put my hands on.
- I slowly learned what all those components were.
- I began to tinker with them, changing some of the connections inside of them and seeing what happened to the device: usually something between an explosion and a puff of smoke.
- I started building some kits sold by electronics magazines.
- I combined devices I had hacked, and repurposed kits and other circuits that I found in magazines to make them do new things.

As a little kid, I was always fascinated by discovering how things work; therefore, I used to take them apart. This passion grew as I targeted any unused object in the house and then took it apart into small bits. Eventually, people brought all sorts of devices for me to dissect. My biggest projects at the time were a dishwasher and an early computer that came from an insurance office, which had a huge printer, electronics cards, magnetic card readers, and many other parts that proved very interesting and challenging to completely take apart.

After quite a lot of this dissecting, I knew what electronic components were and roughly what they did. On top of that, my house was full of old electronics magazines that my father must have bought at the beginning of the 1970s. I spent hours reading the articles and looking at the circuit diagrams without understanding very much.

This process of reading the articles over and over, with the benefit of knowledge acquired while taking apart circuits, created a slow, virtuous circle.

A great breakthrough came one Christmas, when my dad gave me a kit that allowed teenagers to learn about electronics. Every component was housed in a plastic cube that would magnetically snap together with other cubes, establishing a connection; the electronic symbol was written on top. Little did I know that the toy was also a landmark of German design, because Dieter Rams designed it back in the 1960s.

With this new tool, I could quickly put together circuits and try them out to see what happened. The prototyping cycle was getting shorter and shorter.

After that, I built radios, amplifiers, circuits that would produce horrible noises and nice sounds, rain sensors, and tiny robots.

I've spent a long time looking for an English word that would sum up that way of working without a specific plan, starting with one idea and ending up with a completely unexpected result. Finally, *tinkering* came along. I recognised how this word has been used in many other fields to describe a way of operating and to portray people who set out on a path of exploration. For example, the generation of French directors who gave birth to the Nouvelle Vague were called the *tinkerers*. The best definition of tinkering that I've ever found comes from an exhibition held at the Exploratorium in San Francisco:

> Tinkering is what happens when you try something you don't quite know how to do, guided by whim, imagination, and curiosity. When you tinker, there are no instructions—but there are also no failures, no right or wrong ways of doing things. It's about figuring out how things work and reworking them.
>
> Contraptions, machines, wildly mismatched objects working in harmony—this is the stuff of tinkering.
>
> Tinkering is, at its most basic, a process that marries play and inquiry.

From my early experiments I knew how much experience you would need in order to be able to create a circuit that would do what you wanted, starting from the basic components.

Another breakthrough came in the summer of 1982, when I went to London with my parents and spent many hours visiting the Science Museum. They had just opened a new wing dedicated to computers, and by following a series of guided experiments, I learned the basics of binary math and programming.

There I realised that in many applications, engineers were no longer building circuits from basic components, but were instead implementing a lot of the intelligence in their products using microprocessors. Software was replacing many hours of electronic design, and would allow a shorter tinkering cycle.

When I came back, I started to save money, because I wanted to buy a computer and learn how to program.

My first and most important project after that was using my brand-new ZX81 computer to control a welding machine. I know it doesn't sound like a very exciting project, but there was a need for it and it was a great challenge for me, because I had just learned how to program. At this point, it became clear that writing lines of code would take less time than modifying complex circuits.

Twenty-odd years later, I'd like to think that this experience allows me to teach people who don't even remember taking any math class and to infuse them with the same enthusiasm and ability to tinker that I had in my youth and have kept ever since.

—Massimo

Acknowledgments for Massimo Banzi

This book is dedicated to Ombretta.

Acknowledgments for Michael Shiloh

This book is dedicated to my brother and my parents.

First of all I'd like to thank Massimo for inviting me to work on the Third Edition of this book, and for inviting me to join Arduino

in general. It's been a real privilege and joy to participate in this project.

Brian Jepson for guidance, oversight, encouragement, and support. Frank Teng for keeping me on track. Kim Cofer and Nicole Shelby for doing a wonderful job of copyediting and production editing, respectively.

My daughter Yasmine for thinking so highly of me, for her never-ending support and encouragement of my pursuing my interests, and for still thinking that I'm kinda cool in spite of being her dad. I could not have done this without her support.

Last but not least, my partner Judy Aime' Castro for the endless hours she spent turning my scribbles into fine illustrations, for discussing various aspects of the book, and for her endless patience with me. I could not have done this without her support either.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from Make: books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Getting Started With Arduino, Third Edition*, by Massimo Banzi and Michael Shiloh (Maker Media). Copyright 2015 Massimo Banzi and Michael Shiloh, 978-1-4493-6333-8."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at *bookpermissions@makermedia.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like Maker Media, O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

Make:
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

Make: unites, inspires, informs, and entertains a growing community of resourceful people who undertake amazing projects in their backyards, basements, and garages. Make: celebrates your right to tweak, hack, and bend any technology to your will. The Make: audience continues to be a growing culture and community that believes in bettering ourselves, our environment, our educational system—our entire world. This is much more than an audience, it's a worldwide movement that Make is leading—we call it the Maker Movement.

For more information about Make:, visit us online:

Make: magazine: *http://makezine.com/magazine*

Maker Faire: *http://makerfaire.com*

Makezine.com: *http://makezine.com*

Maker Shed: *http://makershed.com*

We have a kit with the components needed to try most of the examples (through the end of Chapter 7) available from the Maker Shed.

We also have a web page for this book, where we list errata, examples, corrections to the code, and any additional information. You can access this page at *http://bit.ly/start_arduino_3e*.

For more information about Arduino, including discussion forums and further documentation, see *http://www.arduino.cc*.

To comment or ask technical questions about this book, send email to: *bookquestions@oreilly.com*.

1/Introduction

Arduino is an open source *physical computing* platform for creating interactive objects that stand alone or collaborate with software on your computer. Arduino was designed for artists, designers, and others who want to incorporate physical computing into their designs without having to first become electrical engineers.

The Arduino hardware and software is open source. The open source philosophy fosters a community that shares its knowledge generously. This is great for beginners as help is often available geographically nearby and always online, at many different skill levels, and on a bewildering array of topics. Example projects are presented not just as pictures of the finished project, but include instructions for making your own or as a starting point for incorporation into your derivative or related projects.

The Arduino software, known as the Integrated Development Environment (IDE), is free. You can download it from *www.arduino.cc*. The Arduino IDE is based on the Processing language, which was developed to help artists create computer art without having to first become software engineers. The Arduino IDE can run on Windows, Macintosh, and Linux.

The Arduino board is inexpensive (about $30) and quite tolerant of common novice mistakes. If you do somehow manage to damage the main component on the Arduino Uno, it can be replaced for as little as $4.

The Arduino project was developed in an educational environment and is a very popular educational tool. The same open source philosophy that created the community which generously shares information, answers, and projects also shares teaching methods, curricula, and other information. Arduino has a special mailing list to facilitate discussion among anyone interested in teaching with or about Arduino.

Because the Arduino hardware and software are open source, you can download the Arduino hardware design and build your own, or use it as a starting point for your own project, based on (or incorporating) Arduino within its design, or simply to understand how Arduino works. You can do the same things with the software.

This book is designed to help beginners with no prior experience get started with Arduino.

Intended Audience

This book was written for the "original" Arduino users: designers and artists. Therefore, it tries to explain things in a way that might drive some engineers crazy. Actually, one of them called the introductory chapters of the first draft "fluff". That's precisely the point. Let's face it: most engineers aren't able to explain what they do to another engineer, let alone a regular human being. Let's now delve deep into the fluff.

This book is not meant to be a textbook for teaching electronics or programming, but you will learn something about electronics and programming while reading this book.

> After Arduino started to become popular, I realised how experimenters, hobbyists, and hackers of all sorts were starting to use it to create beautiful and crazy objects. I realised that you're all artists and designers in your own right, so this book is for you as well.
>
> —Massimo

Arduino builds upon the thesis work Hernando Barragan did on the Wiring platform while studying under Casey Reas and me (Massimo) at Interaction Design Institute Ivrea (IDII).

What Is Interaction Design?

Arduino was born to teach Interaction Design, a design discipline that puts prototyping at the centre of its methodology. There are many definitions of Interaction Design, but the one that we prefer is this:

Interaction Design is the design of any interactive experience.

In today's world, Interaction Design is concerned with the creation of meaningful experiences between us (humans) and objects. It is a good way to explore the creation of beautiful—and maybe even controversial—experiences between us and technology. Interaction Design encourages design through an iterative process based on prototypes of ever-increasing fidelity. This approach—also part of some types of conventional design—can be extended to include prototyping with technology; in particular, prototyping with electronics.

The specific field of Interaction Design involved with Arduino is physical computing (or Physical Interaction Design).

What Is Physical Computing?

Physical computing uses electronics to prototype new objects for designers and artists. It involves the design of interactive objects that can communicate with humans by using sensors and actuators controlled by a behaviour implemented as software running inside a microcontroller (a small computer on a single chip).

In the past, using electronics meant having to deal with engineers all the time, and building circuits one small component at a time; these issues kept creative people from playing around with the medium directly. Most of the tools were meant for engineers and required extensive knowledge.

In recent years, microcontrollers have become cheaper and easier to use. At the same time, computers have become faster and more powerful, allowing the creation of better (and easier) development tools.

The progress that we have made with Arduino is to bring these tools one step closer to the novice, allowing people to start

building stuff after only two or three days of a workshop. With Arduino, a designer or artist can get to know the basics of electronics and sensors very quickly and can start building prototypes with very little investment.

2/The Arduino Way

The Arduino philosophy is based on making designs rather than talking about them. It is a constant search for faster and more powerful ways to build better prototypes. We have explored many prototyping techniques and developed ways of thinking with our hands.

Classic engineering relies on a strict process for getting from A to B; the Arduino Way delights in the possibility of getting lost on the way and finding C instead.

This is the tinkering process that we are so fond of—playing with the medium in an open-ended way and finding the unexpected. In this search for ways to build better prototypes, we also selected a number of software packages that enable the process of constant manipulation of the software and hardware medium.

The next few sections present some philosophies, events, and pioneers that have inspired the Arduino Way.

Prototyping

Prototyping is at the heart of the Arduino Way: we make things and build objects that interact with other objects, people, and networks. We strive to find a simpler and faster way to prototype in the cheapest possible way.

A lot of beginners approaching electronics for the first time think that they have to learn how to build everything from scratch. This is a waste of energy: what you want is to be able to confirm that something's working very quickly so that you can motivate yourself to take the next step or maybe even motivate somebody else to give you a lot of cash to do it.

This is why we developed *opportunistic prototyping*: why spend time and energy building from scratch, a process that requires time and in-depth technical knowledge, when we can take

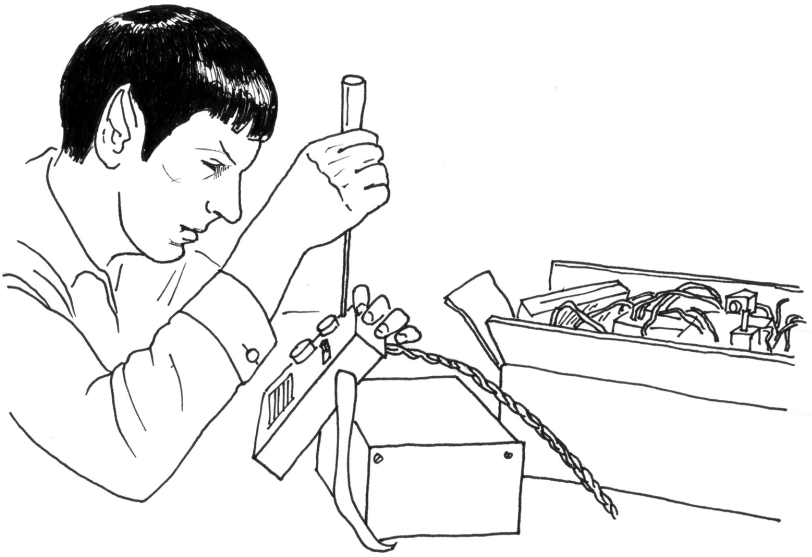
ready-made devices and hack them in order to exploit the hard work done by large companies and good engineers?

Our hero is James Dyson, who made 5127 prototypes of his vacuum cleaner before he was satisfied that he'd gotten it right.

Tinkering

We believe that it is essential to play with technology, exploring different possibilities directly on hardware and software—sometimes without a very defined goal.

Reusing existing technology is one of the best ways of tinkering. Getting cheap toys or old discarded equipment and hacking them to make them do something new is one of the best ways to get to great results.

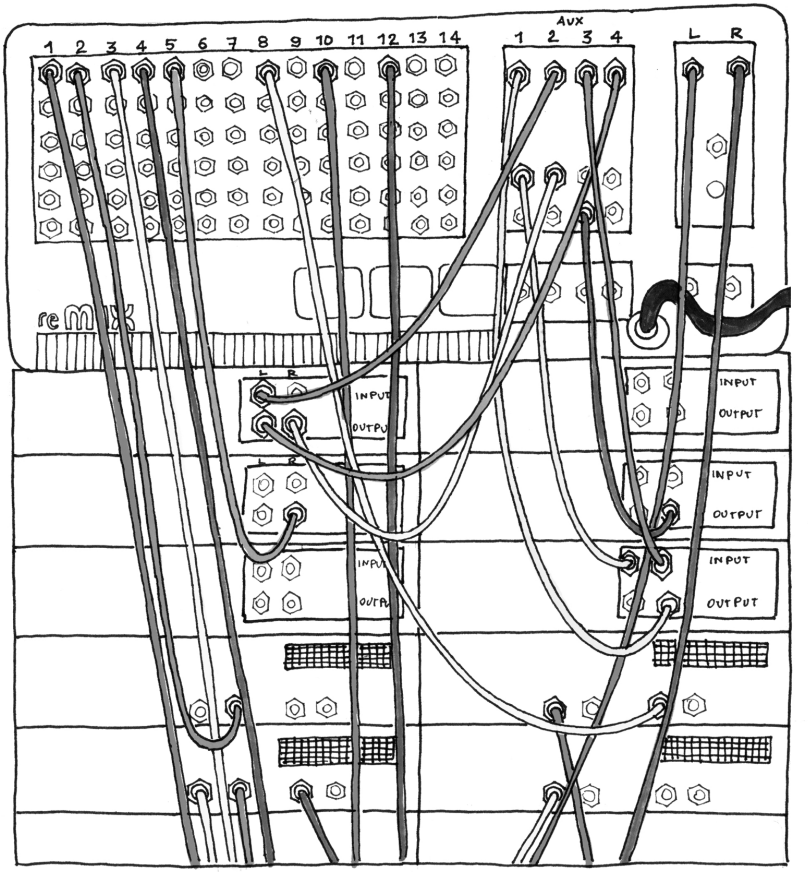


Patching

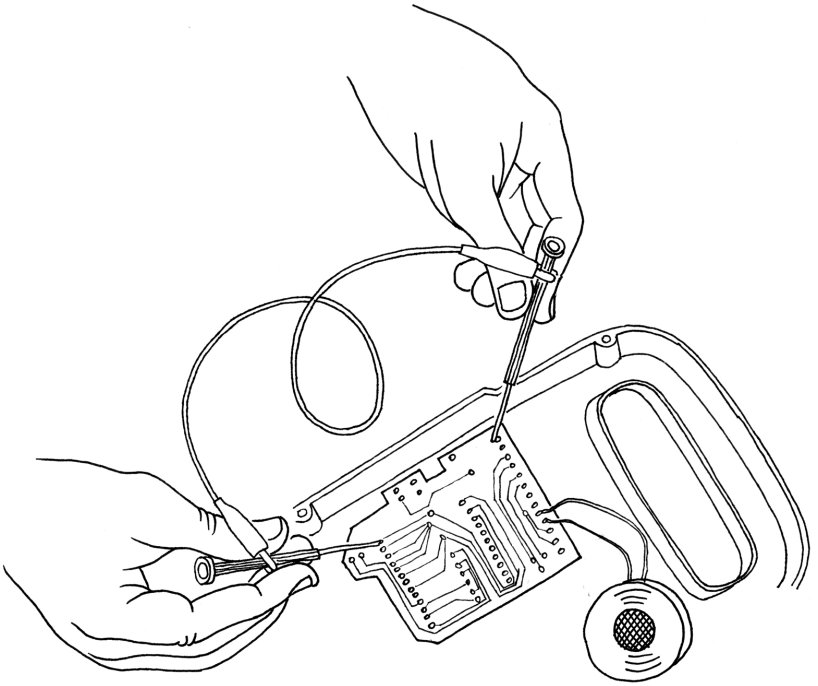
> I have always been fascinated by modularity and the ability to build complex systems by connecting together simple devices. This process is very well represented by Robert Moog and his analogue synthesizers. Musicians constructed sounds, trying endless combinations by patching together different modules with cables. This approach made the synthesizer look like an old telephone switch, but combined with the numerous knobs, that was the perfect platform for tinkering with sound and innovating music. Moog described it as a process between "witnessing and discovering". I'm sure most musicians at first didn't know what all those hundreds of knobs did, but they tried and tried, refining their own style with no interruptions in the flow.
>
> —Massimo

Reducing the number of interruptions to the flow is very important for creativity—the more seamless the process, the more tinkering happens.

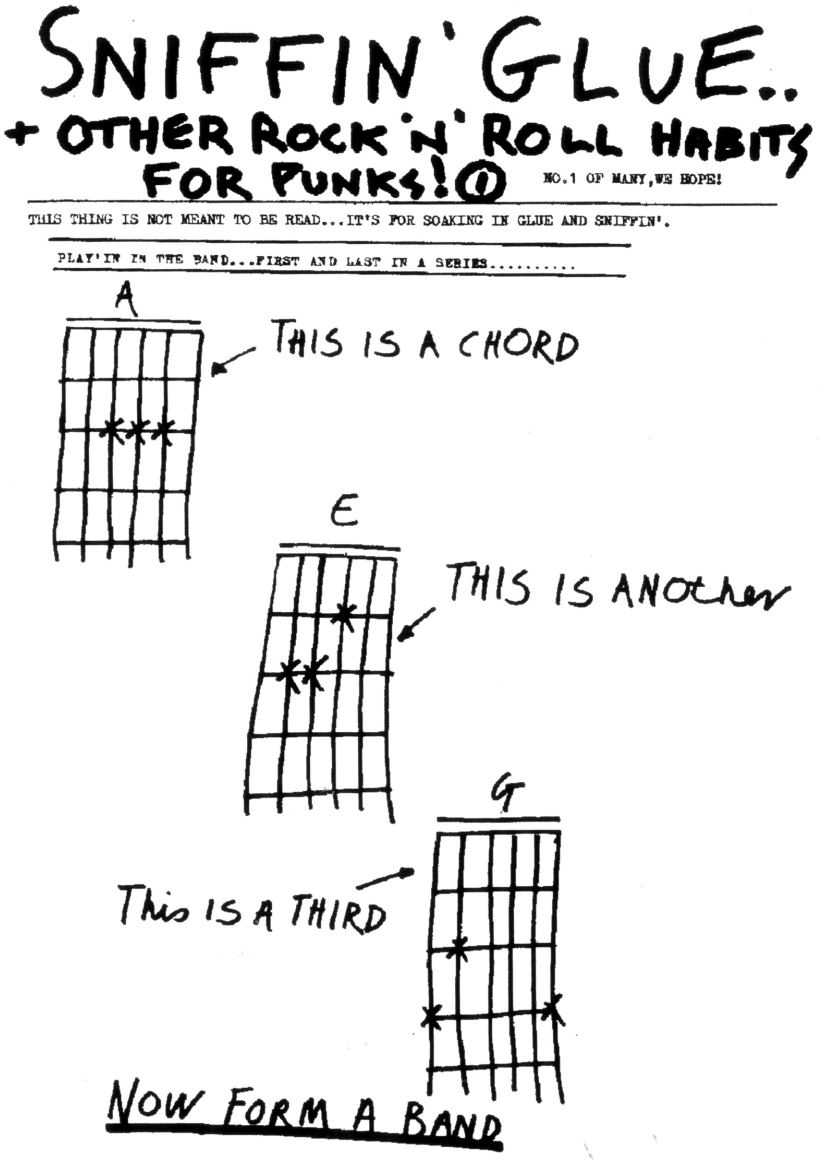
This technique has been translated into the world of software by visual programming environments like Max, Pure Data, or VVVV. These tools can be visualised as *boxes* for the different functionalities that they provide, letting the user build *patches* by connecting these boxes together. These environments let the user experiment with programming without the constant interruption typical of the usual cycle: "type program, compile, damn—there is an error, fix error, compile, run". If you are more visually minded, we recommend that you try them out.


1 2 3 4 5 6 7 8 9 10 11 12 13 14
Aux
1 2 3 4
L R
remix
INPUT
OUTPUT
INPUT
OUTPUT
INPUT
OUTPUT
INPUT
OUTPUT

Circuit Bending

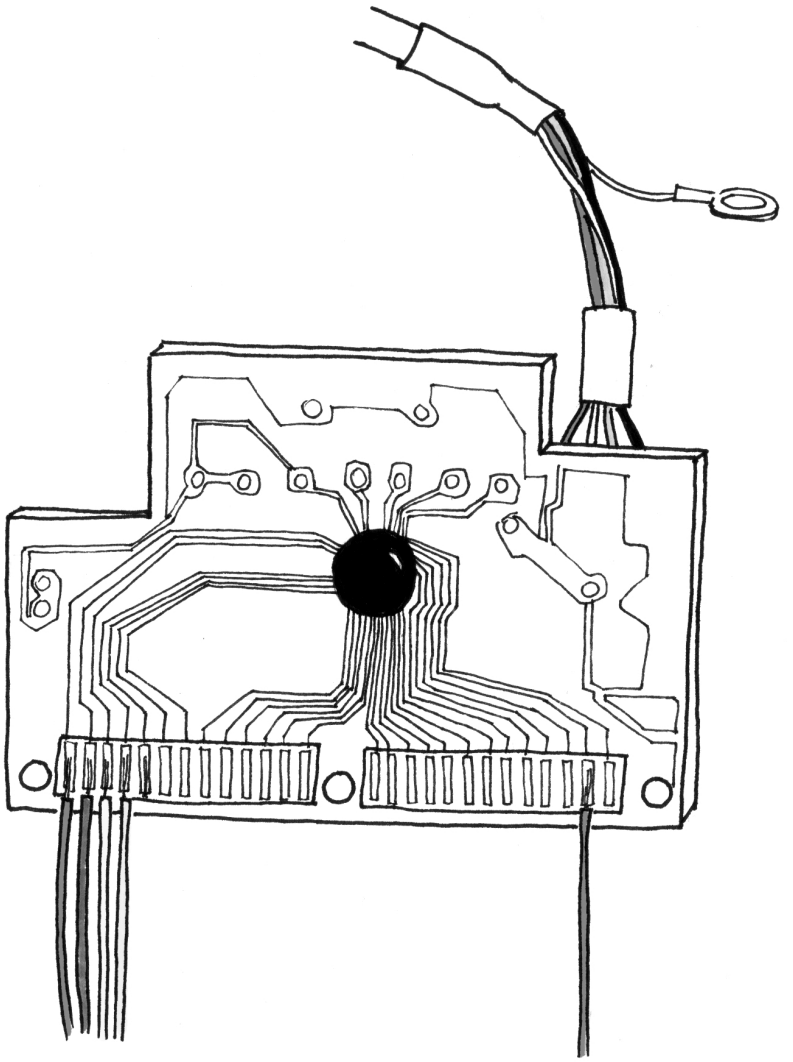


Circuit bending is one of the most interesting forms of tinkering. It's the creative short-circuiting of low-voltage, battery-powered electronic audio devices such as guitar effect pedals, children's toys, and small synthesizers to create new musical instruments and sound generators. The heart of this process is the "art of chance". It began in 1966 when Reed Ghazala, by chance, shorted-out a toy amplifier against a metal object in his desk drawer, resulting in a stream of unusual sounds. Circuit benders excel in their ability to create the wildest devices by tinkering away with technology without necessarily understanding what they are doing on the theoretical side.



It's a bit like the *Sniffin' Glue* fanzine shown here: during the punk era, knowing three chords on a guitar was enough to start a band. Don't let the experts in one field tell you that you'll never be one of them. Ignore them and surprise them.

Keyboard Hacks



Computer keyboards are still the main way to interact with a computer after more than 60 years. Alex Pentland, academic head of the MIT Media Laboratory, once remarked: “Excuse the

expression, but men's urinals are smarter than computers. Computers are isolated from what's around them."[1]

As tinkerers, we can implement new ways to interact with software by replacing the keys with devices that are able to sense the environment. Taking apart a computer keyboard reveals a very simple (and cheap) device. The heart of it is a small board. It's normally a smelly green or brown circuit with two sets of contacts going to two plastic layers that hold the connections between the different keys. If you remove the circuit and use a wire to bridge two contacts, you'll see a letter appear on the computer screen. If you go out and buy a motion-sensing detector and connect this to your keyboard, you'll see a key being pressed every time somebody walks in front of the computer. Map this to your favourite software, and you have made your computer as smart as a urinal. Learning about keyboard hacking is a key building block of prototyping and physical computing.

We Love Junk!

People throw away a lot of technology these days: old printers, computers, weird office machines, technical equipment, and even a lot of military stuff. There has always been a big market for this surplus technology, especially among young and/or poorer hackers and those who are just starting out. This market became evident in Ivrea, where we developed Arduino. The city used to be the headquarters of the Olivetti company. They had been making computers since the 1960s; in the mid 1990s, they threw everything away in junkyards in the area. These are full of computer parts, electronic components, and weird devices of all kinds. We spent countless hours there, buying all sorts of contraptions for very little money and hacking into our prototypes. When you can buy a thousand loudspeakers for very little money, you're bound to come up with some idea in the end. Accumulate junk and go through it before starting to build something from scratch.

1 Quoted in Sara Reese Hedberg's "MIT Media Lab's quest for perceptive computers," *Intelligent Systems and Their Applications*, IEEE, Jul/Aug 1998.