TECHNOLOGY IN ACTION™

Practical Fashion Tech

Wearable Technologies for Costuming, Cosplay, and Everyday

—

Joan Horvath
Lyn Hoge
Rich Cameron

Apress®

Practical Fashion Tech

Wearable Technologies for Costuming, Cosplay, and Everyday

Joan Horvath
Lyn Hoge
Rich Cameron

Apress®

Practical Fashion Tech

Joan Horvath
Nonscriptum LLC,
Pasadena, California, USA

Lyn Hoge
Los Angeles, California, USA

Rich Cameron
Nonscriptum LLC,
Pasadena, California, USA

ISBN-13 (pbk): 978-1-4842-1663-7
ISBN-13 (electronic): 978-1-4842-1662-0
DOI 10.1007/978-1-4842-1662-0

Library of Congress Control Number: 2016954029

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com/9781484216637. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/. Readers can also access source code at SpringerLink in the Supplementary Material section for each chapter.

Printed on acid-free paper

This book is dedicated to Lyn Hoge's family, biological and extended, for the laughter, adventure, support, and love. They have been there through the good and the bad and have always generously shared their intelligence, joy, and creative ideas.

Contents at a Glance

Contents

About the Authors

Joan Horvath and Rich Cameron are the cofounders of Nonscriptum LLC based in Pasadena, California. Nonscriptum consults for educational and scientific users in the areas of 3D printing and maker technologies. Joan and Rich are particularly interested in finding ways to use maker tech to make scientific research cheaper and more accessible to the public.

This book is their latest collaboration, following their earlier works *Mastering 3D Printing* (Apress, 2015), *The New Shop Class: Getting Started with 3D Printing, Arduino, and Wearable Tech* (Apress, 2015), *3D Printing with MatterControl* (Apress, 2015), and *3D Printed Science Projects* (Apress, 2016). They also teach online classes in 3D printing and maker tech for LERN Network's U Got Class continuing education program. Links for all of the above are on their website, www.nonscriptum.com.

In addition to her work with Rich, Joan also has an appointment as Core Adjunct faculty for National University's College of Letters and Sciences. She has taught at the university level in a variety of institutions, both in Southern California and online. Before she and Rich started Nonscriptum, she held a variety of entrepreneurial positions, including VP of Business Development at a Kickstarter-funded 3D-printer company. Joan started her career with 16 years at the NASA/Caltech Jet Propulsion Laboratory, where she worked in programs including the technology transfer office, the Magellan spacecraft to Venus, and the TOPEX/Poseidon oceanography spacecraft. She holds an undergraduate degree from MIT in Aeronautics and Astronautics and a master's degree in Engineering from UCLA.

Rich (known online as "Whosawhatsis") is an experienced open source developer who has been a key member of the RepRap 3D-printer development community for many years. His designs include the original spring/lever extruder mechanism used on many 3D printers, the RepRap Wallace, and the Deezmaker Bukito portable 3D printer. By building and modifying several of the early open source 3D printers to wrestle unprecedented performance out of them, he has become an expert at maximizing the print quality of filament-based printers. When he's not busy making every aspect of his own 3D printers better, from slicing software to firmware and hardware, he likes to share that knowledge and experience online so that he can help make everyone else's printers better too.

Lyn Hoge has been a dance teacher, costumer, and choreographer for over 40 years. In that time, she has designed and created costumes for musicals, plays and various types of dance performances. These include everything from simple period costume plays like *Our Town* to elaborate and quirky versions of *The Rocky Horror Picture Show* and *Bat Boy the Musical*. Lyn has also created unique and functional designs for everything from the T-Rex and Woolly Mammoth in *The Skin of Our Teeth* to stilt walkers at the Edinburgh Fringe Festival. In the past couple of years, she has been delving into the world of wearable tech and is writing about her experiences as a teacher and a student. Lyn has a BA in dance and has studied at UCLA, UCI, and at many private studios.

Acknowledgments

This book draws heavily on the open source hardware and software communities. First, we want to acknowledge the contributions of the Arduino community worldwide, particularly the many useful tutorials and background information at www.arduino.cc, and the community behind the Fritzing software (www.fritzing.org), which we use for many illustrations. The consumer 3D printing ecosystem would not exist in its current form without the open source 3D printing hardware and software community, which we as always are grateful for as the basis of much that we have built upon in our work. We have endeavored to attribute open-source material accurately wherever it appears and apologize for any inadvertent omissions.

The maker community as a whole has also been very supportive. The picture of Joan and Rich in the “About the Authors” section was taken at the 2015 San Mateo Makerfaire by Ethan Etnyre; we appreciate all the inspiration we have gotten by looking at projects made by everyone at maker events.

The Apress production team, past and present, made this process seamless for the most part, and was there with virtual needle and thread for the occasions where it was not. We dealt most directly with Mark Powers, Michelle Lowman, Corbin Collins, Natalie Pao, Jessica Vakili and Welmoed Spahr, but we also appreciate the many we did not see.

We thank the staff, teachers and students of Windward School in Los Angeles, particularly to those in Lyn’s 2015-2016 theater costuming class where many of these concepts were tried out in early form, as well as Lyn’s department chair, Jordon Fox. The owners, past and present, of Make Believe costuming in Santa Monica, California, were very helpful in discussions of what makes a good costume. Bill Doran, cofounder of Punished Props, was also generous with his time in giving us ideas of what to include in Chapter 12. Others have given us permission to use their images or ideas, and we credit those gifts where they occur.

Finally, we are grateful to our friends and families for putting up with the disruption of a maker book in progress and supplying pizza interventions when needed. The book has been a wonderful creative ride for the three of us, and we appreciate everyone on each of our individual roads to this point.

Introduction

This book is a collaboration between two technologists (Joan and Rich) and a veteran teacher, costumer, and choreographer (Lyn). The three of us take turns narrating different chapters and sections. Fashion tech can require skills in design, pattern-making, sewing, electronics, programming, and 3D printing. Besides the tech skills, making a good costume or accessory also requires knowledge of the intangibles of what makes a good costume. We know that people come to fashion tech and wearable electronics from a variety of directions, and that any given reader may know a lot about one part already. We have structured the book so that you can easily skip a chapter or two if you are very experienced in that particular art already.

There are many books in this space that present sets of projects. We have those too, but we wanted to focus on showing why things are done a certain way so that you can figure out how the techniques might apply in other situations. The technology is changing fast, and there are many new components to play with all the time; the key thing is to know the general assumptions behind how they are designed, and where you are most likely to find information on how to use the latest thing. There are many good resources out there on how to sew, how to 3D print, and how to use an Arduino. We felt that the missing part was bringing them together in one readable volume.

We wrote this book with several audiences in mind. First, if you are already someone interested in creating great costumes for theater, or you like to go to cosplay conventions, you will be able to use the material in this book to make your creations interactive, illuminated, or wherever you choose to go with the technologies. If, on the other hand, you are passingly familiar with Arduino electronics but have no idea how to sew or assemble a garment, you can fill in your gaps and learn how to design the overall project.

If you are a high school or college teacher who needs to create a "Fashion Tech," "Costume Tech," or "Wearable Electronics" course, Appendix A has some suggestions that you could use to get started. The material in this book would also work well as the basis for a summer camp session that would mix traditional sewing and crafting with some electronics and coding. One caveat is that the electronic components are delicate and pricey compared to traditional craft materials—they are not toys. Most manufacturers suggest a minimum age of 13 or so to use their products, with adult supervision, and we suggest the same.

One of the challenges in fashion tech is that you need a lot of stuff. First, you need a sewing machine, or you need to stick to projects that start from an existing garment or that are small enough to hand sew.) You will need to purchase the electronic components required, and you will need access to 3D-printing services for those projects. We have tried to design the projects so that you can try things out with as little hardware as possible for any given project.

The big temptation in doing a first project is to do something big and complicated. That is a bad idea, because without experience it can be very hard to debug projects that mix sewing, electronic circuits, and software. We devote almost a whole chapter (Chapter 10) to deconstructing our first collective project, which was far too ambitious, it turns out. To reduce the temptation, we provide first projects that are fun and open-ended so that you can add more if you would like, or stop when you feel you have a product you are happy with.

To cover all this ground, we divide the book in four parts. The first part, "The Big Picture," sets the stage for the rest. Chapter 1 gives our view of what fashion tech comprises and talks about how the three of us work together as a model for you to build your own team. Chapter 2 then gives a broad introduction to what makes a good theatrical costume, assuming that is how many will apply this material.

Moving on to Part II, "The Basics," we introduce the key skills needed for wearable tech. Chapter 3 introduces hand and machine sewing basics, with many references to other resources. Chapter 4 focuses on the art of creating and using a sewing pattern. In Chapter 5 we switch over to the tech side and give an introduction to electronic components. In Chapter 6 we see how to program these devices. Finally, in Chapter 7 we bring it all together in a comprehensive but manageable project to create a hostess apron with a built-in timer that flashes a red light on the apron when the timer is counting down, and a green one when time is up. These chapters are enough to create "blinky" projects—garments that light up with some minimal ability to control the lights.

Part III, "Beyond the Basics," explores more sophisticated topics. Chapter 8 reviews the different types of sensors that are available to make your project react to its environment and introduces some other hardware, such as motors, that is beyond the scope of this book to review in detail but that we think you should know about generally.

Chapter 9 summarizes the 3D-printing process and gives pointers on where to learn about it in detail. Chapter 10 talks about our experiences trying to create an overly complex project (a dress with a mind of its own) without adequate planning.

If you are thinking of going straight to Chapter 11's bigger projects (we know, we would have thought about it too), resist the urge and read Chapter 10 first. Speaking of Chapter 11, there you will find the dress from this book's cover, which uses electroluminescent (EL) ribbon to light up the boundaries between fabric blocks. This is an intermediate sewing project and requires no circuit design or coding. The other project in that chapter takes an off-the-shelf hat and adds circuitry to it so that it lights up red if you shake your head no and green if you nod your head yes. The hat project requires minimal sewing. Thus you can pick a substantial project based on where you feel most secure.

Finally, in Part IV, "Where to Go from Here," Chapter 12 looks at other technologies we did not capture elsewhere in the book but that are frequently used in amateur costuming, like laser cutting, foam armor creation, and vacuum forming. Chapter 13 winds up the main part of the book with a look at some high-end current projects and some speculations on where the field may go in the future.

We have also included two appendices. Appendix A has details about how you might think about laying out one project-focused class of varying length to teach all the pieces of fashion tech. Appendix B captures all the links in the book in one ready reference.

There are several Arduino sketches in this book. They are available for download. Instructions are on the copyright page of this book.

We hope you enjoy trying out the aspects of fashion tech that are new to you, and we hope to see many projects in the future. If you create something cool based on this book, you can tweet it to Joan on her `@JoanHorvath` Twitter account, or contact us at `www.nonscriptum.com` to let us know. Now, start reading and make something awesome!

PART I

The Big Picture

These first two chapters provide some background about what makes a good costume (or other garment) design. Chapter 1 introduces fashion tech and talks a bit about how the rest of the book is arranged. Chapter 2 focuses on what makes a good costume and suggests things to think about in garment design in general.

CHAPTER 1

Fashion Tech

Fashion tech is an interdisciplinary field that merges traditional fashion and textiles with modern electronics, software, and other technologies. In this book, we consider fashion tech to mean interactive garments or accessories that incorporate electronic components, or that were created using digital fabrication technologies like 3D printing. Technologies like these have only recently become available at the consumer level because of advances in the production of electronics that have lowered the cost of computers, sensors, and light-up components that can be embedded into everyday objects. This chapter introduces fashion tech and talks about how you can use this book to get started as a practitioner in this new field.

A Brief History of Fashion Tech

Creating clothing to protect ourselves from the weather has been an inspiration for technology development since antiquity. Tools have gone from bone awls for punching holes in leather, to spindles for creating yarn, to the looms that could produce vast amounts of fabric at industrial scale. Figure 1-1 shows an 1875 stereoscopic photograph of the Amoskeag (New Hampshire) Gingham Mill weaving room. (There is an animation of this stereo image at `http://stereo.nypl.org/view/14480`).

Electronic supplementary material The online version of this chapter (doi:10.1007/978-1-4842-1662-0_1) contains supplementary material, which is available to authorized users.

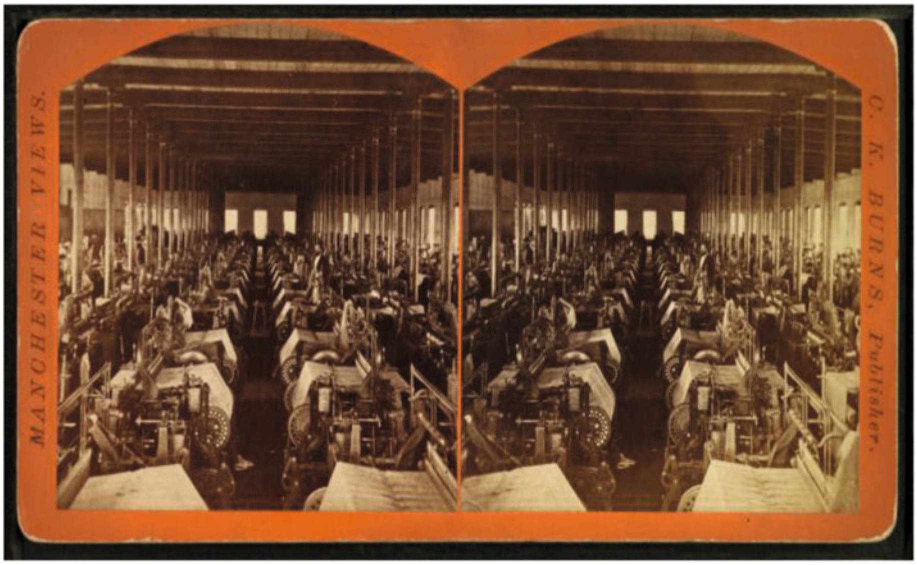


Figure 1-1. *Weaving, circa 1875. From the New York Public Library (`http://digitalcollections.nypl.org/items/510d47e1-7c18-a3d9-e040-e00a18064a99`)*

The desire to create elaborately patterned fabrics led to the development of the Jacquard loom in 1801. A weaver created cards with holes in them that controlled (through an ingenious system of lightly tensioned springs and wires) the patterns the loom was creating. This allowed automatic generation of very elaborate patterns such as brocades.

Fancy fabrics for drapes and wedding gowns is not the end of the story, though. English mathematician and inventor Charles Babbage saw the Jacquard loom punch cards and wondered if a similar system could be used to create a more general computing machine for mathematical problems, which he called the Analytical Engine.

Babbage first described the machine in 1837; the full machine was never actually created (`https://en.wikipedia.org/wiki/Analytical_Engine`). Nevertheless, Babbage is widely seen as the inventor of the general-purpose computer, and the descendants of its punched cards were in use well into the 1970s. And so, the computer has its birth in Victorian fashion tech!

■ **Tip** If you are interested in the history of textile technologies, check out the amazing historical archive with many illustrations and original documents at `www.cs.arizona.edu/patterns/weaving/index.html`. More specifically, if you would like to download a Victorian-era book on how to create the cards for a Jacquard machine, E. A. Posselt's *The Jacquard Machine Analyzed and Explained* (Posselt, 1893) is now in the public domain and available from the Hathi Trust Digital Library at `http://hdl.handle.net/2027/gri.ark:/13960/t26b0d33d`.

The introduction of the integrated circuit in the late 1950s and its rapid evolution have now resulted in affordable computers that are comparable in size to coins. The late Gordon Moore predicted in 1965 that the performance of computer chips would roughly double every two years based on technology improvements; Moore's Law, as it came to be called, has been remarkably accurate (so far). This means that early-1980s supercomputers are outclassed by 2016 single-board computers that are a few inches across and cost $5.

Other digital electronics have kept pace, and tiny processors and sensors developed for smartphones, cameras, and other devices now make it possible to unobtrusively include as part of a hat or apron processing power that would have had been ministered to by dedicated staff in the 1980s. Coming full circle from the Jacquard loom, it is now possible to have circuits woven into clothing. (We talk about Google's Project Jacquard in Chapter 13.)

Another side effect of access to cheap, easy-to-program electronics has been the rise of robotics-based consumer products, like low-cost consumer 3D printers and computerized home sewing machines. 3D printers in turn make it easy to prototype physical objects quickly and cheaply, and are leading to even more innovation. Feature-laden home sewing machines can enable complex projects that would have been too much for the hobbyist in the past.

The bottom line of all this is that you have access to a fantastic array of technologies to make cool projects. In this book we focus on using digital electronics and related technologies (such as 3D printing) in fashion applications like creating costumes and other interactive wearable pieces.

Costuming

Now that it is possible to make a more elaborate and professional-appearing costume at home, it is not surprising that *cosplay*—dressing up as and role-playing a favorite fictional character—has become something of a subculture, initially in Japan but rapidly spreading to the United States and elsewhere. Science fiction and other kinds of conventions often have cosplay fans attend as their favorite character. Anime and video game characters are favorite subjects. Given that, there are many opportunities where an otherworldly effect is desired, and the technologies in this book might be just the thing to take a cosplay costume to the next level.

Historical costuming has always been always popular. Renaissance Faires (`www.renfaire.com`) have popularized historical re-enactment of activities of the late Elizabethan era and dressing up as a person from that time. Visible electronics might be out of character, but there could be a dragon on your arm with glowing eyes or a head that turns toward the light.

You may be reading this book for ideas on enhanced theatrical costumes, either for school productions (as Lyn talks about later in this chapter) or for professional use. At the school level, costumes likely need to be low-budget, assembled quickly, and easy to get in and out of for a ten-year-old with stagefright. A few strategic special effects enabled by electronics can make a costume memorable. For example, Lyn and her students added programmable glowing eyes to fish for *The Little Mermaid*.

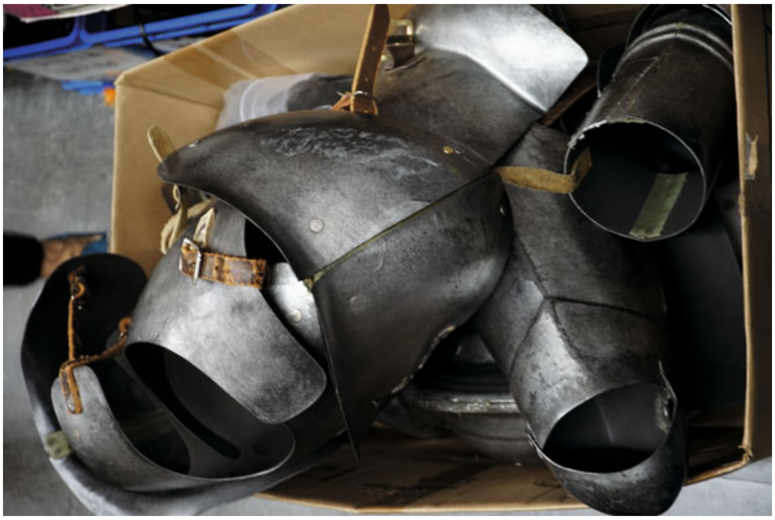


Figure 1-2. *Plastic armor in its box*

Finally, there are just plain dress-up costumes for parties, holidays, and so on. Figure 1-2 shows custom-made, vacuum-formed (Chapter 13) plastic armor as it looks in a box, and in Figure 1-3 it briefly transforms Sir Rich to his true knight-errant identity.

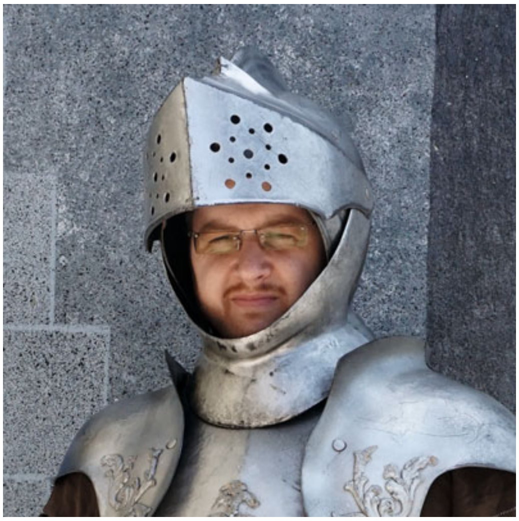


Figure 1-3. *The armor fitted on our knight-author*

Some costume elements such as masks (Figure 1-4) or a period dress can take a person to another place and time. If you have a fantasy costume anyway, why not mix it up and make it interactive? (Armor and masks courtesy of Make Believe in Santa Monica, California; we talk more about them in Chapter 2.)

Figure 1-4. *Masks*

Maybe you are not looking to make a costume, per se, but are looking to do something functional—a wall hanging that lights up as a night light when it gets dark, maybe, or strategic LEDs inside a bag so you can find your keys at night.

Our point in all this is that the first step in applying the technologies in this book is to think about what you are trying to do and how you want it to look. Too often people start out wanting to use a technology for its own sake, but that often does not end well. Just because it is possible to make an interactive garment or art piece, why should you?

If you are reading this book you may be planning to build costumes like those we just described. Or you may have been asked to develop a class for high school or college students. If you are a teacher or parent, fashion tech projects can be a good way to convince students who otherwise might have been scared off electronics or programming to give them a try, motivated by the final product.

Our Design Philosophy

We (Joan, Rich, and Lyn) came into this field from very different directions. Rich is a Millennial who grew up designing electronic projects (including one of the forerunners of today's consumer 3D printers and a small, elegant 3D printer still being sold by a company Rich and Joan used to work for—you can see one in Chapter 9). He likes to make things for their own sake, the usual definition of a *hacker*. (In the circles we travel in, *hacker* does not have a negative connotation. The people who do bad things with their skills are called *black-hat hackers*.) Rich is very detail-oriented and has encyclopedic knowledge of the hardware and software we cover in this book.

Joan comes to this as a recovering rocket scientist, and her role is to keep the big picture in mind and think about how to avoid going into too much detail in any one technical area. She worked on spacecraft to other planets, where one tiny mistake could cause disaster. So she brings structure and experience working on complicated systems, and also the desire to make explanations as simple as possible (while still being correct).

Lyn comes with long experience making costumes for middle and high school productions. Besides the ability to apparently whip up costumes out of a pile of fabric plus thin air, she has a sense of humor and a keen eye for when a small detail might make all the difference. She has also taught sewing in a classroom and so knows what pitfalls might arise.

The three of us take turns guiding you through the book and switch into first person for much of the book when we focus on one person's particular expertise. Sometimes we collaborate too closely for any one of us to take the lead, and there (as we do here) we will just say *we*.

We are walking through our backgrounds here (there is more in the "About the Authors" section at the front of the book) because we suggest that you build a team to work on your first projects that has all these aspects, though not necessarily spread across the team the way it happens to be with the three of us. Good chemistry and a sense of humor are also important for a team to have. Sometimes things just come out looking silly, and you have to laugh, figure out what went wrong, and not do *that* again. At other times you may need to let one of your team members go off in a corner and try things for a while. But if one of us got stuck, we found it valuable to articulate what the problem was to the others, and we could go back to first principles and try to think about what we were trying to accomplish.

Planning Your Projects

There are many books out there that cover different aspects of making a fashion tech project, and many projects on the Internet that you can try. We felt an orderly path was missing, starting with the basics of sewing, plus electronics hardware and software, to allow someone who knows very little about the skills needed to get started on projects.

We always emphasize the idea of *system design*. It is very easy to come up with a great idea for part of a project, but doing that part the way that you would if it was not incorporated into something wearable might be very different than the way you should start out to incorporate it into a dress or hat. We cannot emphasize enough the need to plan out a complex project end-to-end ahead of time. Chapter 10 tells the story of the first project we did together in which we largely ignored this principle—even though all three of us knew better.

In this book we cover the basics of sewing (including making and using patterns), creating electronic circuits, and programming. These are the core skills of most fashion tech projects, and we go into considerable depth on each one. We have laid the chapters out in the order you would most likely create a project: sewing first, then figuring out the circuitry, and then programming it if needed. We guide you in managing complexity in your projects and give you tips on how to avoid being overwhelmed if your project does not work.

The Wearer's Environment

As Chapter 2 advises, you also need to think hard about what the wearer is going to be doing and what environment they will be in when they are wearing the garment. For example, the projects in this book are all intended to be worn indoors, in a dry environment. Even though one project is an apron, we imagine it as more something one would wear to impress friends while serving food at a party, and not so much when cooking or with wet hands.

Prototyping and Testing

As you plan out projects, think about where in the project development you might be able to cobble together mini-prototypes. For the sewing portion, you might make a *muslin*—a version of the project in a very cheap fabric that you will not mind taking stitches out of and resewing multiple times if necessary. When creating electronics, you might initially use *alligator clips* (clips to hold together a circuit temporarily) to create a circuit that you will ultimately sew on with conductive thread. And when programming, you should consider how to create some simple stepping-stone projects that build up the ultimate functionality one part at a time. As we go through the book, we will suggest these techniques. As you are coming up with ideas about what you want to do, consider how to make some parts of it independently testable so you are not left with a complex project that might have multiple interacting problems.

The most important thing, though, is to figure out what you want the costume to be first, and add the technology or animation second. Lyn will now step up and talk about a few of her best creative experiences.

LYN COSTUME STORIES

In every production I have worked on, there seems to be at least one costume that is difficult to get right. The concept was too abstract to understand and/or perfect, or the fitting was troublesome, or it suffered from numerous other issues. This was the case for a turkey for *A Coney Island Christmas.*

Some of the costumes in this production were a particular challenge because the play is set in the Bronx in the 1930s. There is a play-within-the-play in which costumes are supposed to look homemade by children and their parents in Depression-era New York. I wanted a kitsch factor, especially for a turkey that appears in a Thanksgiving Pageant scene. Most of the costumes we found were way too modern. The colors were too vibrant and neon for the time period, and I wanted the look of the fabrics to be authentic.

After much brainstorming with the director, I purchased a turkey costume online. When it arrived I took my scissors to it. I cut the sleeves off and trimmed away most of the front and back of the torso. This allowed me to have straps or suspenders over the shoulders of the actress. It had a stuffed lower body, which I wanted to preserve. I didn't use the leg covers because she would be wearing it over a dress.

After I was satisfied with the shape of the overall costume, I began to cover the bottom half with curled construction paper strips to represent feathers. Using muted brown, orange, and yellow strips of paper, I curled them around a pencil and glued them in rows onto the turkey body. The tailpiece of the costume was a nice shape, so I just covered it with construction paper feathers, and it was my favorite part (Figure 1-5).

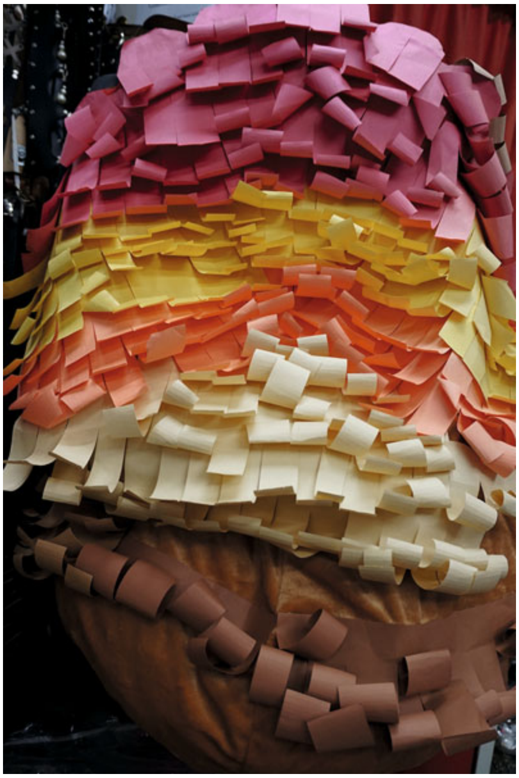


Figure 1-5. *Close-up of the construction-paper turkey tail*

The curling and gluing of the paper feathers was time consuming and a bit tedious, but I was pleased with the outcome and effect. For the wattle and beak, I cut two pieces of construction paper in the shapes needed and used double-sided tape to hold them in place on the actress's face.

I also used the curled construction paper feathers for the wings of the angel Gabriel. Cutting the wing shapes from cardboard, covering them with white feathers, strapping them to the back of the actor, and making sure he went through doorways sideways, was a successful and charming solution for this costume. Most of the time I am working with fabric to create and alter costumes, so this was a refreshing and back-to-basics show to work on.

A production of *The Wizard of Oz* was perhaps my favorite show in terms of the freedom to explore wild and zany ideas. The director, another costumer, and I spent a lot of time tinkering with plans for some of the characters that were a bit different from the usual versions. We put the Wicked Witch in ski boots on a rolling platform pulled by one of her monkeys. She could move without walking and lean forward in a very menacing way. To allow her to melt onstage, we added a hoop skirt under her witchy dress and she could slowly writhe and squat down very small into a heap, leaving the hoop standing with a sunken center.

The monkeys wore thermal underwear dyed brown with furry patches and tails sewn onto them. The Tin Man had metal stovepipe arms and legs, and the Cowardly Lion's mane was made from a mop head. The Munchkins were the most fun to create. I found some pretty ugly and colorful tie-dyed jumpsuits at a budget store. Then I used regular dance tights in bright colors, stuffed them with batting, twisted two pair together for each actor, and attached them to their heads. When the Munchkins were hiding as the house landed in Oz, they looked like a field of strange and wonderful roots.

I have discovered through many years of costuming that often the pieces that seem to be failures at the start of a production turn out to be the best by opening night, or at least the most commented on. My favorite moments of any show are the brainstorming, exploring, and experimenting in the beginning with the production team. Bounce ideas around and gathering new input and different points of view are exhilarating.

■ **Note** As you read Lyn's stories, a few things probably stood out. One was that it sounded like fun. Good design cannot take itself too seriously. Be playful and try incongruous things. That's not a skill we can teach you in a book, but we will intersperse sidebars like Lyn's here and there in the book to give you examples of exercises that went well. Chapter 2 goes into some depth about what makes a good costume, deconstructing more experiences like those that Lyn just shared.

Summary

In this chapter we defined fashion tech and gave a brief history of the converging fields that comprise this interdisciplinary endeavor. From there we moved to introducing different costuming situations that might apply these technologies. We talked about how to use the book and described our design philosophy. We also mentioned a few case studies about great costumes that were decidedly low-tech, to make the point that a good costume will shine through and design is the most important element in a good project. In the next chapter we talk more broadly about what makes a good costume.

CHAPTER 2

Practical Costume Design

In this book we teach you a bit about a lot of things: sewing, pattern-making, electronics, and writing computer code. For many readers, though, the point will be to make a costume to wear. People have always dressed up and masqueraded, but recently cosplay has been popular. *Cosplay*, which has spawned many conventions for devotees, can be loosely defined as dressing up (sometimes in incredible detail) as your favorite character from a movie, video game, or other story. Or cosplayers might start with a style (like steampunk, a mashup of Victorian clothing and quasi-period mechanical decorations) and create something from there.

The skills are important, but it is also just as critical to think about what the final project is for. If you are using this book as the basis for a class to teach these skills, then perhaps each project can be a small exercise building up to a final project to wear or maybe to hang on the wall.

If, however, you are reading this book because you want to create costumes for yourself or perhaps your local theater group, then you need to think a bit about the design issues about what makes a good costume. Lyn has been creating costumes for many years, and the next section is a distillation of her experiences. She and Joan also chatted with past and current owners of a professional costume-making and rental store in Santa Monica, California, to get their take on what makes a good costume.

What Is a Costume?

Costumes not only play a very important role in the performing arts, but they are also significant in our daily lives. What you choose to wear and when can inform your family and friends with many details about your mood or where you are going. These choices define how we present ourselves to the world. Whether it is *haute couture*, workout attire, or quirky combinations, our clothes make a statement about who we are or want to be, and how we want the world to perceive us. A good costume can inform an audience about the nature of a character before a word has been spoken.

J. Horvath et al., *Practical Fashion Tech*, DOI 10.1007/978-1-4842-1662-0_2

The Power of Costume Design

Costumes can grab an audience and engage it in the story in a subtle way, or distract the audience from the plot of the story. Cosplay costumes are often very specific to certain characters, but can also be created to introduce new characters or ideas.

Characters, social status, and time periods help to define what a costume should look like. Is the costume for a man, woman, animal, alien, or some combination of these? What type of character is being created? How is this character different or the same as others? How old is this being, and what time period do they live in? How much physical activity must be performed by this character in this costume? Is the costume being used realistically in a play or story, or is it used in an exaggerated or stylized performance?

Costumers and designers must research how people of all classes and genders have dressed throughout history. Historically, a person's sex has been clearly distinguished by their clothing. But in today's world, all bets are off. Men and women choose many of the same items, colors, and styles. Almost everything is acceptable and experimented with.

A costume can set a character in a historical time period, certain location, and specific country. Most time periods have very definite looks, styles, and shapes, and a costume may also show the climate or the season the character is inhabiting.

How a character is dressed may reveal the nature of a character or how this character wants to be seen, which may be opposite his true nature. A villainous character might attempt to appear simple and honest. Or perhaps a meek, shy type may pretend to be cool and hip. Consider Odette and Odile in *Swan Lake*—the good and evil swan, typically played by the same person, but identified by the switch from white to black tutu. You can also think about the white or black cowboy hat in classic Westerns used as shorthand for "the good guy" and "the bad guy." And, of course, there is Darth Vader versus Luke Skywalker.

Lyn's Tips for Student Theatrical Costuming

If you are in a school theater department, you will also likely have constraints of time and cost and may have one person making, adding to, or altering many pieces (as I am doing in Figure 2-1). Regardless of the resources you have to make a costume, the key part is to start with a clear idea of what the costume is trying to convey.

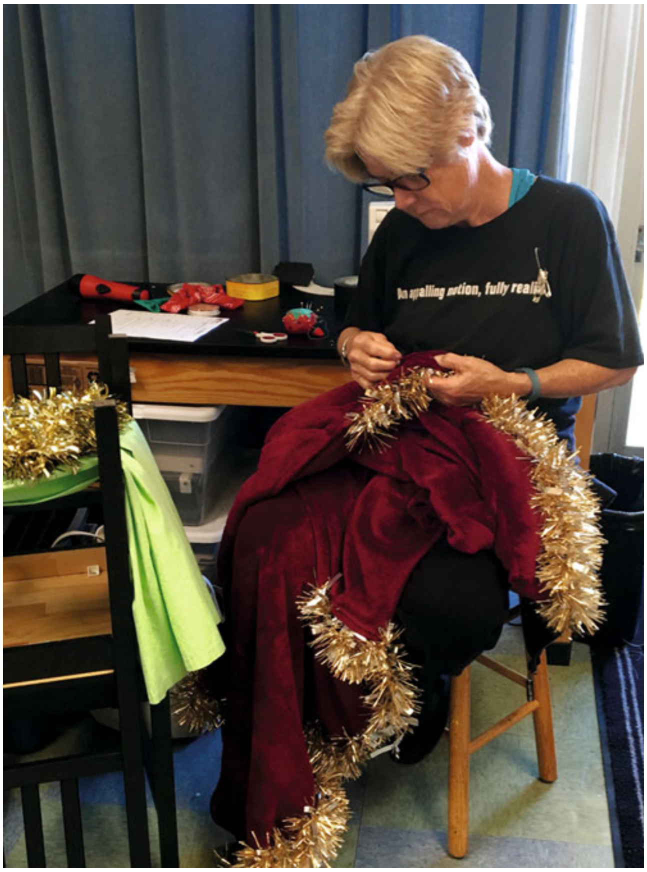


Figure 2-1. *Lyn working on school musical costumes*

I have often relied on donations from parents and lots of shopping trips to secondhand stores to create entire costume plots for a production. It is fairly easy and a lot of fun to add trim or other things to a dress or robe, and invent an entirely different look. The piece I am working on in Figure 2-1 is a secondhand velour robe. I am sewing a gold garland on as trim for a Father Christmas character in *A Coney Island Christmas*.

Some of my best costumes have been garments that I had to deconstruct and resurrect as something new. I used a 1950s-era pink prom dress and added lots of lace and frills for Glinda the Good Witch in a production of *The Wizard of Oz*. For the Wicked Witch of the West, I used a black taffeta cocktail dress from the 1960s, removed the glamorous parts, and added rubber snakes, spiders, and other evil things. I think these two are among my favorite creations. Unfortunately, they were loaned out a few times and never found their way home . . . heavy sigh.

It is possible to costume an entire production on a small budget. It just takes some planning and work, but it can be quite spectacular and rewarding, especially when students help to brainstorm and cut and sew.

It is also a lot of fun to build a costume from the ground up. A few of my favorites were two polka dotted 1960s Twiggy-style dresses for a production of *The Odd Couple*, the costumes for Lola in *Damn Yankees*, several 1940s tap-pant sets for *Anything Goes,* and a turkey for *A Coney Island Christmas*.

Take a garment and turn it into something entirely different from its original incarnation. Give your brain the freedom to explore. Be bold and brave in your choices. Sometimes it fails, but other times it is spectacular. I have learned a lot from every piece altered or created, especially the ones that had problems and turned out to be less than fabulous.

Costume Design Principles

When asked, "What makes a good costume?" Tanisa Fatchett's quick response was that it had to be immediately obvious what the wearer was supposed to be. Tanisa has just taken over the reins of Make Believe (`www.makebelieveinccostumes.com`), a costuming creation and rental store in Santa Monica, near Los Angeles.

Make Believe was founded in the early 1980s, initially as a sort of quasi-captive costuming department for a nearby theater. Founders Doug Spesert and Ruth Talley built up the business to its current inventory of what Tanisa estimates to be thousands of costumes (Figure 2-2).

Figure 2-2. *Professional costuming inventory*

Founder Ruth Talley recently retired but was kind enough to send Lyn some of her key criteria for a good stage costume. Her key points were the following:

- "Costumes only work if they are part of the overall design of the set, the lighting, the make-up, hair, the props, and so on. When it all comes together, it's magic. The costume must fit the period and within that it must fit the actor. I've "cheated" at times with some historical inaccuracies if the actor ended up looking ridiculous. Maybe the bustle has to come in a bit, or the codpiece not quite so padded, or the myriad of buttons top stitched on Velcro for an easy on and off."
- "The costume also has to work for the actor. If they need to stand out, the color, fabric, trims all need to stand out. If they need to be unnoticed, then muted is the way to go. If it's a rip-away skirt, it has to look effortless, or accidental. In period pieces, the audience has to feel that the undergarments are period too, and some of those undergarments can change posture and movement, which adds to the historical feel."
- "The audience should be able to tell very quickly what decade you are in, be it 1840 or 1940, and they should be able to identify class, royalty, and maybe occupation. The audience should be able to determine time of year and time of day or night. If you get all of this spot on, then you have a great costume."

Ruth also sees parallels throughout history between the fashion and architecture of a period. She says, "I wanted to study buildings and clothes of different eras to make comparisons on the silhouettes, austerity, or grandeur. Fashion can tell us a great deal about the times we are living in and times past. I think the transition of the teens to the Roaring Twenties is a great example of a liberal movement in dress, thought, behavior, literature, and how representative was the flapper in this? It's hard to separate the influence of fashion designers on artists and vice versa. Think Erte. Think Mod 60s. Think Pop art. Then there's the music . . . were the Beatles wearing costumes on *The Ed Sullivan Show* that became mainstream fashion? Now it's the costumer's job to capture that on the stage. Establishment, hippies, mods, and nerds. And so it goes for so many eras. It is the costumer's challenge to make the French look French and the English look English in *The Tale of Two Cities* and *The Three Musketeers*. And how about Malvolio? What is he without his crossed garters?"

Making a Costume Work for You

The most important thing for a person to look good in a costume is that they have to be comfortable in it, Tanisa advises. By this she means physical comfort, and that the costume is sturdy and not falling apart around the wearer. But she also means that in a slightly more subtle way. One might imagine that a professional costumer might spend most of her time gently talking people out of age-inappropriate attire.

However, apparently it is common that people think they are not good-looking enough to pull off a particular look. People have to "let down their guard" to wear a costume effectively, Tanisa says, and not everyone can or will do that. And they have to "decide what kind of attention they want to get."

It is important to try all the physical movement that is required of the character in the costume as it is being created. If you wait until it is finished, you may have to do a lot of alterations.

Costume Development Time and Effort

One of Tanisa's favorite costumes of all time is a period gown (shown in Figure 2-3). Tanisa estimates that there are hundreds of work-hours in this costume. The beaded fabric can be bought, but is very expensive and so has to be used carefully so as to avoid wasting it.

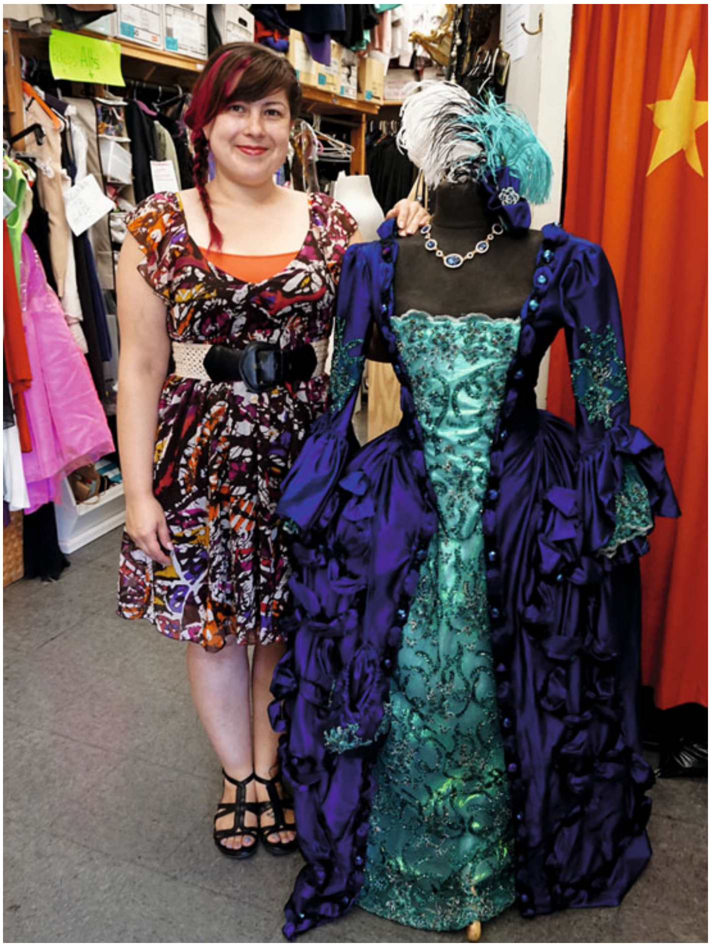


Figure 2-3. *Tanisa Fatchett and a favorite gown with accessories*

Beyond the incredible surface detail, there is a lot of infrastructure under the hood, so to speak (see Figure 2-4). Elaborate costumes like this have a lot of layers, and experience guides what kinds of fabrics to make things out of so that they can be altered for different wearers without being destroyed in the process.

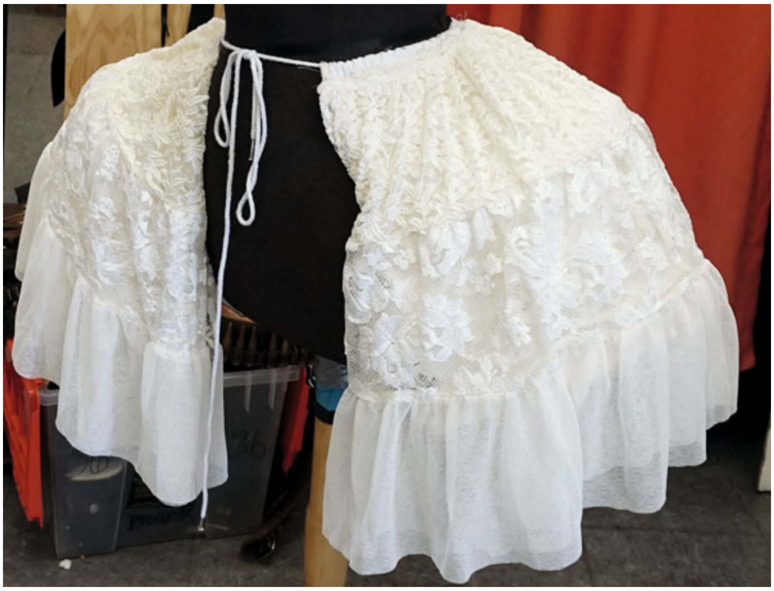


Figure 2-4. *Some of the infrastructure for the gown in Figure 2-3*

WHAT WILL YOU MAKE?

Before you get too far into this book, think about what kind of project you want to do ultimately. This may change as you learn more about what is possible, but having a goal will both motivate you and keep you from going down too many isolated tech explorations, resulting in never having a cool finished product. Flip through the project chapters (Chapters 7 and 11) to see if something jumps out at you.

If you are a teacher laying out a class, you might want to come up with a few types of final projects that will narrow down the possibilities a little. For example, you might suggest that they create props for some part of a school theater production that happens near the end of the course.

Alternatively, the whole class can work on one historical costume piece that has a lot of detail and accessories. They can all participate in different levels of its creation. It is inspiring to work as a group, and they can learn to collaborate and express their ideas. Another good final project is to have each student choose one character in one scene from a play, design the costume and accessories, and then build the entire look.

Beginner Mistakes

In Chapter 10 we tell you about our first major wearable tech project, in which we were far too ambitious. Tanisa agreed that a common error among the inexperienced was to try to do too many things at once. When asked to think of a costume that had failed, though, she could not come up with an example. "There's something to learn from everything," she says. This is an attitude remarkably close to that of our friend Metalnat Hayes, cofounder of Burbank Makerspace (`www.burbankmakerspace.com`), who is fond of the hashtag `#testcase` to describe something that went kerplooie but which he learned something from just the same.

Materials

Tanisa noted that with sewing projects, the commonest beginner mistake is to go for "really pretty, shiny fabrics," that may not be the right quality, or fabrics with no body or texture, that show many mistakes. In this book we try to keep you away from that one by recommending fabrics for the projects. Our aim though is not to guide you through specific projects for their own sake but to make you able to go on from here. Thus, we give reasons for what we recommend so you can expand on it. But do not be afraid to experiment, especially on unwanted garments and secondhand store purchases. Those are terrific and inexpensive ways to learn to work with different fabrics, trims, and garments.

When professionals get together, they talk about materials and how to put them together, and share experiences with glues and similar substances. That type of folklore is not something you can learn overnight, but take advantage of the interactions you do have when you buy things to try to learn more. This is one of the advantages of going to a brick and mortar store versus always buying online. The same applies to the electronics and software world too, but the equivalent of a physical store in that case would probably be hanging around at a hackerspace.

■ **Note** One of the challenges with projects like those in this book is that you might be trying to integrate electronics with materials like those shown in Figure 2-5. Crossing over styles and disciplines is some of the fun part, but be sure to think through possibly incompatible materials early in a project.

Figure 2-5. *The feather boa stock at costuming store Make Believe*

Estimating Time

When professionals are creating a costume, the time they spend is important since they have a business to run. A cosplayer might spend a year making a costume for a convention, but obviously, if stores did that they would go out of business.

■ **Tip** When you think about designing your own costume using the techniques in this book, we suggest that you estimate about how long you think it will take to try something out, and then multiply your best estimate by three. That is a typical number to cover the time for you to learn all the *other* things that you have to sort out first while making something. But with most of these things the only way to learn and make techniques your own is to try things. As we say, `#testcase`!

Do Not Go It Alone

Professional costumers often specialize in one aspect of the design and will hire others to do a part of it. When you are learning, team up with friends. If you are teaching this as a class, perhaps one of your colleagues has some experience that can help. But do not be afraid to experiment with a look, alone or with friends!

Summary

This chapter covers some of the philosophy behind a good costume, based on Lyn’s experience as well as the insights of some professionals in the field. Lyn reviews some of her favorite costumes, and the chapter covers a few key design points to keep in mind during the design of costumes for student theater productions in particular.

PART II

The Basics

In Chapters 3 through 7 we introduce the key skills needed for fashion tech. Chapter 3 covers hand and machine sewing basics, with many references to other resources. Chapter 4 focuses on the art of creating and using a sewing pattern.

In Chapter 5 we switch over to the tech side and give an introduction to electronic components. In Chapter 6 we see how to program these devices.

Finally, in Chapter 7 we bring it all together in a comprehensive but manageable project to create a hostess apron with a built-in timer that flashes a red light on the apron when the timer is counting down, and a green one when time is up. These chapters are enough to create “blinky” projects—garments that light up with some minimal ability to control the lights.

CHAPTER 3

How to Sew

One of the challenging things about wearable tech is that you need to learn to do a lot of different things that would not typically be in the skill sets of one person. You can, of course, collaborate, as your three authors have done. In this book we try to give you all the key basics so that you can in principle do projects end to end yourself, or with your students.

This chapter covers the basic elements of hand and machine sewing, starting you off with hand sewing and then moving to machine sewing. This is Lyn speaking in this chapter. I've taught sewing and costuming for many years, and so I know all the mistakes you are likely to make and can help you skip some of them.

Using inexpensive fabric to learn and to perfect your skills is a good idea because mistakes can become quite expensive when you have to replace fabric. There have been times when I have had to redo a seam several times, which left visible needle holes in the piece. I have had to cut out a new piece from more fabric after too many mistakes.

Note Don't be afraid of making mistakes. It is almost inevitable when you are learning new things. When I began to sew, I would make the same mistake several times in a row, and it would make me crazy. I had to discover when I was too distracted to continue and take a break. Almost every time I was making something with sleeves, I would attach the first one correctly and then attach the other one inside out. I had to make sure I was completely focused before I attempted the second sleeve. It was a matter of discovering my foibles and how to prevent them. I still make mistakes, but fewer of them, and I have become better at fixing them.

Figure 3-1 shows me with my first wearable project—a vest with lights and sensors that we talked about back in Chapter 1 and deconstruct in detail in Chapter 10. Joan and Rich say that if you are coming at this from the electronics side and have never sewn, you will probably find it to be both more forgiving and requiring more design decisions than the technology you are used to. (Color and drape do not come up in circuit design very often!)

Figure 3-1. *Lyn and her first wearables project*

Basic Hand Sewing Tools and Techniques

Sewing has been around a long time—since the Paleolithic era (there is a nice history at `https://en.wikipedia.org/wiki/Sewing`). Hand sewing in the machine era is a way of creating one-of-a-kind objects, and at the moment is the only real way for a home hobbyist to attach sewn components to fabric. It does not take a lot to get started.

I teach sewing, costuming, and design in a class setting, which means when students have questions, or problems occur, I am there to answer and correct. If you are an absolute beginner at sewing of any kind, I recommend asking someone to coach you, or watching some online videos for specific stitches or questions you may have. Here are some sites to check out to get started from zero: `www.Instructables.com/id/How-to-Sew./` or `www.wikihow.com/sew` or the video at `www.youtube.com/watch?v=B2mfJweh8aO`. There are short videos of making many types of stitches on the Instructables site, such as the the slip stitch that we describe in this chapter (`www.instructables.com/id/Hand-Sewing-Basic-Slip-Stitch-Blind-Stitch/`).

Alternatively, just search for "How to Sew" or "How to sew by hand or machine." Sometimes it may take more than one picture or description to visualize what is being described and explained. Figure 3-2 shows the things that you will need to practice the various stitches described in this section. You will also need them to create your projects:

- Light to medium weight cotton fabric.
- Sharp hand sewing needle (for light–medium weight fabrics).
- Sewing shears or scissors (usually found in a fabric store).
- All-purpose thread. There are many brands of thread to choose from. I like Gutermann's polyester thread, which has a bit of stretch and doesn't easily break. For practicing stitches, using a contrasting color of thread relative to your fabric makes it easier to see your stitches on the fabric.
- Straight pins (to hold fabric pieces together while sewing) and a pin cushion (to hold the pins).
- Measuring tape. (A cloth tape is best for measuring your body. If you don't have one, use string or ribbon to wrap around what you are measuring and then use a ruler to measure the string.)
- Tailor's chalk or water-soluble marker (to mark the fabric showing notches, darts, or other symbols from the pattern).
- Seam ripper (to take out stitches if you make a mistake).
- Iron (to flatten seams you have sewn and get rid of wrinkles).

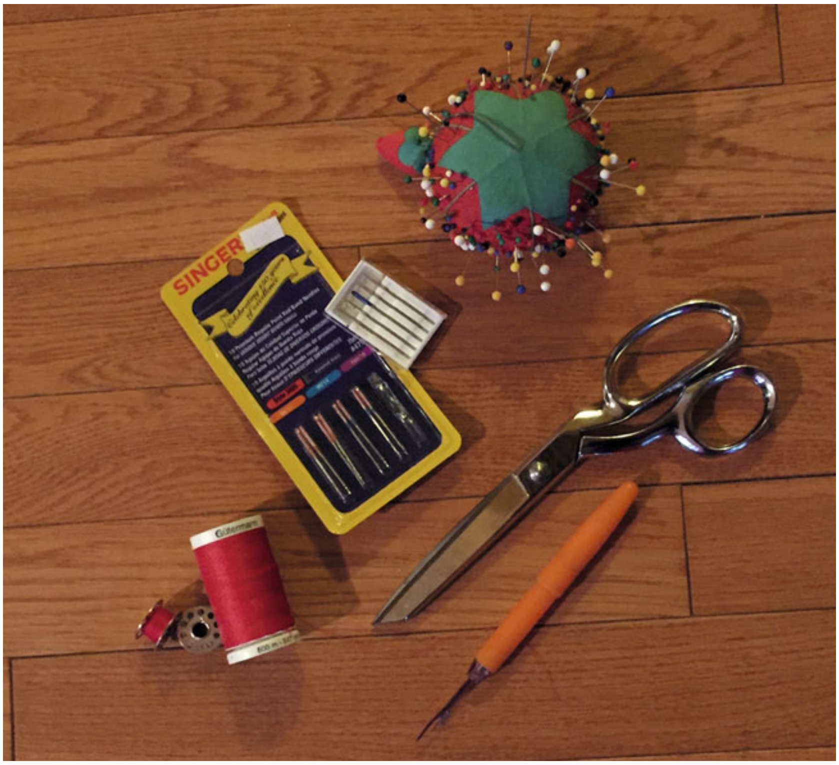


Figure 3-2. *Items you will need*

The following items are also nice to have, but not essential:

- Cutting mat or cardboard (to protect surfaces and keep fabric clean)
- Needle threader
- Magnifying glass
- Thimble (to protect your fingers)
- Beeswax (described later in this chapter)

To begin your first sewing experiment, choose a small piece of light to medium weight all-purpose cotton fabric. There are many types of fabric. Cotton is an easy one to learn on and practice basic skills. You can use something like a bandanna or flour-sack kitchen towel, or buy some remnants from a fabric store (*remnants* are leftover bits of fabric from the ends of bolts and are usually inexpensive).

■ **Tip** You can buy fabric online, but if you aren't familiar with the various properties of fabric, it may be wiser to go to a bricks-and-mortar fabric store at first. There you can feel the fabric and ask questions about the weight or amount of stretch and what type of needle you might need to sew it with on a sewing machine.

Threading a Needle

Take your sharp needle and a contrasting color of thread, as noted earlier, to make it easier to see your stitches on the fabric. Later on, when you are creating a garment, you usually will want the thread to match the fabric. However, there are times you may want to use contrasting thread as a design accent.

Hold the needle with the eye facing you. Cut a small bit from the end of the thread to make sure it isn't frayed, poke it through the eye until you can grab the end from the back, and pull about 3–4 inches through. Alternatively, you can use a needle threader (Figure 3-3). Hold onto the eye of the needle and the thread, pull about 12–18 inches of thread off the spool, and cut the end.

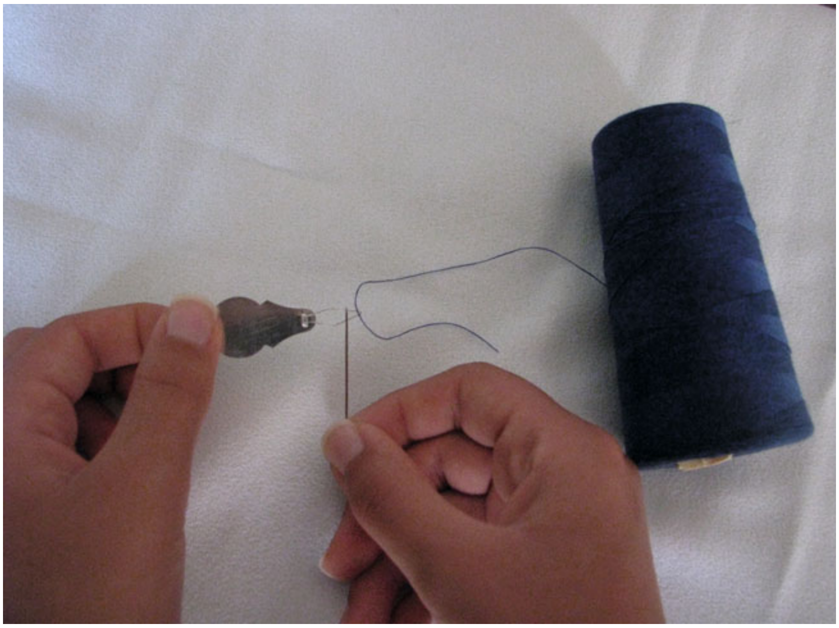


Figure 3-3. *Using a needle threader*

■ **Caution** You will rapidly discover that the ends of the thread tend to fray. Steel conductive thread is particularly hard to deal with this way. Use a bit of beeswax on the end of the thread to keep the ends of the thread from fraying and to make threading needles easier. For cotton thread, you can also moisten the end of the thread with your mouth, if that doesn't seem too gross. Don't do that with conductive thread—both because it won't work and because the ends are a bit sharp.

Knotting the Thread

Make sure you hold onto the eye of the needle and the thread or push it into a pincushion (it is easy to pull the thread out of the needle, and you have to rethread it, which becomes tedious). Using the longest side of thread, hold the end between your index finger and thumb, wrap around your finger once, and push the loop off your finger while holding securely onto the end and twisting between your thumb and forefinger (Figure 3-4). With any luck, you will have a knot.

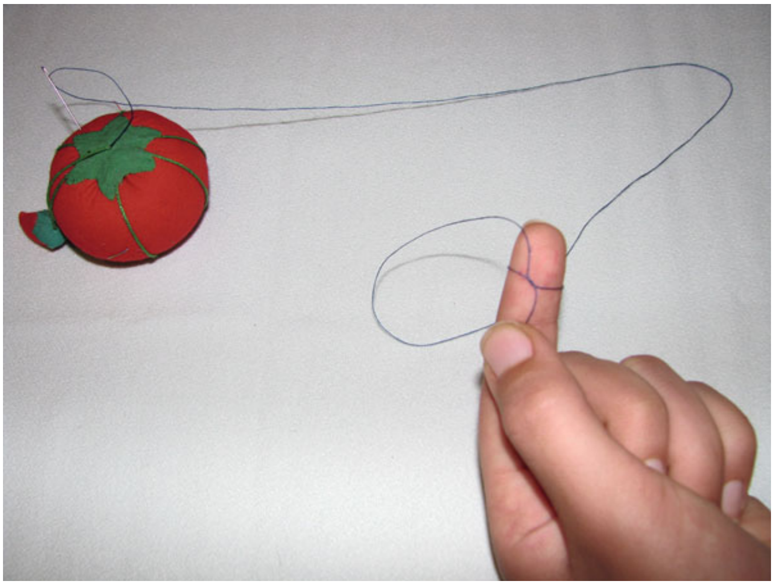


Figure 3-4. *Making a knot*

Sometimes it is best to repeat this process for a slightly larger and more secure knot. If you have a knot that is too small for the weave of the fabric, it will pull through and you must retie the knot and begin again—sometimes again and again. Sewing can try your patience when you are a newbie, but there are ways to correct mistakes and carry on. Deep breathing and a brief walk around the room, house, or block may help.

■ **Tip** If the fabric has a very coarse weave, it may be a good idea to take a small stitch, pull it almost all the way through, and then tie the thread together into a knot. We will be using conductive thread in other projects in this book. *Conductive thread* is made from thin, braided wire, which makes it difficult to create a tight knot. After knotting conductive thread (but not regular thread), I use a drop of clear nail polish to keep the knot tied.

DISTINGUISHING BETWEEN THE RIGHT AND WRONG SIDE

The top or most colorful side of the fabric is called the *right* side. The underneath or less vibrant side is called the *wrong* side. When using fabrics with a *nap* (fabric which has a different appearance depending on which way it is rubbed), such as corduroy, velvet, and velour, the plush side is the right side. Sweatshirt fabric uses the flat side, not the fuzzy side, as the right side.

Using Different Stitches

This section talks about the different ways you can stitch something together. Stitches are your sewing tools, and it helps to have a lot of them at your disposal.

■ **Tip** Always hold onto the needle and thread together at the base of the needle to avoid pulling the thread out of the eye. This will help you avoid constant rethreading.

Running Stitch

Put the two right sides of the fabric together. Holding the needle and thread, poke the tip of the needle up from the bottom side (the two right sides will be together) and pull it through both pieces of fabric until the knot hits the fabric. Approximately a quarter of an inch from where it came up, push the needle through from the top side and pull until the thread is pulled all the way through and the stitch is flat. For clarity, Figure 3-5 just shows the stitch through one thickness of fabric.

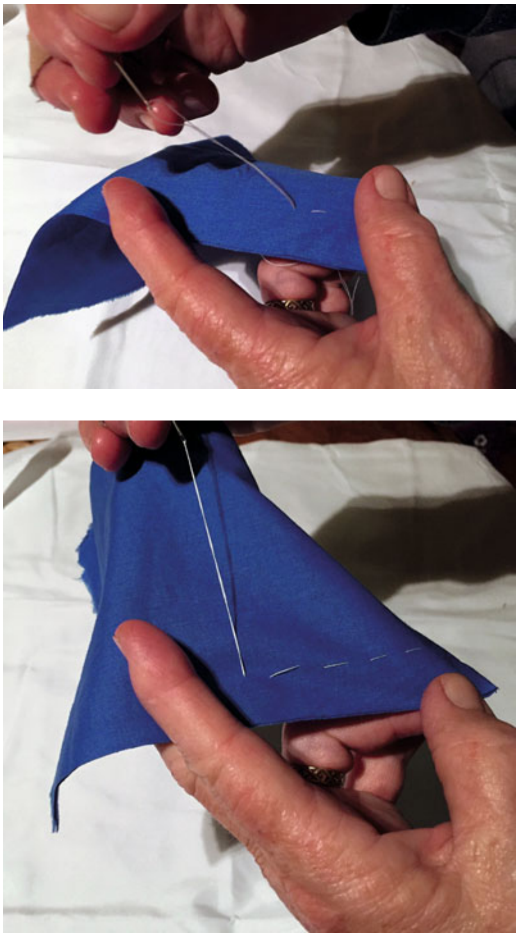


Figure 3-5. *The running stitch in progress*

Continue these steps to make a straight line of stitches. You can make the stitches on the wrong side (or bottom) slightly longer than the right side (or top), or you can make them the same length. Experiment with different sizes until you are comfortable with the technique and able to keep the stitches neat.

Basting Stitch

This stitch is the same as the running stitch but each stitch is quite a bit longer. It is a temporary stitch that usually holds pieces together without pins for machine sewing, for quick hems and other repairs or temporary fixes.

Backstitch

This is the strongest stitch, and perhaps easiest to keep straight. Make a stitch in the fabric, and when you bring the needle up through the fabric for a second stitch, instead of taking a stitch away from the first one, poke the needle through at the end of the first stitch on top and pull the thread through the fabric (Figure 3-6). The next stitch will be longer on the bottom, pull through to the top, and back stitch to the end of the previous stitch. Carry on with this technique, or use a few running stitches in between each backstitch.

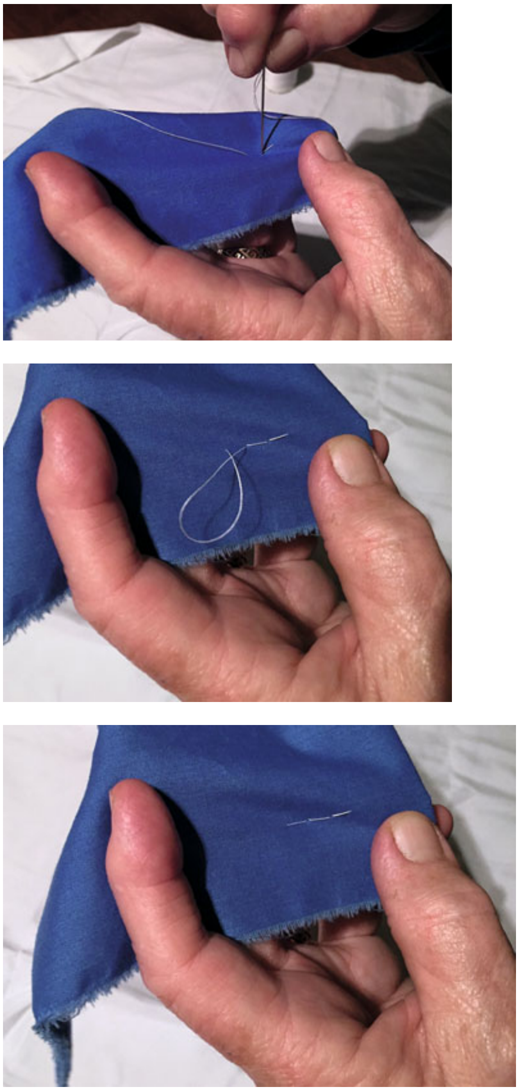


Figure 3-6. *Creating a backstitch*

Slip Stitch or Blind Stitch

This stitch is a bit complicated, but it’s a nice way to give your garment or project a professional look. It is an almost invisible stitch, and once you learn to slip stitch, it is a fairly quick finish.

To practice the slip stitch, cut out a small square of fabric and fold it in half. Open, and fold each side into the center line, press, and then fold down the center line (this and the next sequence of steps are shown in Figure 3-7).

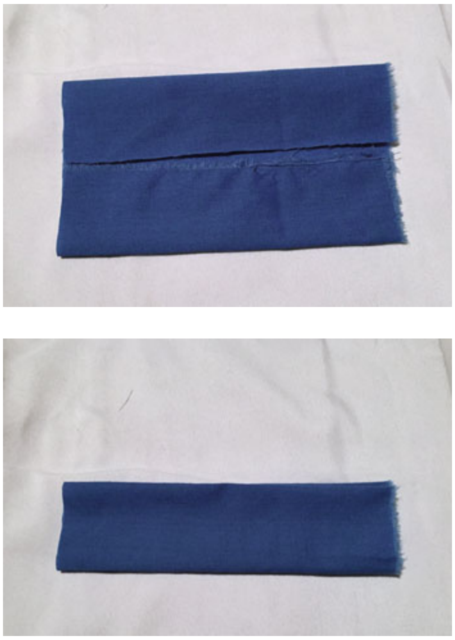


Figure 3-7. *Folding the fabric for slip stitch*

Pull the needle through the top of the fabric by sticking it through the top folded edge until the knot catches. Now move the needle to the bottom fold, stick the needle into the crease horizontally, and push it along behind the fabric for about 1/4 inch. Then pull the needle through and go back up to the top fold. Push the needle horizontally through the top fold and pull through. Continue in this way until the end (see Figure 3-8).

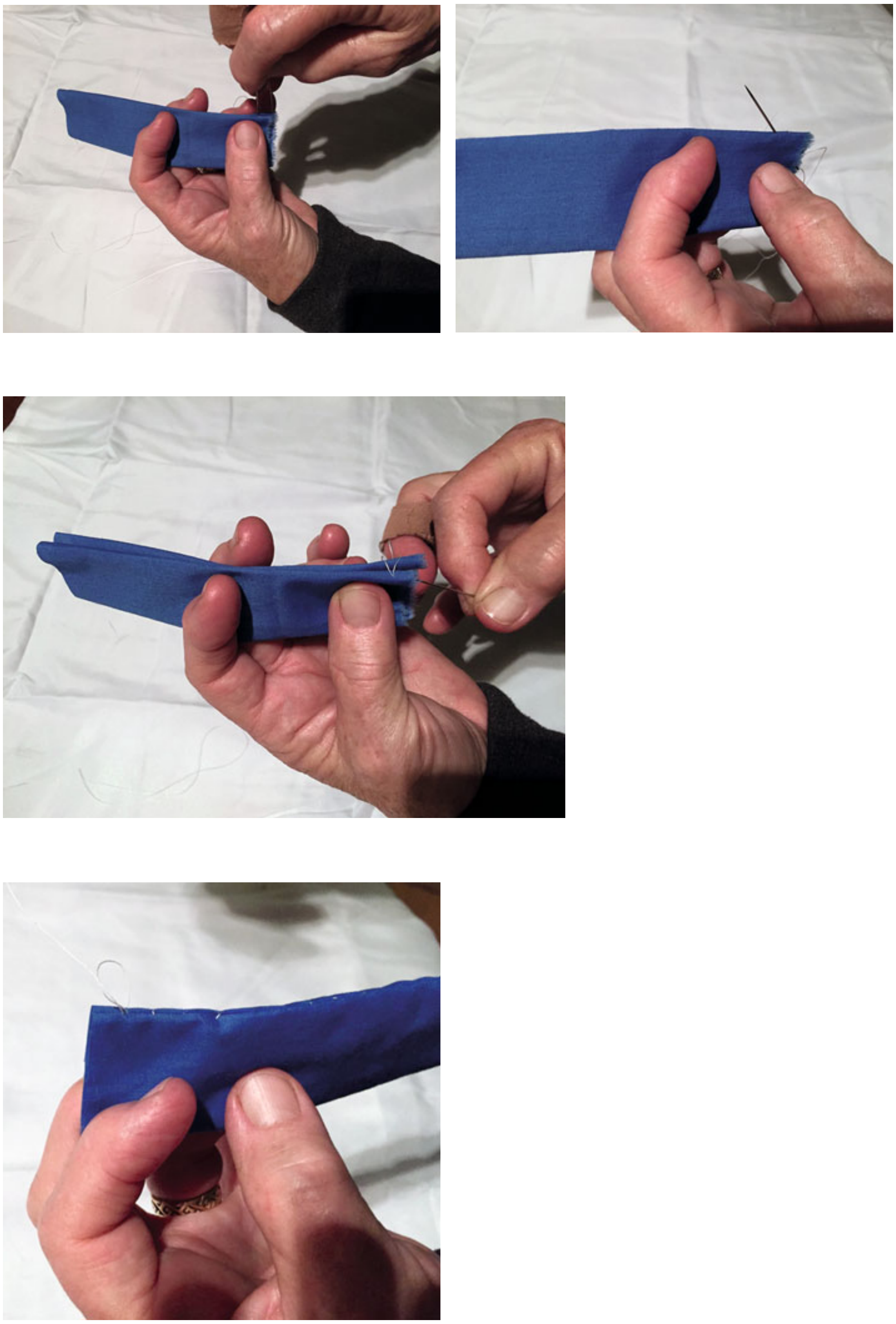


Figure 3-8. *Stitching with blind stitch*

At the end, make a small stitch in the fold opposite your last long stitch. Pull the thread until it forms a small loop and put your needle through the loop. Pull the thread to form a knot. Do this three times to create a strong end knot. This stitch can be used to connect two pieces of fabric or on a single piece of fabric as an edging.

Blanket Stitch

The blanket stitch is also intended to connect an edge. Start as you did for the blind stitch—pull the thread through from the back to the front near the edge of the fabric. Then place the needle diagonally to the direction you are stitching and go to the back of the fabric. Do not pull the thread tight. Rather, stick your needle through this loop first and then pull the thread to create a 90-degree angle. Continue until you have finished the edge. At the end, move the needle to the right of the last vertical line the thread forms, bring it up through the fabric, and form a loop. Put the needle through the loop and pull tight three times to form the end knot (Figure 3-9). Just like the slip stitch, the blanket stitch can connect two pieces of fabric or be used as an edge on a single piece.

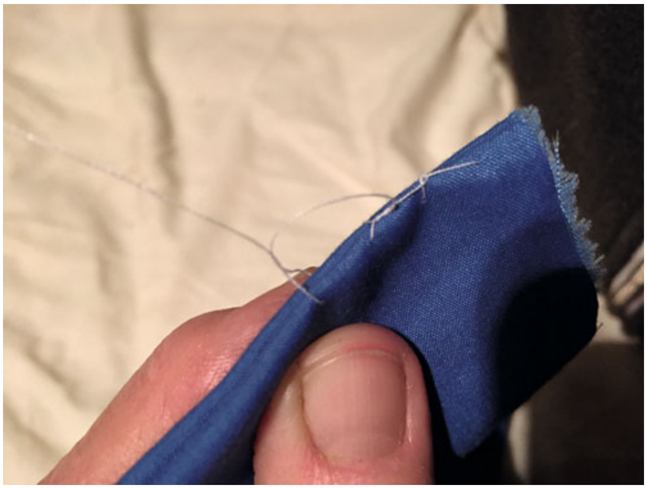
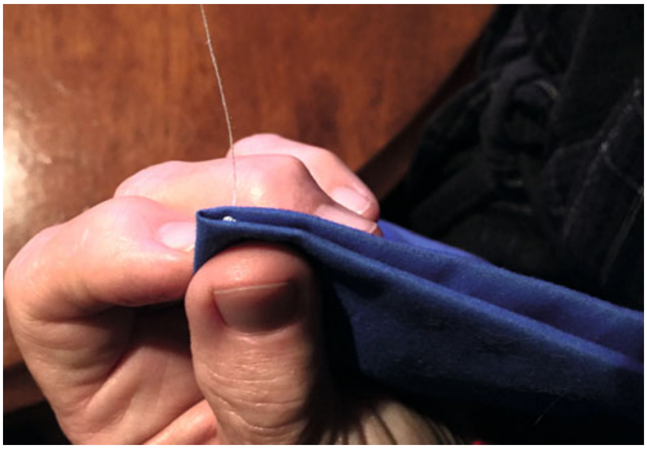


Figure 3-9. *Blanket stitch*

Whip Stitch

Fold a piece of fabric in half and pin it together. Insert the needle through the fold from the inside (Figure 3-10a) so that it comes out the front side (Figure 3-10b). Push the needle through the back and to the front so that it comes out level to the first stitch (Figure 3-10c). Continue to the end (Figure 3-10d) and create a finishing knot.



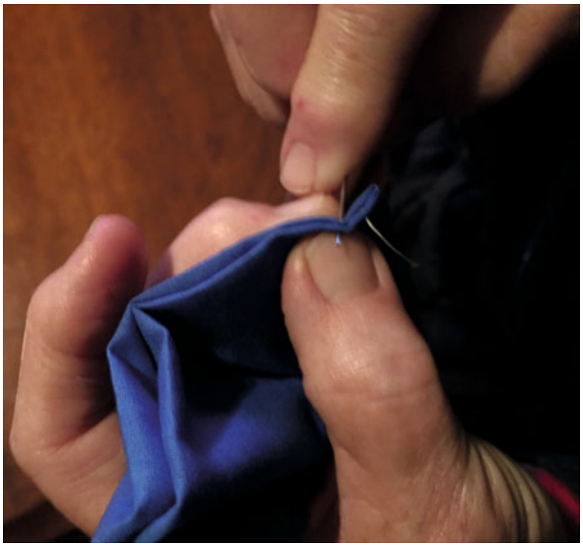
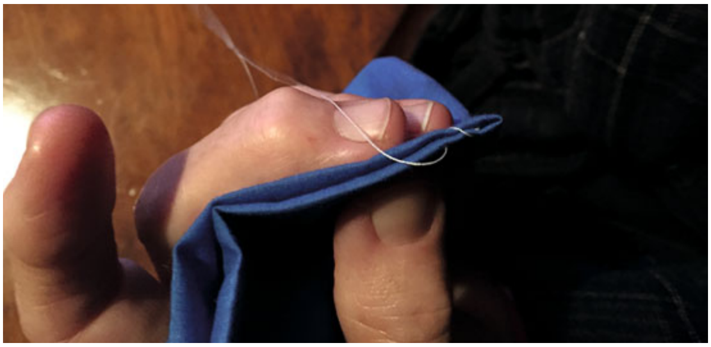


Figure 3-10a. *(top) and b (bottom). Whip stitch steps a and b*



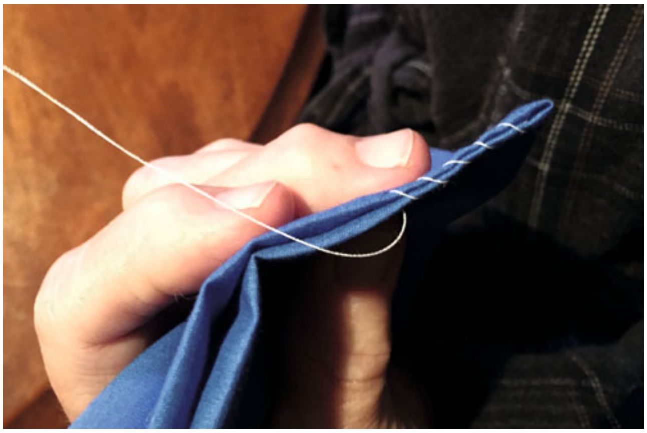


Figure 3-10c. *(top) and d (bottom). Whip stitch steps c and d*

■ **Tip** Use the pictures for reference to help you understand the instructions. These stitches should look the same on both sides of the fabric.

Threading a Sewing Machine

Figuring out how to sew by hand may seem daunting enough, but modern sewing machines are pretty complex robots. Since sewing machines last a very long time, if you have a machine, it may be anywhere from 60 years old (like the cast-iron 1953 Singer Featherweight in Figure 3-11) to a modern equivalent (Figure 3-12). The instructions that follow try to cover this spread of machines wherever possible.

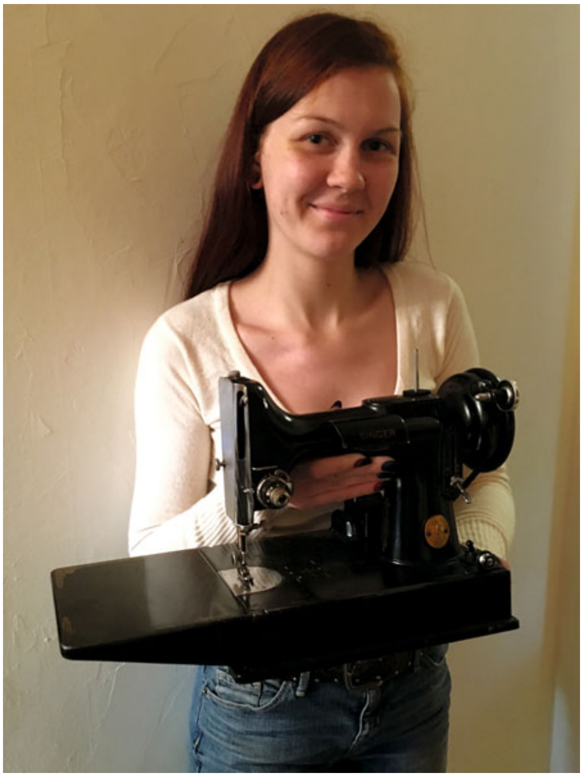


Figure 3-11. *A 1953 Singer Featherweight, still working perfectly in 2015. (Thanks to Rich's sister, Rachel Cameron, for modeling for the pictures and reviewing these instructions.)*

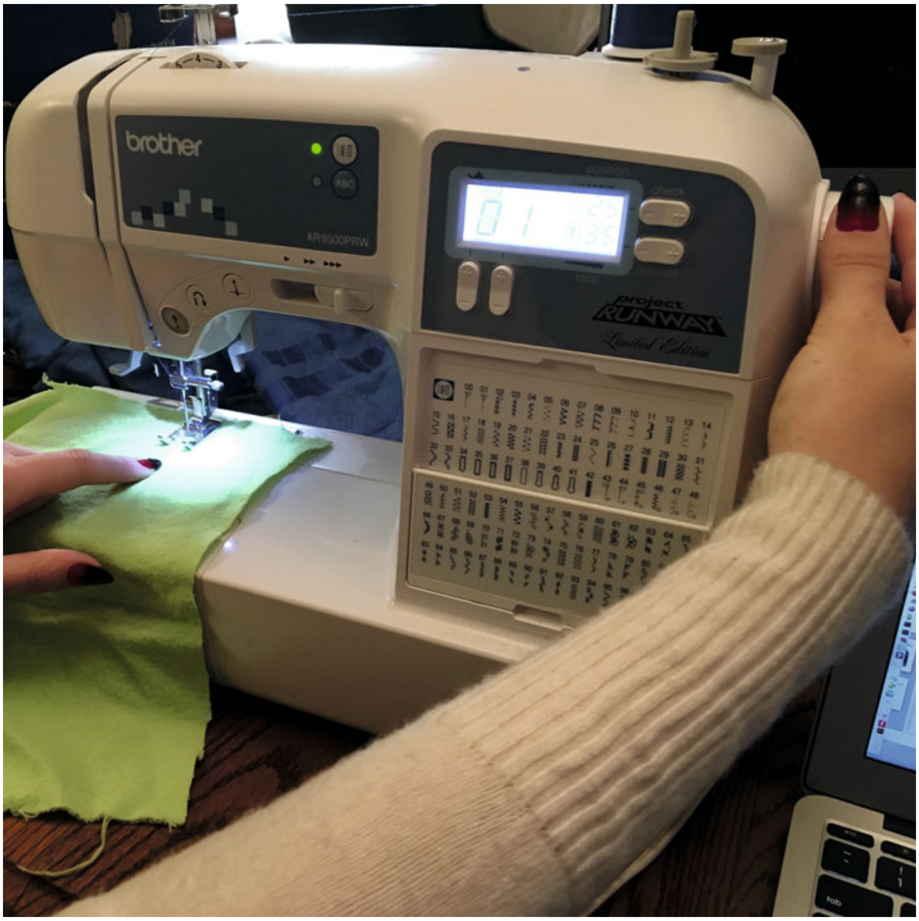


Figure 3-12. *A 2015 Brother automated sewing machine*

Sewing machines have several types of stitches available by turning a knob or pressing a button, depending on the machine. They are, most commonly, straight, zigzag, and embroidery stitches. You can adjust the length and width of them with a dial or computer-screen menu. The more elaborate machines have even more stitches available. The machine can also go in reverse with a button or switch.

If you have a new machine, it came with a manual that explains the steps necessary to thread it (if you can't find the manual, it may be available online). Many machines also show the path of the thread on the machine itself in some way (Figure 3-13).

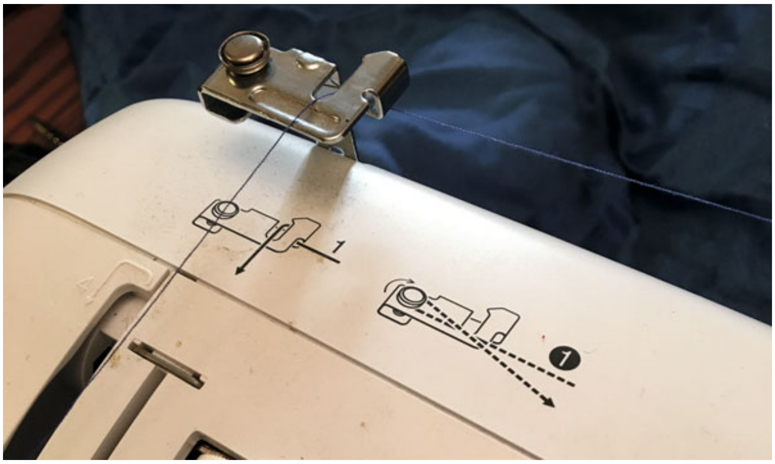


Figure 3-13. *The threading diagram on the top of the machine in Figure 3-12*

Threading a sewing machine can be confusing if you've never attempted it. But once you learn how, it will take only a few seconds on most standard machines, and you'll be ready to begin sewing. You must be able to see the small eye of the needle and have fairly good aim, so get your glasses if you need them to see up close. Many machines have a needle threader attached, which can help with the issues of fuzzy eyesight or shaky hands.

■ **Caution** The eye of a sewing machine needle is at the pointy end, and the eye of a hand sewing needle is at the top. Do not try to use the wrong kind for your purpose.

You will need some test fabric, a spool of thread (as with hand sewing, use a contrasting color to that of your fabric if you want to practice and easily see the stitches). You will also need sewing machine needles and a threaded sewing *bobbin*, a second spool of thread that feeds the stitches on the bottom of the fabric. On most sewing machines you fill the bobbin with thread before threading the top part of the machine. Treat the following instructions (which use two machines examples so you can see some typical variation) as guidelines, and consult your manual for details.

Winding the Bobbin

Using the spool of thread you have chosen, fill the bobbin so you have the same color of thread on the top and bottom of the stitch. If you sew other projects, you will create a collection of bobbins with various colors of thread so that you do not need to start over each time.

To get started, hold the hand wheel with your left hand and turn the inner knob with your right to loosen it. Place the spool of thread on the spool pin at the top of your machine. If your machine does not have an inner wheel, it may have a release button on top instead. Insert the thread through the hole in the bobbin from the inside to the outside and place the bobbin on the bobbin winder spindle. Push the spindle to the right. Hold the end of the thread and depress the foot control slowly to wind a few inches of thread around the bobbin. Figure 3-14 shows this point in the process.

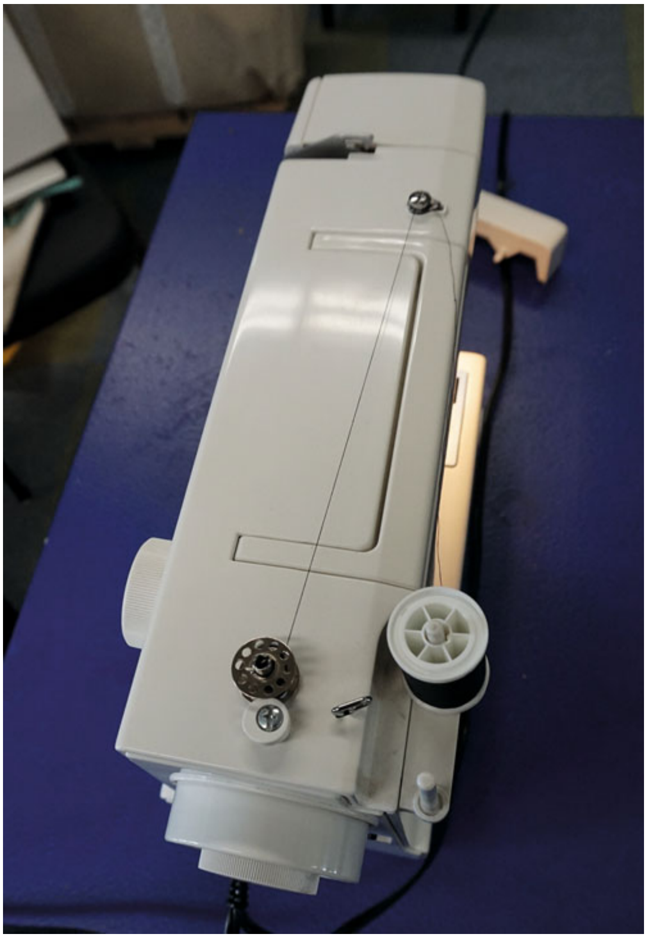


Figure 3-14. *Bobbin-winding*

Next, stop the machine and cut the end of the thread close to the hole in the top of the bobbin. Depress the foot control again until the bobbin is full. Stop the machine and return the bobbin winder spindle to its original position by moving it to the left. Cut the thread. The equivalent spool holder for the modern machine is just ahead of the main spool of thread in Figure 3-12.

Installing the Needle

Check the needle to see if it is straight by laying it flat side down on a flat surface. The gap between the needle and the surface should be consistent and the needle should be sharp. To install a needle, raise the needle holder assembly to its highest position by turning the handwheel toward you (counterclockwise), and lowering the *presser foot,* the foot that holds down the fabric around the needle. Your machine will have a mechanism to release it, such as a lever near the foot. Figure 3-15 shows this step (in a machine that is already threaded on top).

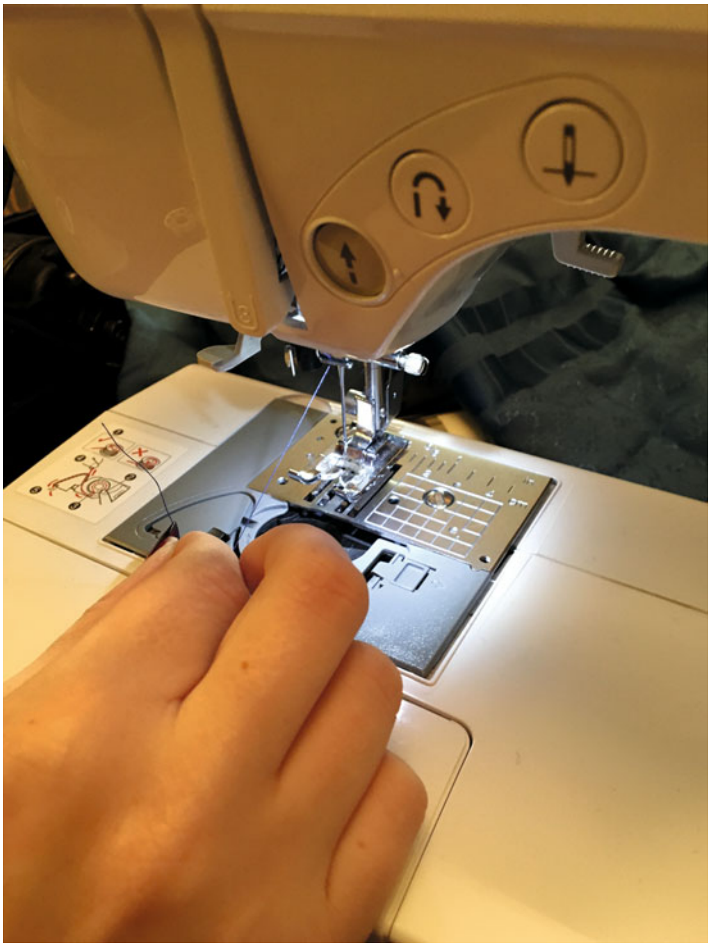


Figure 3-15. *The presser foot*

Loosen the needle clamp screw and insert a needle into the needle clamp with the flat side away from you. Be sure to push the needle up as far as it will go. Tighten the needle with the mechanism your machine has for the purpose.

Threading the Top of the Machine

Raise the thread take-up lever to its highest position by turning the hand wheel toward you (counterclockwise). Make sure the presser foot is in the up position. Place the spool of thread on the spool pin at the top of your machine. If the spindle sits horizontally, secure the spool of thread with the cap provided. Snip the end of the thread with sharp scissors to make a clean end. Take the thread end and pass it through the top thread guide. Draw the end of the thread around the upper thread guide (Figure 3-16).

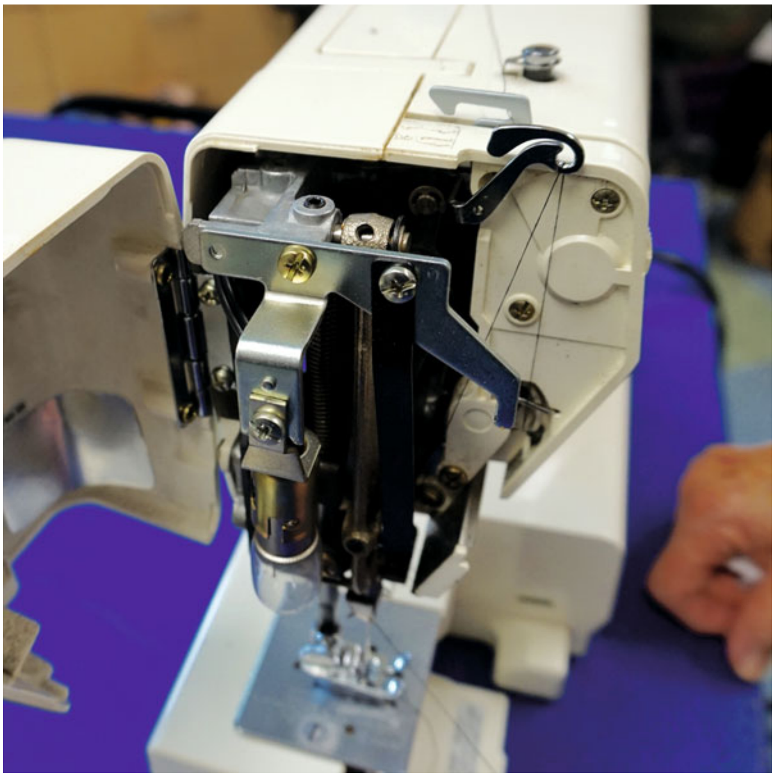


Figure 3-16. *The upper thread guide*

Holding the thread near the spool, draw the end of the thread down around the check spring holder or tension assembly. The tension assembly is located on the top of the machine in Figure 3-16 and controls the flow of thread. Firmly draw the thread up and then follow the numbers or diagram to go through any other clips your machine may have. In some cases, like the machine in Figure 3-16, the area around the thread guides can be opened to give you better access.

Draw the thread down through the remaining thread guides on your machine and slide it behind the needle bar guide—a clip just above the needle, barely visible in Figure 3-17. Thread the needle from front to back and pull a few inches through the eye of the needle and to your left. (Yours may be right to left—see diagram on your machine.)

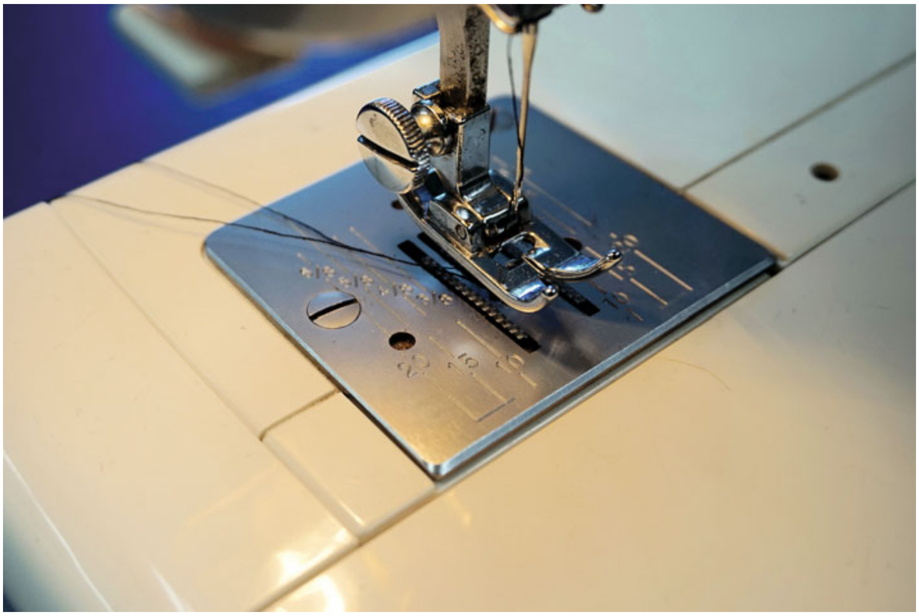


Figure 3-17. *The threaded needle*

Inserting the Bobbin

On a more modern machine, follow the diagram on the machine for inserting the bobbin thread, with the bobbin in place in its cavity under the needle (Figure 3-18). On some machines, the cavity may be tucked onto the front or side of the bottom (Figure 3-19). Machines vary in their arrangement, but you can get the general idea from these two examples.

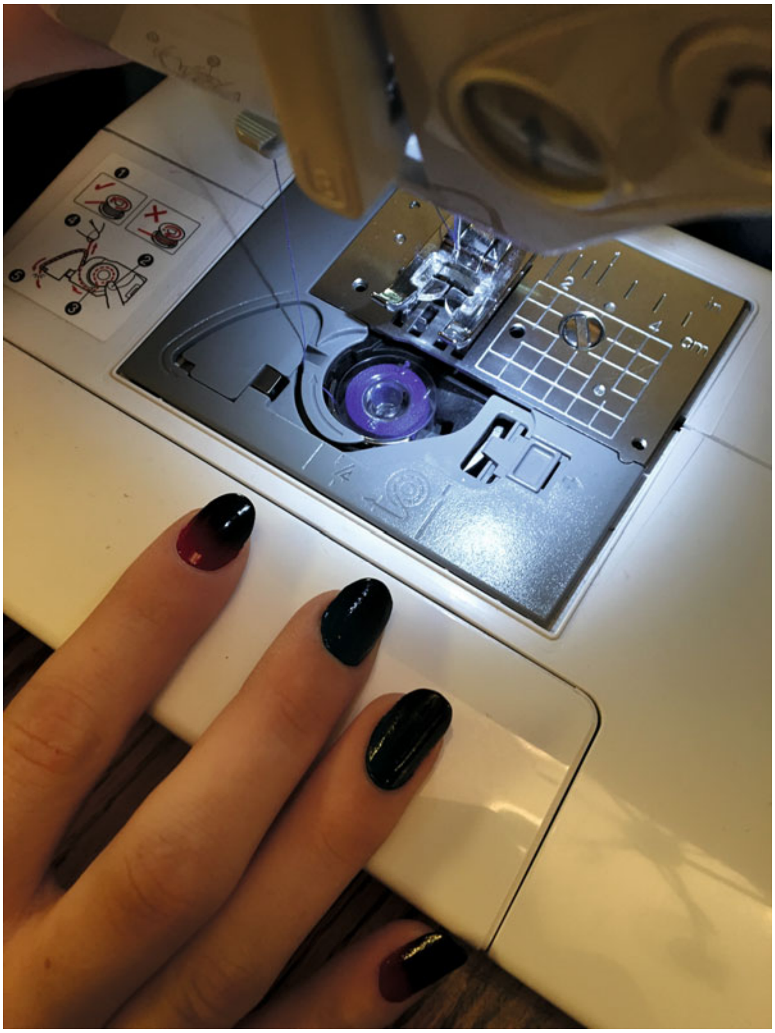


Figure 3-18. *Bobbin threading directions on one modern machine*

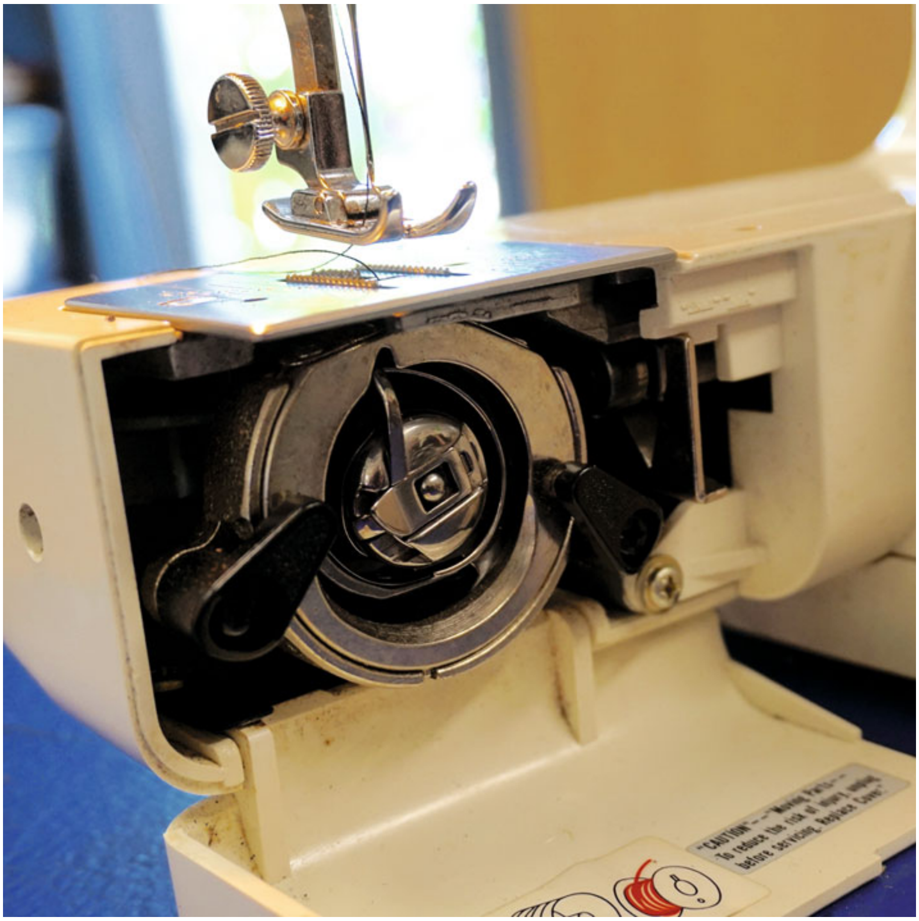


Figure 3-19. *Bobbin placement on front, near the bottom, on a differently designed machine*

Place the full bobbin in the bobbin case inside the cavity, making sure the thread feeds in the direction specified by the diagram for your machine.

Holding the latch on the bobbin case, push the case into position on the machine and release the latch. The bobbin case should lock into place when you release the latch. A few inches of bobbin thread should be pulled out.

With the presser foot in the up position, hold the thread from the needle lightly with your left hand. Turn the hand wheel toward you (counterclockwise) for one complete turn, until the needle disappears into the bobbin case. Keep holding onto the thread and move the handwheel until the needle is once again at its highest position. As the needle rises, a loop of bobbin thread will come up as well. Pull the thread out toward your left to draw the bobbin thread loop further out of the bobbin case. Pull both threads 4 to 6 inches away from you, pulling both the thread from the needle and from the bobbin straight back away from you. You can see a threaded needle at this point in the process back in Figure 3-17. You are (finally) ready to sew!

Trying Out Machine Sewing

It is a good idea to practice using the machine before you begin any project. With the list of basic stitches that follows and some scraps of fabric, practice putting the fabric under the presser foot, lowering the needle by hand-turning the wheel, lowering the presser foot, (with the lever you used to raise it before) sewing, changing stitches, using reverse, removing the fabric, and snipping the threads. Most sewing machines have a thread cutter next to the needle—or use sharp scissors.

To remove the needle from the fabric, turn the knob toward you until the needle is in the up position and out of the fabric, raise the presser foot, and pull the fabric out from under the foot. Leave a 4- to 6-inch trail of thread for next time.

Changing Seam Direction

Some projects will require changing the direction of a seam. To do so, make a seam a few inches long. Stop the machine with the needle in the fabric (lower it by hand if it is up). Lift the presser foot, turn the fabric 90 degrees, and lower the presser foot. Continue sewing for a few inches (Figure 3-20). Practice pinning fabric together and creating seams. Cut a curve and sew a seam along it, and practice clipping the curve. *Clipping* means to make little notch cuts from the edge of the fabric to the seam to make the seam lay flatter.

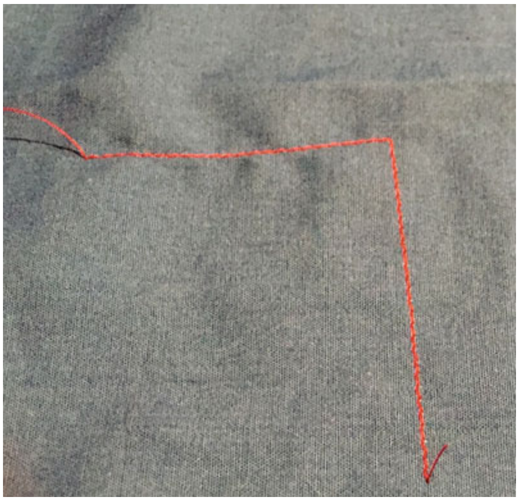


Figure 3-20. *Changing seam direction*

Trying Out Types of Stitches

If your machine has a variety of stitches available, go ahead and try them all. Some stitches may require different presser feet and/or needles.

Double or Stay Stitch

After sewing the stitch on the seam line, sew it again 1/4 inch inside away from the first stitch in the seam allowance using a straight or zigzag stitch (Figure 3-21).

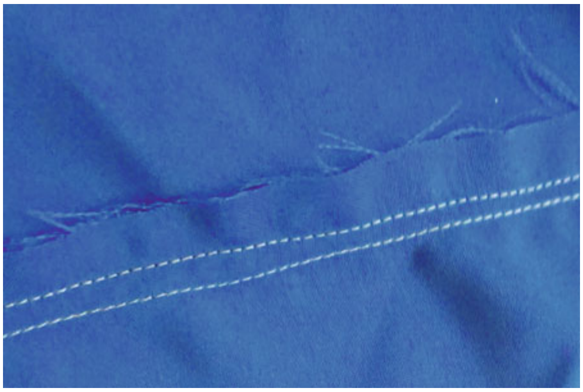


Figure 3-21. *Stay stitch*

Ease or Gather Stitch

Sew along the seamline using long straight stitches, leaving 3 to 4 inches of thread at each end of the seam. Pull the thread ends to adjust for fit (Figure 3-22).

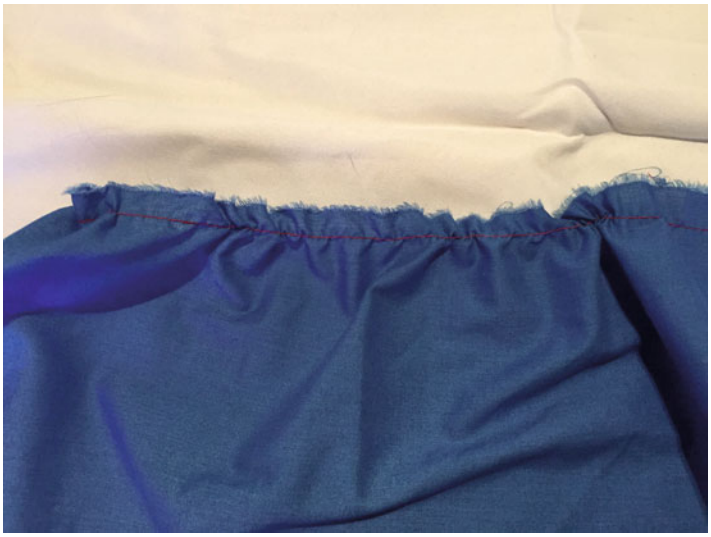


Figure 3-22. *Ease (or gather) stitch*

Edge Stitch

Stitch close to the edge of the fabric and cover the edge with your stitching (Figure 3-23).

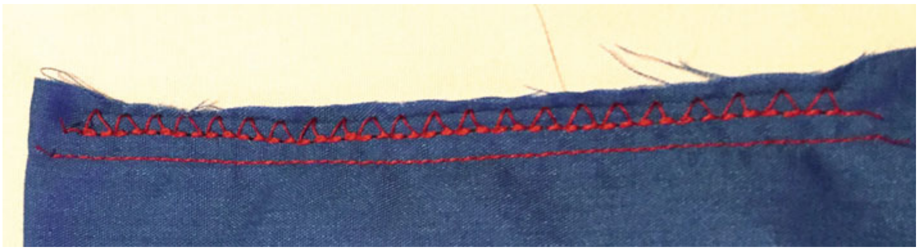


Figure 3-23. *Edge stitch*

Top Stitch

On the right or outside of the fabric, stitch very close to the seam. Use the presser foot as a guide (Figure 3-24).

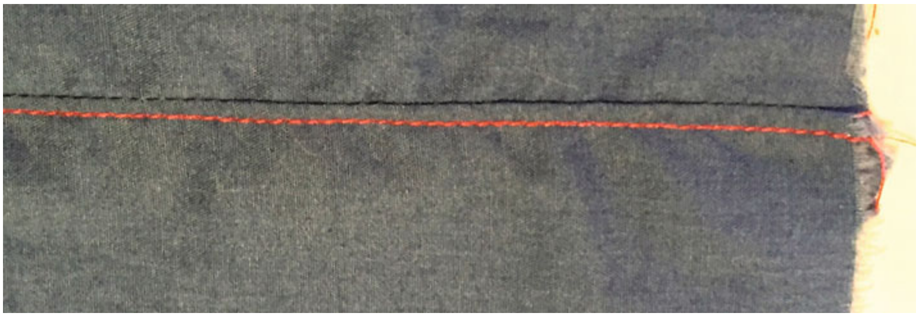


Figure 3-24. *Top stitch*

Back Stitch

After sewing a seam to complete a piece, stop the machine, put the machine in reverse, and make a few stitches along the same seam line. Then go forward again to the end of the piece. Figure 3-25 shows the "reverse" button on a machine—typically it is a button or lever like the one shown. (It is the button on the right with the arrow turning around "over the top.")

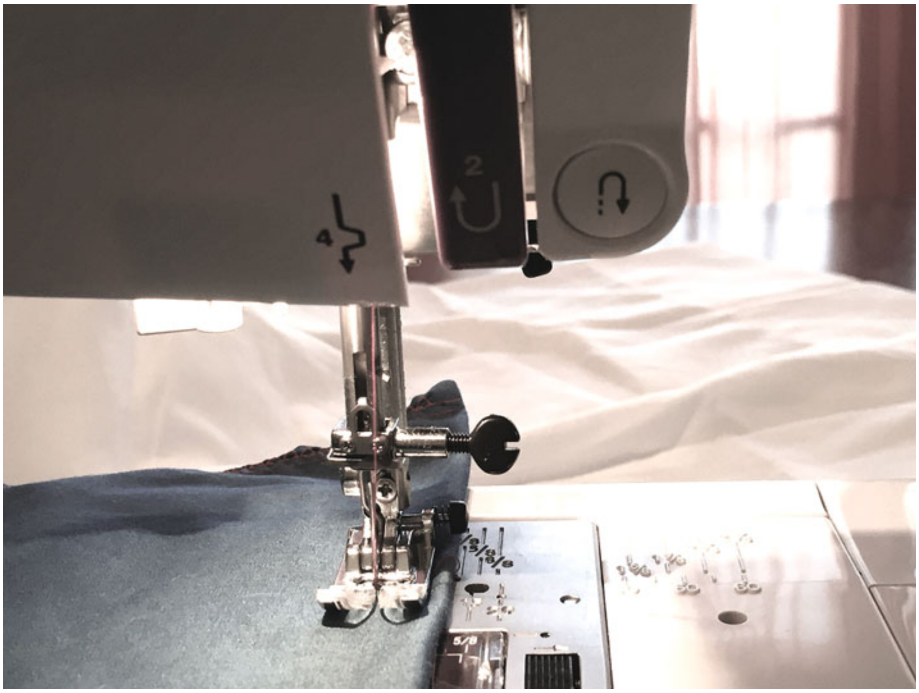


Figure 3-25. *The reverse button (on top right)*

There are many more types of stitches. The best way to see them in context is to take a pattern and try actually making something.

Stitches in Context

Figure 3-26 is a sketch of a vest you will make in Chapter 4. You can see where I will use some of the stitches. Even something as simple as a vest has a few pieces that are not obvious and that require a variety of stitches.

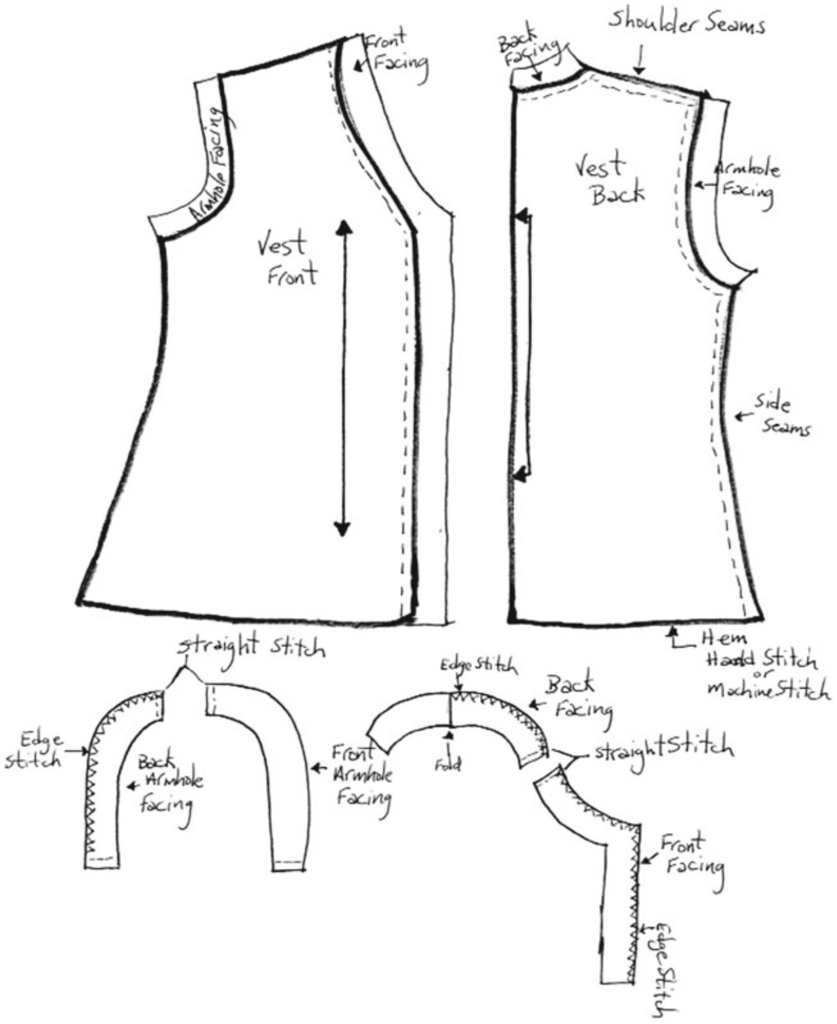


Figure 3-26. *Stitches for the vest in Chapter 4*

In Chapter 4 you will see more of this stitching in action. Most of the time there is no “right” way to stitch something. You can try different things to see what will work. For example, in Chapter 4 we topstitched a few places just to show the effect (not shown here).

Summary

This chapter introduced the basics of sewing by hand and with a standard sewing machine. It also discussed what you will need to begin to sew, showed several common stitches, and talked about how to thread a standard sewing machine and common sewing terms. It will be helpful to practice sewing straight stitches, sewing a curve, edge stitching, and finishing a seam on scraps of fabric before you begin your first project in the next chapter.

CHAPTER 4

Making and Using Sewing Patterns

This chapter talks you through the ins and outs of sewing patterns. This is Lyn writing again, now taking you into using the stitches and techniques introduced in Chapter 3.

The first time you use a standard sewing pattern, it is important to read and follow the instructions that come with it, and to understand and follow the markings printed on the pattern pieces. As with a new cooking recipe, it is best to read the steps through to the end before beginning your work. Commercial patterns come printed on tissue paper and stuffed into an envelope. A pattern envelope is like a magic hat full of scarves in that it seems like you keep pulling out piece after piece—so many that they could not possibly have fit in that small envelope. I can never seem to get all the pieces back in neatly after using the pattern.

There is usually a section in the instructions explaining fabric cutting layouts and symbols. Again, like a recipe, once you understand the steps, you can adjust and/or alter them to suit your style and abilities. Pinning all the pattern pieces you will need onto the fabric first and double-checking all directions and symbols before cutting will help prevent some mistakes.

You can cut out the smaller pieces first in order to make sure you are cutting on the correct lines and cutting out the proper notches. Those are easier to recut if you make a mistake, and you will not waste a large piece of fabric.

In this chapter we will create a simple, unlined vest (Figure 4-1). It does not require a lot of fabric and has a basic shape that is fairly easy to sew and finish. It is possible to sew the entire vest by hand, but sewing seams with a machine is much quicker. You may want to hem the vest by hand if you need some more practice hand sewing.

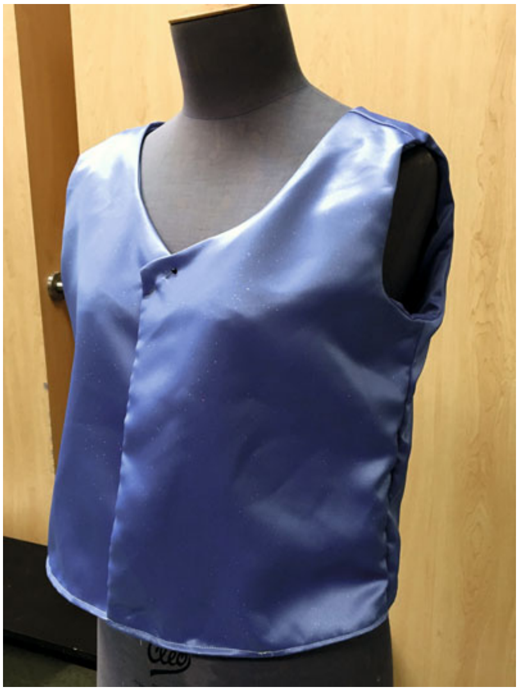


Figure 4-1. *The finished version of the vest we will create in this chapter*

Measuring

The first step in using or creating a pattern is to decide what it is you are going to make. After that you will need to measure the person who will wear it (if it is clothing), the piece of furniture it will cover, or whatever else it needs to fit. In the case of a person, use a fabric measuring tape because it is flexible and will go around the curves of a body. Here are some tips on how to measure various body dimensions:

- *Bust (female)*: Wrap the tape around the bust and back at the nipple line. Make sure it is even all the way around the body. If you are measuring yourself, a mirror is handy.
- *Chest (male)*: Wrap the tape around the front and back directly underneath the nipple, making sure it is even all the way around the body.
- *Waist*: Use the narrowest place on the torso and wrap the tape around the body about an inch above the navel. Make sure it is even all the way around. Note that this may not be where you wear your pants or skirts. Today's styles are worn much lower than the traditional waistline. But this is where the waist measurement is taken regardless of the style.

- *Hips*: Use the fullest part of the buttocks and wrap the tape around the front and back. Make sure it is even all the way around.
- *Crotch*: Hold the end of the measuring tape at the navel; pull the tape between the legs and up to the back even with the navel.
- *Thighs*: Wrap the tape around the fullest part of the thigh, keeping it even.
- *Calves*: Wrap the tape around the fullest part of the calf, keeping it even.
- *Upper arm*: Wrap around the fullest part of the bicep.
- *Wrist*: Wrap around the wrist bones.
- *Length of arm*: Holding the arm straight out to the side, place the end of the tape at the front of the armpit, hold it there, and measure to the wrist.
- *Center front (torso)*: Hold the end of the tape at the notch between the clavicles (top of the sternum), and measure down to the navel.
- *Center back (torso)*: Hold the end of the tape on the spine, just below the neck, and measure down to the level of the navel on the back.

■ **Note** If you prefer, you can also measure with string. To do this, wrap a piece of string or ribbon around the part you are measuring, hold onto or mark the place, and then measure the string with a ruler or yardstick. Some people have also used tape (layered on over an old t-shirt or plastic wrap, then the whole thing cut off) to create a custom dress form. None of the authors has tried the latter process, but you can find descriptions and pictures of it online.

Choosing a Pattern

If this were a class just in learning how to sew, I would probably not talk about making a pattern from scratch. However, in a wearable tech project, it is more likely that you will be using a mix of fabric, 3D-printed pieces, and electronics and so will need to make your own pattern or modify an existing one. I will deconstruct how patterns work here so that you can make a choice about whether you buy or make a pattern for your project.

Buying a Pattern

Sewing patterns are available online, in fabric stores, and on many websites such as `www.mccall.com`, `sewingpatterns.com`, `www.craftsy.com`, and `Etsy.com`. Joann Fabrics (`www.joann.com`) is a national store in the United States and has an online store as well. There may be local fabric stores in your area that stock patterns.

■ **Note** By and large, patterns are still sold as physical envelopes full of tissue paper (as opposed to being made available for download) because of the practical issues of printing out a person-sized pattern on many pieces of taped-together paper. Unfortunately, this means that it is difficult to get an electronic pattern suitable for laser cutting, if you wanted to use a laser cutter to cut out your fabric. I talk about ways around that later in this chapter.

Most patterns have a level of difficulty labeled on the pattern envelope. Choosing a basic or beginner level pattern is best for a first project. By choosing a basic pattern you will learn terminology and basic skills without the frustration of complicated techniques and instructions. Patterns are printed with a *cutting line* and a 5/8-inch (1.6 cm) *sewing margin* or *seam allowance* line. Make sure you understand which is which, or your completed project will not fit properly (Figure 4-2). Note that in Chapter 3 and other places where I created a pattern from scratch, I used 1-inch margins just to give more room for error. Printed patterns usually have 5/8 inch though.

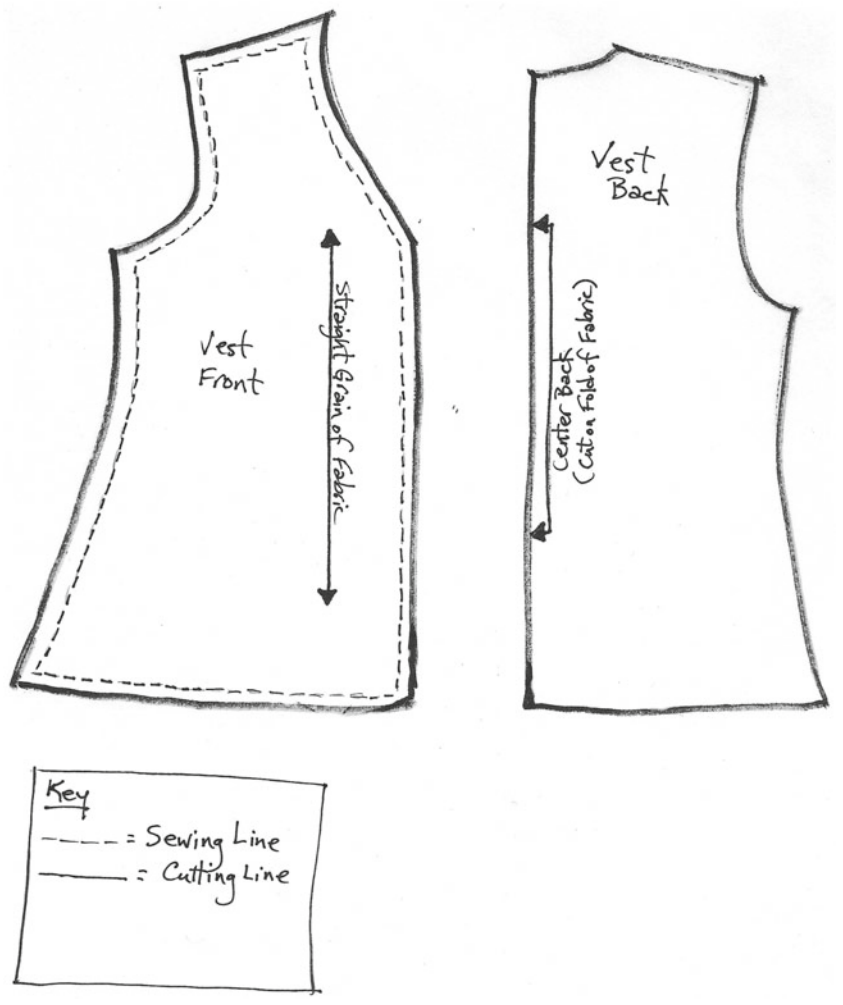


Figure 4-2. *Sewing and cutting lines on a purchased pattern*

Commercial patterns usually are made so that several sizes of garment can be made from the same pattern. Different lines on the pattern (Figure 4-3) show where you would cut for several different sizes. In this case, the pattern covers sizes 6 through 10.

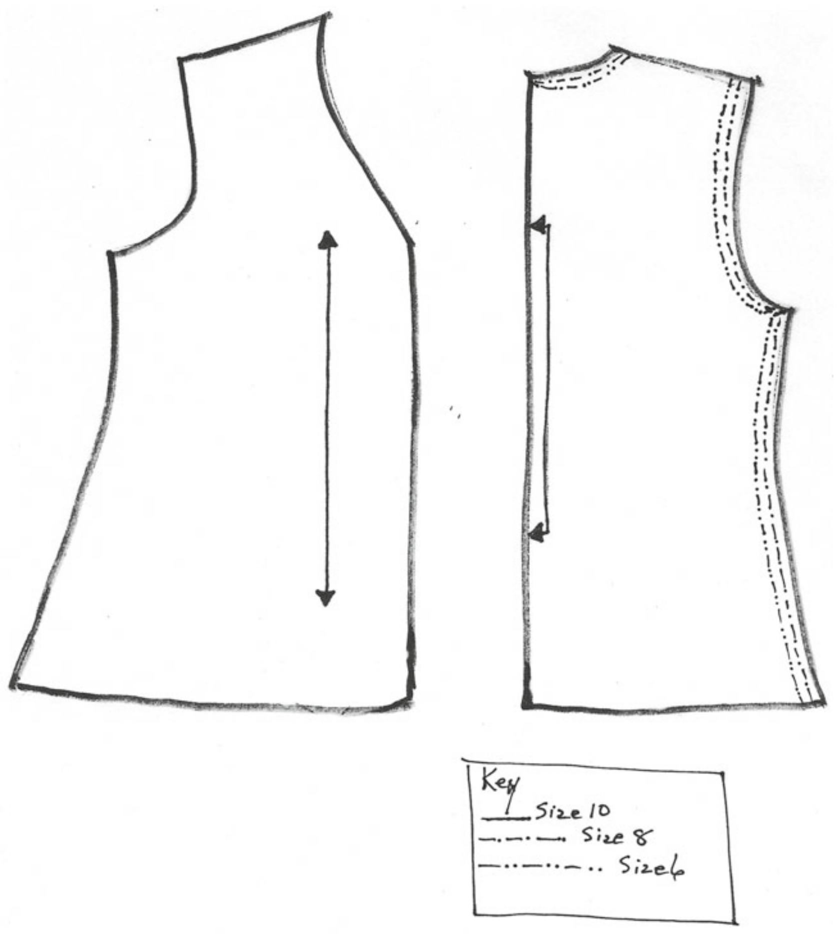


Figure 4-3. *Sizing lines*

■ **Tip** You may want to cut the pattern pieces out in the size you will be using before you pin it to the fabric. This can help eliminate the chance that you will cut on the wrong lines and have a resulting garment that will not fit.

Making a Pattern

It may turn out that either you are inventing something that does not exist yet, or you just can't find a pattern you like. In that case, there are many costume books and websites that include simple pattern drawings and instructions for enlarging the pattern. Some that I like include Barbara and Cletus Anderson's book *Costume Design,* 2nd Edition (Harcourt

Brace College Publishers, 1999); Shirley Dearing's *Elegant Frugal Costumes* (Meriwether Publishing, 1992); Sheila Jackson's *Costumes for the Stage,* Second Edition (New Amsterdam Books, 2001); and Nancy Bradfield's *Costume in Detail 1730–1930* (Costume and Fashion Press, an imprint of Quite Specific Media Group, 1997). I am also a fan of the websites `http://sewing.craftgossip.com` and `www.wikihow.com/sew-using-patterns`.

Another way to create a pattern is to carefully deconstruct a piece of clothing. You can reuse older garments that fit as a pattern for a new piece. Choose a piece you would like to duplicate and use a seam ripper to undo the seams. It is probably best to use something that is old and worn, or that you no longer want. Buying something from a secondhand store just for this purpose is also an option. Use the pieces as a pattern, or use them to make a paper pattern.

After you learn how the garment is put together, you may want to alter parts for other costumes—for example, making shorter or longer skirts, dresses, or pants, or adding longer or blousier sleeves or different collars.

Pattern tissue can be found online and in fabric stores, but you may also use butcher paper, brown paper bags (as I did for the example in this chapter), or any other paper that you can draw on, pin to the fabric, and cut out. Newspaper is not recommended because the ink smears and it is difficult to see any markings you make.

■ **Note** As laser cutters become more common in public makerspaces, you may want to create a pattern that you can use on one of those machines. Laser cutters can create very complex shapes very easily. For example, the Adobe Illustrator software package (`www.adobe.com/products/illustrator.html`) is a good tool for creating electronic sewing patterns, although it has a significant learning curve. You will want to create a file with 2D *vector* graphics (as opposed to *raster* graphics) to be able to cut on a laser cutter.

Because readily available laser cutters are quite new, this is an area that will probably expand significantly in the near future (try searching on "digital sewing pattern"). Note, though, that the electronic patterns intended for downloading, printing out in pieces, and sewing on a sewing machine likely will not be useful for laser cutting. Even if the format is correct, some work on the file will be needed to cut the correct lines.

Basic Pattern Shapes and Pieces

Pants and shirts are created from a few basic shapes. Pants can be made with one pattern piece, cutting it out of two pieces of fabric and adding elastic to the waist or creating a waistband. It can be upgraded by adding a placket for a zipper, hook and loop closures, buttons, or snaps.

Shirts can be made with as little as three pattern pieces—front, back, and sleeves. A more elaborate shirt or blouse will have collar, cuff, and button tabs or zippers.

Choosing Fabric

The back of the pattern envelope will list recommended fabrics for the best results. It is best to choose a fabric from these to ensure that your project fits and hangs properly on your body. For the vest project, do not use stretch knits. There will also be a list of notions needed: thread, zippers, buttons, snaps, and so on.

■ **Tip** Most fabric has *sizing* in or on it when you purchase it. *Sizing* is a solution that is used to keep fabric looking crisp and unwrinkled on the bolt. I recommend washing and drying the fabric, in whatever temperature you will normally wash the garment after it is completed, before cutting out the pattern. This will preshrink the fabric and eliminate the chance of the completed garment not fitting properly. Of course, if you are using a fabric that should not be washed, don't wash it. Dry clean it when necessary.

Figuring Out How Much Fabric You Need

Patterns come with an estimate of how much fabric to buy. If you are creating your own pattern, you can determine how much to buy by first measuring the pattern pieces you are going to use, or the body or object you intend to cover. Fabric comes in different widths. In the United States, the most common widths are 40 and 60 inches wide. Remember all the pattern pieces need to run in the same direction. If you cut things out all over the place just because they fit on the fabric and you want to use as little fabric as possible, the garment or item will look odd and not fit very well. If the fabric has a stripe or pattern you want to match on the front or back, you will need a bit extra to be able to match it up evenly.

Using a Pattern

Once you have a pattern and fabric and are ready to go, you will need to lay the pattern out on the fabric and then cut it. This section talks about that in general, and you can see more photos in the vest creation example later on.

Laying Out the Pattern

Lay the fabric out on a table, cutting board, or floor. It should be folded with the *selvage* (finished, not the cut) edges together (Figure 4-4). In Figure 4-4 the selvage edge runs along the bottom of the picture. You can see where I folded it back so that the right side of the fabric is folded inward. The demonstration fabric has some sparkly bits on the right side and a matte finish on the wrong side (inside of the garment).

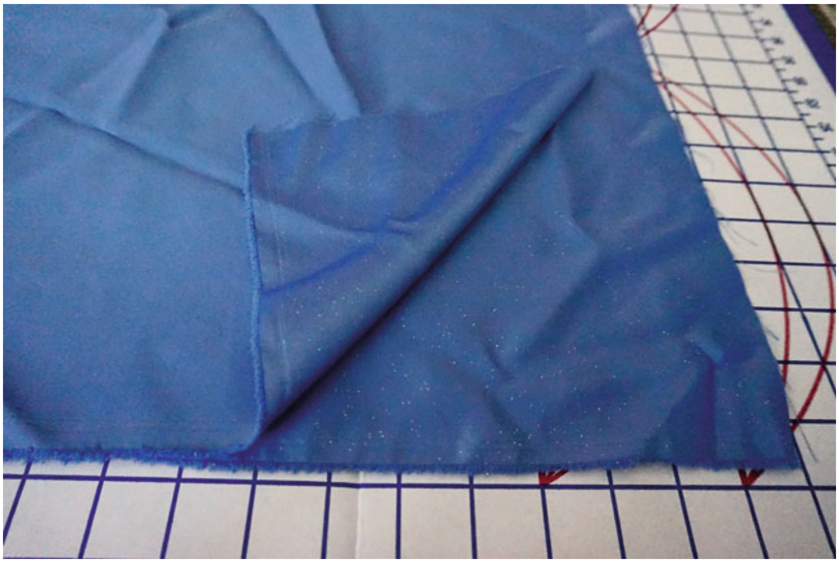


Figure 4-4. *Folded fabric, ready to cut—selvage on bottom, cut edge on the right*

■ **Caution** If the fabric has a *nap* (like corduroy or velvet), the arrows should be parallel to the nap, not horizontally across it. The same goes for a fabric with a strong pattern. Normally the pattern or nap would be parallel to the selvage anyway.

You will be cutting through two layers of fabric. If there is a piece that you only need one of, it will be on the fold of the fabric, and the pattern piece will only represent half of the finished piece. For example, the back of the vest we will make later on in this chapter lies on the fold. Others will need to be placed so that you are making two of them. Figure 4-5 shows the vest pattern that I made by drawing freehand on cut-up grocery bags as it appears pinned on the fabric. The fabric is laid out on a cardboard cutting board, which in this case we are going to use on the floor. If you have a table that is big enough, you may find it more convenient to put it there.

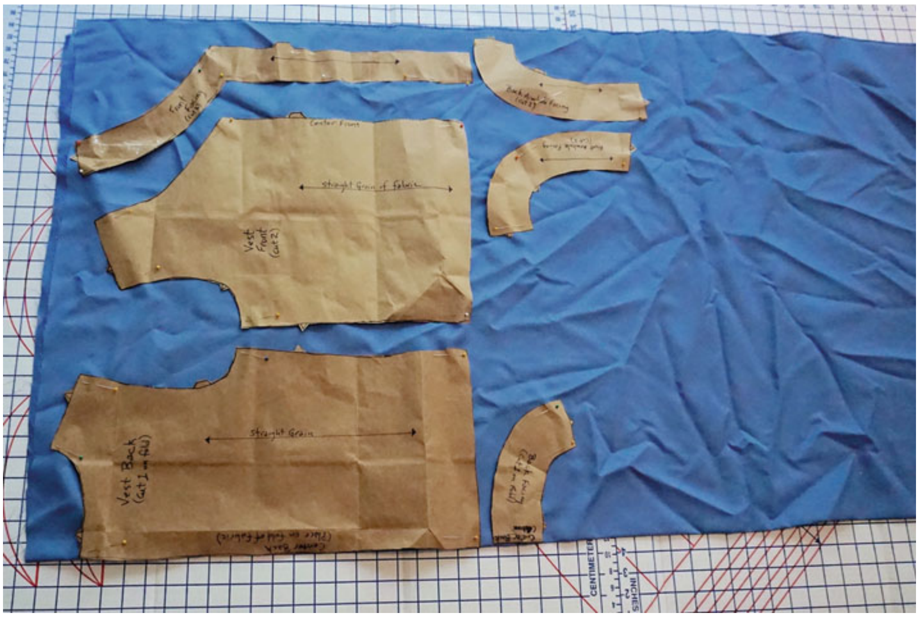


Figure 4-5. *The vest pattern (on a fabric without a nap) pinned on fabric on a cutting board. The fold is at the bottom, selvage on top.*

I did not try to optimize laying out pieces all that much here; if you have expensive fabric and you sew a lot of clothes, it can be worth it to play around with the pieces a bit before pinning. That way you can have as much fabric as possible left over for other projects.

You will notice that there are a lot of pieces on the fabric that do not have immediately obvious uses. These are *facing* pieces—used to finish off armholes, necklines, and other pieces instead of just hemming them. In this case, there is one facing piece (for the back of neckline) on the fold, and three that will be created with two copies apiece.

There are arrows on the pattern pieces indicating the *straight grain* of the fabric (see the arrow on the sketched pattern in Figure 4-2 and again in Figure 4-5). The straight grain runs parallel with the selvage edge.

Patterns are made with *notches* (pieces that stick out of the cutting line of the garment, as in Figure 4-6) to help you assemble the garment and avoid getting confused. I drew the pattern we will be using in this chapter freehand based on experience, but even so, I put in a few notches of different shapes to help us keep track of what goes where. It can otherwise be very easy to assemble things partially inside-out or otherwise incorrectly. These notches poke out of the cutting line of the garment—some patterns will have them indented from the cutting line.

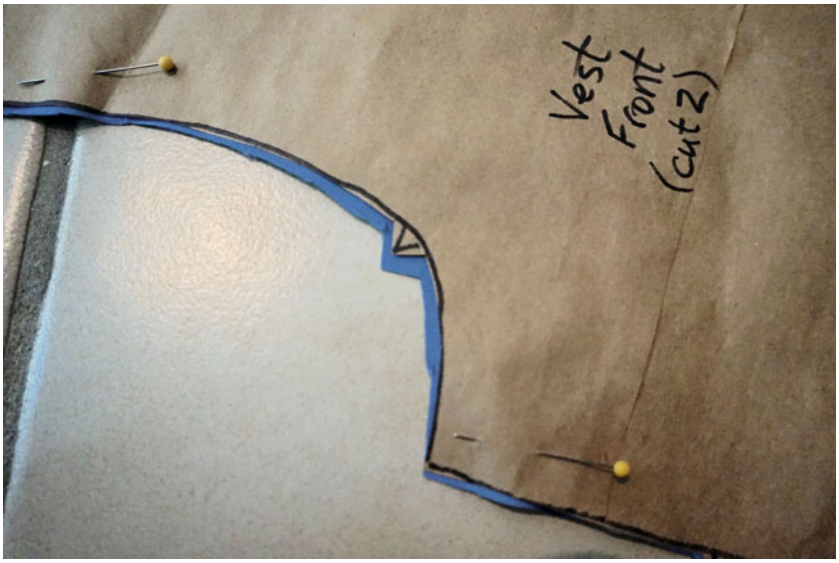


Figure 4-6. *A notch*

Pinning and Cutting the Pieces

After you are happy with the layout of the pattern pieces on the folded fabric, pin the pattern in place with straight pins. Be sure the pins do not stick out where you are going to cut.

When you are cutting, lean on the fabric with your other hand to hold it still (Figure 4-7). Otherwise (particularly with stretchy fabrics) you can find yourself cutting away from the pattern lines. This usually means you are continuously moving around as one hand maneuvers the scissors and the other holds the fabric still.

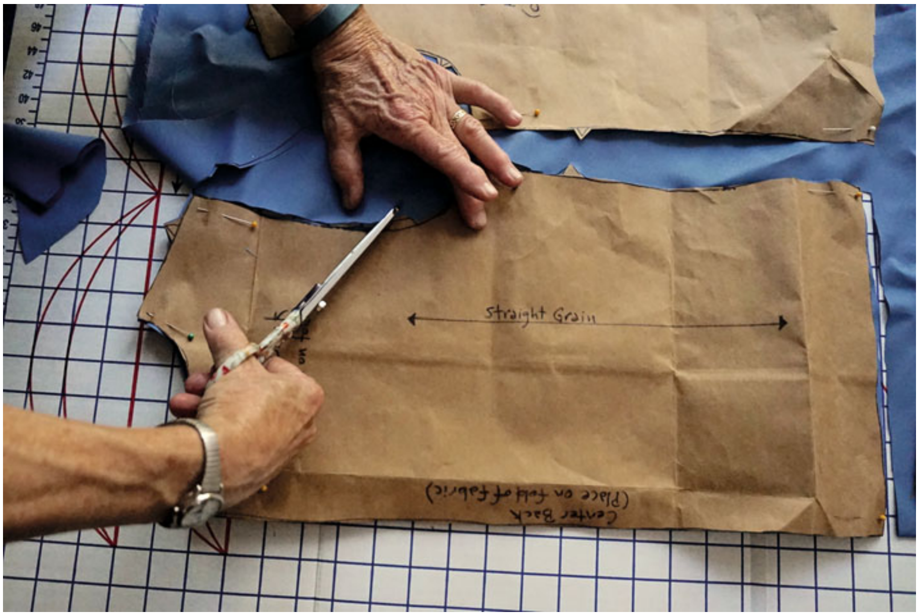


Figure 4-7. *Leaning on the fabric to cut a notch*

■ **Tip** When cutting, follow the cutting lines, not the seam allowance lines. Use long, even scissor strokes, and cut out the notches. It is a good idea to use your fabric scissors only on fabric. They become dull quickly if they are used to cut paper or other things. If you have dull scissors, it is harder to cut the fabric cleanly.

Marking the Fabric

Using tailor's chalk or a washable pen, you can transfer all markings and lines of construction you will use onto the "wrong" side of the fabric before removing the pattern. Remember, the wrong side is the back (the part that will not be visible when the garment is worn, although some of the facing will not normally be visible either). For a simple construction, like the vest example in this chapter, you would normally not worry about doing that.

Creating a Simple Vest

Now I will walk you through creating a simple, unlined vest. We are making a vest first because it is a fairly small article of clothing and requires less fabric than a pair of pants or a shirt. That means if you make a terrible mistake, you will not have ruined a large piece of expensive fabric. It involves using a standard pattern and sewing by hand and machine.

A vest can be convenient for wearables projects because it can be dropped over other clothing and is a bit easier to handle than an entire dress or complex costume. The tools required, illustrated in Figure 4-8, are as follows:

- A sewing machine
- Sewing machine needle
- Spool of thread
- Simple vest pattern (see suggestions in Tip that follows)
- Bobbin with the same thread as the spool
- Sharp scissors
- Straight pins
- Pin cushion
- Seam ripper
- Fabric
- Cutting board (optional)

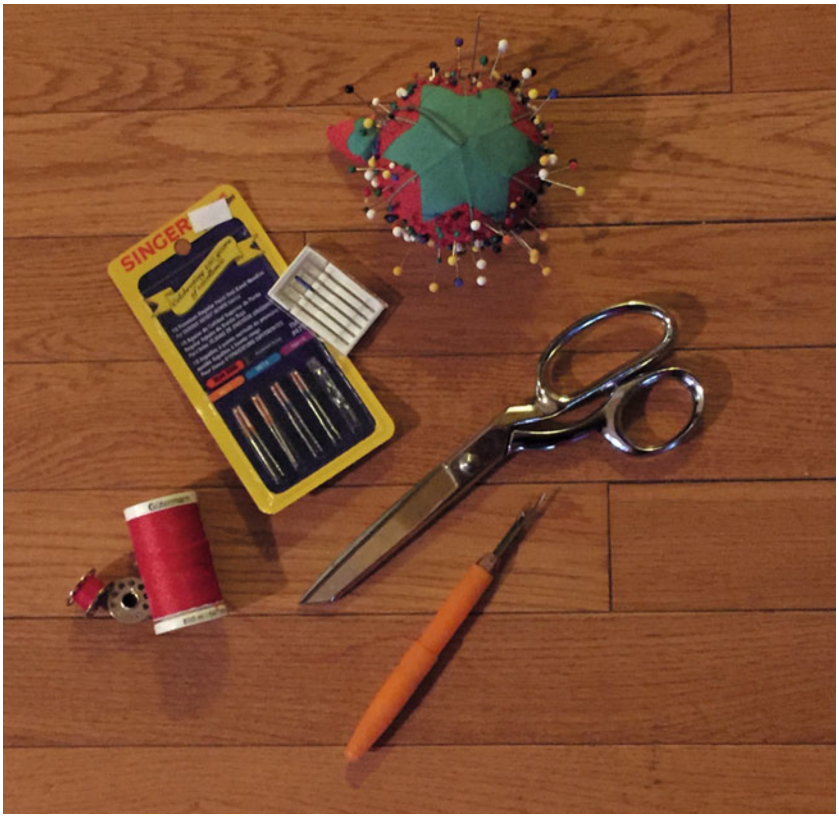


Figure 4-8. *Materials to make the vest, other than the sewing machine, fabric, pattern, and cutting board. The latter three are shown in Figure 4-5.*

■ **Tip** When using a standard sewing pattern for the first time, it is important to read and follow the instructions that come with it and to understand and follow the markings printed on the pattern pieces. The example here uses a pattern that I drew from scratch, but I suggest you buy a simple pattern for your first time out. Some basic vest patterns you could buy to follow along with these instructions are McCall's M6228 (men's and women's vests), McCall's M2260 (unlined women's vests), and Kwik Sew K3899 (easy women's vests).

This vest is a basic design, with few pieces to attach to complete it. You can make it a more advanced project by adding pockets, lapels, lining, buttons, snaps, ties, or other design elements. We show you different electronic elements you might want to add to it in later chapters.

Choosing Fabric

For your first outing here, I suggest you get some very low-cost cotton fabric since it is likely that things will go wrong and there is no point in wasting a lot of money on fancy fabric. Take a look at the discussion earlier in the chapter to see how much fabric to buy. The fabric used here is a light polyester, with a bit of sparkly stuff on the right side.

Laying Out and Cutting the Pattern

Figure 4-5 shows the overall layout of the vest pattern on the fabric. The back of the vest pattern piece will be pinned onto the fabric with the center back edge on the folded edge of the fabric (Figure 4-9).

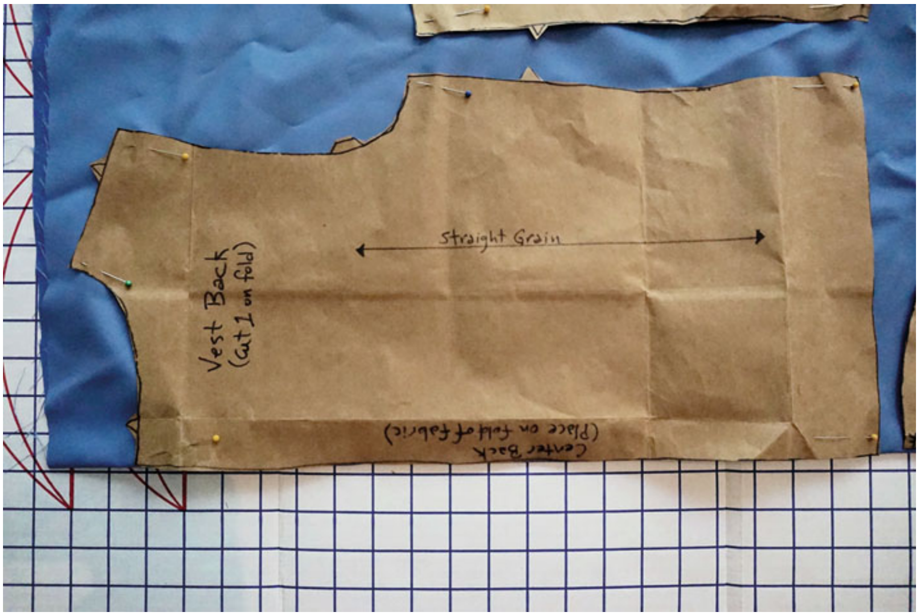


Figure 4-9. *The vest pattern back piece*

The front of the vest pattern piece will be pinned onto the fabric close to the fabric edge, not on the fold. The *facing* (internal finishing) pieces will be laid out the same way as the pieces they are finishing. One facing is on the fold, and the others away from it (again, you can see the big picture in Figure 4-5).

Sewing the Vest

After cutting out and marking your vest pattern, remove the pattern tissue from the front and back pieces of fabric. With *right* sides together, pin the front pieces to the back piece at the shoulder seams (Figure 4-10). Notice how the notches line up across these two pieces when the orientation is correct.

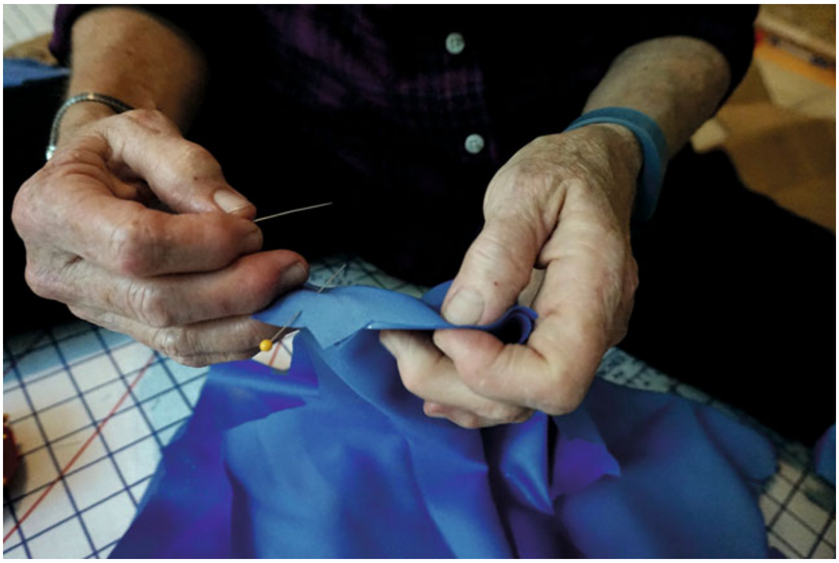


Figure 4-10. *Pinning the front of the vest to the back*

Make the Shoulders

If you have not already done so, thread your sewing machine and fill the bobbin (as described in Chapter 3). This example uses contrasting thread so you can see what I am doing, but normally you would use thread that matched your fabric. Place the shoulder seam on the sewing machine (Figure 4-11) and lower the presser foot onto the material.

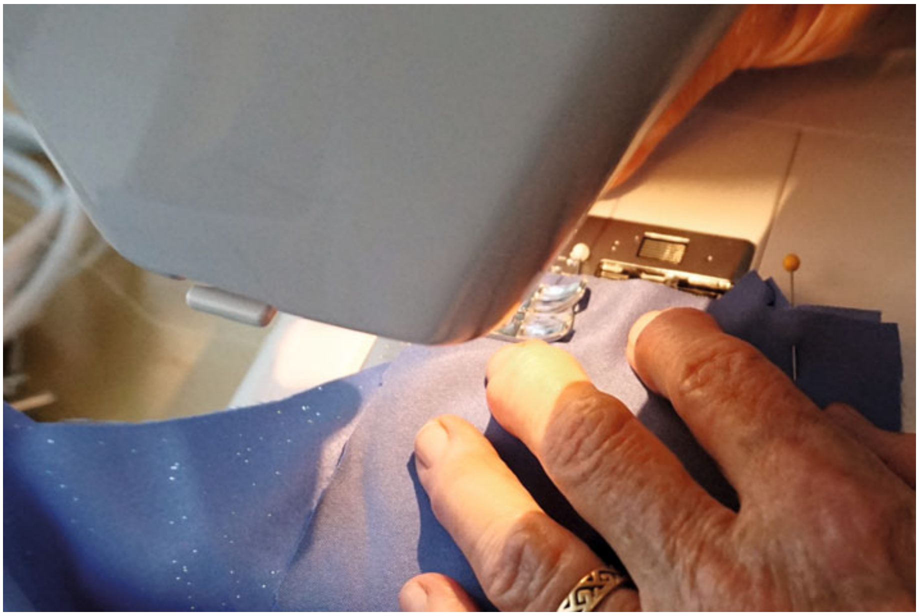


Figure 4-11. *Placing the shoulder seam on the sewing machine*

Manually (usually with hand wheel) lower the needle into the fabric before you start machine sewing. Select a short straight stitch (with short, close-together stitches) and sew along the seam line on each shoulder. Remove the pins as you sew, or after you have finished. (If you leave the pins in when sewing, be careful to sew over the pin, not onto it—that is an easy way to break the sewing machine needle.) Figure 4-12 shows the result.

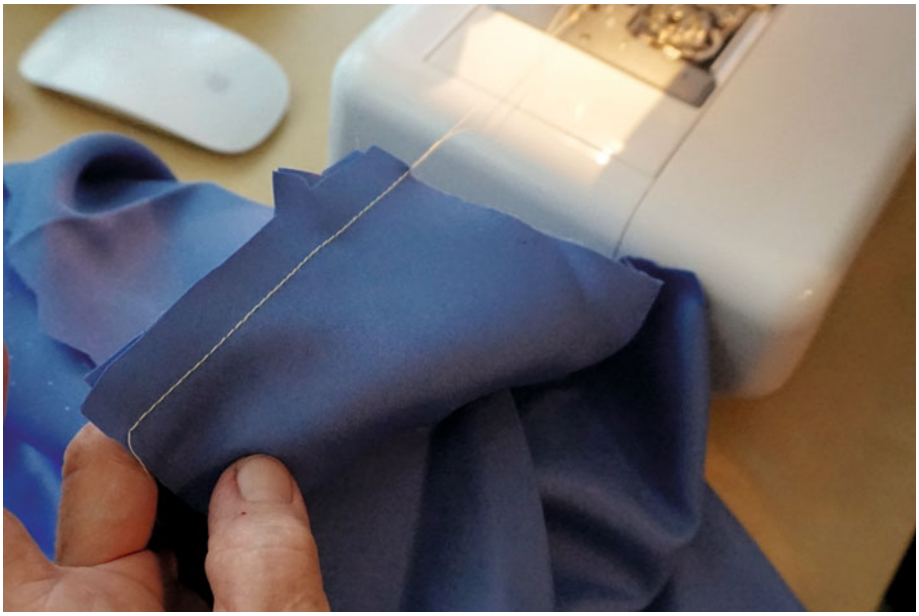


Figure 4-12. *The first pass of the shoulder seam*

■ **Caution** It is very easy to sew together more layers of fabric than you had in mind. Be sure you are keeping the rest of the vest clear as you sew along one seam.

One line of stitches really is not enough for a garment seam, though. If we just left it like this, it might unravel. To prevent that, I did another line of zigzag stitching (Figure 4-13) between the seam we just did and the cut edge of the fabric.

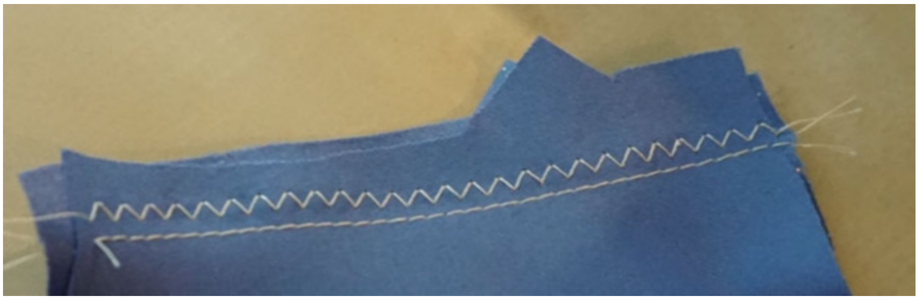


Figure 4-13. *Adding zigzag strengthening*

Finally, I trimmed the seam a bit so that it would lie flatter (Figure 4-14). Do not get carried away here. Leaving a little less margin than your zigzag width is appropriate.

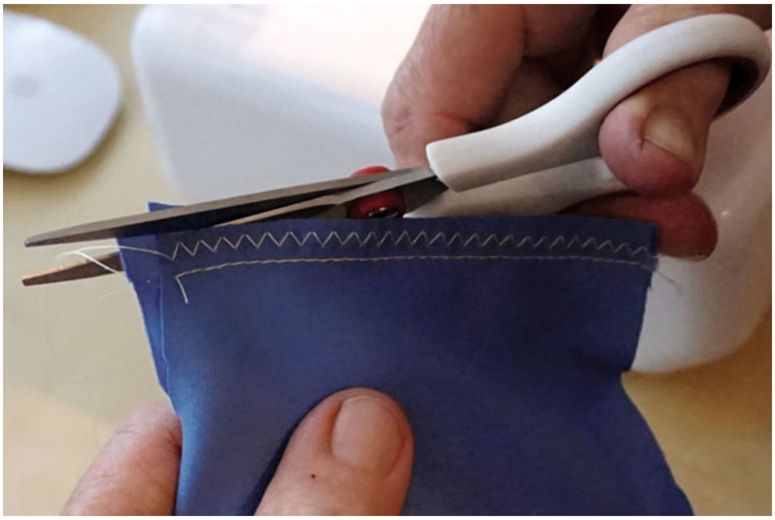


Figure 4-14. *Trimming the shoulder seam*

Make the Side Seams

To make the side seams join the back and two front pieces of the vest, pin, sew (with a single stitch), and then double stitch (two lines of straight stitching) these pieces together. Trim the seams and press them open (Figure 4-15). You can do this with an iron or just by flattening by hand as I am doing in Figure 4-15.

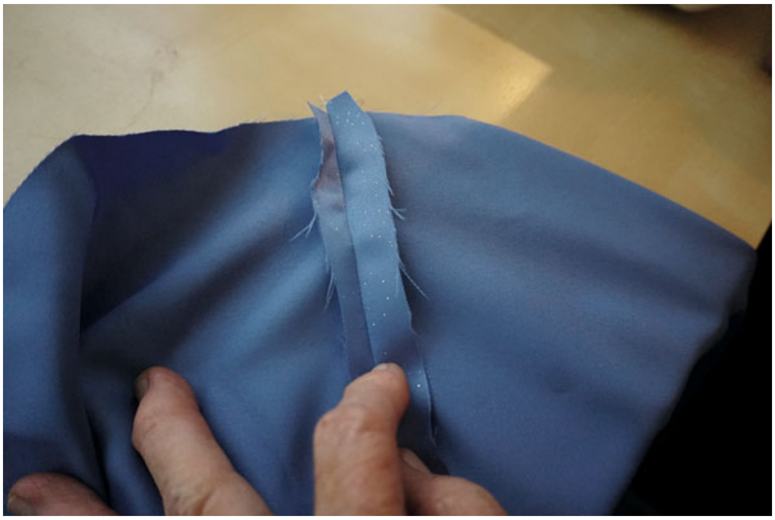


Figure 4-15. *Sew and then press open the seams*

Assemble the Front Facing

Remove the pattern pieces from the pieces of facing. With the right sides together, pin and sew the front facings to the back facing. Trim the seams and press them open. Then, with the right sides together, pin the facing to the vest along the seam line (Figure 4-16). Using the same stitches as before, sew along the seam line. Be careful along the curves. I recommend going slowly and stopping the machine to adjust the angle if necessary.

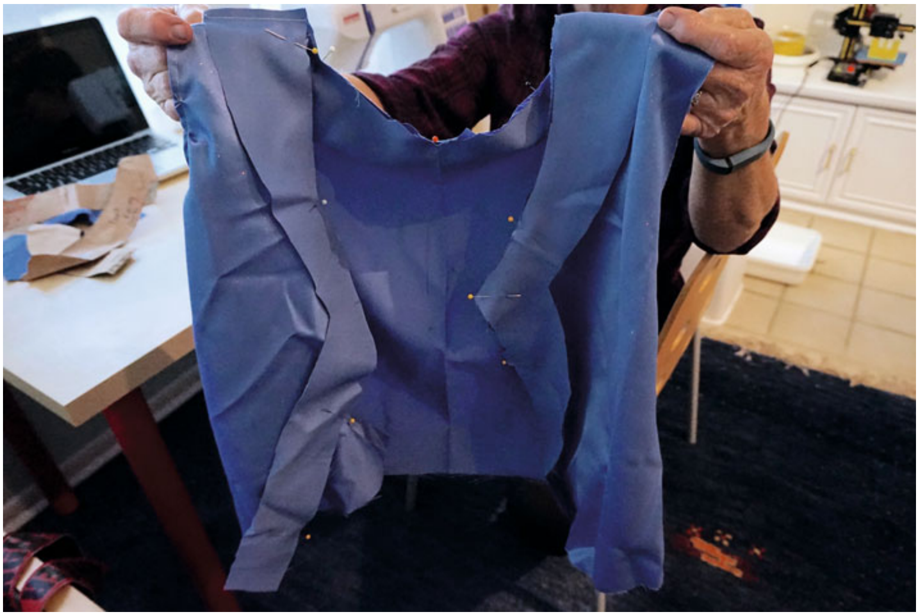


Figure 4-16. *Pinning the facing to the vest*

After sewing the facing to the vest, make tiny clips with sharp scissors at the curves, where necessary, to create a flat seam (Figure 4-17). Trim the seam and turn the facing inside the vest with wrong sides together. Press the facing down neatly.

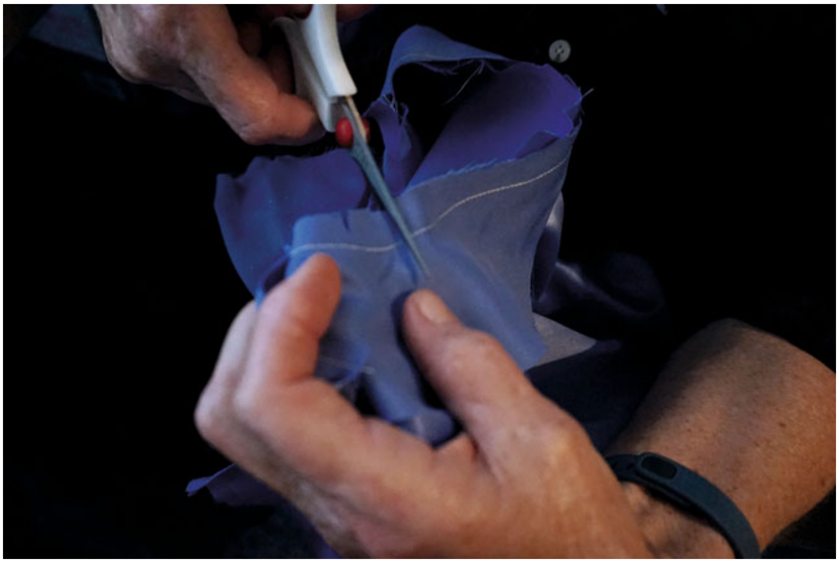


Figure 4-17. *Trimming a curved seam*

Finally, I am going to *topstitch* this seam. That means I will sew over the facing and the outer part of the vest to create a visible line of stitching parallel to the seam (Figure 4-18). Topstitching can be decorative with a different color thread (as I did here) or merely structural if the thread matches the fabric more closely.

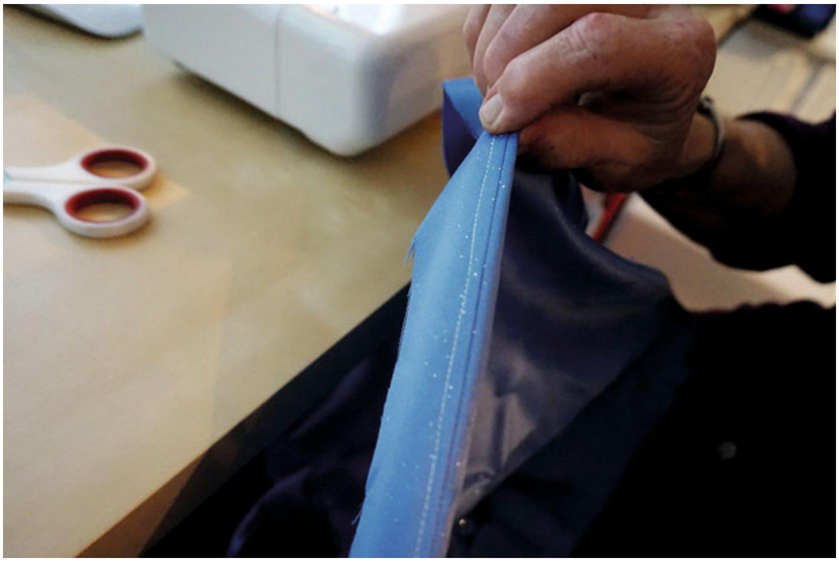


Figure 4-18. *Topstitching a seam*

Assemble the Armhole Facing

Figure 4-19 shows how one end of the two parts of the armhole facing goes together. Remove the pattern from the armhole facings. With right sides together, sew the ends of the facing together to create two donut shapes—one for each armhole.

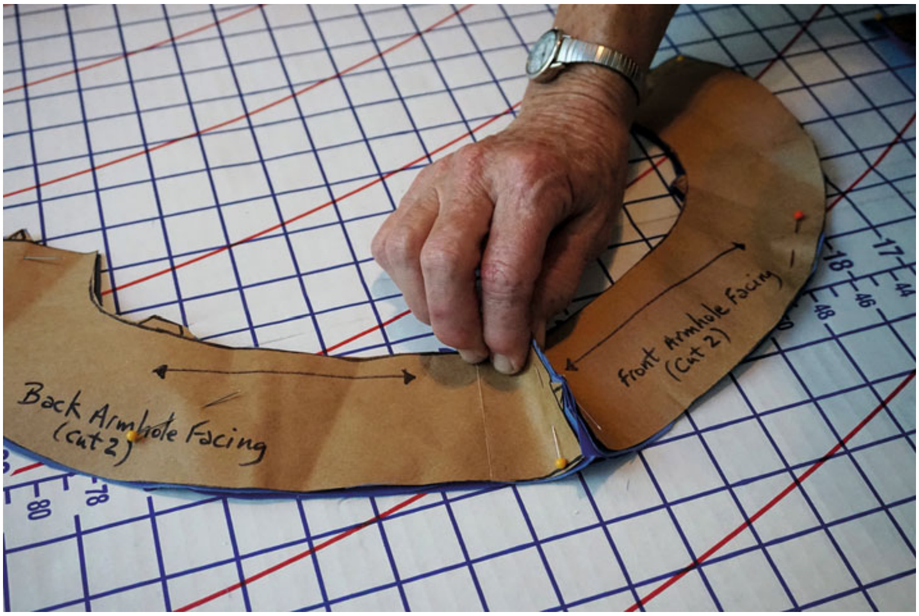


Figure 4-19. *Armhole facing assembly*

With right sides together, pin the armhole facing to the vest armholes. Stitch carefully along the seam line around the entire armhole. Go slowly and stop to adjust for the curve. Clip the curves with sharp scissors and trim the seam. Turn the facing to the inside and press.

Hem

Sew along the bottom edge of the vest with a longer or zigzag stitch (Figure 4-20). Turn the hem edge of the vest under 1/4-inch (.64 cm) and press. Hand or machine sew the hem (Figure 4-21).

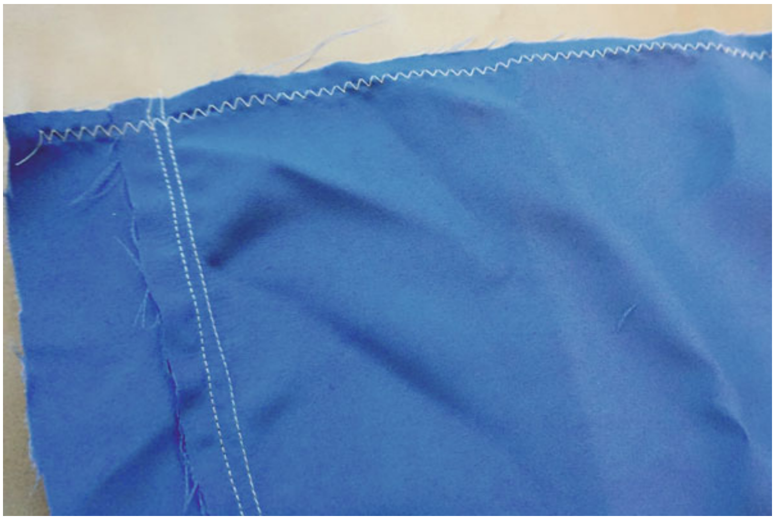


Figure 4-20. *First zigzag on the hem—note the facing double stitching.*

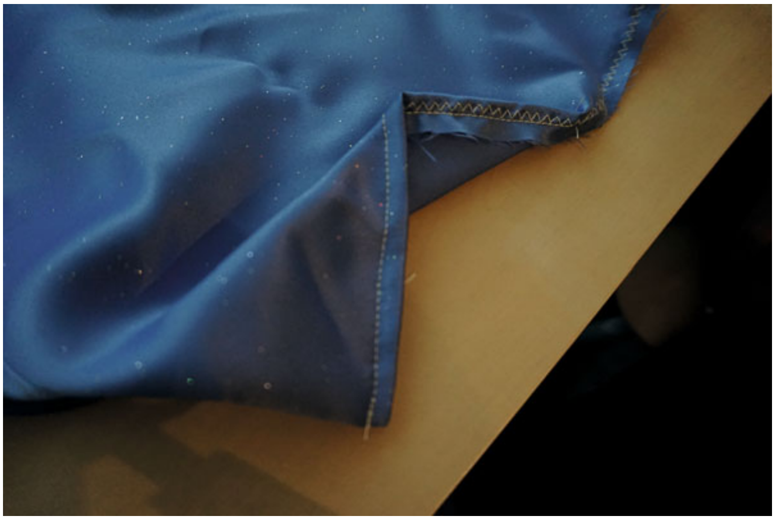


Figure 4-21. *Second pass on finishing*

Other Finishing

You have now finished a basic vest. Go back to Figure 4-1 to see my final result of this process. At this point you could add buttons or snaps or other decoration as you see fit.

Things That Go Wrong

It is easy to make silly mistakes that can become very frustrating. Try to make sure you are attaching the correct pieces at the correct places, and that the right sides of the fabric are facing each other, before sewing. Double-checking before beginning to cut or sew is helpful, especially cutting out the larger pieces because it is possible to waste a lot of fabric if you cut it too small.

I have been sewing for a long time and I still catch myself making goofy mistakes because I'm trying to hurry or am just spacing out. Particular places to pay attention to are where the facings attach to each other, where they attach to the vest, and what stitch you are using. If you have to rip out a seam and do it again, it will be fine. Just pull out all the bits of old thread before you begin again.

Fashion Tech Considerations

Allow space for wiring, batteries, and other components as you are planning, cutting out, and sewing your garment. You may want to add a pocket, a channel of fabric, or some other way to conceal and contain the tech pieces.

Pre-planning the run of any wiring will help prevent some mistakes and, hopefully, allow you to solder or connect them without ripping out seams. Even making a mockup of the piece first with cheaper fabric or scraps will allow you to keep the final garment in good shape.

It is important, too, to consider whether you will need to prevent your runs of conductive thread or fabric from shorting out your circuit. Beware of drapey or loose clothing with a lot of circuit lines.

Summary

This chapter discussed choosing patterns, various types of fabric, and how to measure for correct fit, and suggested some websites and stores where you can buy supplies. We also discussed how to pin and cut out sewing patterns and basic shapes of garment pieces for creating patterns. It included the directions for making the simple, unlined vest—our first project.

CHAPTER 5

Wearable Tech Electronics

In Chapters 3 and 4, Lyn gives you a tutorial on sewing and the skills to make the fabric "wearable" part of a project. In this chapter and Chapter 6 Joan and Rich introduce you to the "tech" part of a wearable tech endeavor. This chapter introduces the fundamentals of the electronics hardware you will need for projects later in the book. Chapter 6 adds the software basics about how these electronics operate.

Note If you want to learn to sew (as Lyn describes in Chapter 3) there are good video tutorials, and the skills are somewhat separable and easier to learn in bite-sized gulps. Here, however, you need a certain minimal base to get going on the electronics side of wearable tech design. Rich and Joan give an overview in this chapter, but there are many (too many?) resources out there you can look at for more. In particular, `http://arduino.cc` and `www.instructables.com` are good places to learn about the basics of the Arduino ecosystem, and many manufacturers, such as Adafruit (`http://learn.adafruit.com`) and SparkFun (`http://learn.sparkfun.com`), provide extensive tutorials supporting their basic products. A basic electronics textbook might help, too; you might check your local library for what they have on hand and see what fits your current level of knowledge.

Circuit Design

The first thing that happens when you start to learn any new field is that you have to learn a lot of vocabulary in a hurry. Neither of us is particularly a fan of lists of words, so in this section we will define key concepts by talking about them in context.

The first word you need to know is *Arduino*. (And yes, it really is named after a bar in Ivrea, Italy). It is an open source *platform* (a standard other people can design on top of) that includes a family of *microcontroller* boards and *Integrated Design Environment* (IDE) software that you will learn in Chapter 6. Figure 5-1 shows one of the common Arduino boards. Figure 5-2 is a version from the Fritzing program (discussed later in the chapter) along with a *breadboard* and some circuit components. We will talk about breadboards in a minute.

J. Horvath et al., *Practical Fashion Tech*, DOI 10.1007/978-1-4842-1662-0_5

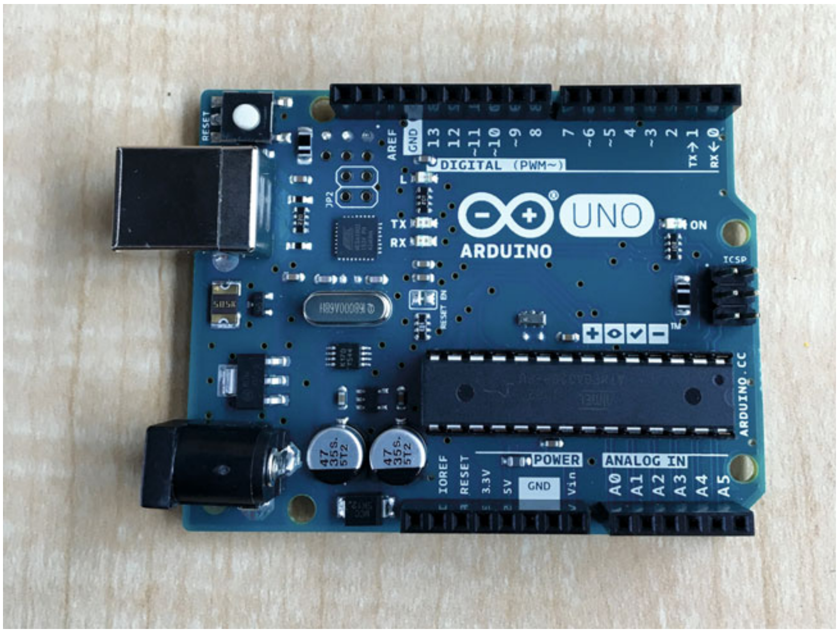


Figure 5-1. *An Arduino*

A microcontroller like an Arduino is not quite a computer like the Mac, Windows, or Linux machine you are used to. Microcontrollers typically control something that is not overly complicated. Although there are ways around it, normally they just execute one set of instructions over and over, and cannot switch back and forth between several programs or apps the way your laptop or smartphone can. However, this capability is plenty to run your item of clothing.

An Arduino has *pins,* electrical contacts on the circuit board. *Female headers* are permanently attached to the board so that you can insert jumper wires or male pins from other boards. (Boards intended to be soldered have different arrangements.)

An Arduino board also has a USB connector that can be used both to power the board and to send code to it. You will learn a lot more about these pins and how they work in Chapter 6. For the moment, we are just going to use the Arduino as a glorified power supply to run a circuit.

Breadboards

The white rectangular thing with holes in it in Figure 5-2 is called a *breadboard*. A breadboard is intended to make it easy for you to create *circuits*—combinations of electronic components that do something.

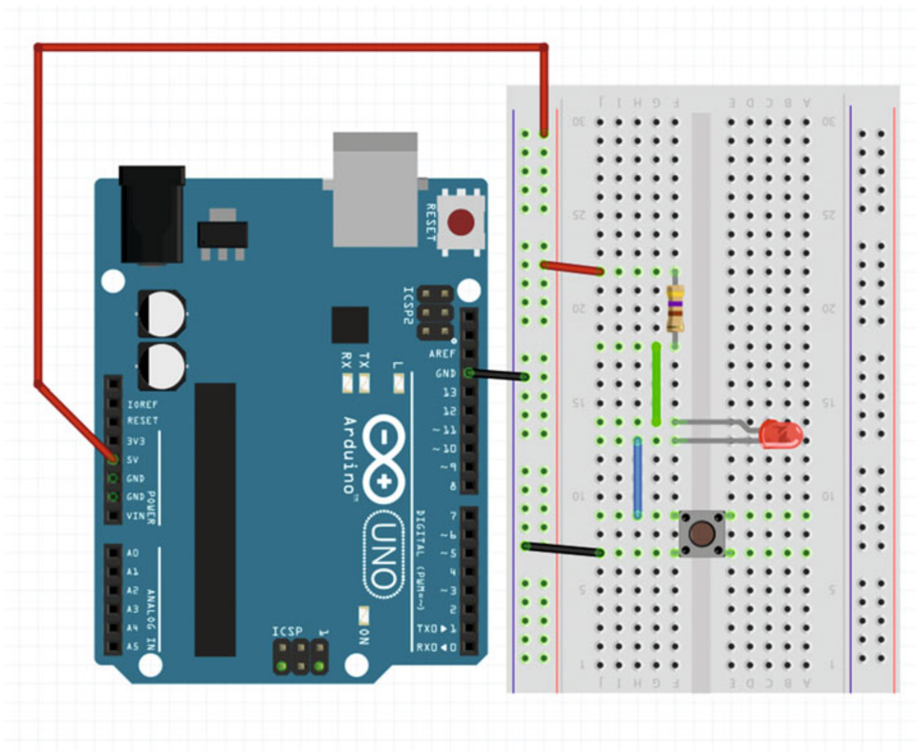


Figure 5-2. *An Arduino (blue circuit board) and a circuit on a breadboard*

The groupings of holes on a breadboard represent *electrical connections.* Circuits work by flowing electricity through components and wires. *Voltage* is the difference in electrical energy between a given place in a circuit and a reference level (which, if you were trained as a physicist, you might want to think of as a difference in potential energy between those points). (The Figure 5-2 diagram and others in this chapter were created in the Fritzing software—see `www.fritzing.org`.)

Current is the flow of electricity from one point in a circuit to another point at a different voltage. Resistance to a current can cause a *voltage drop* between two points. Voltage, current, and resistance are connected together by an equation called *Ohm's Law,* which we talk about in the next section.

The *power rail* (or power bus) and *ground rail* (or ground bus) on a breadboard are the vertical groupings of holes usually marked + and –, respectively, or with red and blue vertical lines as in Figure 5-2. They are used as places with standard voltages that we will call HIGH (typically 5 volts, labeled 5 V on the Arduino) and LOW (ground, or zero volts, abbreviated GND on the Arduino).

The horizontal rows of five pins each are only connected within these sets of five pins. There is no connection between the five pins on either side of the gap. The gap is there so that you can put the pins of *dual in-line package* (DIP) components (a common format for integrated circuit chips) across it. Here you can see a black button crossing the gap.

We go over all the other components you can see here—jumper wires, an LED, a button switch—in the upcoming sections.

■ **Note** You may have a wearable electronics board like a Gemma, Flora, or Lilypad that is "Arduino compatible" and looks different from the board in Figure 5-1. Some of the basic concepts are easier to see with a traditional Arduino and a breadboard than on the wearable boards, where some of the problems we are about to describe are hidden from you *most* of the time. We are preparing you for those other times when you will need to know what the board is trying to do so you can work around it. It is not necessary to have a standard Arduino board and breadboard to follow along with this chapter, but if you want to, you might buy a beginner Arduino kit (so you get all the parts you will need) from a supplier like `www.sparkfun.com` or `www.adafruit.com`. Otherwise, just follow along here and you will be better prepared to appreciate what your wearable components are doing for you.

Ohm's Law

To design a circuit, you need to know a little bit about how electricity works. One fundamental rule that underlies a lot of what we are talking about in this chapter is Ohm's Law. Ohm's Law can be stated very simply: voltage equals current times resistance, usually written $V = I * R$. (We use * to show multiplication, as is the convention in computer code.)

■ **Note** According to Wikipedia, current is I because Ampere, after whom the unit of current was named, was French, and he referred to current magnitude as "*intensité de courant.*" The unit of current is the *ampere* (typically just called *amps* and denoted with a capital A). The unit of voltage is the *volt*, usually shown as a capital V. The unit of resistance is the *ohm,* usually symbolized by the Greek letter omega, Ω. Sometimes R is used instead of Ω in online situations where people either do not want to figure out how to type a Greek letter or are afraid that it will not survive text encoding conversion.

Very often we need to talk about a large or small voltage, or current, or resistance. Then we introduce prefixes like *milli-*. A millivolt (mV) is 1/1000th of a volt, and a milliamp (mA) is 1/1000th of an amp. Other prefixes include the following:

- *mega (M)* = times 1,000,000
- *kilo (k)* = times 1,000
- *micro (μ)* = 1/1,000,000

Voltage varies throughout a circuit and is measured relative to an arbitrary level (referred to as *ground*). Because voltages are denoted in reference to ground, ground is, by definition, zero volts. In most circuits like the ones in our breadboard, it is the lowest voltage in the circuit, thus avoiding negative numbers. Because voltage varies depending on where you are in the circuit, it is a property of a particular point in a circuit, not a whole circuit. Actually, it is better to think about how voltage changes across a component, called the *voltage drop*. Obviously, this is easier to calculate if the low side of the component is zero volts, which is why most circuits are arranged this way.

Circuit Components

A difference in voltage between two points results in a current flowing between those points. If you want to limit the current flowing to a part of a circuit while keeping the voltage across it the same, you need to add resistance to the circuit. Not surprisingly, components that only add resistance to a circuit are called *resistors*. (Other components add resistance too, but resistors have this as their only function.)

Resistors (and other components) can either be in *series*—arranged so that current flows first through one, and then through the next—or in *parallel,* so that current at some point forks and goes through two parts of the circuit at once.

If two paths in a parallel circuit have different resistances, current wants to take the path of least resistance, and more current will flow through that side of the fork. That is, more current will flow across the lower-resistance part of the circuit, but both parts of the circuit will have the same voltage drop.

In a series circuit (say, two resistors one after the other), all the current has to flow through the one path. The same amount of current will flow through both resistors, but each will have its own voltage drop proportional to its part of the total resistance of the circuit.

Resistors

Because resistors are pretty small components, we need some way of labeling them that scales way down, too. A system of colored bands on resistors is in common use. The four-banded resistor in Figure 5-3, for example, has brown-black-red stripes followed by a small gap and then a gold stripe.

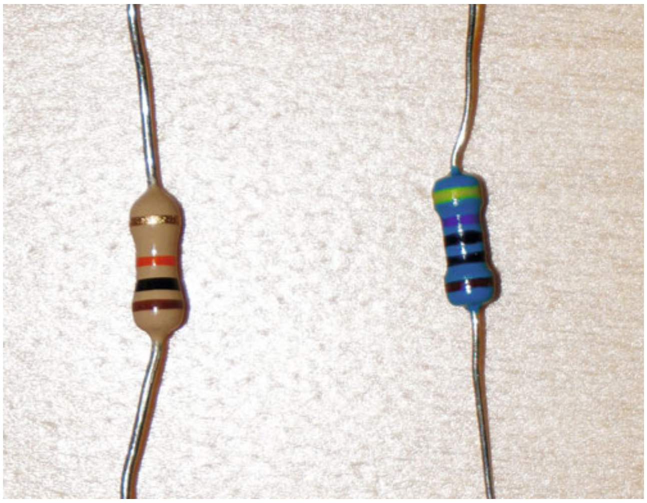


Figure 5-3. *A four-banded and five-banded resistor*

The two main conventions, as shown in in Figure 5-3, are resistors with four bands and ones with five. You can see a good explanation of what the colors mean at `https://en.wikipedia.org/wiki/Electronic_color_code`, or search online for "resistor color chart." We will calculate the values of the two resistors in Figure 5-3 to give you the idea.

On a four-banded resistor, the first two bands give the first two digits of the resistor's value, and the third band tells you how many factors of ten to multiply it by. Brown stands for 1 and black stands for 0. Red stands for 2. So brown-black-red means the value is 10 times 10^2, or that this is a 1000 Ω resistor. The gold stripe is the *tolerance band*, which shows how close the nominal resistance this particular one is guaranteed to be—plus or minus 5 %, in the case of a gold band.

The five-banded one is yellow-purple-black-black-brown, which turns out to be 470 Ω plus and minus 1 %. (Yellow is 4, purple is 7, and black is 0, so this is 470 times 10^0. Any number raised to the zeroth power equals one, by convention. The brown tolerance band means that this is a plus and minus 1 % resistor. Note that in both the four- and five-band cases, the multiplier band is not the number of digits in the final resistor—it's not scientific notation, if you have seen that before. In the case of the four-banded resistor, you are multiplying a two-digit number by some factor of ten; in the five-banded ones, you multiply a three-digit number by some factor of ten.

The tolerance band is at the top of the four-band resistor in Figure 5-3, and at the bottom of the five-band resistor. We arranged it that way to show how hard it can be to tell. Experience will give you a sense of what resistors are in common use. Look carefully when you take some out of the packaging for the first time so that you learn. Hypothetically, there is a gap between the last band of the resistor's bands denoting the value of the resistor and the tolerance band, but often it is hard to see.

Here are some common four-banded resistors (ignoring their tolerance bands):

- *1 kΩ:* Brown black red
- *10 kΩ:* Brown black orange
- *470Ω:* Yellow purple brown

LEDs

Light-emitting diodes (LEDs) are low-powered lights that illuminate when current is passed through them in one direction (but not in the other). LEDs have one leg that is longer than the other. The longer side of a freestanding LED should go on the + (HIGH, connected to Vcc or 3.3 V) side of the circuit. NeoPixels have separate connectors for power (+) and ground (–). The NeoPixel also has a signal-in and signal-out pad that we talk about in Chapter 6. Figure 5-4 shows one of each.

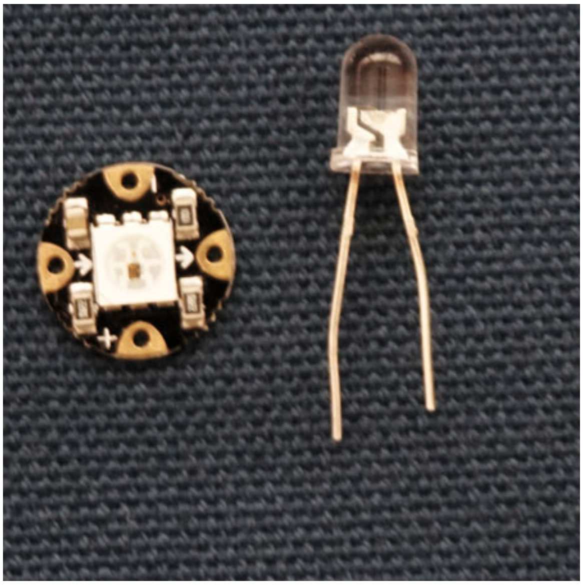


Figure 5-4. *A sewable NeoPixel 3-LED set (left) and a regular single LED (right)*

Different-colored LEDs will create specific, fixed voltage drops across them. There is a table explaining these at `https://en.wikipedia.org/wiki/Light-emitting_diode#Colors_and_materials`. Generally, the values are between 2 and 4 volts. An LED will not survive too high a current across it and will block a current going the wrong way.

Sizing Resistors

Suppose we wanted to create a circuit that would light up an LED without making it blow out. First, looking at the numbers that are usually listed on websites where you buy LEDs, or just by knowing common values for LEDs, we would discover that the maximum current a red LED wants to see is about 35 mA (or 0.035 amps). An Arduino usually puts out 5 volts at about 20 mA max. So we need to put a resistor in series with the LED so that we do not damage it or the Arduino pin controlling it. But how big a resistor?

Ohm's Law says $V = I * R$, or we can make this $R = V / I$. The LED will drop a fixed voltage of about 2 volts. That means that we will drop about 3 volts (5 minus 2) across our unknown resistor.

The maximum current we want to run through the LED is 35 mA, which is more than the Arduino's max output of 20 mA (20 mA = 20/1000 A = 0.02 A). We use the lesser of the maximum current that would be okay for the LED and the current that might damage the Arduino's pin (20 mA). Thus, the resistance we want is $R = V / I = 3 / 0.02 = 150\ \Omega$. So we want a resistor *at least* this large. In this case, bigger is okay too (it will just reduce the brightness of the LED).

■ **Caution** More complex components have resistance too, but it might vary depending on what the component is doing at the time. Do not assume that only resistors have finite resistance.

Jumper Wires

To connect points on a breadboard to each other or to an Arduino, we use *jumper wires*. They are just wires that are insulated in the middle and have exposed stiff wire on either end. Figure 5-5 shows both the common garden variety and the less common "pre-formed" flat jumpers that look like giant staples.

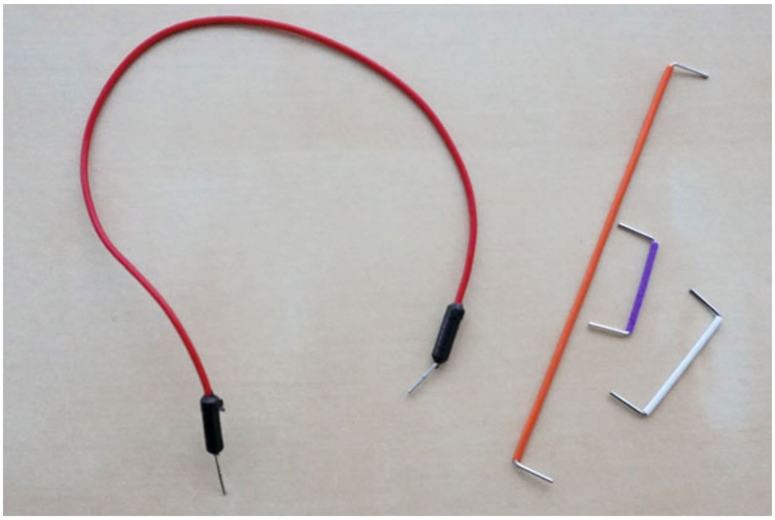


Figure 5-5. *Regular jumper wires and pre-formed ones*

The staple-like ones have the virtue that it is a lot easier for a beginner to trace what is going on. Both of these types are called *male/male* jumpers because there is a male connector on each end. Other types of connectors come up from time to time (male-female jumpers, for instance).

Voltage Divider

A *voltage divider* is a piece of a circuit in which two or more resistors are used to alter the voltage in parts of the circuit connected to the circuit input or output. When two or more resistors are wired in series, their resistances are added to figure out the total resistance.

When voltage drops across a series of resistors, intermediate voltages are created between them. Typically, the arrangement is something like that shown in Figure 5-6 (where we show that resistors can also be drawn as a squiggly line—the two drawings are equivalent).

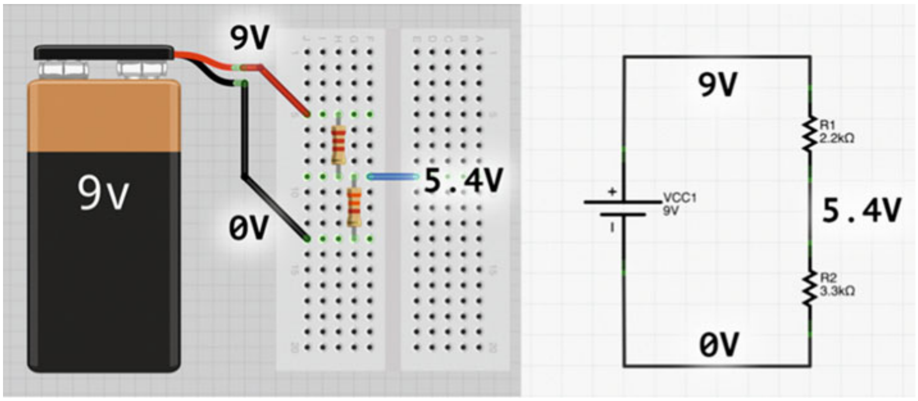


Figure 5-6. *The voltage divider*

INTERFACING THE ANALOG AND DIGITAL

An Arduino circuit can use current to do two things (perhaps simultaneously): power a component, or cause a voltage to change as seen by a particular pin on an Arduino. The latter are often called *logic signals,* and typically the circuit is designed so that a voltage on a digital pin (more about this in Chapter 6) is either unambiguously HIGH or LOW.

Voltage dividers are useful when a digital input needs to "see" a certain voltage reliably to use that voltage as the basis for an action. For example, a pin being HIGH or LOW may cause a program running on an Arduino to take an action. A button or switch with a *pullup or pulldown* resistor is also a type of voltage divider. In this case, one of the resistors (the button) switches from an infinite resistance to zero resistance when pressed (or when released, in the case of a normally closed switch). We talk a lot about this in Chapter 6. Many sensors are used in circuits like this, so you can look in Chapter 8 for examples, too.

Voltage dividers can be used to change the voltage of a logic signal to allow a component like an Arduino with 5-volt output to communicate with another device (such as an accelerometer) that expects data to be sent with a 3-volt HIGH signal.

However, you cannot use a voltage divider to produce the 3 volts needed to *power* the other device. Because the device draws current to power itself, it acts as another resistor in parallel to one of the ones in the divider, changing the effective resistance. Theoretically, putting the right resistor in series with the device should limit the current and drop the voltage appropriately, but in reality, the device needs a constant voltage even when the current flowing through it changes (which is likely to happen, for instance, as its outputs turn off and on).

Voltage regulators are used to produce a constant output voltage for this purpose, and most Arduino boards have 5-volt and/or 3.3-volt regulators built in.

Potentiometers

Sometimes it is useful to vary the voltage at a particular point in a circuit. When one or both of the resistors in a voltage divider can change its value, you get a variable voltage that can be read by the analog input pins on an Arduino.

The most common form of this is a *potentiometer,* also sometimes called a *pot,* which is a variable resistor, or a rheostat. A potentiometer is one big resistor with a wiper that can make a conductive connection anywhere along the resistor.

This means that from one end to the other, the potentiometer may measure 10 kΩ, but the wiper could be 0 kΩ on one side and 10 kΩ on the other, or it could be 4 kΩ / 6 kΩ, or 8.73 kΩ / 1.27 kΩ, or any other ratio. Thus, if you connect the two ends of the potentiometer to 5 V and ground, you can turn the knob on the potentiometer to get any voltage on the wiper between those two.

An Arduino can read this voltage to know the position of the potentiometer or equivalent devices (like sensors in Chapter 8) that are used like voltage dividers.

Example

If the series of resistors is between two different voltage levels, commonly 5 V and ground (zero V), the total voltage drop across the series of resistors will be equal to the difference between those two voltages, but the voltage drop across any particular resistor in the series will be proportional to its fraction of the total resistance. So, with a 2.2 kΩ resistor and a 3.3 kΩ resistor, the 2.2 kΩ will drop 2 volts, and the 3.3 kΩ resistor will drop 3 volts. If the 2.2 kΩ resistor is connected to 5 V and the 3.3 kΩ resistor is connected to ground, the voltage at the point where the two connect to each other will be 3 V.

When making a voltage divider, you generally want to use resistors with values of several kΩ, because current will always be running through the divider. In the earlier example, the total resistance is 5.5 kΩ, so the current is 5 V / 5500 Ω = 0.91 mA. If instead we had used 2.2 Ω and 3.3 Ω, the current would be about 910 mA (which may be more than the power source can provide—a Flora, for instance, only has a 250 mA regulator) and there would be 4.55 watts of power being dissipated between the two resistors (1.82 on the 2.2 kΩ and 2.73 on the 3.3 kΩ). (A *watt* is a unit of power, computed as current times voltage. That is, a resistor with a voltage drop of 1 V and a current of 1 amp will dissipate 1 watt.)

Not only is this wasteful, but the resistors that you use on a breadboard are usually only rated for a quarter watt and would burn out if they tried to dissipate that much power.

On the other hand, if you used a value of several hundred megaohms (MΩ), your signal might be very noisy because radio waves picked up by the wires between the resistors might induce nearly as much current as the voltage across the divider. The total resistance of a voltage divider should generally be somewhere in the range of 1 to 100 kΩ.

Creating a Circuit

Now we will take some of the components that we have described so far and create a circuit. If we look back at Figure 5-2, we see a simple series circuit with a 470 Ω resistor, an LED, and a button.

The first thing to notice is the long red wire going from 5 V on the Arduino to a vertical *rail* of the breadboard, and a shorter wire going from the GND (ground) Arduino pin to the rail next to it, as we talked about in the "Breadboard" section earlier in this chapter.

Tracing the circuit from top to bottom, we have a 470 Ω resistor in series with a red LED and a button. When we look it up at the site referenced in the "LED" section of this chapter, we see that red LEDs have an inherent 1.8 V drop across them, plus or minus 0.2 V.

The Arduino is putting out 5 volts. If we use the example earlier in the discussion of Ohm's Law, we see that the 470 Ω resistor is plenty big enough to protect the LED. If you plugged this into the USB port of a compatible computer and pushed the button, you would expect the LED to light up if the button is pushed.

If the button is pushed, the LED is connected to 5 V through the resistor and to GND through the button. This means that adequate current will flow through the LED, and it will light up. If the button is open, the LED is connected to 5 V on one side and to an unknown, un-set voltage on the other. It is said to be floating. We do not care in this chapter, because we are not trying to use any of the voltages in the circuit to control something else. In Chapter 6 we will care, however, and we will introduce some concepts here to start to explain that.

In Figure 5-7, we show a different kind of circuit. If the Arduino is just being used as a power supply and you push the button, nothing happens. The LED is connected only to ground and weakly to an unknown voltage from pin 12. The button is on a separate circuit. In Chapter 6 we will look at this as an example of a pulldown resistor. We introduce it here to show that you can create unconnected circuits in hardware that you will need to connect in software to do something.

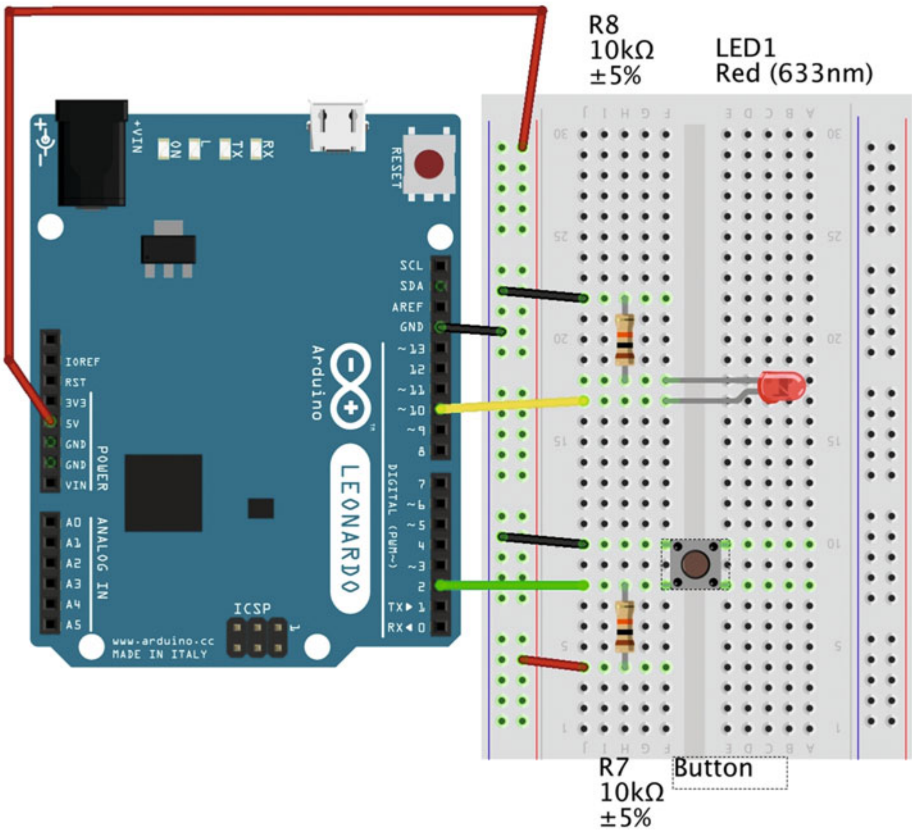


Figure 5-7. *A circuit with a pulldown resistor*

The circuit in Figure 5-7 does nothing in this chapter, since we are just using the Arduino as a power supply and there is no direct electrical connection between the button and the LED. In Chapter 6 we will show the code to have the Arduino manage turning the LED on or off using the button. The button forms a voltage divider with the second 10 kΩ resistor. If the button is pushed, it has a very low resistance resistor. If the button is *not* pushed, its resistance is infinite.

When the button is *not* pushed, pin 2 is weakly connected to 5 V—weakly because it is connected through a significant resistance, and the "other half" of the voltage divider (the open button) has an infinite resistance—thus no appreciable current is flowing. Pin 2 would see what the Arduino would interpret as a HIGH voltage (discussed in the next chapter).

If the button is pushed, the button will have almost no resistance. Pin 2 will now be connected to (almost) ground. We say "almost" because most of the voltage drop will occur across the 10 kΩ resistor, not across the small resistance of the button connection. Pin 2 would thus see what the Arduino would interpret as a LOW voltage.

In later chapters we will do things like this to create HIGH and LOW defined voltages on pins we use to see what is going on in a circuit. This is called using a *pulldown resistor* because the resistor *pulls down* the output to a known value instead of leaving it to float. Until the button is pushed, pin 2 is connected to 5 V, but when you push the button, it creates a strong connection to GND and allows a small amount of current (inversely proportional to the value of the resistor) to flow.

■ **Tip** Pullup and pulldown resistors should have values of no less than 1 kΩ and may be as much as 100 kΩ. Higher values may create a connection that is too weak and may react too slowly or be vulnerable to interference. The chip used by an Arduino has internal pullup (but not pulldown) resistors that can be activated by writing an input pin HIGH.

FRITZING

The Fritzing program allows you to draw diagrams of circuits neatly so that you can share circuits with others in a clean and lucid way. You can share either cartoon "breadboard" views (what we have used in this document so far), or show the circuit as a schematic or how it would look as a printed circuit board (PCB) layout if you had a special-purpose board created. You can download the program (which is free) at `www.fritzing.org`, and there are tutorials at `http://fritzing.org/learning/`.

Using the Inspector

To change component details or to see what the defaults are, click on a component and then go to the Window menu item and be sure that Inspector is checked, which will display the Inspector window. In this example we have selected a resistor. When we look in the Inspector pane on the lower right, we see that we can select the resistor resistance value from a dropdown menu, and it will change the color bands on the displayed part to the correct values for that resistance. This is particularly handy for resistors, but it applies to other components too.

Adding New Components

Fritzing comes with a standard set of components. To add more, you may need to search around a bit to see if someone has created other components. To see how to add the Adafruit components, for example, see `https://learn.adafruit.com/using-the-adafruit-library-with-fritzing/download-the-fritzing-library-from-github` or, for Sparkfun components, `https://learn.sparkfun.com/tutorials/make-your-own-fritzing-parts`.

Fritzing Expert Tips

To add more components in breadboard view, go to the upper right and click on the spyglass. You can type in what you want (a manufacturer, a type of component, and so on) to find a component.

Right-clicking (Control-clicking on a Mac) on a component brings up other options you can change, too (like wire color).

■ **Note** It is easy to forget that any given physical circuit can run different software and vice versa. What your circuit and its software do is a result of both of these aspects, and there usually are many ways to create a circuit to do something you want to do. You might consider figuring out something one way, taking photos, and then trying a different one. The photos are insurance in case you *cannot* figure out another way.

Sewable Components

Arduinos come in various shapes and sizes. The ones in this chapter to this point are good for applications where you might want to build something on a tabletop. But suppose you want to have the components built into something you want to wear? You would want to use sewable components, put together with conductive thread. There are Arduino-based microprocessors that are flat, round, and intended to be sewn onto a garment.

Figure 5-8 is the apron project we create in Chapter 7. The saucepan is made of two layers of conductive fabric separated by a piece of felt with a hole in it. The processor is an Adafruit Gemma (Arduino-compatible board) with a NeoPixel multicolor LED in the "steam." If you tap the saucepan, the LED flashes red for a time set by the programmer and then turns green for another settable period of time. You will see how to put this together in great detail in Chapter 7.

Sewable Arduino Boards

Several sewable Arduino boards are available. As of this writing, the Flora, Gemma, and Lilypad boards are in common use. The Gemma is a cheaper, smaller board than the other two, but it has some operational limitations that we discuss in Chapter 6. The small round board in Figure 5-8 is a Gemma.

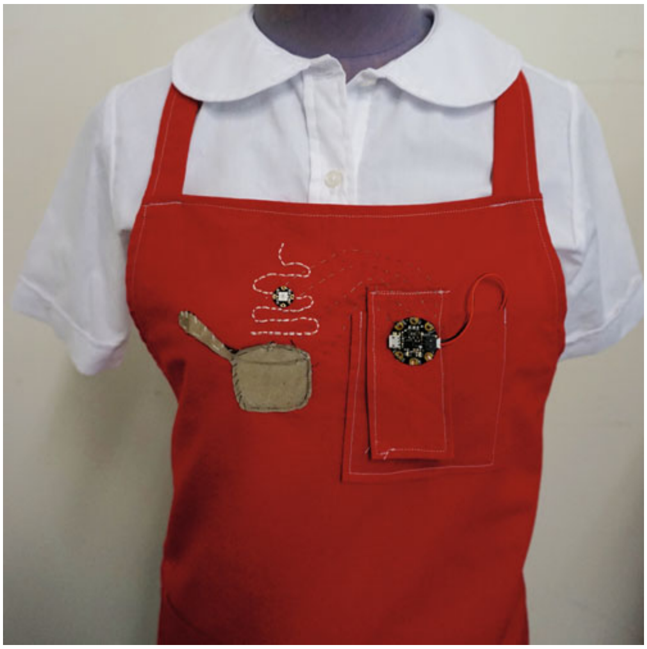


Figure 5-8. *The electronic apron from Chapter 7*

Figure 5-9 shows a Flora processor from Adafruit (`www.adafruit.com`). Instead of female pins, there are pads with big holes that are meant to be sewn with either conductive thread (to create a circuit) or with regular thread (to hold it onto the garment). For prototyping purposes, you can put together a circuit temporarily with alligator clips (Figure 5-10). Convention is to call these pads (with holes) *pins* in analogy with the traditional Arduino board. There are also boards with snaps, and by the time you read this there may be other variations.

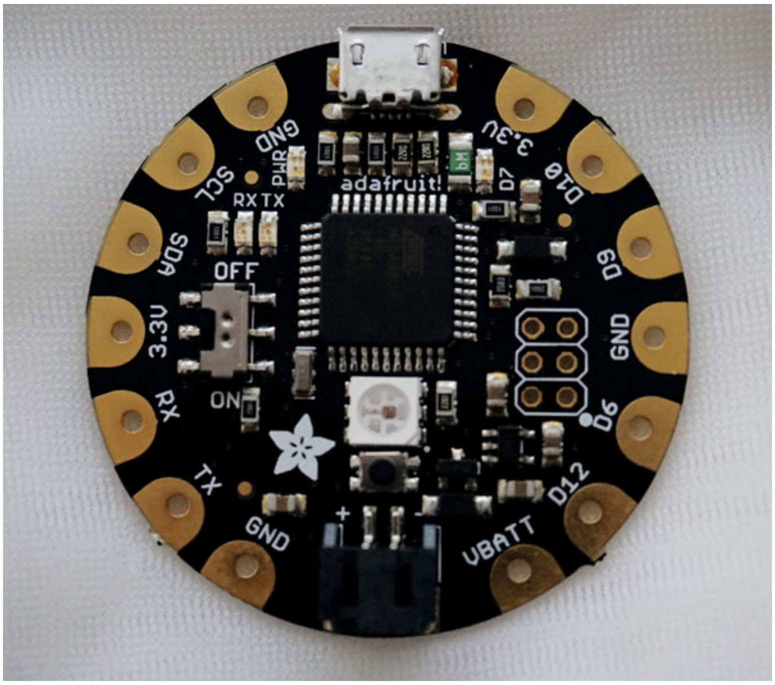


Figure 5-9. *The Flora board*

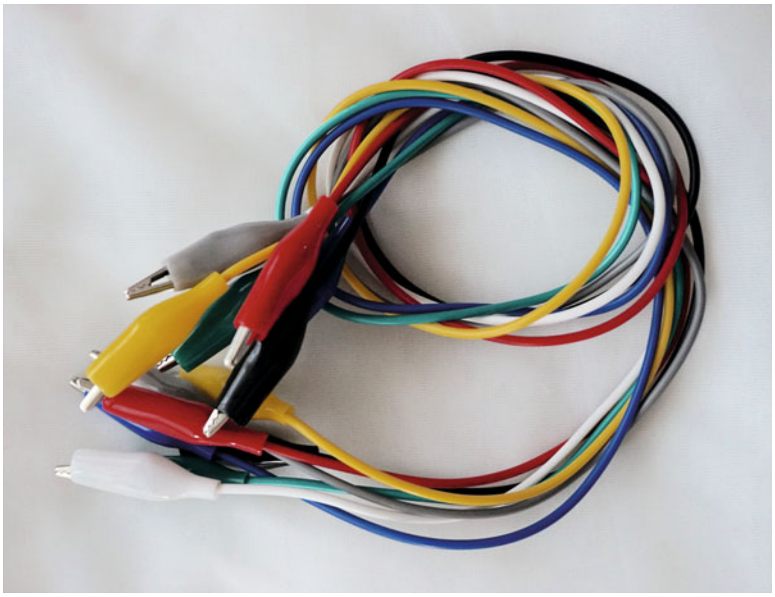


Figure 5-10. *Alligator clips*

Prototyping with Sewable Arduino Boards

Fundamentally a wearable board (we will demonstrate here with an Adafruit Flora) is the same as an Arduino, although it may leave off some pins. Where the differences start to arise, though, is in the smaller components—the LEDs, sensors, and so on that are made specifically to be sewable.

Figure 5-11 shows a Flora board with a NeoPixel LED board, which has a red, green, and blue pixel—and appropriate current-limiting circuitry—all in one package. Figure 5-12 shows this circuit prototyped with alligator clips. Resistors that you would have to size for a regular Arduino are often built-in to the sewable counterparts. The bad news is that sewable components by and large are a lot more expensive than their traditional counterparts.

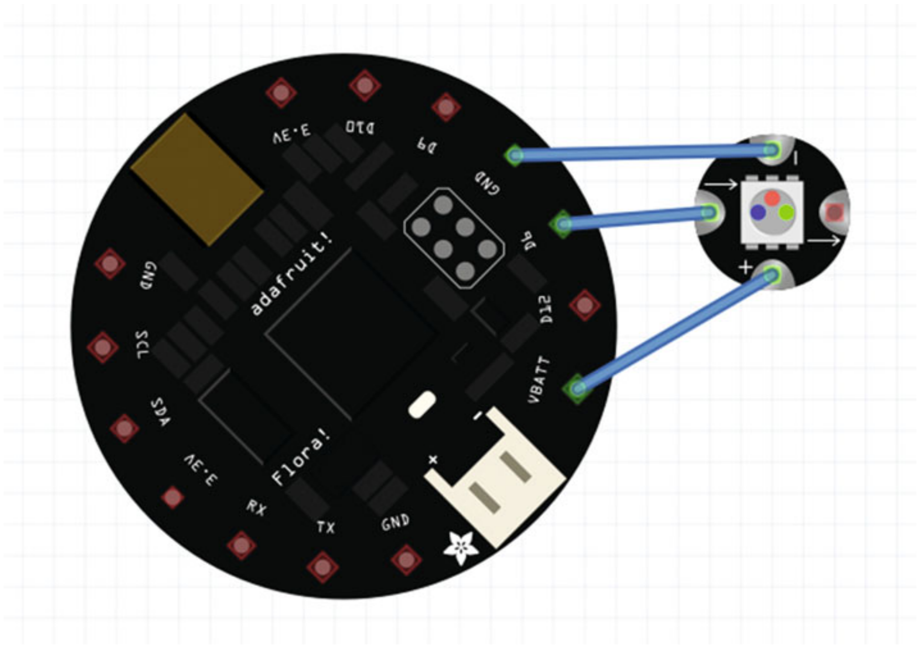


Figure 5-11. *Fritzing diagram of a Flora and NeoPixel circuit*

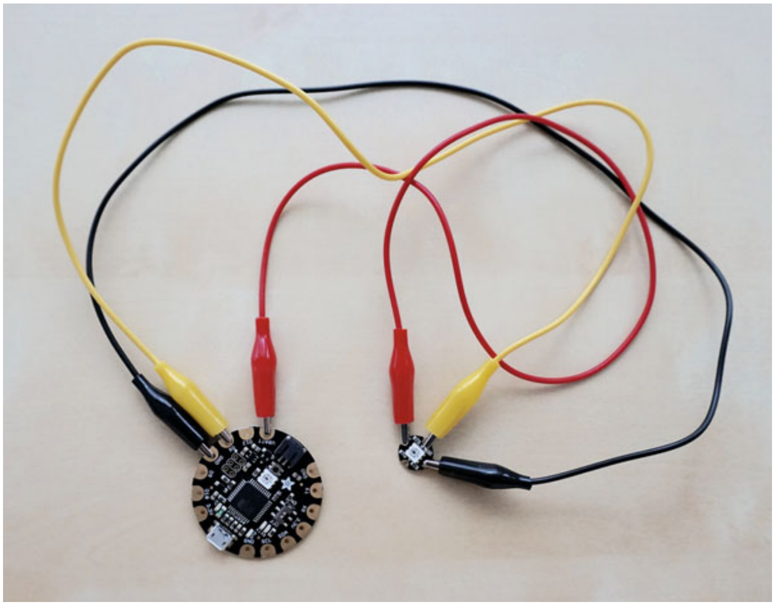


Figure 5-12. *Flora and NeoPixel circuit, with alligator clips*

If you have a Flora and NeoPixel handy, use alligator clips to connect the following:

- GND on the Flora to the – on the NeoPixel
- Vbatt on the Flora to the + on the NeoPixel
- D6 on the Flora to the - > on the NeoPixel

In Chapter 6 we write the code to make this circuit do more than have the NeoPixel light up brightly when the USB is plugged in. Once you are happy that your circuit works, after stringing it together with alligator clips and testing out the software we discuss in Chapter 6, you can sew the circuit with conductive thread. See Chapter 7's discussion of the apron development to see how much thought this may require ahead of time. It can be challenging to avoid having your conductive thread make contacts only where you want it to in a squashy garment.

Batteries

You will most likely power a circuit during development by plugging the Arduino board into a USB port. Once you are wearing a piece or using it in the field, you will most likely need a battery. If a project is just running a few NeoPixels and a board (like the apron project), you can probably get away with coin cell batteries—the small flat metal cells you have seen in watches and other small electronics. These batteries fit well into small cases (Figure 5-12), and the batteries are then pretty easy to remove. AAA batteries (in an appropriate holder with connectors) might be an option too.

Look on the battery's packaging to see what voltage and power are delivered and analyze the circuitry in your project to see what you will need. Many smaller batteries are alkaline batteries that deliver 1.5 V. To drive a component needing three volts, you would hook up two of these in series. Obtain battery packs that arrange this for you. The battery connector is connected to a voltage regulator that requires the input voltage to be higher (as much as 1.5 V higher for some regulators, known as the drop-out voltage) than the output voltage. In this case, we are using two 3 V lithium coin cell batteries.

■ **Caution** Do not use the "pillow" type LiPo (lithium polymer) batteries in wearable projects. They are fragile and can break and catch fire. You may see older designs online using these, but avoid them and use newer, safer, and better-contained batteries. Read the manufacturer's charging instructions if your batteries are rechargeable.

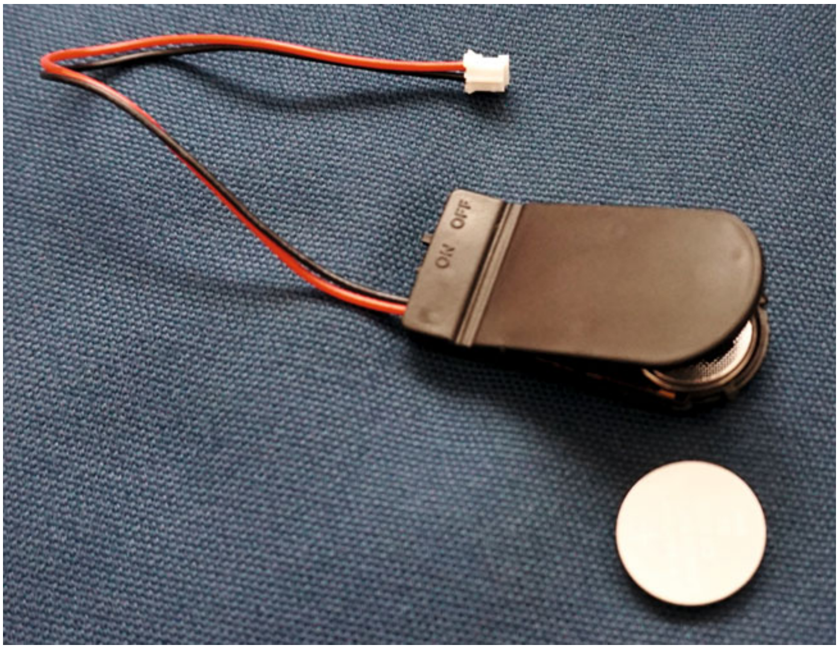


Figure 5-13. *Coin cell battery case (open, with a battery next to it) designed to attach to a sewable processor's connector*

Conductive Ribbon and Thread

To connect up a sewable circuit, you need to use some sort of conductive material. You can sew the circuit on by hand; the projects we will show are best done this way. There are, however, other products out there, like ribbon with five parallel connectors, which you might want to try. Lyn preferred sewing by hand to using ribbon because she found it too easy to short out circuits on the closely spaced conductors running along the ribbon.

Other Components

There are a few other miscellaneous component types that you should know about in case they come up in projects you decide to try.

Some components may come with *shields* or *breakout boards.* Shields are really only applicable to traditional Arduino boards, since they are designed to clip on top of the pins (which will not work on the wearable's pad connectors). Breakout boards, however, are designed to take a set of outputs from a sensor or other device and route them (directly or indirectly) to the Arduino. You may need to get creative to figure out how to connect these boards to a wearable, if there is not yet a sewable version.

Capacitors are components that are used to store small amounts of energy, or to smooth out spikes in current. You might use one if you anticipate voltage spikes in your project for some reason (such as a motor or a large number of LEDs turning on at once).

Electroluminescent (EL) wire is wire with a coating that glows when an alternating current is passed through it. An inverter is required to step up the Arduino voltage to the higher values (but very low current) used by EL wire. Sequencers are sometimes used to switch between pieces of EL wire to give an appearance of motion, like an old-fashioned animated neon sign. We have an EL wire project in Chapter 11.

DO I NEED TO LEARN HOW TO SOLDER?

You will be able to do all the projects we give instructions for in this book without needing to solder. However, you may find sensors or other components that you would really like to use, but are impractical to connect with sewn or other non-permanent connections. There are many learn-to-solder tutorials online and in-person classes, and we suggest you look into one of those if you want to move on to more complex projects.

Laundry

If you are making a piece that you plan to wear, sooner or later you will need to wash it. Some components can be hand-washed, but others should not be. You can put a circuit on a placket that you attach in a removable way, have non-washable pieces in a pocket or shoulder pad that they can be slipped out of, and so on. The manufacturer of your part likely has something on its website about this issue. Note that dry cleaning is actually not "dry," and dry cleaning fluid can damage components.

■ **Caution** Never wash or dry-clean batteries, inverters, sequencers, or other sensitive parts. Design your piece so that batteries, EL wire, and any parts not specifically noted as washable can be removed easily.

Summary

This chapter reviewed the Arduino platform and components in its ecosystem of products. We talked about how to size resistors and how to prototype basic circuits. In the next chapter we will see how to write code that runs on the circuits we talked about in this chapter. We will write some programs in the Arduino IDE (software environment) both for the sewable boards and their generic non-sewable components.

CHAPTER 6

Programming Wearables

One of the things that people find intimidating when starting up Arduino projects is that you do need to learn to program, at least a little. The good news is that a good way to learn computer programming is to look at examples, and there are many examples out there for basic Arduino programming.

As with our sewing and electronics chapters, we only have the space to get you started here, and you will probably need some details fleshed out using other resources. Arduinos use a programming language called C++, a close relative of other programming languages like Java, Python, and C. There are many good C++ programming books; for example, there are many available at www.apress.com/programming/c-c?p=1. You can also follow equivalent online tutorials on C++ or the simpler, older language C that C++ is based on to back up the quick tour we provide in this chapter. For the purposes of the rest of this book, if you can figure out how to tinker with existing examples that are available online or in the downloadable material in this book (the link is on the copyright page of the book), you should be good to go.

Joan and Rich take you through this chapter. Joan learned to code in traditional classes and then worked on and managed some quite large programming projects. Rich has more of a maker background. When the two of us diverge on approaches, we may show you more than one way to think about something and let you decide what approach fits your style.

This chapter is in three parts. First we give a general introduction to programming. Then we show you how to find examples for the regular (not sewable) Arduino boards and make a few points about programming those. Finally, we move into the sewable boards.

Programming Basics

Programming can seem intimidating because the actual code looks arcane and there seem to be a lot of rules to learn. However, once you know why those rules are there, anticipating the right way to develop a program becomes easier. Programming is just coming up with a very detailed set of instructions for the Arduino to use to do something in a particular sequence, such as turning on LEDs if a button is pushed, taking sensor data and turning something toward a light source, and so on.

Another way to think about a program is that it is a detailed script for a computer (or a processor like an Arduino) to carry out. If you were writing a script with stage direction for an experienced actor, you might not put in very much detail. But if you were explaining stage direction to kindergarteners for the school holiday pageant, you would likely break it down into small steps. Programming is like writing a script for preschoolers, because a computer does not know what you meant to do, nor does it have knowledge of the real world beyond what you tell it.

■ **Tip** The classic way to introduce programming to beginners of any age is to have them write a very detailed set of instructions for a robot to do something, like make a peanut butter and jelly sandwich. Typically one person writes down the instructions and then someone else plays the "robot," following the instructions very literally. If you have never coded before, you might do this as an exercise just to think about the steps. How might you define "above" and "below" and "turn the lid" in terms a computer without a vision system or any force feedback can understand?

Open vs. Closed-loop Control

A *closed-loop* system checks that whatever the Arduino (or other computer) commanded something in the real world to do actually happened. So, one might check turns of a wheel, the angle of a lever, the temperature of a heating element, and so on.

Open-loop control systems do not check things like this. You might be surprised to know that most consumer 3D printers are managed by an Arduino or something roughly equivalent, and run almost entirely open loop. Stepper motor precision (discussed in Chapter 8) is good enough that you can get away with not checking where you are, though there are "end stops" on some axes to allow the machine to find its starting position.

Planning a Program: Flowcharting

If you get a little ways into the robot coding thought experiment we just suggested, you might have already discovered the importance of planning out a coding project in meticulous detail before writing a line of code. Otherwise you may never get something to work. A common way to plan out code is with a flowchart, like the one in Figure 6-1. This diagram shows a program that starts, has some things set up and defined, and then runs as long as nothing makes it stop.

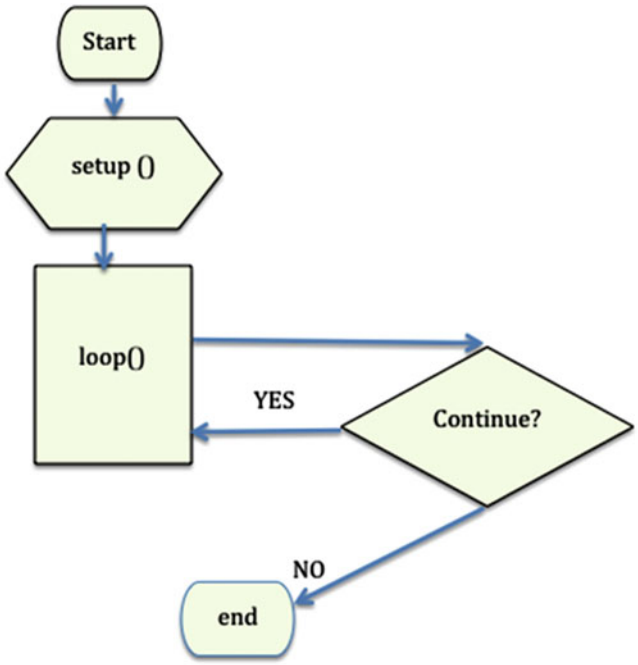


Figure 6-1. *A flowchart*

Joan likes to draw out a high-level flowchart of code first and then write some notes on the hand-drawn sketch. For bigger projects, she will list out all the functions first before starting to code, and then might look for an example to pull pieces out of for what she wants to do once she gets to that stage. For Arduino projects, Rich usually starts by searching online to find a pre-existing open source example that is close to what he wants and then adapts it, or combines several such examples. As a beginner, you will probably use a mix of these styles.

Arduino Code Conventions

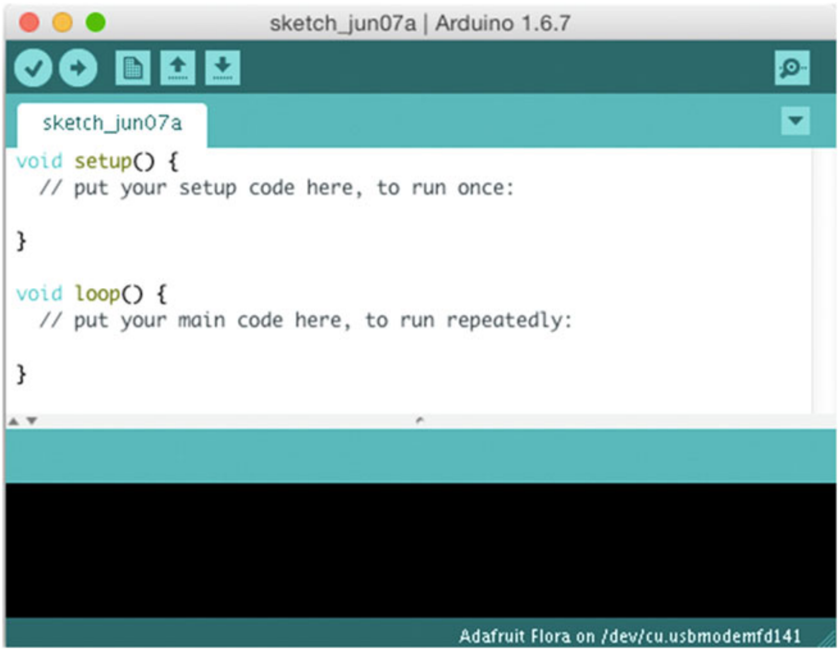
The flowchart in Figure 6-1 actually is how Arduino code is laid out. All programs contain `setup()` and `loop()`, which are functions, or pre-defined words that are special.

■ **Note** Virtually all modern computer code is case-sensitive. `Setup()` has nothing to do with `setup()`, and it is a bad idea to mix names of things that are very similar because it is easy to confuse yourself.

The `setup()` function in an Arduino includes everything you do once. That might include setting up things that need to start at zero, telling the Arduino which pins it might be reading data from or writing data to, and doing similar housekeeping. The `loop()` function is where the action is. The Arduino will keep going around and around in `loop()` (starting over at the top of the code when it gets to the bottom) until something interrupts it (turning off the power, hitting the reset button, and so on).

For historical reasons, programs to run on Arduinos are called *sketches* by some people, and just *programs* by others. The environment we will program in is called the IDE (pronounced "eye-dee-eee"), short for Integrated Development Environment. This environment provides some structures "behind your back" to load and run your programs. Free and open source, the IDE is available at `www.arduino.cc`, along with instructions for installing it.

The Arduino IDE originated as a programming environment called Processing, which was designed to do quick drawings and animations (hence the idea of programs as sketches). Processing is still around (also free and open source) and available at `www.processing.org`. A screen shot of the IDE is shown in Figure 6-2. There are various different types of Arduino-compatible boards, and the IDE will create different code for different boards. We talk about this later in the chapter.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Adafruit Flora on /dev/cu.usbmodemfd141

Figure 6-2. *The Arduino IDE default sketch. Note that it tells you on the bottom right what type board it thinks that you are using.*

■ **Note** At this point we want to stop and acknowledge all the people who have put time and effort into the Creative Commons Arduino ecosystem. You can read more about its history and appropriate licensing for it at `Arduino.cc`.

Format Conventions

Most programming languages follow some conventions that you just have to learn, like vocabulary in a new language. We show some examples to get you started. In general, it is a good idea to start with an example and edit it until syntax is second nature. You might consider making yourself a little template that you always start with, like the following (we explain what a *comment* is in a minute):

```
// Write comments here about what the code does
// Say who wrote it and when the last edit was
// List license terms or any restrictions on how this can be copied
  void setup() {
    // code to be done once here
  }// end setup()
  void loop() {
    // code to be repeated here
  }// end loop()
//end
```

You can see what a similar sketch looks like in the Arduino IDE in Figure 6-2.

Things People Find Intimidating

People often look at computer code and panic, because it looks pretty obscure and they are afraid of breaking something. If you are orderly about it and learn the conventions of how programs are constructed, you will pick it up reasonably quickly. If you think of it as a foreign language created by people rather than something dropped down by robot overlords, you may get there faster.

Syntax Pickiness

Computers do not know what you mean, and they are very literal. Typos in code will generally cause the code to do something you did not intend. Some problems are caught by the *compiler* (the program you run to turn your human-readable code into something the machine can understand), but not everything. Be very careful typing in code from examples. In some languages like FORTRAN and Python, formatting and the position of tabs and line breaks matters to the code execution. The C++ language underlying Arduino coding, fortunately, does not have these issues.

Notice that the curly brackets { and } say, “Everything between these brackets is part of the same block of code.” Different people have different ways they like to use and lay out brackets, name variables, and so on. It is often very hard for people who learn one way to work with people who have learned another, trivial though it sounds. Sometimes the end of a function like `loop()` may be pages away from the start, and Joan learned the convention (shown in the preceding example) of always noting with a comment what piece of code each ending bracket closes. Rich, on the other hand, learned to keep track of these things with precise and consistent use of indentation and white space. Good programming practices include:

- Pick a formatting standard and stick to it. If you are writing code with someone else, come up with a style guide before you start to avoid arguments later on. For example, Rich and Joan are both really bad at consistently using each other’s conventions, and we have to cross-check to avoid errors.
- Flowchart and plan ahead; do not just jump in and start coding. Usually whatever you write without planning needs to be thrown away, particularly in the beginning. Look to see if an open source version already exists, too, or perhaps a *library* (more about that in a later section).
- Avoid single-letter variable names; use names that say what the variable does, like `inputPin`. The lone exception to this is for loop counters (which we get to later in this chapter), which are often, by tradition, single letters.
- Never use a 1 or a 0 in a variable or function name because it is hard to distinguish from the letters l and o.
- Variable names cannot have spaces in them. People get around this with underscores or by using “camel case” (names like `cornerValue`, or `inputFile` with a “hump” in the middle).
- Comment your code. Your future self will thank you!

Programming Vocabulary and Ideas

As we mentioned, the Arduino uses a variant of the C++ programming language, which is a relative of some other languages like C, Java, and Python. We give a brief vocabulary lesson here and then go on to examples.

Comments

Programs usually are not meant to be read like a novel, despite the protestations of Rich and other strong coders who claim their code is “self-documenting.” To understand what is going on, people insert *comments* for later readers. The computer ignores these when the code is *compiled*, that is, translated from human-readable to computer-hardware-readable form. Comments start with a double slash if they are all on one line.

```
// This is a comment.
```

It is conventional to have a block of comments at the top of a program that says what it does, who wrote it and when, and any restrictions on copying it. Comments are useful to leave yourself notes about why you did something, too. You can "comment out" parts of a piece of code by putting // at the beginning of each line you want the compiler to temporarily ignore. Alternatively you can have a multiline comment by starting your comment with /* and ending it with */, like this:

```
/* This is a comment
Blah blah
Which ends here */
```

The compiler ignores anything between /* and */ and so the Arduino or other processor does not see it.

Variables and Loops

A program has *variables*—which Joan likes to think of as boxes where things will be stored. In C++, a variable needs to be given a *type*—is it an integer, floating point, character, or some other type of variable? You can read about variable types that are supported for Arduinos at `www.arduino.cc/en/Reference/HomePage`. As we note earlier in the chapter, variable names are case-sensitive, and `APPLE`, `apple`, and `Apple` are all different variables from each other. Giving a variable a type looks like this:

```
int apples = 15;
```

That code creates the variable `apples` as the integer type and sets its value initially to 15. Note that you only have to define a variable's type once, and then it is set for the entire code. Alternatively you can just give it a type:

```
int apples;
```

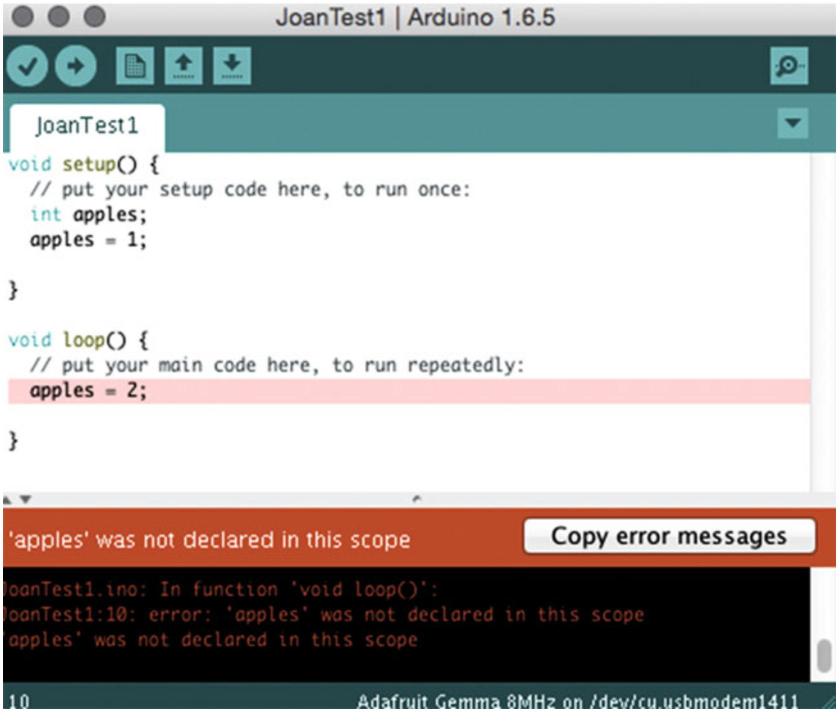
Or you can build the type into more complex code:

```
for (int apples = 0; apples < 10; apples++) {
  // insert code here you want to run ten times
} // end apples loop
```

This fragment is a piece of code that we want to execute ten times. The first time it runs, `apples` would equal 0. Every time we run the code inside the loop, the variable `apples` is incremented by 1 (that is what the `apples++` does). The program will set `apples` to 0, run the code between the curly brackets, increment `apples`, then check whether `apples` is less than 10. It will repeat that process until the answer is no and then it will continue executing whatever comes after the loop. The variable `apples` here is what is known as a *loop counter*— a variable that keeps track of how many times we have done something inside a loop.

Variables have a *scope* as well. The scope of a variable refers to how much of the code will recognize it. If you define a variable up at the top of the code before (not inside the curly brackets of) your `setup()` and `loop()` functions, they will be *global* variables and visible to your whole program. If you define a variable inside the curly brackets that start and end a function, only that function will know about that variable.

Global variables can be a little dangerous in general because you might lose track of what the variable is being set equal to if you are bouncing around between different functions and making assumptions about the value of a particular variable. But things like which pin is an input and an output and similar things are typically set to be global variables. Otherwise, a variable's scope is within the function where it is defined. If you define `int apples;` inside `setup()`, you will get an error if you then use it in `loop()`, as you can see in Figure 6-3.


JoanTest1 | Arduino 1.6.5
JoanTest1
Copy error messages
10
Adafruit Gemma 8MHz on /dev/cu.usbmodem1411

```
void setup() {
  // put your setup code here, to run once:
  int apples;
  apples = 1;

}

void loop() {
  // put your main code here, to run repeatedly:
  apples = 2;

}
```

'apples' was not declared in this scope

```
JoanTest1.ino: In function 'void loop()':
JoanTest1:10: error: 'apples' was not declared in this scope
'apples' was not declared in this scope
```

Figure 6-3. *Scope compiler error*

Reserved Words

Arduinos define a few words that are handy to have around. `HIGH`, `LOW`, `true`, and `false` are all defined. The full set is on `www.arduino.cc/en/Reference/HomePage`. There are also a few oddball conventions just for Arduino, like the fact that names of pins (numbers as well as A0 through A5) are always considered to be int variables, even though they contain letters. Arduino sketches convert these special values into integers for you.

Functions

Functions are sets of commands that you want to do over and over again, or that you want to isolate so that you can pull those commands out and replace them with something else in some circumstances. You can *pass* a value to a function in very predefined ways. The built-in functions `setup()` and `loop()` are "void" functions—they do not return any values to their code when they start.

A function call looks like this:

```
digitalWrite(pin, value);
```

This function call says to use the commands defined for the function `digitalWrite` with two inputs, `pin` and `value`. What function inputs do is usually laid out in the comments. For Arduino built-in functions, the definitions are on the Arduino standards page cited earlier in the chapter. You can also create your own functions. This is handy if you have to do something over and over, but with different values (for example, rotating an arm a variable number of degrees).

Assignments, Comparisons, and Basic Math

Some conventions in computer programs are different than those in algebra class. In programming, an equals sign is an assignment operator—the two sides of the equal sign are *not* the same. For example, the following means "take whatever value *a* is, add one to it, and overwrite the current value of *a* to store the result in it":

```
a = a + 1;
```

Or equivalently:

```
a =  a++;
```

Or simply:

```
a++;
```

Here is a comparison which is asking whether *a* is equal to 1:

```
a == 1
```

Here is a comparison asking whether *a* is *not* equal to 1 is:

```
a != 1
```

Other common comparisons are less than (<), less than or equal to (<=), greater than (>), and greater than or equal to(>=).

In programming languages, multiplication is typically shown by an asterisk, and division with a slash, as in the following respective examples: `a = b * c;  a = b / c;` For more complex math functions, you may need to add a library to your code. See the section on libraries farther along in this chapter.

If, Else, and While

You can have a program do one thing if a certain thing happens, and do something else otherwise. The following code fragment checks to see whether the variable `darkness` is greater than 3. If so, the code calls the function `light()` which does something. Otherwise, we set the variable `alternate` equal to 2:

```
darkness = 5;
if (darkness  >  3) {
  light();
}// end darkness conditional
else alternate = 2;
```

You can also have something happen while a condition holds. The next code fragment calls the functions `light()` and `spin()` as long as `darkness` is greater than or equal to 3. Note that for this to ever end, somewhere else in the code the variable `darkness` would need to get set to less than 3:

```
while  (darkness >= 3) {  //darkness greater than or equal to 3
  light();
  spin();
}
```

Walking Through Some Examples

To download the Arduino IDE on your computer, follow the instructions at `www.arduino.cc/en/Main/Software`. Download the appropriate Mac, Linux, or Windows version. On a Windows machine, you may also need to install drivers corresponding to the type of board you're using. When you launch it, the IDE creates a window that you can use to edit code, like the one in Figure 6-2.

You can also get a lot of examples from the File ➤ Examples menu and look at them without having a physical Arduino board attached. To fake it out that you have a board attached for compiling purposes, go to the Tools ➤ Board menu and select Arduino/Genuino Uno, which is a pretty generic board.

We suggest you take a look at the code in the following examples to see the code syntax and get the general idea:

- File ➤ Examples ➤ 01.Basics ➤ Blink
- File ➤ Examples ➤ 05.Control ➤ ForLoopIteration
- File ➤ Examples ➤ 05.Control ➤ IfStatementConditional

In the next section, we get into code for one regular Arduino example and transition into the wearables from there.

Programming an Arduino

You load programs from your computer onto your Arduino with a USB cable. To tell your computer what sort of Arduino you have, go to the Tools ➤ Board menu to set the board. If you do not see yours listed, use the Tools ➤ Board ➤ Board Manager to find and install specifications for it (you need to be connected to the Internet for this to work).

You may also need to set port with Tools ➤ Port and, in a few cases, you may need to change other settings with Tools ➤ Programmer. The Arduino download page referenced earlier in the chapter offers a lot of assistance for specific circumstances.

How the Arduino Thinks

As mentioned, an Arduino can only do one thing at a time. Unlike a full computer (or even a Raspberry Pi), which can share resources to effectively go back and forth among a lot of tasks running more or less at the same time, an Arduino can just keep track of one program.

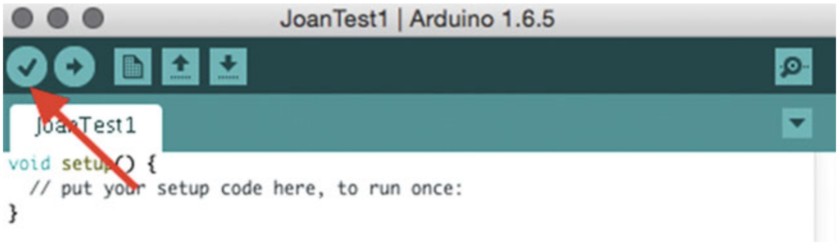
An Arduino program comes in three parts. The first part does not have an official name, but it is the code ahead of the `setup()` function. This is where you define “global” variables (see the “Variables and Loops” section earlier in this chapter). This part does things like bring in libraries of code that someone else has already written, or perhaps defines things to be used later. You can create and set variables in this area, but you cannot do things like turn pins on and off.

The next part of the code is in a function called `setup().` This is for all the things that are done once and is generally where you'll want to set the modes and initial states for your pins.

Finally, we need to run the `loop()` code. This is where the Arduino does whatever you want it to do on an ongoing basis—for example, monitor a sensor, run a 3D printer, or wait for a button to be pressed.

Compiling, Loading, Running

To compile a program and just check that it works, click the check mark icon on the IDE (where the arrow points in Figure 6-4). To compile and load a program onto the Arduino, click the run arrow icon next to the check mark icon. Compiling takes the text version of the code you wrote and converts it to commands the computer can understand.



```
void setup() {
  // put your setup code here, to run once:
}
```

Figure 6-4. *Click the check mark icon to compile code and check that it works.*

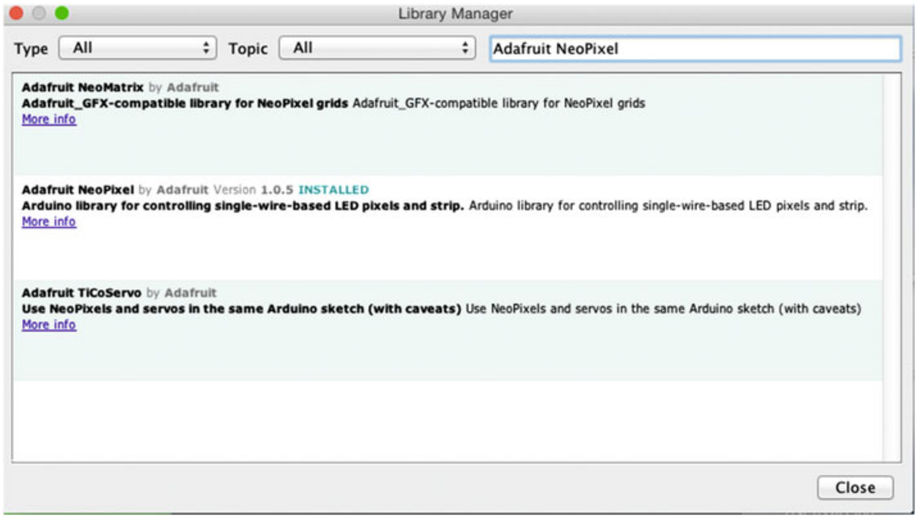
To run a program on an Arduino, attach a USB cable from your computer to your board. Be sure you set the Board and Port (and, for some wearable boards, the Programmer) settings correctly according to the specs for the board, as discussed at the beginning of the "Programming an Arduino" section. Click the "compile and load" right-pointing arrow icon (to the right of the check mark icon) to compile and check the code and upload it to your board.

■ **Note** Once you upload a program onto an Arduino, it is there until you load in something else. Disconnecting the power or resetting the Arduino does not erase code, though pushing the reset button does run `setup()` again.

Adding Libraries

Code libraries are collections of code that someone has already written and is sharing under some terms of use. You can use code libraries in your own code. There is some background at `www.arduino.cc/en/Guide/Libraries` that you may find helpful.

The IDE has a Library Manager that can help you add libraries to your code. If you have a component that says it needs an outside library, go to Sketch ➤ Include Libraries ➤ Manage Libraries and search on what you need. For example, if we search on "Adafruit NeoPixel" (which we use later in the book), we get the window shown in Figure 6-5 (note that you need to be connected to the Internet for this).



Adafruit NeoPixel

Adafruit NeoMatrix by Adafruit
Adafruit_GFX-compatible library for NeoPixel grids Adafruit_GFX-compatible library for NeoPixel grids

Adafruit NeoPixel by Adafruit Version 1.0.5 INSTALLED
Arduino library for controlling single-wire-based LED pixels and strip. Arduino library for controlling single-wire-based LED pixels and strip.

Adafruit TiCoServo by Adafruit
Use NeoPixels and servos in the same Arduino sketch (with caveats) Use NeoPixels and servos in the same Arduino sketch (with caveats)

Figure 6-5. *Installing a library*

If you click one of the library options that come up as the result of a search, you will have an option to install it if you have not already done so. Clicking Install will download the library for you to use on that computer (whether you are connected to the Internet or not). To add it to your code, use the following syntax:

```
#include <libraryname>
```

For example, add the Adafruit NeoPixel library to your code by putting the following line at the top of your file (above anything other than other `#include` lines):

```
#include <Adafruit_NeoPixel.h>
```

■ **Tip** Installing a library will also often add more examples to the `File` ➤ `Examples` menu that you can use to see how to use the library. If no examples appear, clicking the link for more information will usually take you to a Github repository where you can look for a Readme file that may explain how to use the library.

Using Preprocessor Directives

You may have seen code that looks like this at the top of a sketch:

```
#define PIN 3
```

Lines like that are called *preprocessor directives*, or sometimes simply *pound defines*. This construct in this case replaces every instance of PIN (case sensitive) with 3. That is convenient if you want to be able to set a pin number or similar constant everywhere. But it can be a little hazardous. If you had a variable named PINAWAY, it would become 3AWAY, for example, because every time the string PIN appeared in the code it would be replaced with 3. For this reason, it is common practice to use all-caps for names in preprocessor directives, and not for anything else.

Why would you want to use a preprocessor directive, given the hazards? You may want to avoid it as a beginner, because of the possibility of unintended consequences. But for advanced coders, it can be an efficient way to make something that appears in a lot of different places in the code easy to change.

Writing Code For Figure 5-7

In Chapter 5, the example circuit in Figure 5-7 did nothing. The LED did not light up when the circuit was plugged in, because it was only connected to ground. Pushing the button did nothing. Now we are going to add code to the same circuit to make an *inverting circuit*—if you push the button, the light goes on. If you are *not* touching the button, the LED is off.

As noted back in Chapter 5, this circuit requires a pullup resistor so that the voltage that pin 2 will see is known even when the button is open. Until the button is pushed, the pin is connected weakly to VCC, but pushing the button creates a strong connection to GND and allows a small amount of current (inversely proportional to the value of the resistor) to flow. The chip used by the Arduino board has internal pullup resistors (about 200 kΩ) that you can activate by writing HIGH to a pin that's in input mode. Listing 6-1 shows the code.

Listing 6-1. Program to Run the Circuit in Figure 5-7

```
//Code will turn on the LED if the button is pressed.
//Otherwise the LED is off except for initializing it on

int buttonPin = 2;
int ledPin = 10;
int press = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);
  pinMode(buttonPin, INPUT);
  digitalWrite(buttonPin, HIGH); // activates internal pull-up resistor
} // end setup
```

```
void loop() {
  press = digitalRead(buttonPin);
  if (press == HIGH) digitalWrite(ledPin, LOW); // button NOT pushed, off
  else digitalWrite(ledPin, HIGH);
}// end loop
```

Arduino Input and Output

Arduinos need to both be able to read the voltage on their pins and write out a voltage to make things happen, as we just saw in the previous example. Note that *in* and *out* are from the Arduino board's perspective (that is, *out* is writing to a pin, and *in* is reading from one). This means that if a sensor is sending data to an Arduino, it is a digital `INPUT` to the Arduino, which Joan feels is backwards. But there it is.

Before doing anything else with a pin, you should tell the Arduino whether you want to use it as an input or an output using the `pinMode()` function (this is typically done in `setup()`). There are several different kinds of input and output on an Arduino. You tell the Arduino which pins you are using with the `digitalRead`, `digitalWrite`, `analogRead`, and `analogWrite` functions.

Digital Read and Write

You can configure any numbered pin on the Arduino to send or receive a digital signal. A *digital* signal can have just two values—in this case, called `HIGH` and `LOW`—as opposed to an *analog* signal, which can be any voltage between some upper and lower bounds.

Pins defined as an output can be set to `HIGH` (which connects the pin to the Arduino VCC) or `LOW` (connects the pin to the Arduino GND). Writing a value to a pin can be used to change the voltage across another component (for example, to turn an LED off and on). It can be used in specific patterns to send control signals to other devices, though in most cases a library handles this communication protocol for you.

If a pin is defined as an input, you can use it to read a voltage, which will be either `HIGH` (close to VCC) or `LOW` (close to GND). This can be used to do things like check the state of something like a button or switch. A digital input may also be used (again, usually by a library) to receive digital signals back from other components, such as a sensor, or sometimes a device that sends a response back after a digital signal has been sent to it. For instance, an Arduino may write data to an SD card and request data back from the card.

Analog Read and PWM

An Arduino can read an analog voltage on its analog pins. These pins are referred to as A0, A1, and so forth. They can be used as input pins that do not just read `HIGH` or `LOW`, but can read a range of voltages between VCC and GND using a built-in analog-to-digital converter (ADC). Most Arduinos have a 10-bit ADC, which means they can read 1024 (2 to the 10th power) distinct voltages ranging from 0 at GND to 1023 at the analog reference voltage. By default, the Arduino uses a reference voltage that is equal to VCC, though it can be configured to use a different reference voltage using the built-in Arduino function `analogReference()`.

There is also an analog write function, though the Arduino does not actually have a digital-to-analog converter (DAC). Instead, the `analogWrite()` function generates a pulse-width modulation (PWM) signal that is often even better. The PWM signal sent by `analogWrite()` is an oscillation between VCC and GND hundreds of times per second, and the analog value specifies the proportion of the time that the voltage is `HIGH` in each cycle (known as the *duty cycle*). The PWM function takes an 8-bit value, with 256 distinct values (0–255). A value of 0 represents a 0% duty cycle (`LOW` all the time), and 255 represents a 100% duty cycle (`HIGH` all the time). Giving it a value of 127 would result in a 50% duty cycle (`HIGH` half the time, and `LOW` the rest of the time).

A PWM signal can be used to do things like dim an LED or signal a motor to turn more slowly. This is actually more useful than sending a lower voltage because PWM has a more linear relationship to LED brightness or motor speed. PWM can also be used to signal devices such as hobby servos that expect a PWM signal (servos expect a pulse 1–2 milliseconds long every 20 milliseconds to determine their position).

The Arduino can maintain these pulses while the program is doing other things when the `analogWrite()` function is used, but only on certain pins. On most Arduinos, pins 3, 5, 6, 9, 10, and 11 (and sometimes others, depending on which chip is used) can be used for PWM, and they are usually marked with a tilde symbol (~) next to their pin number.

Writing to the Serial Port

Sometimes it is handy to see what values a particular pin is getting as input. You can write that out to your computer over the USB cable by enabling the serial port. To do that, you click on the magnifying glass icon on the right side of the IDE (you can see it on Figure 6-4). Add the following line to the `setup()` function of your Arduino program:

```
Serial.begin(9600);
```

Then, at the point in your `loop()` function where you want to see the output, add `serial.print` or `serial.println` (adds a line feed after whatever you want to print). `serialprint` or `serialprintln` can transmit one number or string (set of letters) at a time. You can see a full example at File ➤ Examples ➤ 0.3Analog ➤ AnalogInOutSerial. Once your program is running while your Arduino is connected over USB, click the serial window icon (the magnifying glass) to open the serial monitor, and be sure that the baud rate at the bottom of the window matches the number you used in `Serial.begin()`.

Programming Sewable Boards

The common Arduino-derived sewable boards (like the Flora, Gemma, and Lilypad) work more or less like a basic Arduino. In the case of the Gemma and some other smaller boards, there are some tricks to uploading code, which we describe shortly.

A bigger difference involves the components, which are more sophisticated by and large than their non-sewable counterparts. They tend to come with appropriate resistors built-in, for starters.

You can use non-sewable sensors and the like in a wearable piece by curling up the ends of the leads you would have pushed into breadboard holes and sewing on the resulting loops. Obviously, you need to think about things like being able to remove these non-washable components, but on the other hand they are a lot cheaper.

The NeoPixel sewable LEDs (which we use in later chapters) are coming down in price and are good for incredible effects, since both their brightness and color (they are really sets of three tiny LEDs, one each of red, green, and blue) can be controlled for many pixels using a single digital pin. More recent Flora boards have an integral NeoPixel attached to pin 8.

■ **Note** Getting code onto a Gemma is a little tricky. You need to connect the USB cable and press the reset button (a tiny black button on the board). When the red light on the board is flashing, you can upload code. If it is not flashing, you cannot.

To use NeoPixels, you must bring in the Adafruit library for the NeoPixels. To find it you, will need to select "All" in both pulldown menus in the Library Manager (see Figure 6-5). Once you have downloaded the libraries and the board definition, click Tools ➤ Board ➤ Adafruit Flora, Port ➤ (Flora port) so your computer will recognize the board. If the Flora is not shown as an option, use the Board Manager to find it as described in the section "Programming an Arduino."

When you bring in the Adafruit NeoPixel libraries, you will get some pretty complicated examples. Listing 6-2 shows a simpler one you can play with which is pretty close to the minimum code that will change the color of the NeoPixel in a systematic way.

You change the color and brightness by changing the values of `i, j,` and `k` in the function `pixels.setPixelColor(0, pixels.Color(i, j, k))`. The value of `i` corresponds to how bright (from 0 to 255) the red LED should be. The value of `j` sets the green, and `k` sets the blue.

An equal mix of all three colors gives you white light. For example, `pixels.setPixelColor(0, pixels.Color(100,100,100);` will give you white light, whereas `pixels.setPixelColor(0, pixels.Color(0,0,200);` will give you bright blue light. After each time you use the `pixels.setPixelColor` function, you also need to use `pixels.show()` to command the NeoPixel to actually display the color you picked.

Listing 6-2. Flora and NeoPixel Example

```
#define PIN   6
#define PIXELCOUNT   1
#include <Adafruit_NeoPixel.h>
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(PIXELCOUNT, PIN, NEO_GRB +
NEO_KHZ800);

int persist = 50; //keep pixel lit for 50 ms
```

```
void setup() {
  pixels.begin(); // This initializes the NeoPixel library.
  Serial.begin(9600);
}

void loop() {

  //Do a test with just one
  pixels.setPixelColor(0, pixels.Color(0,0,200)); //bright blue
  pixels.show(); // Pushes the value just set out to the pixel
  delay(5000); // Delay for a period of time (in milliseconds)

  for(int i = 0; i < 50; i = i + 5){ //red
    for(int j = 0; j < 50;j = j + 5){ // green

      for(int k = 0;k < 50; k = k + 5){// blue
        // pixels.Color takes RGB values, from 0,0,0 up to 255, 255, 255
        // Next three lines were for debugging
        Serial.print(i);
        Serial.print(j);
        Serial.println(k);
        pixels.setPixelColor(0, pixels.Color(i, j, k));
        pixels.show(); // Pushes the value just set out to the pixel
        delay(persist); // Delay for a period of time (in milliseconds).
      } // end k
    }// end j
  }// end i

}// end loop
```

Similar code using this same function can control a series of NeoPixels. The `0` in the example in Listing 6-2 means that you are controlling pixel number 0 (in this case, there is only pixel 0). If you had two pixels (as in Listing 6-3), you would have values `pixels.setPixelColor(0, pixels.Color(0, 200, 0))`, and `pixels.setPixelColor(1, pixels.Color(200, 0, 0))` would turn the first pixel in the series green (0, 200, 0) and the second, blue (0, 0, 200).

This code is shown in Listing 6-3. Note that every NeoPixel needs to be separately connected back to VCC and GND. The signal connections are chained one behind the other, so that the in arrow of one connects to the out arrow of the previous one.

Listing 6-3. Driving Two NeoPixels in Series

```
#define PIN  6
#define PIXELCOUNT   2
#include <Adafruit_NeoPixel.h>
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(PIXELCOUNT, PIN, NEO_GRB +
NEO_KHZ800);
```

```
void setup() {
  pixels.begin(); // This initializes the NeoPixel library.
  Serial.begin(9600);
}

void loop() {

  // Test using two pixels.
  // The in arrow of pixel 0 should be connected to pin 6
  // The in arrow of pixel 1 should be connected to out arrow of pixel 0

  pixels.setPixelColor(0, pixels.Color(0, 200, 0));  //bright green
  pixels.show(); // Pushes the value just set out to the pixel
  delay(5000); // Delay for a period of time (in milliseconds)
  pixels.setPixelColor(1, pixels.Color(0, 0, 200));  //bright blue
  pixels.show(); // Pushes the value just set out to the pixel
  delay(5000); // Delay for a period of time (in milliseconds)
}// end loop
```

Summary

In this chapter we went over how to program a wearable Arduino. We started off by introducing how to write computer code in the first place and then discussed programming an Arduino in particular. We ended with some ideas on how to program a wearable piece, and went over some issues with this type of board.

CHAPTER 7

Your First Project

Creating a wearable electronics project requires you to juggle several different areas of expertise at once. Chapters 3 and 4 focus on sewing techniques, and Chapters 5 and 6 introduce electronics and software. This chapter walks you through a (relatively!) simple project that you can adapt to a variety of different applications and that combines aspects of all those chapters. For the most part, Lyn leads you through this chapter, although Joan and Rich chime in some with suggestions for circuit design and programming.

In this chapter you will learn how to make an apron with an embedded one-minute timer. It is a relatively simple circuit, but it brings up a lot of design issues that you will need to think about in general. The one thing we want you to take away from this chapter is that planning out the project before you do anything is the key to success. In Chapter 10 (after we teach you about sensors and 3D printing) we talk you through an overly ambitious first wearable project that we just jumped into without planning very much ahead of time. This chapter is the sober, more realistic version of a first project, designed knowing what we know now.

The Egg-Timer Apron

You are holding a party for friends and want to keep track of things you have cooking in the kitchen. Wouldn't it be nice to have something you could just tap lightly to time something for one minute? For our simple example we will create an apron with a built-in timer. When you tap a bit of conductive cloth, a NeoPixel will turn red while the timer is running, then turn green for a period of time you specify once the time is elapsed, and then turn off. The whole thing is controlled by an Adafruit Gemma processor, which is something of a stripped-down Arduino (see Chapter 5 for details) powered by two coin cell batteries. Figure 7-1 is what the apron looks like when it is done, with the Gemma and its battery in their protective pocket.

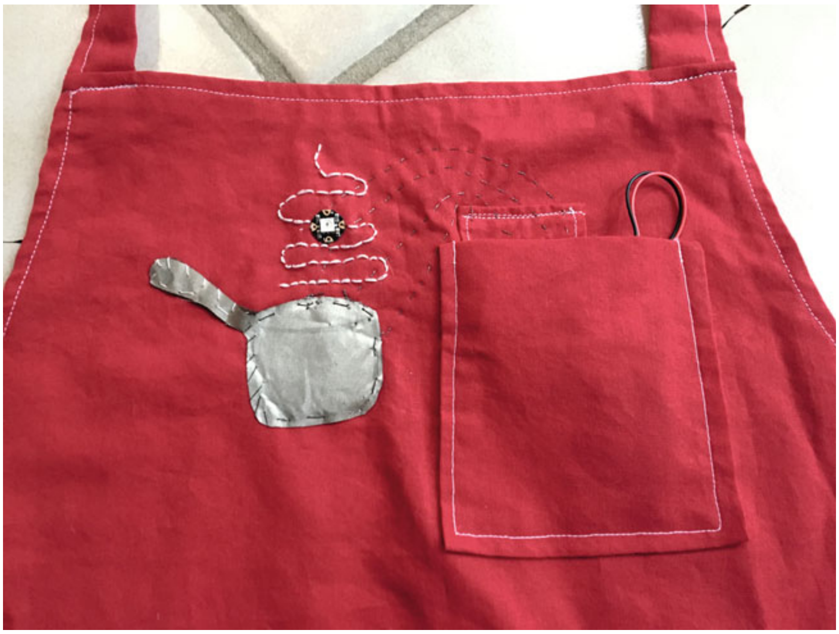


Figure 7-1. *The completed egg-timer apron*

The saucepan on the apron is made of three layers: a top layer of conductive fabric, a middle layer made of felt with a hole in the middle, and a bottom layer also of conductive fabric. When you press on the top layer, it connects to the bottom layer through the hole in the felt. This allows for a very imprecise ability to start the timer with your wrist or anything dry. (Note that the apron should not get wet while you have it turned on.)

For this project, you will need the following materials and tools:

- 1 to 1.5 yards of light to medium weight cotton fabric
- Spool of thread and bobbin
- Paper for pattern and/or tailor's chalk
- Scissors
- Measuring tape
- Cotton webbing for ties (optional)
- Bias tape (optional)
- Embroidery needle
- White and black embroidery thread
- Small piece of felt
- Gemma microprocessor

- NeoPixel LED
- Conductive fabric
- Conductive thread
- Two (3-volt) coin cell batteries
- Battery holder (designed for two coin cell batteries)
- Five alligator clip leads (for testing purposes)

■ **Note** This chapter runs through a simple way of doing this project. In Chapter 11 we talk about several projects and give you some tips about ways to make this project power efficient. We focus the implementation here on being easy to create, debug, and understand. However, this project will use up batteries pretty quickly unless you disconnect them (we guess a couple of days, depending on how often you push the button).

Planning the Project

The temptation with a wearable project is to just start sewing something. However, it is best to resist that impulse and first figure out what you want your control circuit to do. Then you can lay out the circuit, first with Fritzing (see Chapter 5) and then with alligator clips, and write the code.

We suggest doing all that before you start sewing, in case you realize you need another component (and a place to attach it) or realize the circuit is going to need to be laid out in a particular way that might be more convenient when the garment is not assembled yet.

■ **Tip** Joan and Lyn both find it a lot easier to visualize where the circuits will go with an alligator clip version of the circuit than with a Fritzing diagram, even if the clipped circuit can be a little unwieldy sometimes. Rich does not agree, so these things are a matter of style and visualization skills. You will develop your own path with experience.

Because this was going to be used in an environment where we would want to protect the Gemma, we decided to hold the Gemma and its connections on a little tab that could slide into a pocket, along with the batteries. The “saucepan button” required some careful design to make it easy to push but hard to push accidentally.

Control Design and Software

In this case, we wanted the circuit to do nothing as long as the timer had not been started. Once the timer is started, the NeoPixel glows red. When the timer has finished, the NeoPixel glows green for a second amount of time. Both amounts of time are specified at the top of the code in `#define` statements. If you wanted to have the circuit do more—maybe cycle through several colors for different amounts of time—you could add more calls to the two Adafruit library functions (see Chapter 6):

```
pixels.setPixelColor(0, pixels.Color(r, g, b));
    pixels.show();
```

r, g, and *b* are numbers between 0 and 255 that determine how bright a red, green, or blue pixel illumination you want. Note that the relative brightness of these three primary colors determines the hue of the resulting color. The entire Gemma sketch is shown in Listing 7-1.

■ **Note** Listing 7-1 has the timer set to run for just one second and then leaves the NeoPixel on for two seconds. We suggest you leave it that way for the debugging process and then make it whatever value you want (a minute would be 60,000 milliseconds). `TIMER_DELAY` is the length of the timer countdown (during which the NeoPixel is red) in milliseconds; `PERSISTENCE_DELAY` is the amount of time that the NeoPixel stays green after the timer is done.

Listing 7-1. Apron Program

```
// Program to turn on a neopixel red when a button is pushed
// And then green after a specified delay passes
// And finally turns off after a second specified delay

#include <Adafruit_NeoPixel.h>

#define NEOPIXEL_PIN 2 // pin controlling the Neopixel
#define BUTTON_PIN 1 // pin reading the state of the button
#define TIMER_DELAY 1000
  // in milliseconds - one second for the demo
#define PERSISTENCE_DELAY 2000
  //time to wait before turning off, in milliseconds

Adafruit_NeoPixel pixels =
  Adafruit_NeoPixel(1, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);
```

```
/*
  Last parameter depends on the neopixel version you have
  See https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library
  Remember for Gemma you need to push reset button
  to load in your program (no loading after red light stops flashing)
*/

void setup() {
  pixels.begin();
  pinMode(BUTTON_PIN, INPUT);
  digitalWrite(BUTTON_PIN, HIGH);
  /*
    BUTTON_PIN is set to INPUT with the
    internal pull-up enabled and is
    connected to the first layer of the button (one closest to apron
fabric).
    When the button is pressed,
    this pin is connected to GND and goes LOW,
    and the circult begins its countdown.
    Here we use pin D1 which will light up the red LED
    when D1 is HIGH (the button is NOT pushed)
  */
} // end setup

void loop() {
  if (digitalRead(BUTTON_PIN) == LOW) {
    pixels.setPixelColor(0, pixels.Color(0, 150, 0));
    pixels.show();
    // first have red light while timer is counting down
    // this lets you know button was pushed and the timer is running

    delay(TIMER_DELAY);
    pixels.setPixelColor(0, pixels.Color(150, 0, 0));
    pixels.show();
    delay(PERSISTENCE_DELAY);
    // turn green after timer is done
    //and stay green for PERSISTENCE_DELAY seconds
  } else {
    pixels.setPixelColor(0, pixels.Color(0, 0, 0));
    pixels.show();
    // pixels turned off
  }
} // end loop
```

Laying Out the Circuit

Once we have the sketch written and know how everything will work, we are ready to lay out the circuit. Figure 7-2 shows the Fritzing diagram for this project. This circuit uses a Gemma board, a NeoPixel, and two pieces of conductive fabric arranged so as to be equivalent to a button switch.

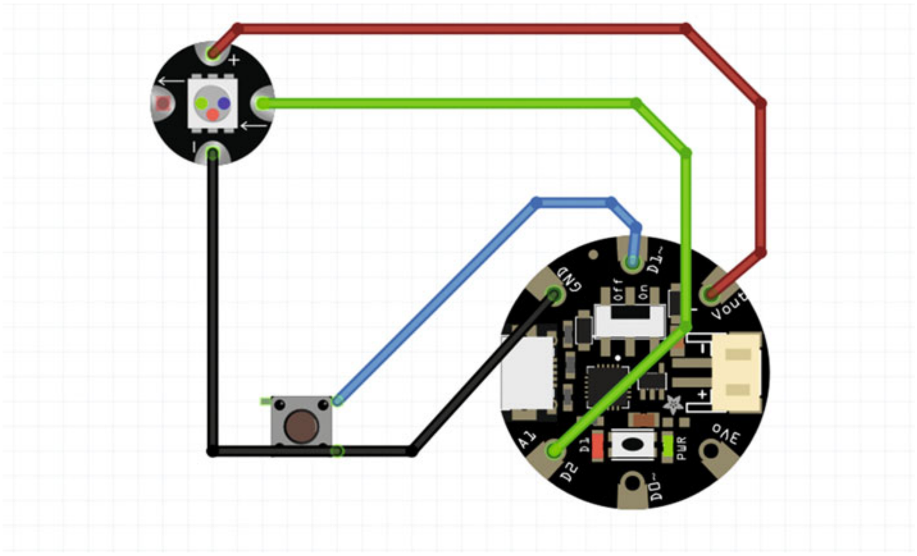


Figure 7-2. *Fritzing diagram of the circuit for the apron*

It works like this: the Gemma sets up pin D2 as an output that will control the state of the NeoPixel. The ground wire of the NeoPixel is tied back to the top of the conductive fabric "button." Pin D1 is a special pin that lights up the attached onboard light if it is set to `HIGH`. We use this pin to detect the state of the conductive-fabric button as follows:

- When the switch is open, the red board light is on. Pin D1 is set to `HIGH` by default by the code and is acting here as a pullup resistor (see Chapter 5).
- When the switch is closed, pin D1 is pulled down to ground, the Gemma sends a signal to the NeoPixel to turn itself on, first one color and then the next. After the timers clock out for the two timed lights, the Gemma shuts the NeoPixel off by setting the brightness of all three colors to zero.

Figure 7-3 shows how this looks laid out with alligator clips. The red wire goes from Vout to the + on the NeoPixel. The blue wire goes from Gemma pin D2 to the input of the NeoPixel (->). The grey wire goes from the ground (–) side of the NeoPixel to the top piece of conductive fabric, which the black wire connects to GND on the Gemma. Finally, the green wire goes from D1 on the Gemma to the piece of conductive fabric forming the bottom of the button. (We used two scraps here to simulate the top and bottom of the saucepan button we create in the project.)

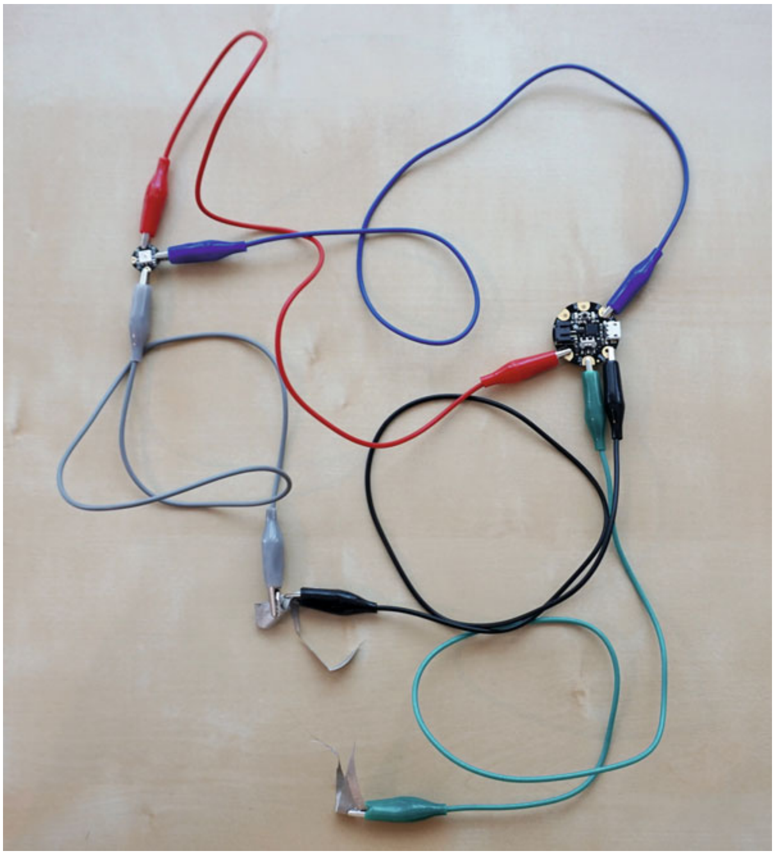


Figure 7-3. *Circuit testing with alligator clips*

By touching the green wire's conductive fabric to the piece connecting the grey and black wires, we pull pin D1 down from `HIGH` to `LOW`, and the code running on the Gemma turns on the NeoPixel.

Debugging

Once you are happy with the alligator-clipped circuit, you can upload the sketch to it (see details in Chapter 6) and see if it works as advertised with power supplied via the USB cable to the Gemma. If not, check that your alligator clips are firmly attached, that they are attached to the correct places, and that your USB is getting power to the Gemma (the green light on the Gemma board itself, which is labeled PWR, should be on).

■ **Tip** Gemmas are a little finicky to get started. You need to upload code right after you apply power, while the red light is flashing because the "bootloader," which allows the Gemma to receive code from the USB, is running—we cover this in Chapter 6. Otherwise it will not load correctly. If you are too late, you can get the bootloader to run again by hitting the (very tiny!) reset button.

You can also use a Flora for this circuit. If you have older (v1) NeoPixels, you will need to change the last parameter in the line `Adafruit_NeoPixel pixels = Adafruit_NeoPixel(1, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800).` This last parameter depends on the vintage of NeoPixel you have. For details, see `https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library`. For v1 NeoPixels, change `NEO_GRB + NEO_KHZ800` to `NEO_RGB + NEO_KHZ400`.

Sewing the Apron

Once you feel you have the circuit and software under control, it is time to make the apron and sew the circuit onto it. You can look at the alligator-clipped version and think about how it is going to lay across the fabric. We highly recommend you read all the way to the end of the chapter before starting any of this so that you have a mental picture of where you are going. We are going to switch to Lyn narrating the creation of the pattern now, since she is the expert here.

Creating a Pattern

Start by measuring from about 2 inches below the top of your sternum to where you want your apron to end. I am five foot three inches tall, so if I want the top of my apron to start below my sternum and to fall to about 3 inches above my knees, I need the following measurements—you can vary to fit your height. I measured and cut my fabric 12 inches wide across the top of the bib, and curved out to make it 24 inches at its widest part (at the bottom). In my case, the measurement from top to bottom of the center of the apron was 28 inches. This allows for approximately an inch on every side for double hemming, or you may want to use *bias* tape (strips of fabric you can purchase to finish off a garment), in which case you would probably only add a half inch to each edge.

Using the diagram in Figure 7-4, create a pattern out of tissue or butcher paper, or draw it with tailor's chalk directly on your fabric. The body of the apron should be about 24–27 inches wide, and the bib can be 8–10 inches wide after hemming. It all depends on how much of your torso you want to cover with the apron. Fold the fabric in half so the *selvage* (finished borders of the fabric) edges are together. Draw half of your apron out on the fabric, or use the paper pattern you have created, and cut out your apron. Using the fold will allow you to have even curves on both sides.

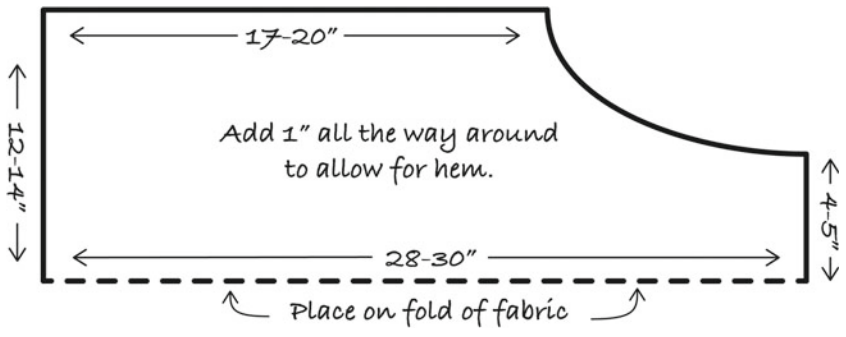


Figure 7-4. *Pattern for the apron (scale up as needed)*

We are making a small pocket on the left side of the bib (meaning on your left as you are wearing it) to hold the battery and Gemma. The saucepan button will be sewn just to the wearer's right of this pocket. The small pocket will be 2.5 inches wide and 4 inches deep, so we will cut a piece of fabric 3.5 inches wide and 5 inches deep, to allow for hemming (Figure 7-5). The large pocket on the body of the apron will be 6 inches wide by 7 inches deep, so we will cut a piece 7 inches by 8.

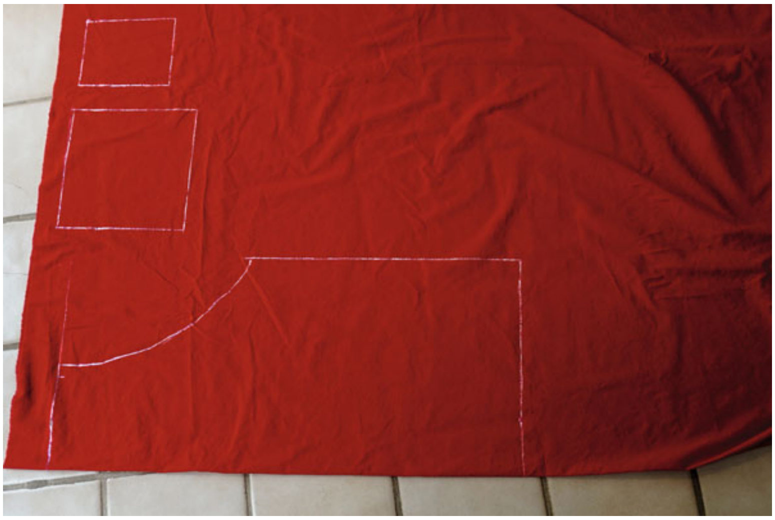


Figure 7-5. *Apron pattern laid out with chalk. Fold is along bottom of the photo.*

If you want to make the strap and ties out of the apron fabric instead of cotton webbing, cut one piece of fabric 4 inches wide by 22–24 inches long and two pieces of fabric 4 inches wide by 24–26 inches long. Make sure to cut these pieces on the straight grain of the fabric, which is parallel to the selvage edge—otherwise your apron will be cut across the weave of the fabric, and the fabric will not fall evenly or may bunch up.

Creating the Saucepan Button

We will cut two pieces of conductive fabric and one piece of felt in the shape of a saucepan to create the timer button (Figures 7-6 and 7-7). The felt will insulate the conductive fabric, except in the center. We will cut a hole in the felt to allow the conductive fabric to make contact when we press the center of the saucepan. The “steam” can be created by pieces of felt, fabric sewn on, or by embroidery (see how it looks in Figure 7-1).

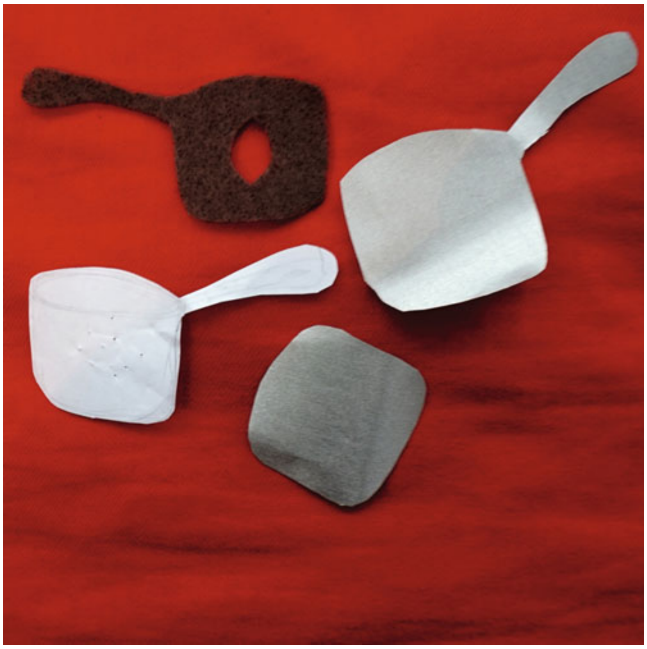


Figure 7-6. *The felt inner part of the button (top left), the saucepan top (conductive fabric—top right), the paper pattern (bottom left), and the conductive fabric bottom of the button (bottom right)*

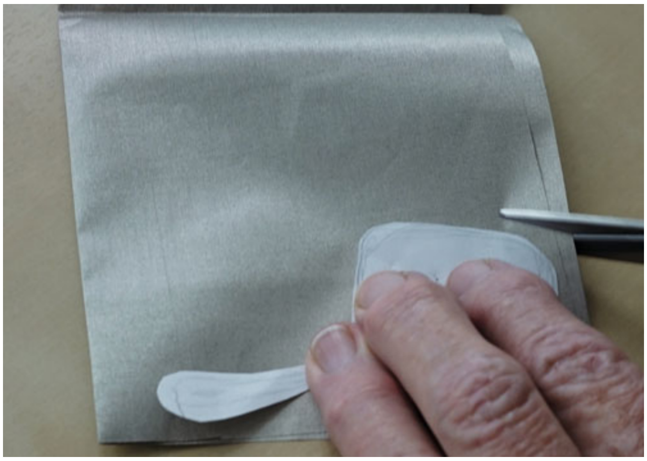


Figure 7-7. *Using a simple paper pattern to cut out the top of the saucepan.*

Creating the Pockets

Using tailor's chalk, measure and mark where you want the pocket s to be on the front side of the apron. Remember to add half an inch on all sides that you will fold over when hemming around the edges of the pockets. First, sew around all the edges of the pockets to prevent the fabric from fraying. Then, fold the top edge of the pockets down half an inch on the wrong side of the fabric and press. Sew across the edges. Turn the fabric right side out, using the tip of your scissors or something with a tip to gently poke the corners out. Press the sides and bottom with an iron to create a half an inch hem (see Figure 7-8).

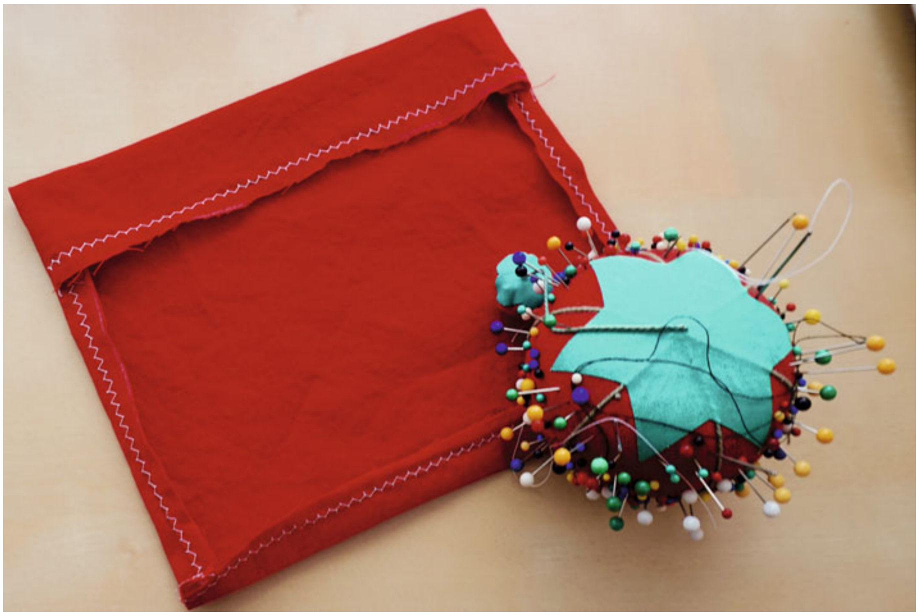


Figure 7-8. *The finished-off piece—this will be the inside of the large pocket.*

Pin the pockets in place on the apron. Sew from the top of one side of each pocket to the bottom and turn the fabric without removing it from the machine. Sew across the bottom, turn the fabric again, and sew up the other side. Remember to add a few back and forward stitches when you begin on the first side and when you finish sewing the other side (Figure 7-9).

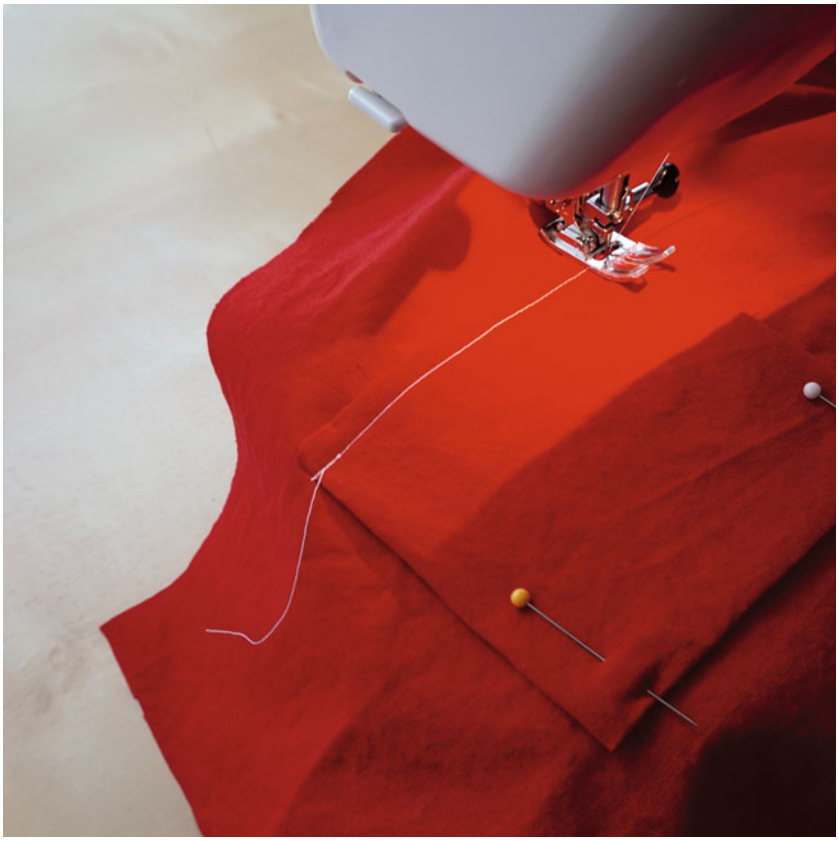


Figure 7-9. *Sewing on the pocket near the bottom—the other pocket goes on the same way*

Hemming the Edges

Sew around the edges of the whole apron with a zigzag stitch to prevent fraying (a zigzag is shown in Figure 7-8). Press the edge under half an inch all the way around the apron. Sew a straight stitch to secure it.

Sewing the Ties and Neck Strap

Fold the edges of the ties and strap into the center, fold them over again lengthwise (Figures 7-10 and 7-11), and tuck them under similarly at the ends. Sew the edges and the ends of the ties and the strap. Pin them in place and sew them onto the apron. It is a good idea to sew over the attached edges several times to prevent them from coming off.

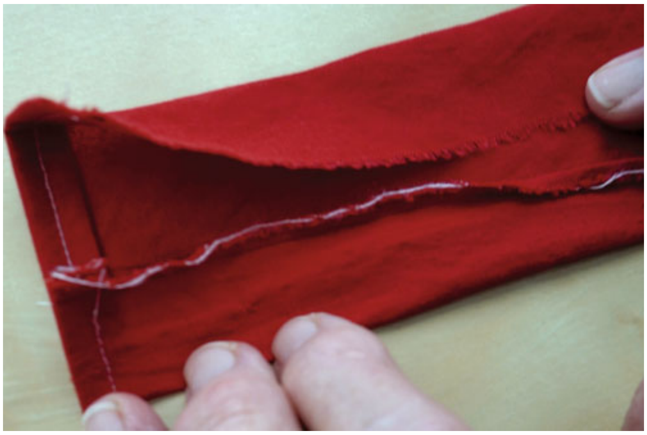


Figure 7-10. *The first fold of the ties or strap*

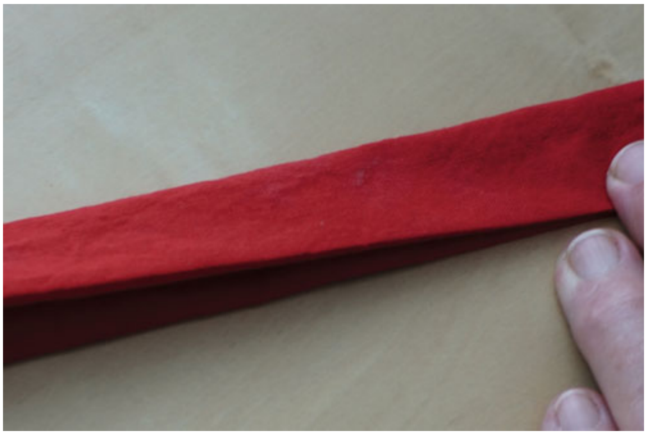


Figure 7-11. *The final fold (it will be stitched on one side to finish)*

If you use cotton webbing for the strap and ties, cut them to the same length as the fabric versions and hem all the ends. Pin them to the apron and sew them on.

Assembling the Saucepan Button

Take the pieces of conductive fabric and the piece of felt in the shape of the saucepan that you cut and set aside earlier. Trim the felt to be slightly smaller than the conductive fabric version. Cut one more piece of conductive fabric without the handle of the saucepan and trim it to be slightly smaller than the felt piece. Cut a hole out of the center of the felt saucepan. This will be your button to add time to the timer. Hand sew the bottom layer of the saucepan pieces (the one without the handle) onto the apron. You can see the pieces back in Figure 7-6 and on the apron as I finish up in Figure 7-12.

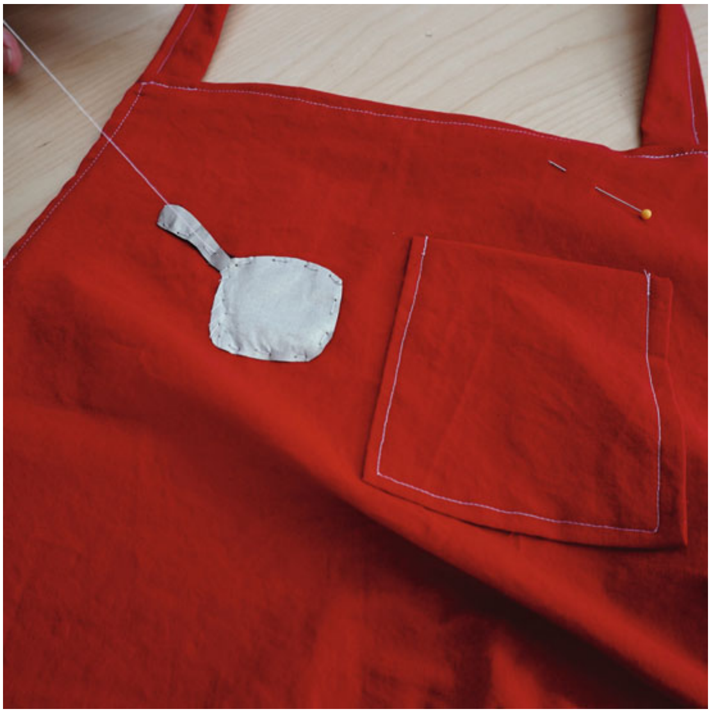


Figure 7-12. *Finishing up the saucepan (assembling the pieces in Figure 7-6)*

Sewing on the Electronics and Circuit

Cut a small rectangle of the apron fabric 2.5 inches by 4.5 inches. Using a zigzag stitch, sew around all of the edges, turn each edge under a quarter inch, press down and hem with a straight stitch. This will create a placket to hold the Gemma and will allow it to slip in and out of the pocket for easy access (Figure 7-13).

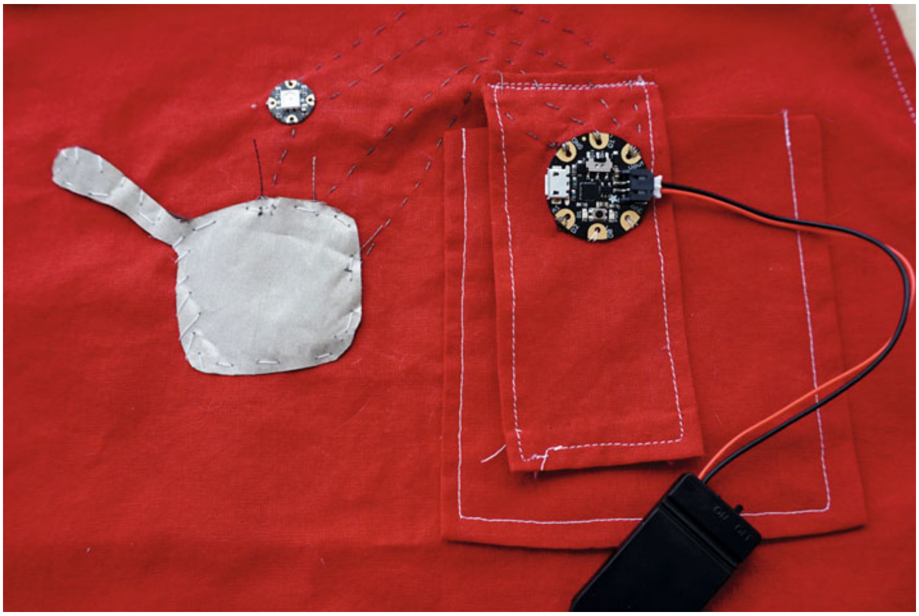


Figure 7-13. *Layout of the electronics placket, battery holder, Gemma, NeoPixel, and conductive hand stitching*

Use the 3Vo and D1 holes (which we are not otherwise going to use) to sew the Gemma onto the placket with regular—not conductive—thread. Figure 7-13 shows the apron with all the rows of conductive thread that this section describes.

The NeoPixel will attach to the wearer's right of the pocket that will hold the Gemma and batteries. Using the "arrow out" hole in the NeoPixel, use regular (not conductive) thread to sew the component onto the apron. The –, +, and "in arrow" will be sewn with conductive thread later on.

Using the Fritzing diagram back in Figure 7-2, lay out your components on the apron bib and lightly draw your circuits with tailor's chalk onto your fabric. These will be the paths you sew to connect the components with conductive thread.

■ **Note** This circuit requires a pullup resistor on pin D1, because we want the input on D1 to be `HIGH` until the button (saucepan) has been pushed, and `LOW` (at ground voltage) when the button is pushed. In other words, we want to complete the circuit when the button has been pushed. We write the input pin `HIGH` in the code to enable the chip's internal pullup resistor (this would not work if we tried to connect the button to 3Vo, because the chip does not have an internal pulldown). See Chapters 5 and 6 for more on this.

You are going to connect the components as follows:

- Bottom layer of conductive fabric to D1 on the Gemma
- Top layer of conductive fabric to GND on Gemma
- "In" Arrow on the NeoPixel to D2 on the Gemma
- + on the NeoPixel to Vout on the Gemma
- – on the NeoPixel to the top layer of the button (which is connected to GND)

We will now walk through each of these conductive thread runs in turn. All the runs are visible in Figure 7-13 except for the connections that run on the inside of the apron or under the placket.

First Conductive Thread Run

This run connects the bottom of the saucepan button to D1 on the Gemma. Thread your needle with conductive thread and tie two or three knots at the end of the long side of the thread. Stick your needle up through the edge of the saucepan closest to the Gemma placket and pull the thread through until the knot is tight. Do not sew too close to the edge of the conductive fabric because it will cause the fabric to fray.

Using small, straight stitches sew a path to the top of the placket and sew down the side to the D1 pin on the Gemma. Stick your needle up through the hole and pull the thread taut. Stick the needle down through the placket very close to the edge of the Gemma pad and pull the thread taut. Make two or three stitches like this to create a secure connection. Do not attach the placket to the pocket.

When you are making these stitching runs with conductive thread, make sure to pull the thread tight around the connection to the Gemma and NeoPixel, but not tight enough to pull the fabric out of shape. A connection that is too loose will not work properly.

Make a knot on the underside of the placket by making three tiny stitches in the same place. Cut the conductive thread, leaving a small tail. After all the connections are made, use clear nail polish to seal the knots (conductive thread is made from wire and does not like to stay in a tight knot). Allow the nail polish to dry and cut off any tails of thread.

Second Conductive Thread Run

This one goes from the top of saucepan to Ground (GND) on the Gemma. Use the same techniques from the previous run for knotting the conductive thread and sewing the connection from the edge of the top layer of the pan to the top of the placket. Make this run parallel to the first one, without allowing the conductive thread runs to touch each other. Sew down the top of the placket to the GND pin on the Gemma, attach to the pad, and create a knot.

■ **Note** The holes on the NeoPixel are smaller than the holes on the Gemma. Check to see that your needle will pass through them with conductive thread before you start the next run.

Third Conductive Thread Run

This run goes from the "in" arrow on the NeoPixel to D2 on the Gemma, and is a little tricky because it needs to pass under the Gemma on the placket. Knot the thread, stick the needle up through the "in" arrow pin from the wrong side of the apron, pull it taut, and stick it down through the fabric very close to the pad. Repeat this action two or three times. Using small stitches, create a run to the top of the placket above the center of the Gemma, making sure it does not touch the other runs already created. Sew down to the Gemma and under it to the bottom D2 pin. Do not allow the stitches to go through the pocket. Attach to the pin and create a knot.

Fourth Conductive Thread Run

This run connects the + on the NeoPixel to Vout on the Gemma. Repeat the same process as in the third run, sewing a run to the top of the placket and down to the Vout pin on the Gemma. You do not need to sew underneath the Gemma for this connection. Attach to the pin and knot the thread.

Fifth Conductive Thread Run

Finally, you will connect the – on the NeoPixel to the top layer of the saucepan. Attach to the – pin and make a short run of thread straight down to the top layer of the saucepan. Make sure you only sew through the top layer of conductive fabric, not through the felt or bottom layers. Create a knot at the edge of the pan.

Finishing It Off

Draw a swirl of steam onto the apron with tailor's chalk. Using an embroidery needle and white embroidery thread, follow the chalk lines with small, straight stitches to create the steam effect. You could also use black embroidery thread to create details on the saucepan. However, we were concerned both about making spurious connections through the button and also about causing the conductive fabric to fray, so we left well enough alone.

Check your connections to make sure they are tight enough and not touching any other connection. Use clear nail polish to secure all the knots and trim any tails of conductive thread.

Put the coin batteries into the holder, plug it into the Gemma board, and check to see if everything is working. If it isn't, doublecheck for any areas of possible shorts in the conductive thread runs and the conductive fabric. Push the reset button on the Gemma (assuming you loaded the code onto it before you sewed it, as summarized in the earlier Note in this chapter). Voila! An apron with a timer—nice work!

Practicalities

We built this example around an apron because it is easy to sew and we thought it would be fun to show off if you were entertaining. The same circuit will work for sports applications where you might want to show a couple of colors for lap split times or other timing applications.

If you really used this as an apron (and therefore would want to wash it a lot), you would sew the circuitry onto a placket with snaps or hook and loop attachments so you could wash the rest of the apron while keeping the electronics out of it. Adafruit says its Gemma processor can be washed as long as you remove batteries; check your manufacturer's specs if you use hardware other than that shown here. The apron should not get wet when the battery is attached, so you should consider it more as something to wear as a conversation piece rather than something to wear in the thick of messy cooking.

This simplistic code will eat a battery faster than is necessary because it never goes into any sort of sleep mode. Sleep mode is tricky on a Gemma because it has a limited ability to handle interrupts (signals that tell it to stop what it was doing and resume doing something else).

Arduinos in general are not really computers—they can only do one thing at a time, unlike your laptop or desktop, which can be sharing resources across different programs at once. Gemmas have a particularly difficult time dealing with multitasking, for a variety of technical reasons. It would also be interesting to try to add more time by pushing the button repeatedly, but that would require filtering out spurious button presses (known as *debouncing*), which is outside the scope of this chapter.

Summary

This chapter walked you through a simple example that used skills from the previous chapters on sewing (Chapters 3 and 4), electronics (Chapter 5), and coding (Chapter 6). The key takeaway is that the order in which you do a project is important. Certainly you should do some planning first, but you should also decide in detail what the project is going to do, lay out any circuits with alligator clips, and write and test the code—in that order.

Only then should you start cutting fabric, because once you understand what your electronics are going to do and how circuits will need to be routed, it is likely that you will need to adjust the garment. We talked through all these steps in detail for an apron that incorporates a timer which lights a NeoPixel one color while the timer is running, and another when time is up. This project should give you an appreciation of the issues you will need to consider in the upcoming chapters covering more complex components and projects.

PART III

Beyond the Basics

Chapters 8 through 11 explore more sophisticated topics than those covered in the earlier chapters. Chapter 8 shows you how to use a few different types of sensors to make your creation react to its environment, and shows you where to go next if you want your project to be able to move on its own.

Chapter 9 summarizes the 3D-printing process and gives pointers on where to learn about it in detail. Chapter 10 shows you what can go wrong (and did, for us) when an overly-ambitious first project tries to do too much at once.

Finally, in Chapter 11, you will find two bigger projects that come at wearable tech from opposite directions. The first is a big sewing project (the dress on the cover) that uses electroluminescent (EL) ribbon to light up the boundaries between fabric blocks, but requires no coding and minimal circuit design. The other project requires more circuit design and a substantial piece of code, but minimal sewing. It takes an off-the-shelf hat and adds a sewn circuit to make it light up red if you shake your head no and green if you nod yes. Thus you can pick a substantial project based on where you feel most secure.

CHAPTER 8

Sensors and Other Hardware

Accessories that light up are pretty cool, but we can go a step farther. In this chapter Joan and Rich introduce you to a few different types of *sensors*—components that can detect some property of the real world, like temperature or light level. You can then program your Arduino to take the sensor reading and do something based on it. For example, in Chapter 11 we teach you how to make a hat that lights up different colors based on the angle of your head. In this chapter Joan and Rich talk you through some common sensors and how you might use them.

You can also make soft objects or clothing that can move on its own—the "haunted dress" described in Chapter 10 is an example of this. But as we point out in that chapter, clothing that moves is pretty challenging. Using motors is more difficult than we really want to get into in detail in this book (mostly because of the power requirements), but we give you an overview of the issues and suggest some places to learn more in the "Making Things Move" section of this chapter.

Sensors

There are now many Arduino-compatible sensors on the market, devices that measure anything from ambient temperature and light levels through percentage of methane present in the surrounding air. Adafruit and Sparkfun sell many different kinds, and online retailers like Amazon offer a variety as well. One challenge, though, is that the hobbyist DIY user (that would be us!) is a miniscule market compared to the consumer products that these sensors are intended for, so the documentation sometimes can be sketchy at best.

There are far fewer sewable sensors, and the ones that exist tend to be relatively expensive. However, the wearables are intended to be sewn on, and may be sturdier and more appropriately documented. What you decide to do will depend heavily on your application, experience level, and budget.

We try out a photodiode, used for measuring ambient light, and a thermistor, for measuring temperature. Some amount of hunting around is involved to find out what to do with any particular sensor. In this chapter, we walk through examples of the process.

Chapter 11 has an example of a gyroscope in a hat that knows the angle of the wearer's head. We talk about gyroscopes, accelerometers, and magnetometers briefly in this chapter and then in Chapter 11 we apply some of that knowledge.

■ **Note** We avoid needing to solder in the projects in this book. However, if you want to use sensors other than the ones we describe here, soldering might be the most practical option. We will show some ways to prototype without it, but the resulting circuits will be fragile. If you want to do a lot of work with sensors or (more so) with motors, you probably should find an online tutorial on soldering (for example, `https://learn.sparkfun.com/tutorials/how-to-solder---through-hole-soldering`) or perhaps a class at a makerspace near you.

Creating a Circuit with a Sensor

Depending on the sensor, you may need to know a bit about how a sensor works before you start searching for specifics, since there may be a standard equation to use to convert the signal the sensor reads to the actual temperature, pressure, or whatever the quantity of interest is. Many sensors (including thermistors and another similar type of resistive sensor called a photoresistor) change resistance in response to their environment—in the case of a photoresistor, to light levels.

Because of that, they can act as part of a voltage divider. The sensor is placed in series with a known resistor (usually specified by the device's manufacturer), and one reads an output voltage at a point between the two resistors. However, it can be a little tricky, because the voltage drop across a sensor that is a variable resistor is a function of *both* the series resistor's resistance *and* whatever the sensor's own resistance is at the moment (as well as the fixed voltage across the whole divider).

The only voltage drop it "sees" (the voltage difference between its two leads) is the one it is creating, determined by whatever it is sensing. This makes picking the right *series resistor* (the "other resistor in the voltage divider") crucial. Fortunately, there are some standards to help us.

Other sensors work differently. A photodiode, for example, produces a small current in response to light. We will see one of these in action shortly.

Sewable Sensor Alternatives

In this chapter we lay out our two examples with a "regular" Arduino (not a sewable one) and low-cost breadboard plugin components, because that way it is a lot easier to see what is going on, and because as of this writing a lot more sensors are available in this format. If there is a sewable version of the sensor, there will likely be a very explicit tutorial telling you what pins to use to connect it and a sample Arduino sketch you can start with. Typically, the wearable sensors include the series resistor we talk about here as part of the component, and there may be other simplifications.

If there is no wearable version, you have a few options. If you are not actually wearing the project (if it is a piece of cloth art, or a theater prop, or a nightlight) you might just use a regular non-sewable Arduino and create a base or some other enclosure to hide the electronics and sensors. When designing such an enclosure, though, bear in mind that the sensor will need to be exposed to the ambient temperature or light or whatever environmental thing you are trying to measure. In that case you can either breadboard the circuit, as shown in this chapter, or create an equivalent soldered circuit.

But if you want to have a sewable circuit with non-sewable components, you can use a sewable electronics board (a Gemma, Flora, or Lilypad). Curl up the leads of the resistors and LEDs into loops and sew them onto the fabric. Or you can solder the circuit together and find a way to support it on the fabric, depending on your particular design (see our examples in other chapters for ideas). Be careful with these connections, though, because wire-to-board soldering with stranded wire can be brittle. Stranded wires may break where they are soldered, though a little hot glue to anchor the insulation will help. If you use solid-core wire instead in your soldered construction, it will tend to break with repeated flexing, and may break in the middle of a cable. Solid wire is better for applications where flexing is not required.

■ **Caution** Although there is “an” Arduino standard, some of the wearable boards depart slightly from it, or the libraries for the sensors might use a capability that is not compatible with some of the sewable boards. For example, the chip used by the Gemma board is not, as of this writing, supported by the communications library underlying the libraries for most of the Flora sensors. Before you buy a board and a sensor that is not explicitly branded as working together with it, you might do a bit of online searching to see whether there are known problems (or ways to work around known problems). Not all sensor libraries (which we get to in a minute) support all Arduino-derived boards.

Many sensors have resistance that changes in response to their environment. In the first example, we use a thermistor, which is a resistor that changes resistance as the ambient temperature changes (hence the name, *therm*(al) res(*istor*)).

In the second example, we use a photodiode to see whether the surrounding atmosphere is dark enough to turn on an LED nightlight. A photodiode is a simple sensor that is essentially an LED in reverse—it generates current when a light shines on it, rather than lights up when you run an external current through. We will read the amount of current being produced by putting it in series with the Arduino’s internal pullup resistor and seeing how much it has changed a voltage after a short delay.

Both sensors (and a force-sensitive resistor, described in the next section) are shown in Figure 8-1. The main issue with these in a classroom setting is avoiding losing them, as you can imagine. Pill bottles or boxes with separators for sorting work well to store them once they are out of the original packaging.

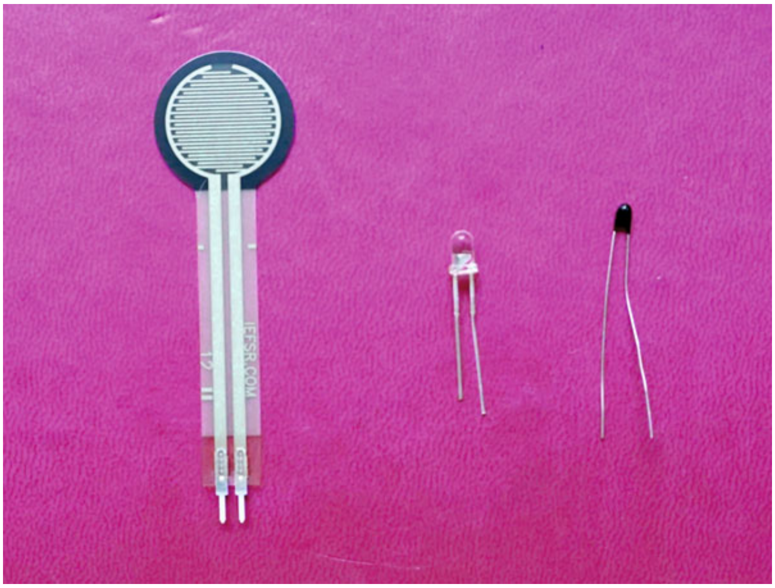


Figure 8-1. *A force-sensitive resistor, photodiode, and thermistor*

■ **Tip** It is important to keep track of where you ordered your parts from so you can track down spec sheets or examples of how to use them. Take a screenshot or bookmark the page and develop a filing system to keep these so you can go back to them.

Some Other Common Sensors

There are too many sensors for the Arduino environment available now for us to list them all. This section can only give you a taste of some of the common types of sensors.

Infrared sensors are photodiodes or *phototransistors* (another type of light-sensitive semiconductor that may provide a stronger signal) that are tuned to respond to lower frequencies than visible light. They are usually dark in color to filter out visible light. These are the types of sensors used by most TV remote controls and can create a simple way for one device to communicate with another. Other types of infrared sensors can be used to detect heat at a distance, like the onese used in non-contact thermometers.

Force or pressure can be sensed by several types of devices. A *force-sensitive resistor* (FSR) is a simple pad of a conductive plastic or rubber that becomes more conductive when it is compressed. Though an FSR is not precise enough to measure weight or the magnitude of a force, it can be used to determine whether a force is present or not, like a big button. FSRs do not handle continuous force well, though, so if you're planning to stand on top of one all day, it may break down prematurely. A variation on an FSR is a *soft potentiometer*, which will give you a different resistance depending on where you press on it.

If you start searching online for Arduino sensors, you will discover exotic fare like tilt switches, vibration sensors, gravity/acceleration sensors, magnetic field sensors, distance sensors, humidity sensors, muscle sensors, heartbeat sensors, radiation sensors, and many more.

■ **Note** In Chapter 11 we use a sewable sensor—a gyroscope that can detect motion. We decided it would be easier to understand in context of a project and so this chapter does not go into it. If you want to read about detecting and acting on motion of a garment, see the hat example in Chapter 11.

Thermistor

A thermistor is a resistor that varies a lot in resistance with temperature (usually a bad thing, if you were counting on your resistor keeping its value at different temperatures). When you buy a thermistor, you usually get a model number, something called a B (or beta) value, the maximum power it is rated for (usually not relevant when used in a voltage divider because the current is negligible), and a value for its reference resistance at a particular temperature (usually 25 degrees C).

In the case of the Negative Temperature Coefficient (NTC) Thermistor MF52-103 we use in this example, the rated power is 0.05 watts; it has a resistance of 10 kΩ at 25 degrees C; and the B value is 3950 K. But what does all that mean?

To proceed from here, you could search online for the type of product and "Arduino." This approach has a lot of obvious problems:

- What you find might be wrong.
- You might not find anything.
- It might not be clear enough what the Arduino sketch out there is doing to feel comfortable that the sensor will work predictably.

As a rule, more expensive sensors (the ones that cost about $15 each, versus the ones that are one hundreth or so that price) will come with better documentation and libraries. We have the cheap ones here to illustrate the process.

Using the Thermistor

Searching on “thermistor” and “Arduino” takes us to the very useful website `http://playground.arduino.cc/ComponentLib/Thermistor2`. The site tells us that the circuit for the thermistor is just a voltage divider (Chapter 5). We will put the thermistor in series with another 10 kΩ resistor to form a voltage divider, as shown in Figure 8-2 in diagram form and in Figure 8-3 as a photograph of the circuit.

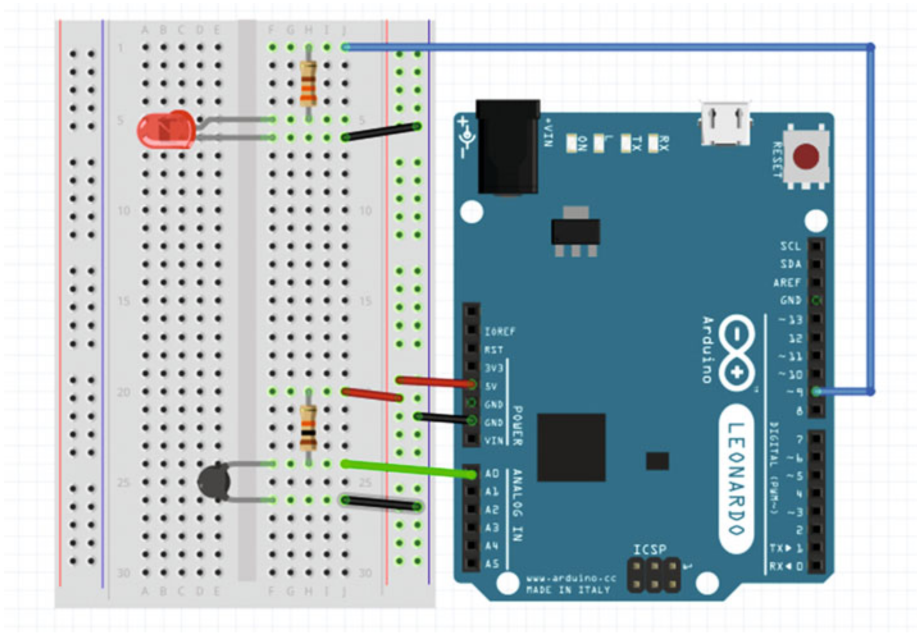


Figure 8-2. *Fritzing diagram of thermistor circuit*

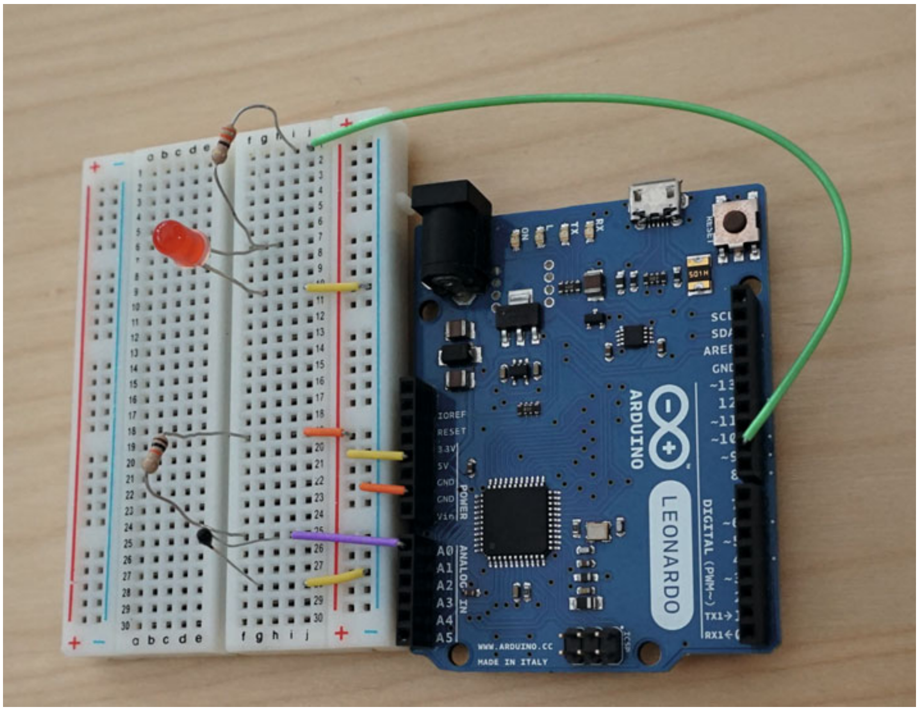


Figure 8-3. *The thermistor circuit in real life*

On the Arduino Playground website just mentioned, we learn that thermistors have a resistance defined by the *Steinhart-Hart equation* (`https://en.wikipedia.org/wiki/Thermistor#B_or_.CE.B2_parameter_equation`) which uses three standard parameters for an equation that converts resistance into temperature:

- B, measured in units of degrees Kelvin, K (in this case, 3950 K)
- Baseline temperature (in degrees Kelvin, which really means degrees above absolute zero—usually the baseline temperature is 25 degrees Celsius, which is 25 + 273.15 = 298.15 K)
- The sensor's resistance at that temperature (this is the nominal resistance of the thermistor, in this case, 10 kΩ).

For the example here, we have also added an LED that will be on when the thermistor reports a temperature within a given range. The circuit in Figures 8-3 consists of a 330 Ω resistor in series with a red LED. The red LED circuit is connected to pin 9 and ground. When pin 9 is HIGH, the LED will come on.

The voltage divider (see Chapter 5) is read between the 10 kΩreference resistor and the thermistor. Its value is picked off between the two and sent as input to the Arduino on the A0 pin. This is an example of how you might set up a sensor to turn on a light or otherwise send a signal to do something to the Arduino. Note that the sensor is outputting to an analog pin, which is necessary to have values other than just LOW and HIGH on the pins.

We want to get an actual temperature out of this sensor and not just a relative number, so getting the actual conversion right is important. For other types of sensors, we may only care where a value falls relative to some arbitrary value (like the light level in a room, or the sensor's own average values over the last minute), so calculating the correct value in specific units may not be necessary.

Arduino Sketch to Interface with the Thermistor

Now that we know how to hook up the circuit, we can go two ways. We can go into our Arduino IDE (see Chapter 6) and see if we can find a library where someone has already created this Arduino sketch. We can do that by going to Sketch ➤ Include Library ➤ Manage Libraries and searching on "thermistor" as described in Chapter 6.

If you find something, you can then choose to Install the library. If several choices appear, you might click the "More info" link first to see which of several choices you like better, or look at the ReadMe file if it has one. Most of the time these "More info" links take you to a Github repository, a common open source way to share an Arduino sketch. Joan likes to see the source for such things in detail and so will usually pick the one that seems the most transparent in what it is doing.

When you install the library, with luck it will also install some examples. To establish whether this is the case, after you install the library, click File ➤ *Examples*.

This may all sound good, but you may discover that you cannot find a library that matches your particular board and sensor hardware. In our case, for this example we went back to the `http://playground.arduino.cc/ComponentLib/Thermistor2` page we started on. There were several examples there. We decided to try the final one because it seemed to be clear about what its assumptions were and it was easy to check that the assumptions were right by cross-checking with the Wikipedia page that described the equaiton; we thank the author of this page for putting his or her Arduino sketch out freely.

Listing 8-1 shows this Arduino sketch, with an additional piece that turns on an LED in the circuit if the temperature is between a given range.

Listing 8-1. Thermistor

```
// A sketch that turns on an LED attached to LEDPIN
// If the thermistor at THERMPIN is in a given temperature range
// Thermistor calibration from
// playground.arduino.cc/ComponentLib/Thermistor2
// modified to make a bit more general and add output

#include <math.h>
#define THERMPIN A0
#define LEDPIN 9
```

```
// enumerating 3 major temperature scales
enum {
  T_KELVIN=0,
  T_CELSIUS,
  T_FAHRENHEIT
};

// Temperature function outputs float, the actual temperature
// Temperature function inputs:
// 1. AnalogInputNumber - analog input to read from
// 2. OuputUnit - output in celsius, kelvin or fahrenheit
// 3. Thermistor B parameter - found in datasheet
// 4. Manufacturer T0 parameter - found in datasheet (kelvin)
// 5. Manufacturer R0 parameter - found in datasheet (ohms)
// 6. Your balance resistor resistance in ohms

float Temperature(int AnalogInputNumber, int OutputUnit, float B,
  float T0, float R0, float R_Balance) {
  float R, T;
  R = R_Balance * (1023.0f / float(1023 - analogRead(AnalogInputNumber)) - 1);

  //reminder: in C, log is ln and log10 is base 10 log, so this is ok
  T = 1.0f / (1.0f / T0 + (1.0f / B) * log(R / R0));

  switch(OutputUnit) {
    case T_CELSIUS :
      T-=273.15f;
      break;
    case T_FAHRENHEIT :
      T=9.0f*(T-273.15f)/5.0f+32.0f;
      break;
    default:
      break;
  };

  return T;
}

void setup() {
  Serial.begin(9600);
  pinMode(THERMPIN, INPUT);
  pinMode(LEDPIN, OUTPUT);
} // end setup
```

```
void loop() {
  float sensorData = Temperature(THERMPIN, T_FAHRENHEIT,
    3950.0f, 298.15f, 10000.0f, 10000.0f);
  Serial.print("Temperature in degrees F = \t");
  Serial.println(sensorData);
  Serial.println();
  delay(500);

  //turn on LED if the temperature data is between 50 and 75
  if (sensorData > 50 && sensorData < 75) digitalWrite(LEDPIN, HIGH);
  else digitalWrite(LEDPIN, LOW);

}// end loop
```

Photodiode: Night Light Example

A *photodiode* creates a current when light shines on it, making it essentially an LED (see Chapter 5) acting in reverse. Reading this current is a little trickier than reading a resistance, but it can be done with a simpler circuit.

To figure out how to use a photodiode, searching online for "photodiode Arduino" is a good place to start. You will find a few examples, mostly pointing out that really, a photodiode is just an LED that you are sort of running backwards. Instead of running current through it to make it light up, you light it up and a small amount of current will run in the opposite direction than an LED would.

Many of these examples involve using a *transistor* (a component that can amplify a signal) to amplify the output. This would be a difficult circuit to build with conductive thread, so we are going to use a simpler solution. A photodiode has an internal *capacitance* (ability to hold a small charge). If we charge up this capacitance by putting a reverse voltage across the LED, then the current created by the photodiode will discharge the capacitor. If we slow down this discharge with a resistor, we can measure the speed (by sampling voltage after a set time or by measuring the time before it reaches a set voltage) to find out how much light the photodiode is seeing.

Using a Photodiode

People tend to use photodiodes in a relative way—to see if something is "dark enough" or "bright enough" to change the state, or to move toward something bright. So we do not really care what the output is, exactly—we just want to know how it is changing. This means we have to play with setting some variables when we create the Arduino sketch (Listing 8-2). Figure 8-4 shows the circuit in Fritzing, and Figure 8-5 shows the photodiode being shadowed to light up the LED.

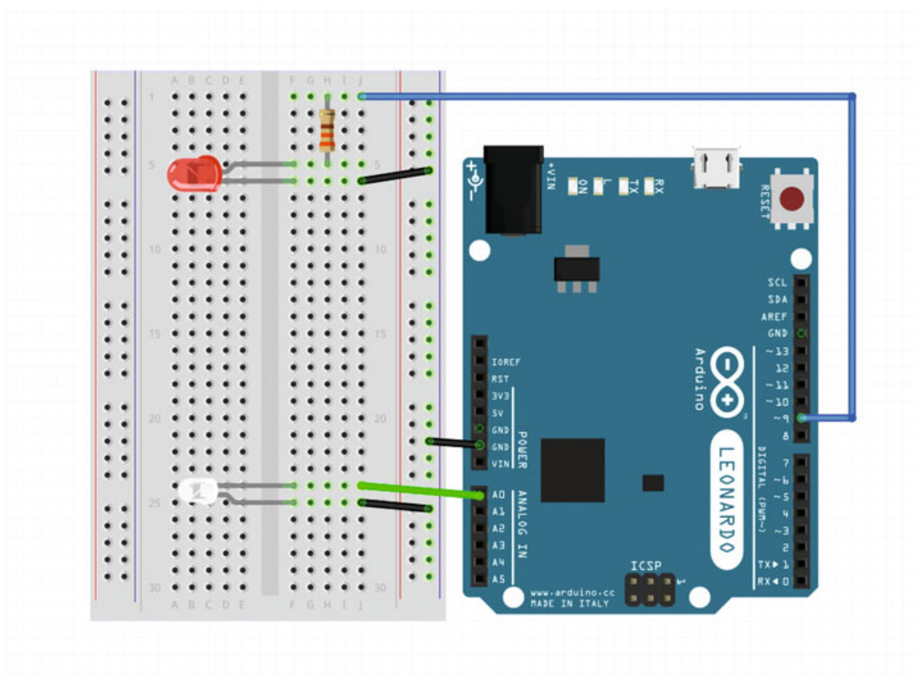


***Figure 8-4.** Fritzing diagram of photodiode and LED nightlight*

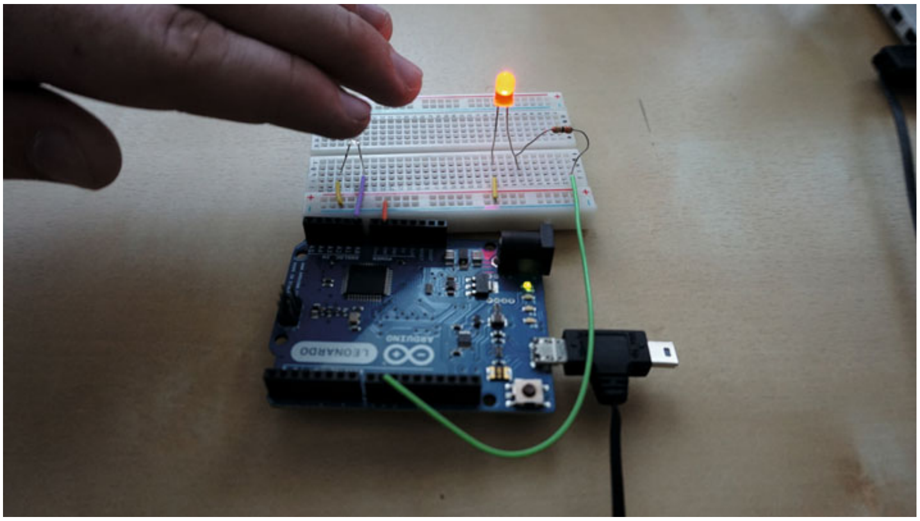


***Figure 8-5.** The photodiode shadowed to light up the LED*

Listing 8-2. Night Light Arduino Sketch

```
// A sketch to light up an LED if a photodiode registers
// that it is getting dark

#define PHOTOPIN A0 //connected to photodiode cathode, anode to GND
#define LEDPIN 9
#define THRESHOLD 1010 //0-1023, use a higher value to trigger when darker
// if the light is always on, THRESHOLD is too high. If it's always off, too
low

int photoRead(int pin) {
  digitalWrite(pin, HIGH); // write high to analog pin
  pinMode(pin, OUTPUT); // fully charge the photodiode's internal
  capacitance
  delay(1);
  pinMode(pin, INPUT); // switch to input with pullup enabled
  // photodiode current will discharge the capacitor, slowed by the 200 k
  pullup.
  // More light makes the voltage drop faster.
  delay(2);
  return analogRead(pin); // see how far the voltage has dropped after 2 ms
}

void setup() {
  Serial.begin(9600);
  pinMode(PHOTOPIN, INPUT);
  digitalWrite(PHOTOPIN, HIGH);
  pinMode(LEDPIN, OUTPUT);
  digitalWrite(LEDPIN, LOW);
} // end setup

void loop() {
  int darkness = photoRead(PHOTOPIN);
  Serial.print("Photodiode reading: ");
  Serial.println(darkness);
  delay(20);

  //turn on LED if darkness is above threshold
  if (darkness > THRESHOLD) digitalWrite(LEDPIN, HIGH);
  else digitalWrite(LEDPIN, LOW);

}// end loop
```

Using More Than One Sensor in a Project

If you are using more than one sensor, normally you will isolate their circuits on separate pins (similar to what you did with the LED and the sensor in these examples) rather than have them in series. Be careful that you do not draw more power from the Arduino than it can provide, but that will not normally be a limiting factor.

■ **Tip** Many types of sensors (and other hardware) are covered at `http://playground.arduino.cc/Main/interfacingWithHardware`.

Making Things Move

An Arduino (including the sewable versions) can control moving parts with one of the following:

- *Small direct-current (DC) motor*: This is used to move something without particularly precise control over where or how fast the motion will be. If you wanted to have a toy car just rotate its wheels and race around, you would probably do it with one of these.
- *Stepper motor*: These are the motors used in 3D printers and other places where precise speed and position control are needed; these motors move in tiny steps (hence the name) of a known size, and their position can be controlled by counting steps.
- *Servo*: This motor wants to hold a particular programmed position. You might couple it with appropriate sensors to hold something in a particular position. A hobby servo has an internal feedback loop that allows it to seek and hold a specified position. Servos contain their own control circuitry and are thus simplest to hook up to an Arduino. They are also commonly used in robotics, when you want to move something precisely.

If you are interested in motors in general, you might look into a book about Arduino-controlled robotics, such as *Arduino Robotics,* by Warren, Adams, and Molle (Apress, 2011). Incorporating moving parts into a garment poses some challenges, as we talk about in Chapter 10. You need to have any motor anchored firmly so that the thing you are trying to move does so safely and as expected, rather than just bunching up the fabric under the motor's attach point.

■ **Tip** There is a long list of motors and and advice on which one(s) to use for various circumstances at `http://playground.arduino.cc/Main/InterfacingWithHardware#Physical_Mechanical`.

Power Management

Compared to just about anything else you would do with an Arduino, controlling a motor takes a lot of power—typically too much to be delivered directly from the Arduino. If you try to power a motor with an Arduino, the Arduino can "brown out" and act unpredictably, or components can get too hot and fail.

To avoid this, use an independent power source other than the Arduino itself. You connect this other power source directly to the motor. You also need to hide batteries somewhere nearby the motor. "Lipstick" USB–connectable rechargeable batteries have become pretty easy to come by now, but you still need a place to put them and a way to connect them.

The easiest way to connect a regular (not sewable) Arduino to a motor is to purchase an appropriate *shield*. This is a circuit board that plugs into the Arduino, and, in the case of motor shields, usually also will let you plug in an independent power source plus the motor. There are no (as of this writing) equivalent sewable boards, so you are somewhat stuck with solutions that involve soldering, or you need to find a way to use a regular Arduino board with a shield on it, which can be a little bulky.

Servos

A *servo* is a small motor that turns its shaft to one particular angular position. Usually a servo comes with a variety of small plastic pieces, called *horns*, that you can use to attach a string or something else you are using the servo to push or to pull. Figure 8-6 shows a servo with a horn attached. This particular horn has two long arms and two shorter ones, but they come in a variety of shapes, and most hobby servos include several different options. Most likely this is the type of motor you would use on a wearable to animate some feature or to react to conditions detected by a sensor (to close up something when it gets dark, for example.)

Powering both a servo and an Arduino can be tricky. A servo can draw significant current, especially when it is prevented from reaching its target position (known as *stall current*). Trying to draw more current than your power source can provide usually results in a current drop (or in some cases, a blown fuse, which may be self-resetting or may need to be replaced).

To use a servo, you want to avoid having the power come through the Arduino's *voltage regulator*, a chip on the board that manages power. This would happen if the motor were plugged in to the Arduino power bus. It may be too much current for the regulator to handle, causing the controller to brown out and reset.

To avoid this problem, most Arduinos have a connection that will allow them to plug into the wall or a battery. Connect your servo using the Vin pin for the positive voltage (V+) side to bypass the regulator, and then plug in the power adapter *before* plugging in the USB connector. Finally, plug in the USB connector so that you can download a sketch to control the servo with the Arduino. (Servo control examples are in the set that is downloaded with the IDE—see File ➤ Examples ➤ Servo.)

In Figure 8-6, note the three wires coming out of the servo. The convention is that brown connects to ground (GND), red connects to 5 V, and orange is the signal (control) wire. (Sometimes black is used for ground and white for the signal wire.) How you hook this up will depend a lot on what you are trying to do exactly and how much power you will need. Take a look at example projects out there that look like yours and see how they did it, bearing in mind the caveats we just reviewed.

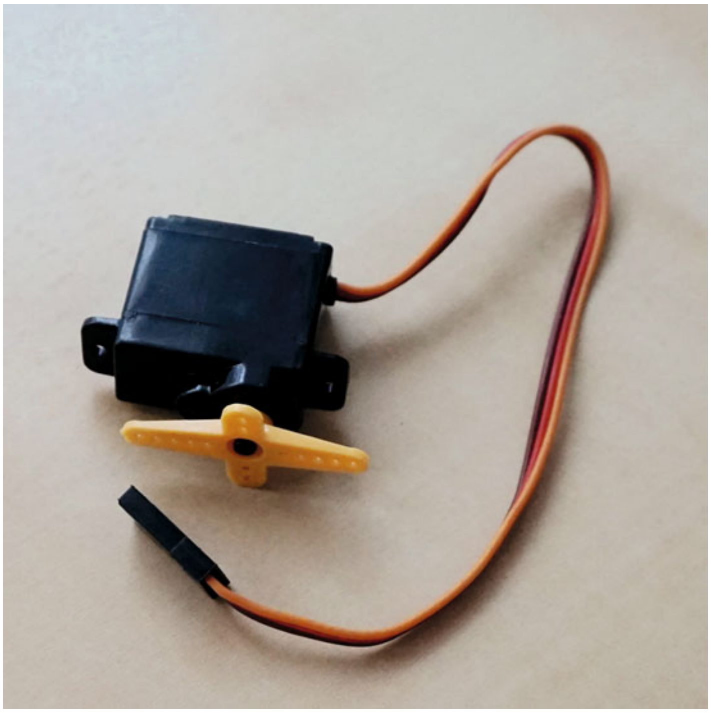


Figure 8-6. *A servo*

Summary

In this chapter we give the general principles behind hooking up a sensor to an Arduino, with details on two examples. We also talk you through how to find more information for a particular sensor, and how to deal with it if a sensor is not available in wearable form. We also give a brief introduction to the different types of motors that an Arduino can control, and include a tutorial on using a servo motor for simple mechanical control. In Chapter 10 we have more complex examples of both, and in Chapter 11 we use a gyroscope to track head motion.

CHAPTER 9

3D Printing

3D printing has entered haute couture in a big way lately (as we touch on in Chapter 13), but much of that has used higher-tech machinery than the consumer 3D printers that are now broadly available. In this chapter Joan and Rich give a quick tour of 3D printing and show you some pragmatic and decorative uses for it.

How 3D Printing Works

In brief, 3D printing technologies create an object by laying it up one thin layer at a time in a process controlled by a computer model. What is being laid up varies by the type of printer (whether it starts with a powder, liquid resin, or spool of filament), but fundamentally all the processes are similar. The layers typically are around 0.2 millimeters thick—which, according to Wikipedia, is twice the diameter of an average human hair or thickness of a coat of paint, or about the length of an average single-celled life form like a paramecium (`https://en.wikipedia.org/wiki/100_micrometres`).

3D printing is not new. It has been around in some form since the 1980s, and the first company, 3D Systems, was formed by Chuck Hall in 1986. However, until the mid-2000s, the 3D-printing process was covered by patents, and the patent holders elected to keep it an expensive technology aimed at industrial users.

When the key patents expired, Adrian Bowyer, then at University of Bath in England, created a design for a 3D printer that could produce other 3D printers, with the addition of some parts that could be purchased at a hardware store. Then, he put this design out as *open source* (meaning it was available to anyone to build on, as long as they, too, put out their improvements for everyone) so that everyone could build on it.

This started the *RepRap* (REPlicating RAPid prototyper, `www.reprap.org`) printer revolution that led to the array of existing consumer printers, fueled by freely available technology and later by the fact that the historical accident that crowdfunding also came along shortly thereafter.

Note The term *3D printing* is somewhat misleading because people immediately make an analogy to printing a document on paper, and the processes do not really have all that much in common. The more generic *additive manufacturing* is more descriptive and accurate, although perhaps not as catchy.

J. Horvath et al., *Practical Fashion Tech*, DOI 10.1007/978-1-4842-1662-0_9

Consumer printers, for the most part, use spools of *filament* made of plastic (see the section "3D-Printing Materials" later in the chapter). Figure 9-1 shows a typical small 3D consumer printer and its spool of filament. This printer, as it happens, was designed by Rich. It has four small motors: one to drive mechanisms in each of three dimensions, and one to push the filament through the *extruder*.

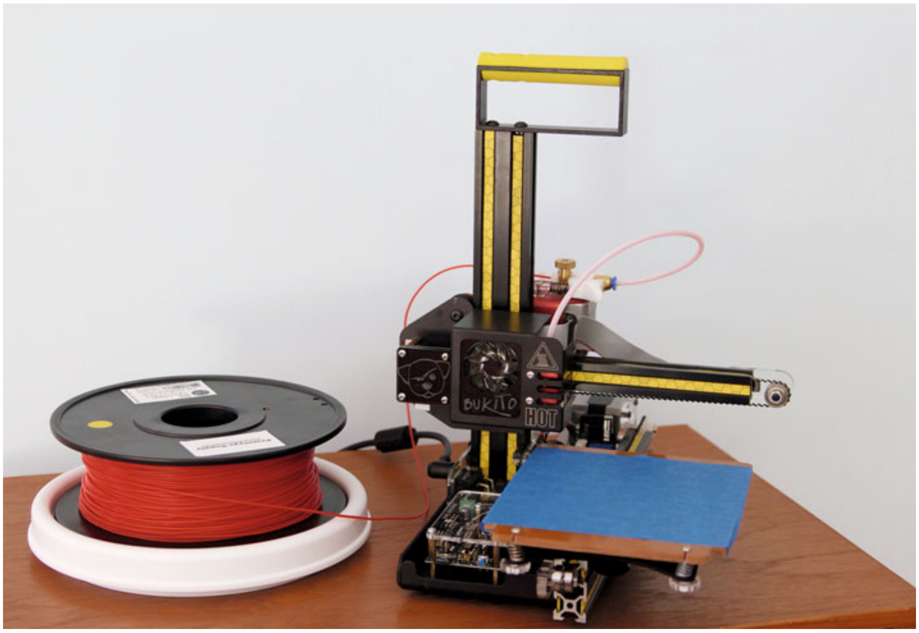


Figure 9-1. *A consumer 3D printer*

The extruder consists of two pieces. One is the *drive mechanism* (which includes a motor and parts that grip the filament). The other is the *hot end*, where the filament is melted and put out through a *nozzle* (in this case, half a millimeter in diameter) that lays down a fine stream of melted plastic. The object builds up a layer at a time on the platform (the area covered by blue tape in Figure 9-1.) Like many consumer printers, this one is controlled by a microprocessor that is more or less an Arduino with some custom features added.

Resin printers use light to selectively harden a light-sensitive liquid resin. The consumer-priced ones are becoming more common and are particularly useful for jewelry and fine prints, but the resin is messy and requires some careful handling. Printers using powder use either a laser to sinter particles together or an inkjet that deposits a binding agent (like a glue or solvent) to make them stick. These create relatively smooth-looking products, but they are expensive, and the materials-handling issues are far more challenging than with filament. We focus on applications of filament-based printers in this chapter and talk a little about the higher-end applications in Chapter 13.

Making something on a 3D printer requires several steps. First, you need to create a 3D model using a 3D CAD (computer-aided design) program. Next, you take that model and use a *slicing* computer program to convert the CAD file into commands for the printer to create the layers. Finally, you load that file onto the printer (which might happen by writing the file from a computer onto an SD card or by connecting a computer via a USB cable or wirelessly) and print your object. The next sections look at each of these steps in a bit more detail.

Filament-based 3D printers create models all in one color—the color of the filament. There are some printers that have more than one extruder, which can use as many colors as they have extruders. There are multi-color printers using other technologies, but they are, as of this writing, not at the consumer price point.

Here are a few things you should know about the limitations of filament-based 3D printing:

- It takes a while to print anything. Even small objects can take half an hour, and big prints can take days. Most printers cannot restart if they have an issue partway through.
- 3D-printed pieces are usually pretty small. Printers have print areas ranging from 4 or 5 inches on a side up to about twice that. Printers that can build objects more than a foot in any dimension get expensive rapidly.
- The smallest feature size one can make reliably is about a millimeter, and we usually suggest that details not get that small. The reason is that the line of filament the printer puts down can approach that size—it is like a scaled-down version of a cake-decorating squeeze nozzle. The nozzle opening in the printer we use for our demonstrations in this chapter has a 0.5 mm nozzle, but it squishes down the lines it makes to have them thinner in the vertical direction, so the lines it puts down are thicker than that.
- Prints build up from the bottom, and if they have parts hanging out in space they need to be supported somehow, and then this support has to be cut off later. There are programs that handle creating support, but it is good design practice to think this through ahead of time. Similarly, prints need to have some hefty amount of contact with the build platform so that they do not get knocked off during the process.
- 3D printing takes tweaking, maintenance, and the occasional cleanout of a clog. Look at user reviews before you buy to get a sense of a printer's reliability and ease of troubleshooting.

3D Modeling

The first step in the 3D-printing process is to either download or create a 3D computer model. There are a lot of sites with free downloadable models, including `www.thingiverse.com` and `www.youmagine.com`. Free download sites have the issue that anyone can (and does) upload models, and many of these will not print well or at all. Look for models that have a lot of downloads and some sample prints posted.

If you want to create your own models, you have a variety of options, many of them free and/or open source. This section goes over two of them, but there are many more. The modeling process will produce a file ending in .stl, normally referred to as “an STL file.” STL stands for *stereolithography*, which is the process used by resin printers. The STL format is a de facto standard for consumer 3D-printable models. Some vintages of Windows do not like the .stl extension and think it is some sort of security file, so you may get some strange complaints from Windows when you save STL files.

■ **Note** There are some other 3D-modeling standards around besides STL—notably OBJ, used by many graphics programs, and AMF, which is not yet widely supported but is designed to replace STL and can track the use of different materials in a model, but in this chapter we will focus on creating an STL file.

TinkerCAD and the Other 123D Apps

Autodesk has a whole suite of CAD software (`www.123dapp.com`), the simplest of which is Tinkercad (`www.tinkercad.com`). Tinkercad allows for drag-and-drop design. This is good because you can design things very quickly. It is a little limiting in the area of drawing freehand, although there are ways around it. If you can create what you want to model by building it up from a variety of basic shapes, Tinkercad is a good choice.

There are many community-contributed shapes, including a utility that allows you to print text in 3D and another that prints gears. Figure 9-2 is a simple little pendant that was developed in Tinkercad. It is shown here on a 3D printer with a *brim*, a few 3D-printed lines that help the model stick to the printer securely, and then off the printer to show how much of the print was the brim (Figure 9-3). You can see the layer lines in the closeups, and see the limitations on feature size. This pendant is about 50 mm across at its widest point and is publicly available at `https://tinkercad.com/things/kQD9mpZ1BLf`.

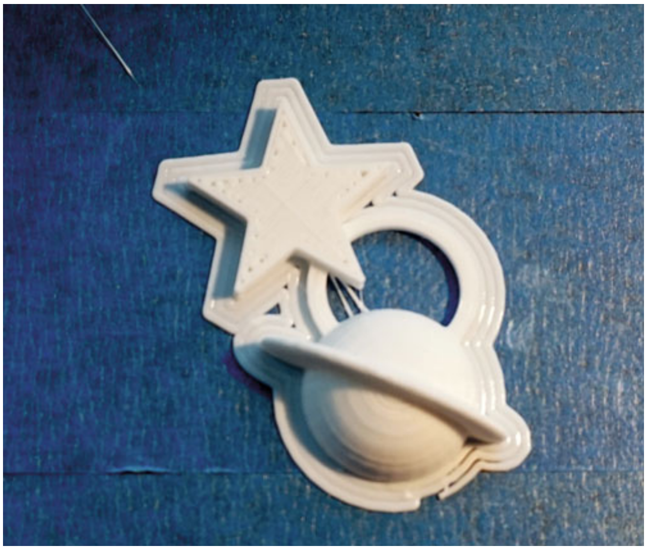


Figure 9-2. *Tinkercad pendant on the printer, with brim still attached*

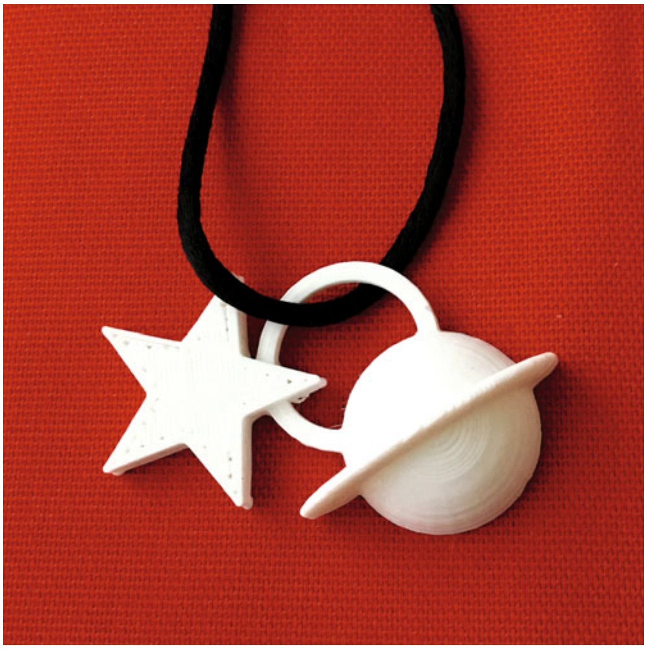


Figure 9-3. *Tinkercad-created pendant*

If you just want to have an art class whip up a little piece to make into a pin or pendant, Tinkercad may be a good choice. Letters, numbers, and some symbols (like the star in this example) are available, and if you go into the Community selections you can find things from user-specified gears to text in various fonts.

■ **Tip** Tinkercad offers good tutorials on its website that should help get you up and running quickly.

OpenSCAD

The OpenSCAD program allows you to develop models in a style that sort of looks like the C/Java/Python family of programming languages. It is free and open source, and we want to acknowledge and thank Marius Kintel and the many other contributors and maintainers of the program. You can download OpenSCAD from `www.openscad.org`, and an excellent user manual is available at `www.openscad.org/documentation.html`.

Download OpenSCAD and install it per the instructions on the download site. OpenSCAD is available in versions for Linux/UNIX, Windows, and Mac OS X. Figure 9-4 shows a few "jewels" Rich generated in OpenSCAD. You can create quite complex models if you know a little bit of programming, or start from an existing example. Many downloadable models that are user-changeable are written in OpenSCAD.

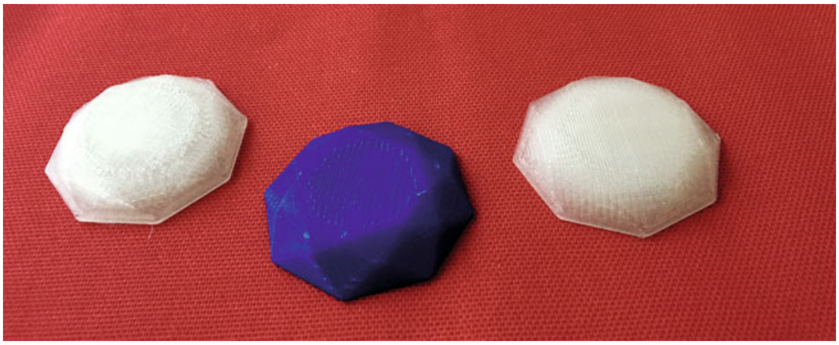


Figure 9-4. *"Jewels" created in OpenSCAD*

Other CAD Programs

There are a lot of other CAD programs out there as well. Some of the programs designed for animation, like the free, open source program Blender (`www.blender.org`) or the commercial program Maya (`www.autodesk.com/products/maya`), can make very sophisticated models. However, they have long learning curves and require special attention to how your model is constructed because they are not intended to make models that are 3D printable.

If you want to make engineering-type parts (dimensioned mechanical pieces), you might consider Onshape (`www.onshape.com`), which is a relatively new program and novel in being a relatively affordable engineering program.

Slicing and Printing

Creating (or downloading) a 3D model is just the first step, though. Once you have the STL file, you still need to run it through a program that will create the detailed commands for your printer to actually make the object, and you also need to transfer this file to your printer. Most printers can be connected via USB, and some also have a wireless option. Others can use an SD (or microSD) card to transfer the printable file from the computer where you run this software to the 3D printer itself. These steps vary somewhat from printer to printer, but some standards have emerged.

One program that works on a wide variety of printers is the open source program MatterControl (`www.mattercontrol.com`). MatterControl (or your printer's equivalent software) takes in an STL file and outputs a *G-code* file. The G-code format (or an equivalent format, such as X3G) is what actually runs on your printer. If your printer uses proprietary software, that software may or may not reveal this file to the user.

You can check whether or not MatterControl is supported on your printer by looking at your manufacturer's instructions. If your printer uses its own proprietary software, it is likely doing something similar to what we describe here, but the details will be different.

■ **Tip** If you want more detail about the 3D-printing process, with a focus on open source printers, you might consider Joan's book *Mastering 3D Printing* (Apress, 2014). This book also includes material on post-processing your prints. If you want to focus on using the MatterControl software in particular and want more of a detailed user guide, you can instead get Joan and Rich's *3D Printing with MatterControl* (Apress, 2015). Both books review how to get started in 3D printing, including creating models. Because the subject is large and complex, we are only giving the general outline of the process here to give you an idea of what is involved and what is possible. There is documentation at `www.mattercontrol.com` as well, and your printer manufacturer may provide some links or a manual to get you started.

MatterControl

Consumer 3D printers fall into two broad types: open source and proprietary. Open source printers have some heritage back to Adrian Bowyer's RepRap printer mentioned earlier in the chapter, and these printers are the ones that MatterControl is designed to support. MatterControl is also an open source, free program that takes an STL file and turns it into the commands (G-code) that run on the 3D printers that support it. MatterControl requires:

- A file that defines your printer (provided in a drop-down menu for the printers that are officially supported).
- An STL file.
- Settings for the specific STL file. For example, one might have the printer generate support structures or not, might use different temperature settings for different materials, and so on.

Some of these settings may be the same all or most of the time, and some may change frequently (such as whether or not a model needs support). The state of the art is that some knowledge of the process is necessary to produce a good print. The resources suggested in the preceding Tip can get you started.

MatterControl also has some capabilities to send a print job to a printer connected to a computer running MatterControl. MatterHackers, the company that created MatterControl, also sells a tablet computer (the MatterControl Touch) that can run compatible 3D printers instead of tying up a computer.

Proprietary 3D Printers

Other manufacturers have elected to build their own software and file formats from the ground up rather than be a part of the open source 3D-printer community. If your printer comes with proprietary software, you should refer to its documentation.

■ **Caution** We recommend using a 3D printer in a well-ventilated area (but not outside or directly in front of a window). Follow the manufacturer's suggestions for your printer's environment.

3D-Printing Materials

Consumer 3D printers use a variety of materials. Probably the commonest one is polylactic acid (PLA), a plastic typically made from sugar/starch (in North America, usually from corn). PLA is one of the easiest materials to print and does not require that the printer's platform be heated, as some others do. However, it becomes soft at the kind of temperatures you will encounter in a hot car in summer, so you have to be careful that you do not end up with an unintentionally surrealistic dress if you are making anything structural out of PLA.

Acrylonitrile butadiene styrene (ABS) is used to make many toys, and thus is often called *Lego plastic*. It is harder to print; it requires a heated printer platform to keep it from peeling away from the bed during printing and an extruder that can handle higher temperatures than those required for PLA. On the plus side, ABS is less vulnerable to getting soft in warm temperatures.

Nylon filament is an interesting choice for fashion tech applications. It is flexible when printed in slender parts, but quite stiff if printed thicker. It can be dyed with dyes appropriate for nylon. However, like ABS, it has to be printed at temperatures that are too high for some printers, and it requires a bed designed for nylon.

There are many other types of filament, with more appearing all the time. PET (a clear plastic often used for water bottles) is pretty transparent and has a temperature tolerance somewhere between those of PLA and ABS. There are also filaments that have sawdust or metal dust in the filament, and so print to look like wood or metal, and some glow in the dark. However, these filaments can have a tendency to clog and abrade nozzles since they contain small particles. There are also flexible filament materials (such as thermoplastic elastomers, or TPEs) that are challenging to print but give a result that is flexible.

■ **Caution** Some printers can only print one type of material. Be sure you purchase filament that is compatible with your machine. Also, note that filament commonly comes in diameters of 1.75 mm and 3.0 mm, and the two are not interchangeable.

Applications

Now that we have gone through some of the ways in which 3D printing can be challenging, why would you want to use it in a fashion tech project? The key advantages of 3D printing are its flexibility and the ability to create a physical object in a pretty arbitrary shape very quickly.

Straightforward applications for 3D printing can include printing ornamental pieces intended to be glued or sewn on, such as buttons or decorative pieces (like those shown earlier in this chapter). 3D-printed pieces can also be useful in support structures for batteries and control wire routing or to supplement traditional *boning* (dress structural support). The small size of consumer 3D-printed pieces can limit its utility in boning, however, unless you glue pieces together or glue them onto the garment.

■ **Tip** Lyn is partial to Aleene's textile glues. The company website (`www.ilovetocreate.com`) has a lot of information about what product to use for different situations. Lyn's experience is that the glues she has used will hold very well, but may not survive a ton of washing.

Making a Mold or Casting Pattern

Another way that 3D printing can be useful in fashion is to create a mold for casting some other substance. The traditional sand-casting method, for example, starts with a positive mold (the same shape as what you ultimately want) and makes a hollow in a sand-oil mixture that is then filled with molten metal. These positives (called *patterns* in this application) have traditionally been made of wood, but because they are only used to press down sand, plastic works as well.

Another traditional casting method, the *lost-wax* process, also starts with a positive, traditionally made of wax. This model is then encased in plaster or another high-temperature material and the combination is heated up so the wax melts and flows away. Metal can then be poured into the void. People have tried "lost-PLA" casting, substituting PLA 3D-printed models for the wax ones.

■ **Caution** Because these processes involve molten metal, they are not typically do-it-at-home enterprises, but we note them here for interest.

Using a 3D-Printing Pen

Moving from metalsmithing to crafting, in principle one could use a *3D-printing pen* to draw on fabric. These pens are not computer-controlled; they put out a plastic stream similar to a short section from a 3D printer, but the user just draws with them freehand. We have not tried this ourselves, but it is not that different from using hot glue on cloth.

3D Printing on Fabric

We were curious about how it might work to try to 3D print nylon on nylon, with an eye to creating some texture in a garment. This is not something that a consumer 3D printer is designed to do, and as a rule of thumb if you did not assemble your printer yourself (and thus know its mechanics very well), you probably should not attempt this. In particular, if your printer has an autoleveling feature, you may hit the binder clips. In our experiments we have found that we pretty much have to stand and watch the process the whole time, or the material can get dragged into the printer mechanisms.

In our case, we have a printer that has a very exposed platform. We just stretched fabric on the platform with small binder clips. Figure 9-5 shows the same design printed on three different fabrics, most using nylon filament, but one using PLA. The PLA stuck quite well to the cotton duck, although attaching the stiff duck to a platform was tricky. Perhaps someday someone will create a 3D printer just for this application. Right now it is very hard to print on anything other than a rectangle of fabric that just covers the platform, or a long strip that is hitched along for each iteration, as in the project in Figure 9-6. In the meantime, you might print structures and glue them onto fabric.

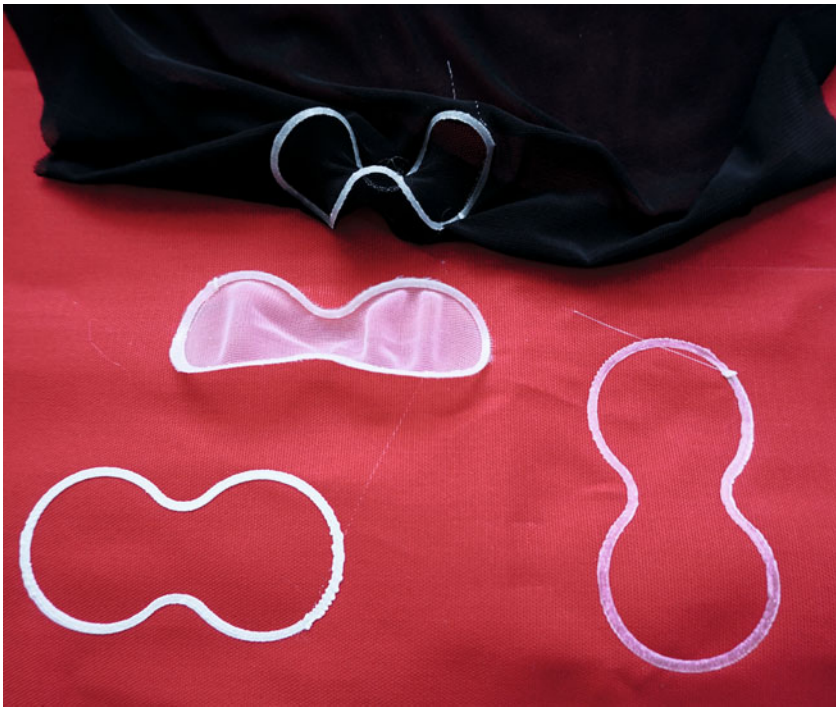


Figure 9-5. *The same pattern printed with nylon filament on a thin, stretchy 95% nylon/5% spandex (top), nylon filament on nylon (possibly blend) fabric that stretches more in one direction than the other (middle), PLA on stiff cotton duck (bottom left), and nylon on cotton duck (bottom right).*

Make magazine tried a somewhat different tack in its article on the topic (`http://makezine.com/projects/how-to-3d-print-on-tulle-net-or-lace-fabrics/`) that involved printing part of a design, spreading the fabric on top of that print, and then continuing the print. This has the advantage that you will know exactly where your design will go on the fabric, but is more complex to implement. If your printer platform is enclosed or is tightly packed with other components, you probably will not be able to print on fabric. You will also need to know how to pause and resume your printer, which requires getting into G-code in some cases.

There has been some commercial experimentation in this space. For a while, 3D Systems sold “Fabricate” kits of pieces of fabric and cartridges of filament that were designed to go together. Nike has filed some patents in the area of 3D printing shoes (`https://3dprint.com/1331/nike-awarded-two-major-3d-printing-footwear-patents/`). So the future may get interesting (as we discuss in Chapter 13).

Summary

In this chapter we broadly reviewed consumer-level 3D printing, with a lot of pointers to resources for learning the detailed process. We covered the workflow, with an emphasis on 3D design software. We looked into materials a little and construction techniques that include 3D printing.

CHAPTER 10

The Importance of Planning

One of the things that makes wearable tech difficult is that it brings together many skill sets. People may come into it from sewing, from electronics, or perhaps from writing computer code. If you come into it from the virtual world (3D modeling or programming), one of the big surprises is that physical things always take up more room than expected. Figuring out where to hide (or how best to display) circuitry, how to avoid shorting out those circuits, and how to deal with all the inconvenient things that happen if someone is wearing the piece take a lot of forethought.

Lyn has more than 30 years of experience in dance and theater as a teacher and a performer, as well as her costuming experience. She will be the primary voice in this chapter, but Joan will bring some of what she will call "system engineering" along to the story, and Rich will chime in on some of the technical problems that can be avoided.

We start the chapter with a case study (well, a cautionary tale, really) of our first large collective project. As we discovered, it was far too ambitious. Then we deconstruct our experience there, and create a laundry list of things that you should think about in the planning stages of your projects. We give an assortment of tips that we have come across, and finally end with some philosophy about making your design easy to test and maintain.

In Chapter 11, we move on to walking through a few examples and case studies of wearable tech projects. In this chapter (after our tale of ambition), for the most part we focus on dealing with component-level decisions that affect the entire project.

If you are a teacher using this book as a text for a class, this chapter is intended primarily for teachers defining projects. These are big-picture design issues that you may not want to go over in detail with students. We want to give you ideas about what to try and avoid in your guidelines for student projects—this is in part a chapter about what *not* to do. We start off with Lyn's point of view as a costume class teacher incorporating wearable tech ideas for the first time.

The Too-Ambitious First Project

My first foray into wearable tech was exciting—and terrifying. I had almost none of the knowledge required to solder wires, write code for microcontrollers, or work with the many components available to create these pieces. But I had been making and altering costumes for many years, so I was pretty confident I could figure some stuff out. I was fortunate enough to have two wonderful and supportive mentors in Joan and Rich.

It surprised me how simple some things were and how incredibly challenging others were. It changed my way of thinking by expanding the areas I felt competent in—or at least felt able to learn and develop new skills. As I was working through some of the Arduino tutorials, I began to understand what some of the code was saying and why. It was shocking and enlightening for me.

Making mistakes, large and small, is a good way to learn what works and what doesn't. It can even spark a new creative idea and/or solution. I have made some silly mistakes and some horrible ones (like the wiring in Figure 10-1), but hey, live and learn.

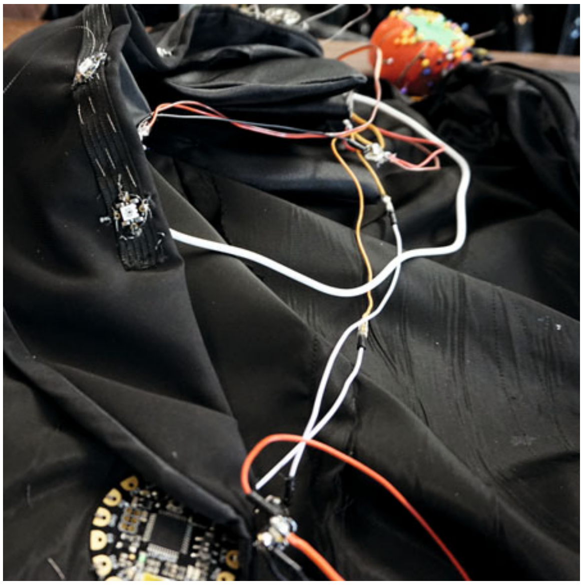


Figure 10-1. *Unplanned wiring gets out of hand*

The first project I am going to describe here came out of a costuming class brainstorming session. The students were supposed to incorporate wearable tech into a steampunk-style piece of some kind. (*Steampunk* is a science-fiction genre in which the general premise is that the world somehow comes back to Victorian England and has a mix of eras of technology, but has steam-powered mechanisms and only lighter-than-air flight). The first project the class converged on was a steampunk "haunted dress" that would have a mind of its own.

In the end, it wound up spilling over way beyond the end of the school year and became my personal summer project and self-tutorial on wearable tech. We knew we were going to present it at an educational maker technology conference at the end of that summer, which gave us a deadline to aim for. It helped me prepare for my first full-year costuming and wearable tech class, but it was certainly a baptism by fire that consumed a lot of my summer. Tackling a complex project does force you to learn all the aspects at once and quickly.

The Haunted Dress—the Original Idea

The "Haunted Dress" was intended to be a dress that would be a little performance piece. I wanted to evoke a dress coming to life around a surprised wearer. The sequence would start when the wearer made a gesture to start up the dress. Then, lights would flash down the front of the dress (to evoke lightning). Next, fans would kick in under the skirt to blow it around. Then as the wearer looked down at the skirt, the sleeves would begin to pull up on their own.

The dress was designed as a steampunk/Victorian long chiffon skirt with a bustle overskirt of taffeta and a taffeta short jacket with chiffon sleeves. Figure 10-2 shows the dress in early development. (Our first mistake was that we assembled most of it before putting on the electronics.) Figure 10-3 shows the completed dress, and Figure 10-4 is a closeup of the top as the servos on the shoulders were being added.

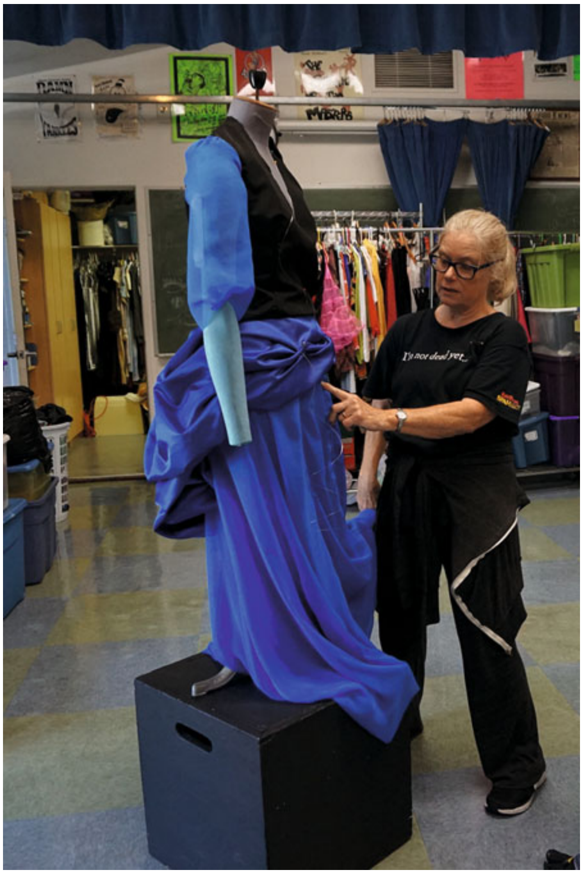


Figure 10-2. *(left) The dress in early construction*

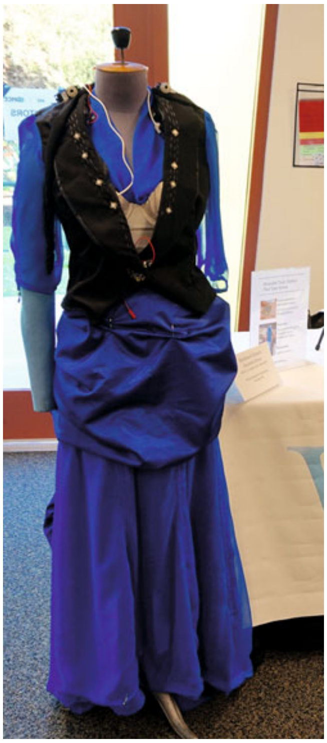


Figure 10-3. *(right) The finished dress*

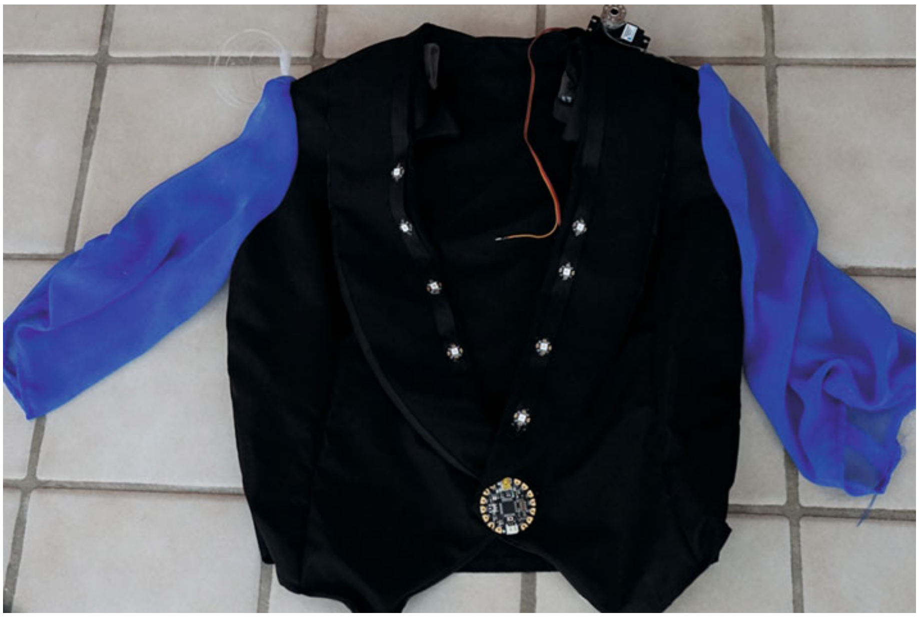


Figure 10-4. *The dress jacket showing NeoPixels, Flora, and some of the servo/fishing line attachments*

The sketch in Figure 10-5 lays out the key parts of the jacket, which supported most of the electronics. I decided to use a Flora(see Chapter 5 for more on the Flora hardware and Chapter 6 for software) as the dress microcontroller and NeoPixel multi-color LEDs for the lightning effect. Small fans were mounted under the skirt to blow it around, and a mechanism with servos (see Chapter 8) and string pulled up the sleeves. (I decided that pulling them up was not too hard, but dropping them back down again would be too difficult.) The bustle would be a good place to hide batteries and anything else large, and shoulder pads would hide the mechanisms for the sleeves. I was all set—or so I thought.

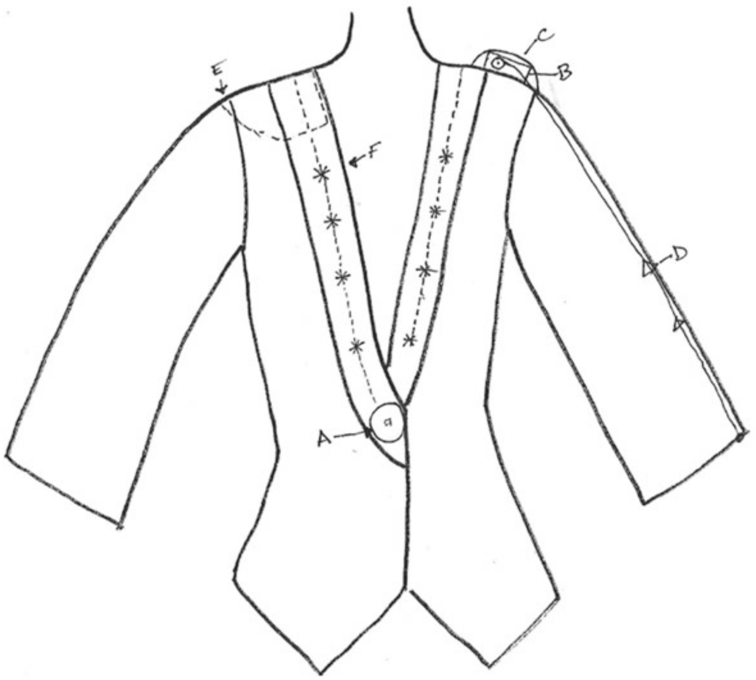


Figure 10-5. *Sketch of the jacket of the Haunted Dress, showing the Flora (A), servo (B), and elastic strap that holds the servo on top of the shoulder (C). Inside the jacket is shoulder pad (E, shown on other shoulder for clarity). Fishing line to pull up the sleeves is spooled on a bobbin glued to the servo and pulled through triangular guides (D).*

There are shoulder pads in the jacket, which hold lipstick batteries to power the servos. A sewing bobbin is glued to the top of the servo and is wound with enough nylon fishing line to reach the end of the sleeve, plus a bit more. Two small triangle-shaped loops are sewn onto approximately three inches of bias tape. The bias tape is sewn to the inside of the chiffon sleeve, and the triangles create a channel for the nylon line, which is knotted and hand-sewn to the edge of the sleeve.

The Flora board is hand-sewn to the front of the jacket, where a button would go, with hook and loop tape on the inside to fasten the jacket. The NeoPixels are hand-sewn onto the lapel, and conductive thread is used to connect them to the Flora. Conductive thread is also used to attach the wires from the servos and lipstick batteries to the Flora. The wires run from the shoulder pads inside of the jacket.

Mistakes

My biggest mistake on the Haunted Dress was making the skirts and jacket before I fully understood all the other aspects of the project. As an advanced seamstress, I am used to diving in and adapting projects a bit as I go along, but these projects take a lot of up-front planning.

For example, I did not allow for a clean wire run from the servos to the Flora. Consequently, the finished product was quite sloppy, and I even dripped hot solder onto the chiffon skirt. Not a very pretty look!

I also had to take the taffeta bustle apart in several places to create the pocket for the LiPo battery and have it in the best place to allow for movement. In hindsight, I would have created channels for the wiring inside the jacket out of lightweight cotton or lining fabric for a more finished look and a more comfortable feel while wearing it.

■ **Tip** I wish I had made a muslin version of the overskirt to figure out where the other elements should live before using the expensive (and fragile) taffeta. Lightweight, inexpensive cotton muslin is a great fabric to use for creating a prototype of a garment, figuring out the best route for wires, and discovering the best locations for the other components. It allows for mistakes on an inexpensive fabric before cutting and sewing the final version, which may be a more expensive fabric. It is often used by designers to create the first mockup of a piece, which is why a mockup is often called a “muslin.” Those from the electronics world might think of it as the equivalent of putting together a breadboard or alligator-clip version of a circuit before you create a permanent, soldered one.

Flora and NeoPixel Placement

I used conductive thread to sew four NeoPixels on each side of the lapel. The thread ran from the board to the first NeoPixel and then to each of the others in turn. It ran up the right side, around the back of the neck, and down the left side of the lapel, as shown in Figure 10-5.

It was a long run, so I had to make sure the thread was tight enough to make a connection, but not tight enough to pull the lapel out of shape. It took some experimentation to get it right.

In a first iteration, the jacket had NeoPixels sewn onto conductive ribbon, which was sewn around the lapel. The conductive ribbon was attached to the points on the Flora with conductive thread. I did not connect the ribbon to the NeoPixels and Flora correctly, so only some of them worked. I then took the ribbon off the lapel, sewed the NeoPixels on individually, and connected them to the Flora and each other with conductive thread. This worked better.

■ **Caution** Using conductive ribbon (Chapter 5) can be a time saver, but make sure to connect the proper wires in the ribbon with conductive thread to your components. My first attempt at attaching NeoPixels to conductive ribbon was pretty clumsy and time-consuming. I was making a long run around the lapel of the jacket for the Haunted Dress. The NeoPixels ran up the right side of the lapel. After a gap around the back of the collar, they then ran down the left side of the lapel. NeoPixels (Chapter 5) are directional, so for signal to come out of one and into the next, you need to make sure to cut the middle two sections of the ribbon so that the signal in and out paths are maintained from NeoPixel to NeoPixel. I attached them to each other along the ribbon, following the arrows and using conductive thread to attach them to the Flora board. When I tried to light them up, only the right side worked. Arrrgh! That was because I had sewn the last NeoPixel on the left side in the in the wrong direction! It was easy to solve, but quite frustrating for a while until we figured out my mistake.

The Magical Sleeve

I used fishing line, spooled up on a sewing-machine bobbin, to automatically draw up the jacket sleeves. I ran the line down the sleeve through small triangle-shaped rings and attached it at the edge of the sleeve. I used continuous rotation servo motors, with a sewing-machine bobbin glued to the top to reel in the fishing line. The servos were a bit heavy to balance there, so I attached a sturdy piece of black elastic on the outside of the jacket shoulders to hold the servos in place. This also allowed for movement of the bobbin, and for easy removal if needed (Figure 10-6).

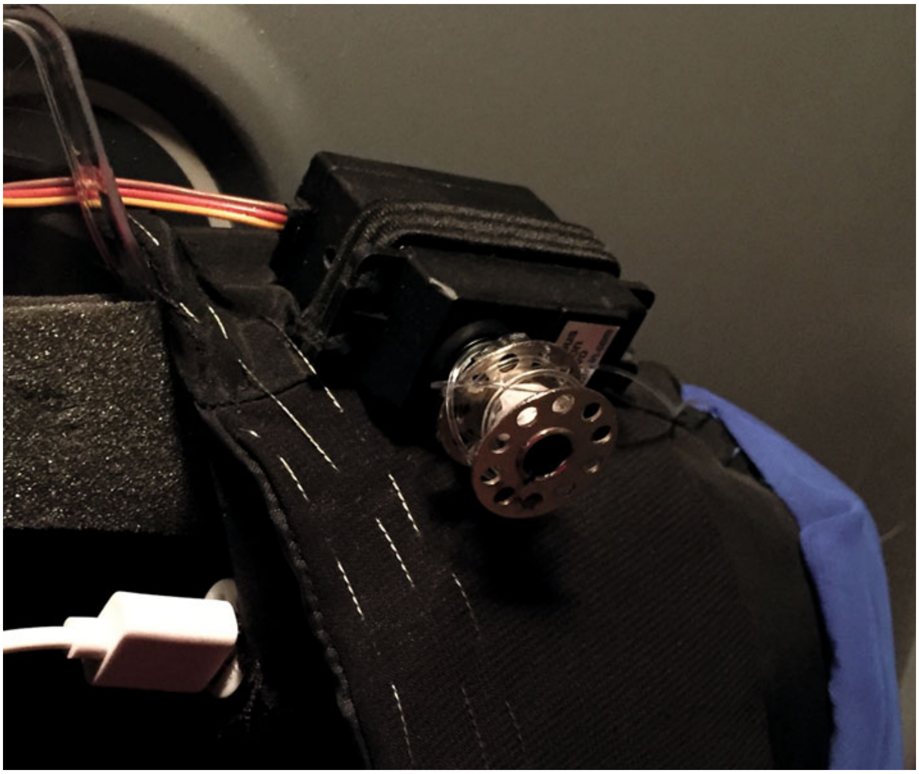


Figure 10-6. *Servo attached to the jacket*

I used two small triangle-shaped loops sewn to a 3″ piece of bias tape and attached along the sleeve as a channel for the fishing line. By knotting the end of the line and using clear nail polish on the knot, I was able to hand sew the line to the edge of the sleeve. I also attached a decorative piece of an old necklace at the end of each sleeve to add weight to the sleeve and help bring the sleeve back into place after the servo had drawn it up (Figure 10-7).

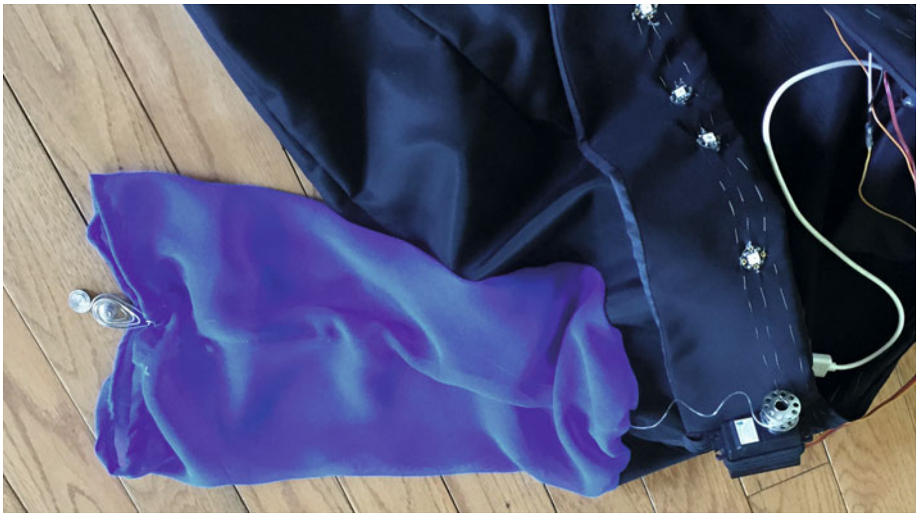


Figure 10-7. *The sleeve mechanism and servo*

The lipstick batteries to drive the servos were hidden in pockets in the shoulder pads (you can see the white USB cable going into the battery in Figure 10-6) and the USB plugs were soldered to wire from the servos. The servos were also wired to the Flora board.

INTEGRATING HEAVY COMPONENTS IN A CHIFFON SLEEVE

USB cables have an A side (the side that normally plugs into a computer) and a B side (the connector that in our case would normally be plugged into the Flora, Gemma, or other board). The batteries needed a USB connector, so Rich jumped in and stripped the B side of the cables and then soldered it to wires running to the servos. Normally you would not do this; see Chapter 8 for the more typical case. Rich and I used this approach here because we needed to deliver a lot of power to the servos and wanted to give them their own power source to ensure that they would not cause the microcontroller to brown-out. We ran wire down the inside of the jacket from the battery to the servo. We used conductive thread to stitch the wires in place and connect the servos and the Flora to signal and ground connections, as you can see in Figures 10-6 and 10-7.

For the Haunted Dress, I wanted to hide the lipstick batteries in the shoulder pads. Shoulder pads are usually made from cotton batting or foam and covered with a lightweight fabric. I made a cover for the pads from black taffeta to match the bodice of the jacket. Leaving a small opening in the taffeta, I was able to hand stitch the pouches that had come with the batteries inside the shoulder pad covers (Figure 10-8).

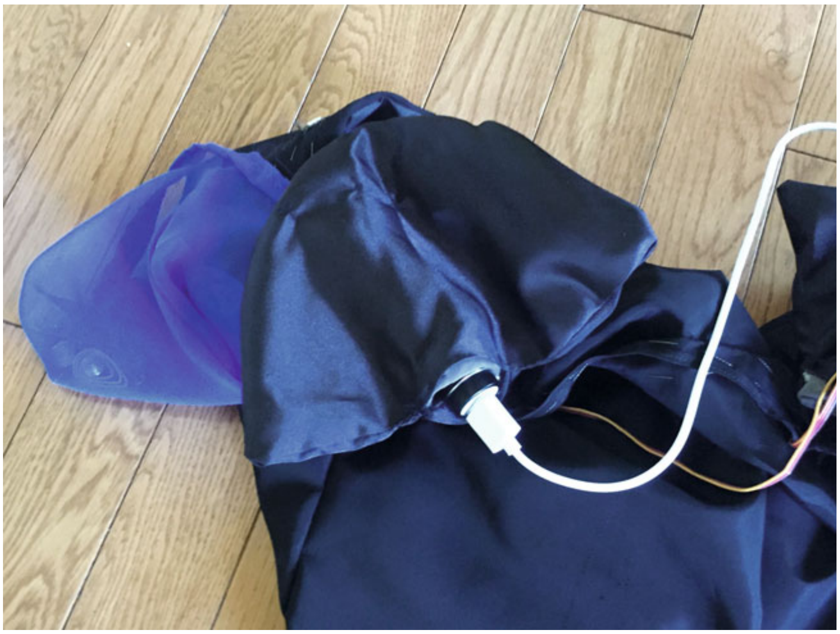


Figure 10-8. *The shoulder pads used to contain the lipstick battery*

I attached the shoulder pads inside the shoulders of the jacket by tacking them down in two places on the shoulder seam. *Tacking* means to take a few stitches in the same spot to hold things in place.

The Billowing Skirt

The three of us wanted the chiffon skirt to billow as if a draft of wind were blowing by. We chose to use two small radial blower fans, and they needed a place to live that would allow them to move the chiffon from inside the skirt while staying away from the person wearing it. I made a small pocket for each one and attached them to the underside of the chiffon (Figures 10-9 and 10-10).

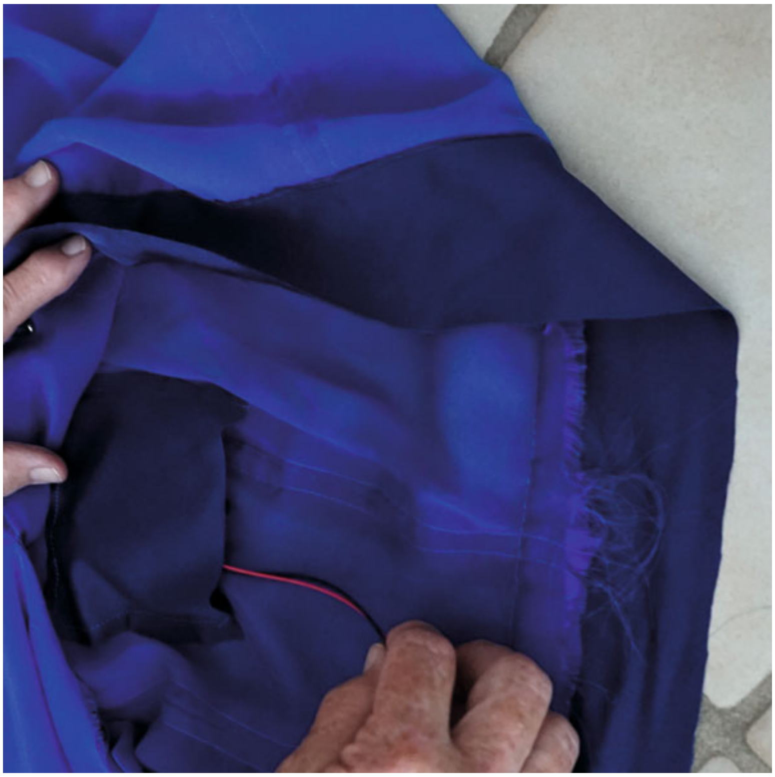


Figure 10-9. *The pocket for the fans*

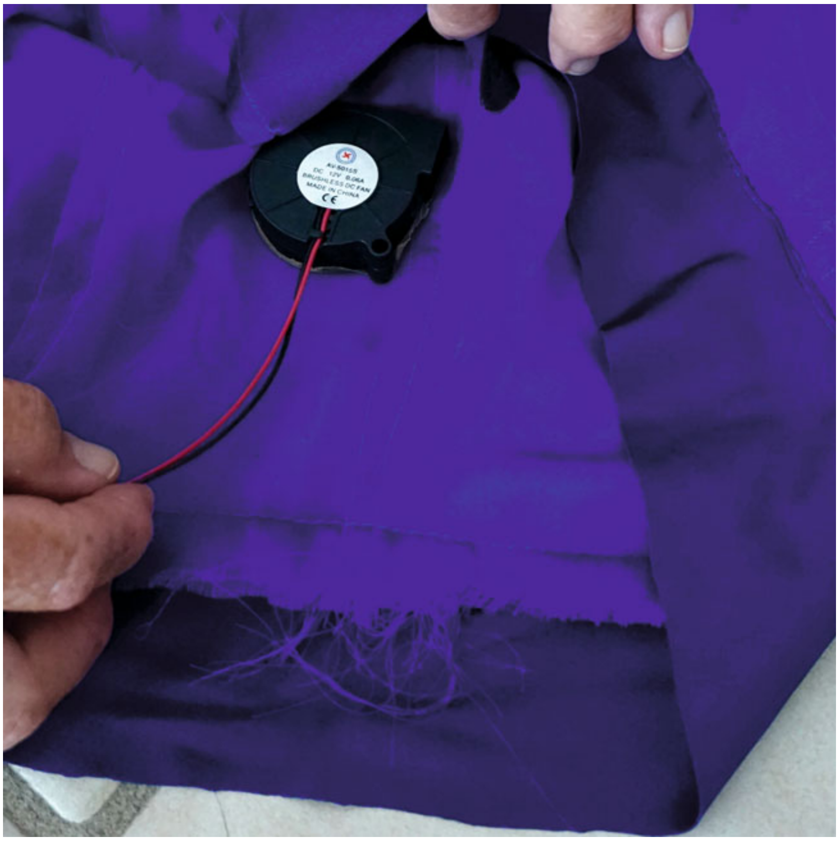


Figure 10-10. *The fan being slipped out of its pocket*

I used two layers of chiffon, one navy blue and one a brighter blue, to make the skirt a bit more opaque (see the pictures of the dress in Figures 10-2 and 10-3). This was also intended to minimize blockage of the fans' intakes. The wires from the fans were soldered to connectors and attached to a battery. The other wires were soldered to the Flora board. The wires ran between the skirt and the overskirt. I made another pocket inside the front of the overskirt for the battery, which also powered the Flora.

Software

By browsing many tutorials from `Arduino.cc` and `Adafruit.com`, I learned a teeny bit about coding (Chapter 6) and was able to program the Flora to control the NeoPixels. The servos (Chapter 8) and fans were more sophisticated, and Rich and Joan stepped in to coach and debug. I have learned more about coding since that first attempt, but still have a lot more to learn.

How It Turned Out

As with many grand ideas, the project did not entirely work out as planned. The original design used an accelerometer to detect a grand gesture that would start and reset all of the components, but unfortunately we ran out of time to implement this. We lowered our expectations to having the wearer push the reset button on the Flora. In the end we wound up just showing the dress as a demonstration on a dress form (as you can see in Figures 10-2 and 10-3), so this was less of an issue than it would have been in the original theatrical concept.

The fans did not provide enough wind to billow the skirt from inside their pockets, so we removed them before the final presentation. In hindsight, we should have run a few tests with more fabric, rather than a brief test with one layer of the cloth.

The servo on the left shoulder only wanted to work about half of the time because of some electrical issue that popped up at the last minute, and we did not have time to debug it. But the right-side servo and the NeoPixels were terrific.

■ **Note** We have not provided the Flora sketch for the dress in this book because the project was never quite completed or debugged. We have referenced (simpler!) NeoPixel and servo examples in Chapters 6 and 8, respectively. We provided this example to show the pluses and minuses of trying something "too hard" the first time out, and as a bit of a cautionary tale.

In the rest of this chapter we will revisit the decisions we made in this project, plus talk about a lot of other "system design" issues that did not come up in this project but that you will encounter.

What We Learned

The first thing we took away from this experience is that it is essential to make a sort of map of your garment before beginning—at a minimum, create a butcher paper pattern and sketch out what goes where. (We say *garment* in this chapter, though you may be making anything from a stuffed toy penguin to a hat to an evening gown.) This pattern will allow you to figure out where to hide components, determine the path to run the wires, and make the most effective and efficient connections. It is a way to eliminate some of the mistakes before you put the piece together.

■ **Tip** To lay out a simple sewable circuit, you can use the Fritzing software package (Chapter 5) to create a full-scale drawing of the circuit layout. You can then either hold the drawing next to what you are sewing or use it as a guide to draw with tailor's chalk on the garment.

Creating some sort of minimalist model will allow you to make mistakes, correct them, and discover the best places to hide batteries and the most efficient and comfortable paths for running wires before you create the final project. I did not do this for the dress project, and consequently had to redo many of my first attempts.

■ **Tip** When you sew together your circuit, you might want to sew some of the conductive thread runs or lay wire inside the garment, which means you may be using the pattern in the mirror image of the way you drew it. It is very easy to sew parts of the circuit backward that way, so be sure you are projecting the pattern onto the fabric maintaining things the right way round, where that matters (like signal and power into a NeoPixel - Chapter 5).

Materials Considerations

Next, we realized that the type of fabric and notions will help make your project successful (or not). For the Haunted Dress, I used chiffon and taffeta—in retrospect, not the best fabric choices. Chiffon can be difficult to sew because it is a fine fabric and frays easily, and taffeta is a stiff fabric that shows every stitch you have to take out as a visible hole. It also frays easily. I had to alter the circuit several times, which caused fraying on the chiffon and visible holes from removed seams on the taffeta. I learned a lot, but would have had a better end product if I had created a muslin prototype first, as noted earlier.

Good fabrics to use include lightweight cotton (particularly for beginners) or t-shirt fabric (a bit harder, since it is stretchy). It depends a bit on what you are making, of course. Nylon or any kind of poly-cotton-rayon mix is good. Avoid very stretchy materials because they can be challenging for a beginner to sew.

Hook and loop fastener tape (that is, Velcro) is sewable and will not cause shorts in your circuits. Snaps are easy to use, but make sure they don't interfere with any conductive thread, fabric, or soldered bits. Creating channels for wire runs out of lining fabric or lightweight cotton is a good idea, but they should open up easily to allow for access if there is a problem or short circuit. If you line the garment, you can attach the channel to the lining so it is not visible from the outside, or hand stitch the channel fabric in strategic places like seams or hems. A hot glue gun is handy for insulating any soldered joints. Just a drop will help protect the connection.

■ **Tip** A pin or magnetic closure hand sewn onto fabric or glued onto the back of a 3D-printed object makes a great fashion accessory. You can add servos for movement or NeoPixels to make it light up. Voila!

Hiding and Supporting Batteries and Mechanisms

Belts, shoulder pads, epaulettes, bustles, pockets, and facings are handy ways to hide batteries, connectors, switches, controllers, and other parts you do not want visible. You can also incorporate these into your design with appliques or embroidery, or let them appear just as they are if they are interesting and/or you like the look.

■ **Caution** It is important to remember to create a secure and protected place for the batteries and controllers that also allows for easy access for recharging and/or replacing. See the discussions of battery issues in Chapter 5.

The sidebar about the chiffon sleeves described how we supported batteries in shoulder pads. These batteries are a bit heavy, so putting them in the shoulders helped to give them some support.

■ **Tip** If something has to actually move, it will probably take a fair amount of power, which may mean hiding a lot of mechanisms and batteries. You may be able to avoid having a costume part move with clever attachments to another body part (like something attached to a shoe that moves as the person walks). In some cases, this might be simpler than trying to be high tech.

Conductive Thread, Wires, and Cables

Conductive thread is a little harder to work with than regular sewing thread because it is stiffer and thicker. It is made from wire, after all. It likes to kink up and twist itself around, so use a fairly short length when you sew. Use just enough to complete each run separately and then rethread your needle. It requires patience and attention to each stitch, but you will get the hang of it fairly quickly.

The stitches need to be tight enough to create a secure connection without distorting the fabric and/or garment. The stitches you make with conductive thread can be longer than you might make with regular sewing thread, but they should not be too long or they will become too loose for an adequate connection.

As you are sewing a run of conductive thread, keep checking the previous stitches, making them taut without pulling the fabric out of shape. After you have finished the project and it is working, move the areas with conductive thread a bit to make sure the connection is strong enough to allow for the movement. If it works intermittently with the movement, you may need to pull some of the stitches a bit tighter.

The bigger issues come into play because you are wiring up a circuit, which has more exacting rules to follow than normal sewing of a garment does. This section suggests some good practices to follow.

Avoiding Short Circuits

To avoid *short circuits* (having conductive thread create paths for electricity to flow where you did not intend it to), you need to plan where you will sew each run of conductive thread. The thread is not insulated like coated wires are, so if it touches another run of conductive thread as the garment moves, it will short out the circuit.

Creating a map for the layout of your wire and conductive thread runs will help eliminate some frustrating issues with your project. Fritzing (Chapter 5) is good for this because it is easy and creates a clear picture or map to the right scale. You may also draw a circuit plan freehand, but it is easy to forget things or to sketch vaguely and then realize you have problems. Either way, it is best to do it before you put the pieces together, whether you are soldering or sewing.

■ **Caution** Pay particularly careful attention to any directional connections (like those for LEDs in general or NeoPixels in particular), which only work in one direction in a circuit. At best the circuit will not work, and at worst you might fry a component. It is frustrating and time-consuming when you have to take out stitches and replace parts that are glued or sewn on. Fritzing diagrams can be time savers and mood stabilizers.

Routing Physical Control Cables

If you are using a control cable to pull on something (like the fishing line used in the Haunted Dress sleeves) you have to manage routing that as well. Obviously movement of something made of fabric can loosen connections, so it is important to keep the controllers, servos, and other components stable relative to their electrical connections when you push or pull part of the garment.

Mapping out the route your wiring will take gives you a chance to really think about what parts move the most and where your components and wiring runs will best serve the project and be most comfortable for the wearer. Using lining, or similar lightweight fabrics, you can create sleeves or channels for your wire runs and pockets or straps for the components.

Wire vs. Conductive Thread

If you were creating an Arduino circuit on a breadboard, you would probably use insulated wire. Sewable circuits, though, might most simply be attached to a garment by sewing them on with conductive thread (typically steel). Wire is not as flexible as conductive thread, so for most garments, the thread is a better choice. Some connections really need to be soldered and insulated. For instance, it is best to use insulated wire for circuits carrying too much current for the conductive thread, or if the circuits are connected directly to a battery that will allow too much current to flow if they short out. Keep in mind where the connections will be, how much movement the area will have, and whether shorts will be a problem if the circuit is moving around on the garment.

■ **Tip** A hot glue gun is very handy to reinforce and insulate key connections. Clear nail polish is great for securing knots in the conductive thread and adding some protection on top of some connections. It is also easier to use in tight areas than a glue gun.

Attaching Servos

Integrating servos (Chapter 8) into a wearable project poses some particular issues. If the servo is pushing on or pulling something, it must be securely attached to something stable. For example, as described earlier in the chapter, the servos in the Haunted Dress project were attached to the shoulder seams on the outside of the jacket of the Haunted Dress to give it a bit of a steampunk feel. More complex or heavier mechanisms might need the wearer to have some sort of harness or more significant support for the mechanisms.

Placing Switches and Sensors

Placing switches and sensors is very project-dependent (and sensor-dependent). Chapter 8 covers the details of including sensors or switches from a circuitry point of view. From a project-construction perspective, the main thing to consider is whether where you attached your sensor or switch might be prone to false readings.

If a sensor is measuring a person's movement, you might not want to attach it to a swishy skirt. Our basic advice here is to think through what the sensor's environment will be (motion, light, whatever else might be relevant) and then be sure that its data will not be corrupted somehow.

There are many different ways to trigger a function on your controller, as Chapter 8 discusses. In a pinch, the reset button on your processor can do the job. However, that button is small, and fumbling with it might ruin the effect you are seeking.

Resistance-Varying Components

There may be times when you may want to control something with more options than just on and off, or when you may want to take action based on an analog input. A rather sophisticated (in terms of fabrication difficulty) way to implement that is to use a metal zipper as a voltage divider (see the discussion of voltage dividers in Chapter 5). For example, you might want to create a purse that has a light that goes on when you open it; we mention it here as another option for a switch. As you open the zipper, you can arrange to have the resistance in part of your control circuit change. We say it is sophisticated, though, because it can be pretty fiddly to get to work and to control.

Fiber Optics

Chapter 5 covers LEDs and NeoPixels in particular, as well as EL wire, at the component level. But you may want to move light around passively—with fiber optics.

Because the middle school was doing a performance of *The Little Mermaid*, we were looking for sea-creature projects that could be incorporated into the production. We found the project Fiber Optic Jellyfish Skirt by Linawassong (www.instructables.com/id/Jellyfish-Skirt/) and decided to use it as a starting point to create a jellyfish costume. The student playing the jellyfish wore the skirt and a black leotard and walked across a dark stage. Because this was meant to be worn only in the dark (by a middle-schooler) we made it a bit sturdier than the delicate concept in the Instructable. The skirt was created with fiber optics, (a lot of) hot glue, a NeoPixel strip, a leather belt to support the weight of all the components, a Gemma board, a lipstick battery, and a chiffon overskirt cut into strips.

It had a lot of components that could be worked on separately and then brought together to assemble. Therefore it was a good project to occupy several students and it created a lot of excitement when we brought them all together for the finished product.

Gluing together the fiber optic strands took a very long time and a great deal of hot glue. We had considered a purely mechanical attachment from the skirt to make it pulse as the wearer walked, but this turned out to be too complex to finish by showtime and was not really needed, since all the loose ends floated around anyway.

Wearing Tech

Student theater groups are natural partners for wearable tech classes in an academic environment. In our first year we were fortunate that the middle school musical (*The Little Mermaid*) lent itself to light-up costuming. As we thought about projects, there were things we avoided doing just because we were concerned that there might be issues. Thus, these are things we consciously did not do and thus did not encounter. With some thought, you might find ways around these—we list them here in no particular order as things you should consider (and avoid to keep things simpler) if you are building a costume.

People sweat and they use hairspray and makeup, and water might get tossed around during a performance. All of that needs planning for before and during the performance. Do a dress rehearsal with the costume on to be sure you do not interfere with wireless microphones or other stage electronics. For costumes that will be worn for multiple performances, consider attaching the circuitry to an overgarment that will not need frequent washing.

Costumes for dance numbers have particular issues. The dancers might connect to each other with considerable force, and any controls, batteries, and so on have to be someplace that will not be in the dancer's (or her partner's) way. Heavy hardware might rip off in a jump or a fouetté turn. A stately jellyfish pulsing slowly back and forth in a straight line across the stage is more achievable.

Finally, like putting together furniture in a room and discovering you cannot get it back out the door, it is easy to forget that people have to get in and out of costumes. This can require some thought and strategic snaps and nonconductive closures.

All that said, some people have done some stunning projects, notably Lisia Trubat's E-traces project. She put Lilypad processors (similar to Floras) and accelerometers on a ballet dancer's pointe shoes and then captured the dancer's motion in an art piece (http://cargocollective.com/lesiatrubat/E-TRACES-memories-of-dance).

Designing a Testable Project

One of the things that can be overwhelming in a wearable tech project is how many things have to go right for it to work. You need to do physical assembly of the garment/piece, plan paths for control circuits, create a circuit, write code, and so on. How can you avoid drowning in all that detail when things do not work? Some of it is just experience, but here we talk about some good design practices that may help.

First, see if you can find ways to have your overall project be a collection of smaller parts, particularly smaller parts that you can test incrementally. Sometimes that is not possible, but consider whether, for instance, if you have a servo and lights, you can build just one or the other, test that much, and then add the other. For the code part, save incremental versions and create a clear naming convention.

For example, you might name something `JoanProjectLightsWork.ino`, save that much, and then rename the next version `JoanProjectLightsServo.ino` to continue from there. That way you can always backtrack to something that did work. Keeping tidy computer file folders helps a lot, as does creating Fritzing diagrams ahead of time and using them. It is very easy to sew things on backwards or to create shorts if you do not plan your conductive thread runs carefully.

Never assume that any particular component, USB cable, or computer port is working. If something *should* work based on all your understanding, try swapping out components that are easy to swap out. We have had a lot of time wasted by low-quality USB cables that appear fine but are not. The earlier chapters in this book have advice on debugging individual aspects, and you should refer there for particular components. Generally, though, never assume that anything *must* be working. Consider whether your batteries or other power source is putting out the right voltage and can supply enough current (Chapter 5).

Similarly, if you get in trouble and things are not working, be systematic. Change just one thing at a time, rather than five things, and save intermediary versions of any code with clear filenames (and/or good code comments.) If something was working and stops working, think very carefully about what changed. The weather? Vibration? Those things may not be the cause, but they may be correlated with it.

And most of all, if you just cannot see what is wrong, step away from the project, even for a few minutes. When you get back, talk it through with someone, even if they know nothing at all about electronics or coding. Sometimes just the process of talking it through to someone else will show you the problem. Even if you don't have anyone to discuss it with, you can engage in *rubber duck debugging*, which is where you figure out why your project isn't working as expected by explaining what it's doing, in detail, to a rubber duck.

Summary

In this chapter we covered a case study of an overly ambitious first wearable tech project of ours. Then we extracted some ways to make projects like this easier. In particular we focused on creating the non-electronics part of a garment in a way that makes laying out the electronics as easy as possible. We talked about ways to integrate components that might be relatively heavy into garments using strategic pockets or shoulder pads, and how to avoid short circuits. We have some pointers back to details in the earlier component-level chapters, as well as some forward to the next chapter on example projects you can attempt on your own.

CHAPTER 11

Two Bigger Projects

So far in this book, we have talked about different aspects of a fashion tech project. Chapter 7 walks you through a relatively simple project, and Chapter 10 tells a cautionary tale about an overly complex one. In this chapter we give you two challenging but not extremely difficult projects that you can use to stretch yourself a little.

The first project uses a hat that you buy and moderately complex circuit attachment and coding. The second project is a fairly sophisticated sewing project involving creating a pattern based on your measurements. It requires plugging electroluminescent (EL) wire into an inverter/charger box, without any coding at all. Depending on whether you have more issues with sewing or programming, you might pick one or the other to start with. If you are teaching a class, these might be two types of examples for final projects where students can pick one or the other path. Joan and Rich guide you through the hat, and Lyn talks you through the dress.

The Yes-No Hat

We thought it would be fun to create a project that used a pre-existing garment rather than creating one, but which would be interactive and flashy. We came up with the yes-no hat: if the wearer shakes her head from side to side to say "no" (in most cultures, anyway), a NeoPixel ring lights up red. If the wearer moves her head up and down (nodding "yes"), the ring lights up green. Figure 11-1 shows the completed hat.

The Sensor

This project uses a sensor that has a three-axis gyroscope, accelerometer, and magnetometer. That means it can sense rotation around any of three axes at right angles to each other, whether that rotation is getting faster in any direction at the moment, and its absolute direction (relative to north, if there is no interference from other things). The way the sensor was mounted, any one of these three types of sensors could have been used, but the gyro turned out to be the easiest by far because it did not require comparing new data to previous data. We use the sensor in a way that does not count on knowing the values output by the sensor to any great precision, but rather on knowing whether the garment is rotating in one of two axes (corresponding to shaking one's head "yes" or "no.") The sensor needs to be at the front center of the hat, oriented vertically, as shown in Figures 11-1 and 11-2, for this sketch to work properly.

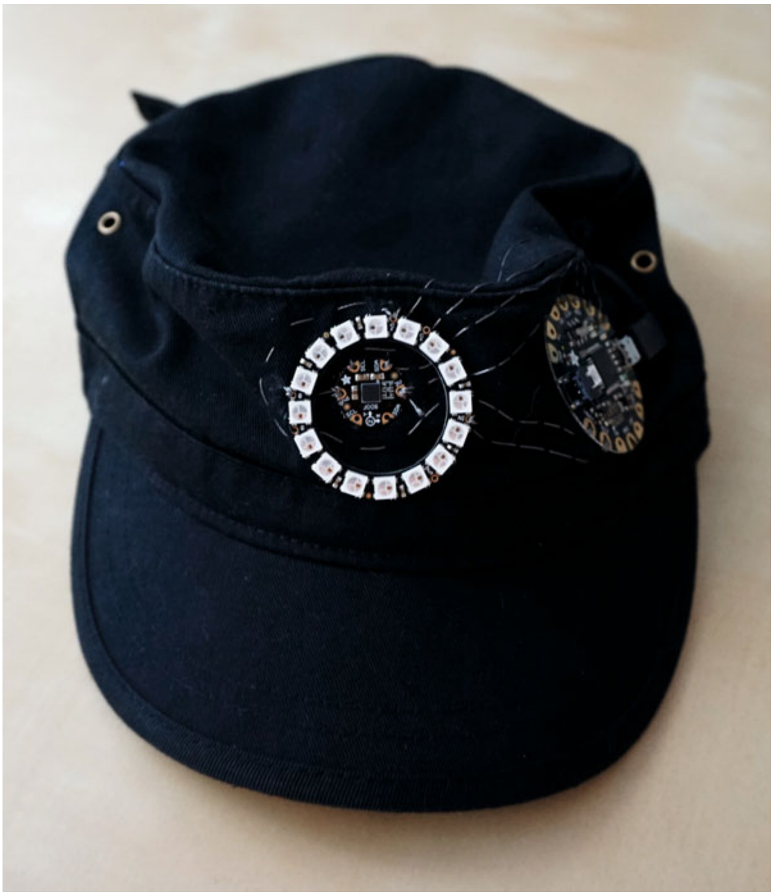


Figure 11-1. *The yes-no hat (with NeoPixel ring off)*

Materials

The hat requires the following materials:

- A hat of substantial canvas or cotton (not conductive fabric).
- An Adafruit 16 NeoPixel ring (a strip of 16 pixels or a smaller number of individual sewable pixels would work too).
- A Flora microcontroller board.
- Conductive thread and a needle.
- Ordinary (nonconductive) sturdy thread and a needle.
- USB cable to connect the Flora to a computer (to upload the program).
- A Flora sewable LSM9DS0 accelerometer, gyro, and magnetometer 9-degree-of-freedom sensor.
- A battery that can plug into the Flora (with a micro USB output) that can deliver 5 V and at least 400 mA. Here we used a small, flat micro USB power bank, but you could also use a lipstick USB battery (you will need a USB-to-micro USB cable).

■ **Tip** For more on this sensor, see `https://learn.adafruit.com/adafruit-lsm9ds0-accelerometer-gyro-magnetometer-9-dof-breakouts/pinouts`.

Creating the Circuit

This sewable sensor is pretty simple to use. However, you will need to put it in the orientation shown in Figure 11-2 for it to work correctly. The *z* axis marking should be at the lowest point, and the sensor should be on the hat such that it will be straight up and down when the hat is worn. Figure 11-2 shows the circuit layout as sewn, and Figure 11-3 gives a Fritzing diagram of the connections.

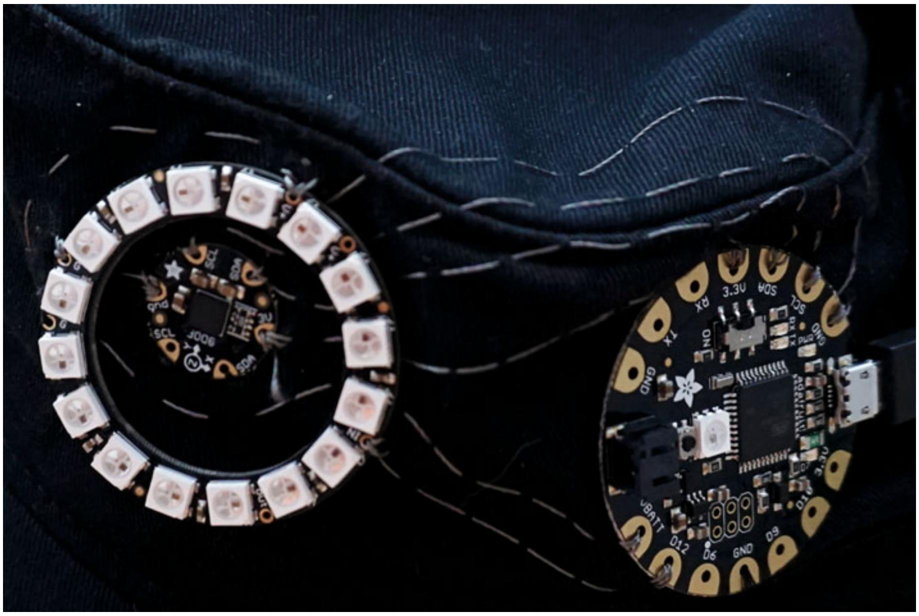


Figure 11-2. *Orientation of the sensor and circuit layout on the hat*

First sew the Flora, NeoPixel, and gyroscope onto the hat with regular, nonconductive thread, using some of the holes you will not be using for electrical connections. You can use your judgment on this, based on where the hat is simplest to sew. The only critical thing is to maintain the orientation of the gyro such that it will be straight up and down when the hat is worn, and the axis marking (a circle and a Z) is at the bottom.

To make the circuit (shown in a Fritzing diagram in Figure 11-3), make the following connections by sewing with conductive thread. Think it through first, look at Figure 11-2 and the Fritzing diagram, and perhaps draw it out with chalk to be sure you do not cross any wire leads:

- Vbatt on the Flora to Vcc on the NeoPixel ring
- D12 on the Flora to IN on the NeoPixel ring
- GND on the Flora to gnd on the gyroscope
- gnd on the gyroscope to GND on the NeoPixel ring
- SCL on the Flora to SCL on the gyroscope
- SDA on the Flora to SDA on the gyroscope
- 3.3 V on the Flora to 3 V on the gyroscope

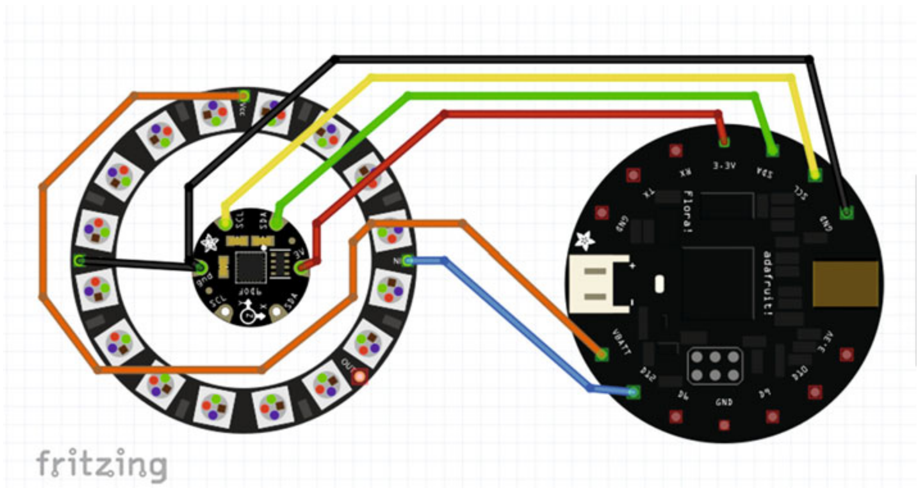


Figure 11-3. *Fritzing diagram of the hat*

You are connecting up the circuit such that the SCL (serial clock) and SDA (serial data) pins on the Flora are receiving data from the gyroscope, and the Flora processes that data and then sends out a signal on pin D12 to the NeoPixel ring about what color the pixels should display. The Arduino sketch for this is shown in Listing 11-1.

Attaching a Battery

Once you are happy with how the circuit is sewn, punch a small hole in the hat next to the Flora to pass through the power cord from the battery holder—on the micro USB side if you are using that type of battery, or the power connection (white connector) side if your battery case has that type of connector.

Depending on the hat, you will need to invent some way (tape or maybe sewing a small pocket on the inside) to keep the battery pack you select in place in the crown. Use only sturdy consumer batteries for this application because they are going to be banged around a lot and against your head.

Libraries

This hat requires the Adafruit NeoPixel, Adafruit LSM9DS0, and Adafruit Unified Sensor libraries. As always, go to Sketch ➤ Include Libraries ➤ Manage Libraries… and search on the library name. Note that it usually does not work to search on the name in the `.h` file; sometimes it is the same, but often it is not.

■ **Caution** Not all libraries for this project are supported on a Gemma. You may be tempted to try and make this project smaller, but the libraries for the sensor are not supported on the Gemma.

Listing 11-1. Sketch for the Hat

```
// This sketch reads the values from an LSM9DS0 Adafruit gyro
// and turns a NeoPixel ring green if there is rotation about the x axis
(gyro[0])
// and red if there is rotation about the y axis (gyro[1])
// Axes are relative to the sensor board orientation, NOT gravity
// i.e. relative and not absolute orientation

#include <Adafruit_NeoPixel.h> // from "Adafruit NeoPixel" library
#include <Adafruit_LSM9DS0.h>  // from "Adafruit LSM9DS0 Library"
#include <Adafruit_Sensor.h>   // From "Adafruit Unified Sensor" library
#define SERIALDEBUG

#define PIN 12       //output pin for NeoPixel controls
#define NUMPIXELS 16 //number of pixels in ring
#define MAXBRIGHTNESS 30

int delayval = 200; // milliseconds between samples
int gyro[] = {0, 0, 0};

Adafruit_LSM9DS0 lsm = Adafruit_LSM9DS0();
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_
KHZ800);

//Adafruit sensor setup routine definition

void setupSensor() {
  lsm.setupAccel(lsm.LSM9DS0_ACCELRANGE_2G);   // set up accelerometer
  lsm.setupMag(lsm.LSM9DS0_MAGGAIN_2GAUSS);    // set up magnetometer
  lsm.setupGyro(lsm.LSM9DS0_GYROSCALE_245DPS); // set up gyro
}

void setup() {
  Serial.begin(9600);
  pixels.begin(); // initialize the NeoPixel object

  while (!lsm.begin()) { // blink red if sensor initialization fails
    for(int j = 0; j < 2; j++) {
      for(uint16_t i = 0; i < NUMPIXELS; ++i)
        pixels.setPixelColor(i, j * 10, 0, 0);
      pixels.show();
    delay(500);
    }
  }

  for(int i = 0; i < NUMPIXELS; ++i)
    pixels.setPixelColor(i, 0, 0, 0);  // turn pixels off
  pixels.show();
}// end setup

void loop() {
  lsm.read();  // read the value of the gyro

  gyro[0] = abs((int)lsm.gyroData.x / 1024);
  gyro[1] = abs((int)lsm.gyroData.y / 1024);
  gyro[2] = abs((int)lsm.gyroData.z / 1024);

  for(int i = 0; i < NUMPIXELS; ++i)
    pixels.setPixelColor(i,
      constrain(gyro[1], 0, MAXBRIGHTNESS), //check one axis for red
      constrain(gyro[0], 0, MAXBRIGHTNESS), //check other axis for green
      0
    );
  pixels.show();
  delay(delayval);

// Rest of the code puts out debugging values to serial port
  #ifdef SERIALDEBUG
    Serial.print("Gyro X: "); Serial.print(gyro[0]);
    Serial.print("     Y: "); Serial.print(gyro[1]);
    Serial.print("     Z: "); Serial.println(gyro[2]);
  #endif
} // end loop
```

Loading the Code and Using the Hat

To send the Arduino sketch to the Flora, plug the Flora into your computer via USB and upload it (Chapter 6). Plug it into the battery and click the reset button on the Flora. You should see it flash red and then turn off if you hold it still. Shaking it up and down should give you green lights in the NeoPixel ring, and side to side should give you red. If you like giving monosyllabic answers, now you can do so from across the room!

The Light-up 60s Mod Dress

The other project in this chapter takes a very different approach from the first one. In this section, Lyn walks you through a project that requires creating a pattern from her specifications, sewing a dress, and then adding some electroluminescent (EL) ribbon for effect. *EL wire* (`https://en.wikipedia.org/wiki/Electroluminescent_wire`) is a conductive wire with a phosphor painted on it. When alternating current is applied, the phosphor glows. *EL ribbon* is a flat version of EL wire that is good for attaching to costumes for light-up effects. Lyn first talks about the heritage of the design and goes over how to create a pattern and measure yourself. Then she walks through how to sew the dress and add the EL ribbon.

The Design

In 1965 Yves St. Laurent designed a collection of dresses that were A-line, almost shapeless shifts with blocks of white and primary colors separated by black lines. They were made from wool jersey and heavy silk, which allowed them to keep their shape without draping the body of the wearer. It was called the Mondrian Collection—an homage to the modern artist Piet Mondrian and to modern art in general. Throughout the rest of the 1960s, variations on this look appeared everywhere, and it became iconic of the era.

I won't be using wool or silk, and the dress will have a fuller cut with a more comfortable neckline. We are updating the look a bit, but trying to keep the feel of the original dress. This is a great 1960s costume and is an intermediate level sewing project. Figure 11-4 shows the completed outfit.

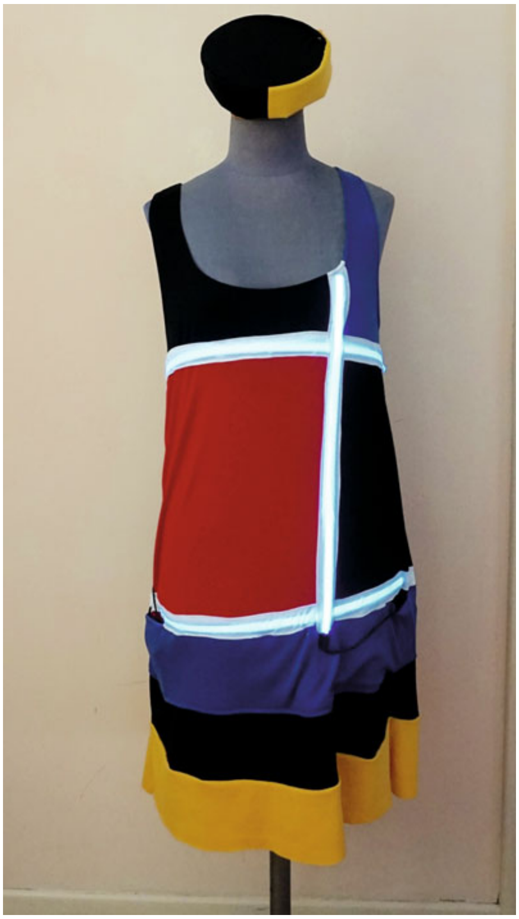


Figure 11-4. *The completed mod dress, with the EL ribbon illuminated*

There are several patterns available for this style dress, like this one: https://betsyvintage.com/index.php?main_page=product_info&products_id=120.

Materials and Tools

To make the dress, you will need the following:

- Fabric (this pattern is designed for stretch knit fabric only—recommended fabrics include interlock, matte jersey, fabrics with spandex, cotton t-shirt fabric).
- Thread.
- Straight pins/cushion.
- Scissors.
- Sewing machine.
- Butcher or pattern paper.
- Tailor's chalk or water-soluble pen.
- Measuring tape.
- Two pieces of EL ribbon (also known as EL tape), each one meter long, with a connector on each end (most EL ribbon comes with two connectors, one on either end).
- Two inverters/battery holders intended for use with EL ribbon/tape and rated for at least 1 meter of EL tape, which is equivalent to 4 meters of EL wire. (If the description mentions a length of EL wire, divide that number by 4 to get the rating for tape.)
- Batteries (typically AA or AAA).
- Male-female connectors from inverter to EL ribbon, one long enough to get from the blue band to the upper horizontal strip. If your inverter doesn't have multiple outputs, you will also need a "Y" connector that can attach two strips, unless you want to have three inverters. (Look carefully at the connectors on the EL ribbon and battery/inverters you get, to be sure you get the right kind.)

The dress is a sleeveless, A-line pullover garment with no zippers or buttons. That is why it is very important to use stretch knits. You will not be able to put it on if you use fabric with less than 35% stretch across the grain (ask the people in the fabric store for advice, or order online with this specification).

I am playing with the original design and colors to make it our own. We are going to use white EL ribbon to enhance the seams, so the lines on our dress will be white. EL ribbon will be inside white fabric channels that divide the color blocks on the piece.

I chose red, blue, and yellow as the primary colors, but they can be any color you like. I also used black for the back, upper right chest, left mid-torso, and lower skirt portions. I added a dropped waist or hip band, which enabled me to create pockets for the electronics and batteries that would power the EL ribbon.

To make the dress washable, I made channels out of the white fabric. These make it easy to slip the EL ribbon in and out.

■ **Caution** EL wire and ribbon generate a high-pitched tone that upsets some pets and people who can hear it. To lessen this, avoid powering too much wire with one power supply. They can also generate RF interference, so don't use in environments where RF interference could cause an issue, such as around medical devices or scientific hardware. For theatrical applications, test this (and other electronics) out carefully in full dress rehearsal.

Taking Your Measurements and Creating the Pattern

The first thing you need to do is to measure yourself carefully. Figure 11-4 is a sketch that labels each of the measurement points you will use in the following directions.

■ **Tip** If these instructions get too intimidating, you can also buy a plain A-line black dress that fits you and add on some of these pieces by folding and pressing under at the edges and topstitching them on. Or, you could just add EL ribbon to another dress, but you will need to find a place to store the inverter and batteries (or attach them to a belt).

Measurements for the Front of the Dress

First we will take all the measurements for the front of the dress. The next section will take care of the back. The letters refer to the labels on Figure 11-4.

For the upper chest and yoke sections, measure from

- The top of the shoulder to just above the bust line (*A* to *B*).
- The left side of the body (where side seam would normally be on a slightly loose-fitting t-shirt) to directly under the left clavicle (*C* to *D*).
- The right side seam to directly under the left clavicle (*D* to *E*).

For the mid-torso sections, measure from

- Just above the bust line to the hip (*F* to *K*).
- The left hip at the side seam to the hipline under the left clavicle (*K* to *N*).
- The right side seam to meet the left side measurement (*K* to *N* on the other side).

For the skirt sections and lower band, measure

- From the hip line to the mid-thigh (*K* to *Q*).
- Around the hips and add 3 inches (for seam allowance and a loose fit, at a level roughly that labeled *P* in Figure 11-5).
- The bottom of the front and back skirt pieces to be the width you want them to be. Cut one piece of fabric 4 inches wide by this measurement for the bottom band (the piece marked *S* in Figure 11-5).

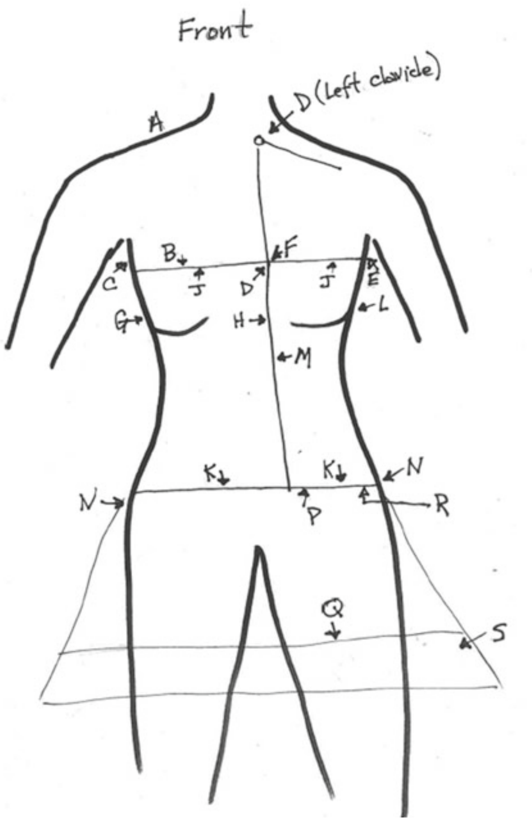


Figure 11-5. *Measurement labels for the front of the dress*

Measurements for the Back of the Dress

The back of the dress is pretty simple. You only need one measurement, really, from the neckline you want in the back to mid-thigh (shown in Figure 11-5). But make sure the back and the front measurements together allow for correct fit on the person wearing the dress by measuring all the way around the body and checking that the pieces collectively will fit (shown in Figure 11-6). If in doubt, make them a little larger and you can trim later after checking fit partway through.

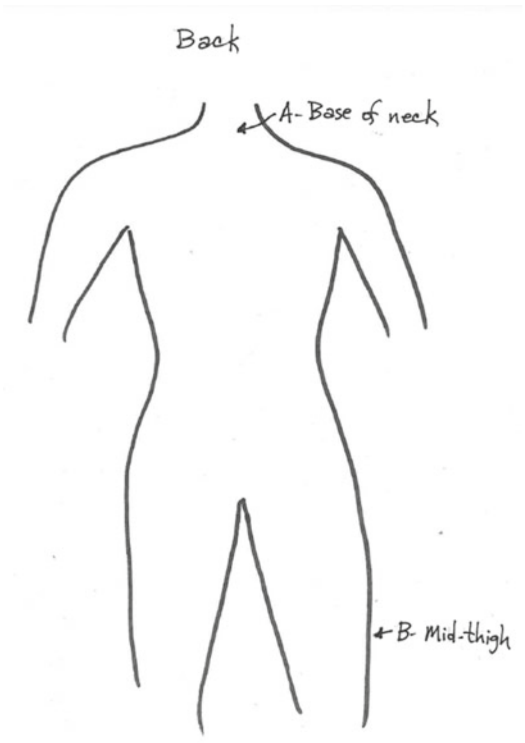


Figure 11-6. *Measurement labels for the back of the dress*

■ **Tip** I bought about 1 yard of black fabric, 1/2 yard of red, 1/4 yard of blue, 1/4 yard of yellow, and 1/2 yard of white. Buy a bit more than you need in case of mistakes.

Creating the Pattern Pieces

After taking all the measurements mentioned so far, use the drawings in Figures 11-7 through 11-12 to draw the pattern pieces onto butcher paper or other suitable paper. Make sure to add a seam allowance (5/8 inch or 1 inch, whichever is easier for you) on the paper before cutting it out and pinning the pattern pieces to the fabric. Note that the *A* and *B* labels on the front and back do not indicate the same point on the garment.

■ **Tip** Cut out the pattern pieces on paper and try lightly taping them together to see whether you like the effect before you commit to cutting out the pattern on fabric. Or you can try creating a smaller paper version to tinker with (at 1/12 scale, for instance) to see if you want to make modifications. See Chapter 4 for details on using a pattern.

The neckline on this dress is somewhat higher than the dress shown in Figure 11-4. I designed it so that you can either have a relatively conservative neckline or trim it some to be a little more rakish.

■ **Tip** Unless otherwise stated, always pin and sew the fabric pieces with the right sides together. Always trim any seams that are more than 1/4 inch.

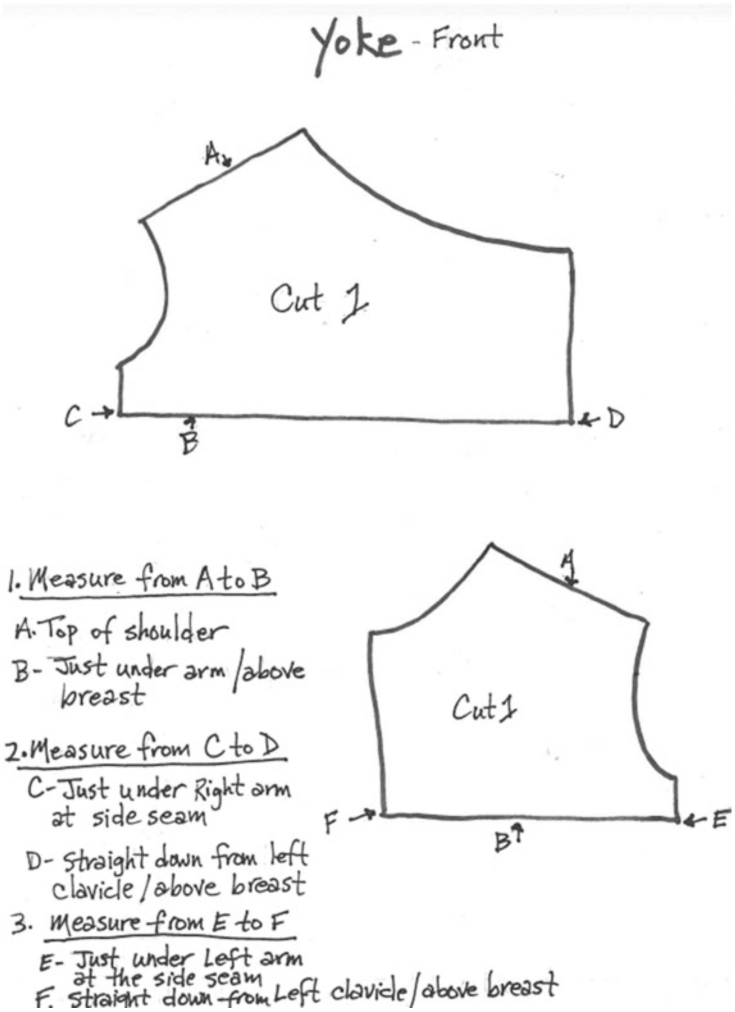

Yoke - Front
A
Cut 1
C
B
D
A
Cut 1
F
B
E
1. Measure from A to B
A. Top of shoulder
B- Just under arm / above breast
2. Measure from C to D
C- Just under Right arm at side seam
D- Straight down from left clavicle / above breast
3. Measure from E to F
E- Just under Left arm at the side seam
F. Straight down from Left clavicle / above breast

Figure 11-7. *Pattern pieces for the yoke: top piece black, bottom piece blue*

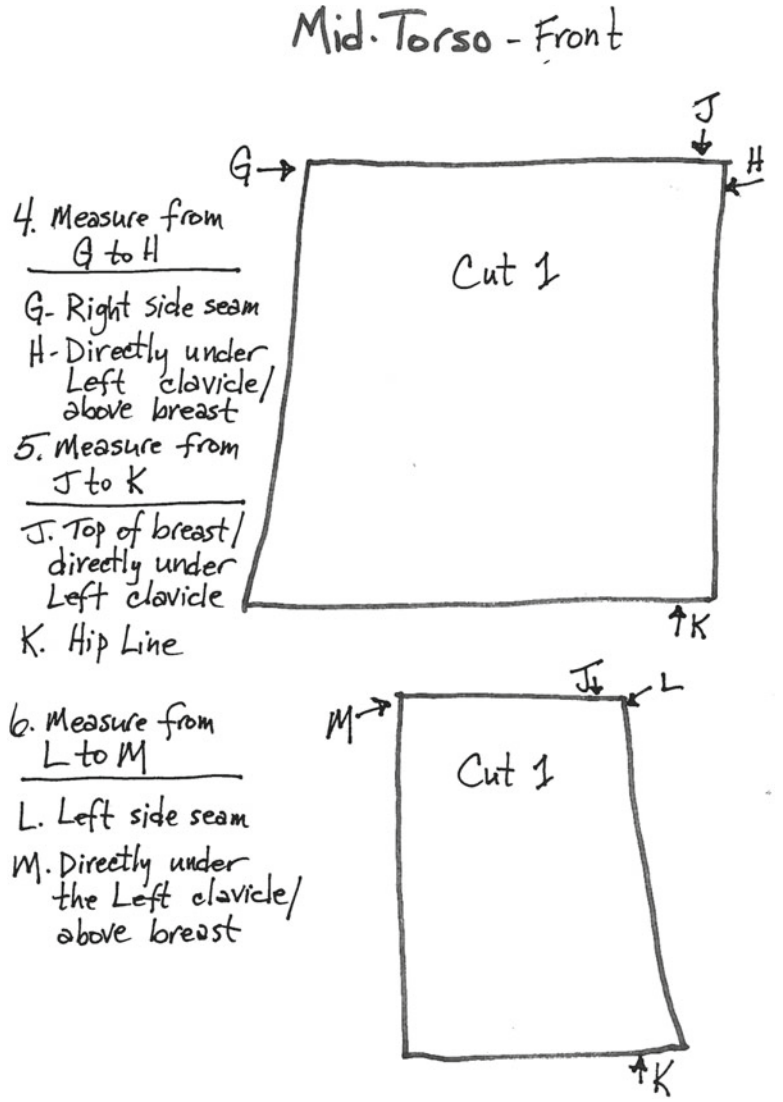


__Figure 11-8.__ Pattern piece for mid-torso: top piece red, bottom piece black

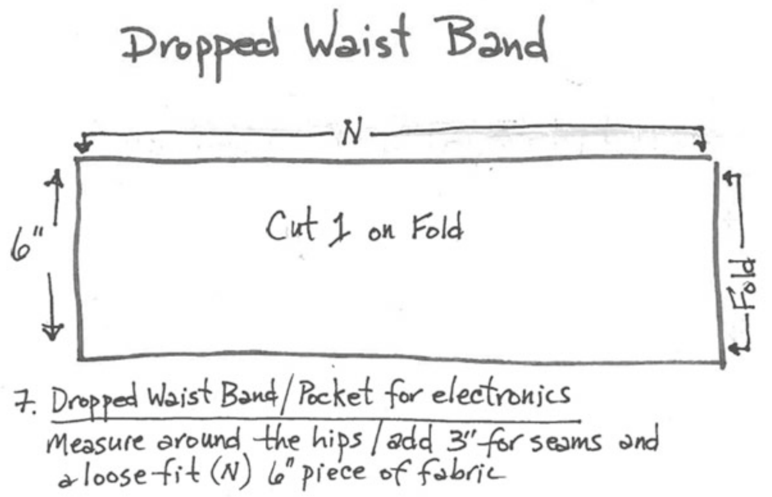

Dropped Waist Band
N
6"
Cut 1 on Fold
Fold
7. Dropped Waist Band/Pocket for electronics
Measure around the hips/add 3" for seams and a loose fit (N) 6" piece of fabric

Figure 11-9. *Pattern piece for dropped waist band (blue) for pockets to hold electronics*

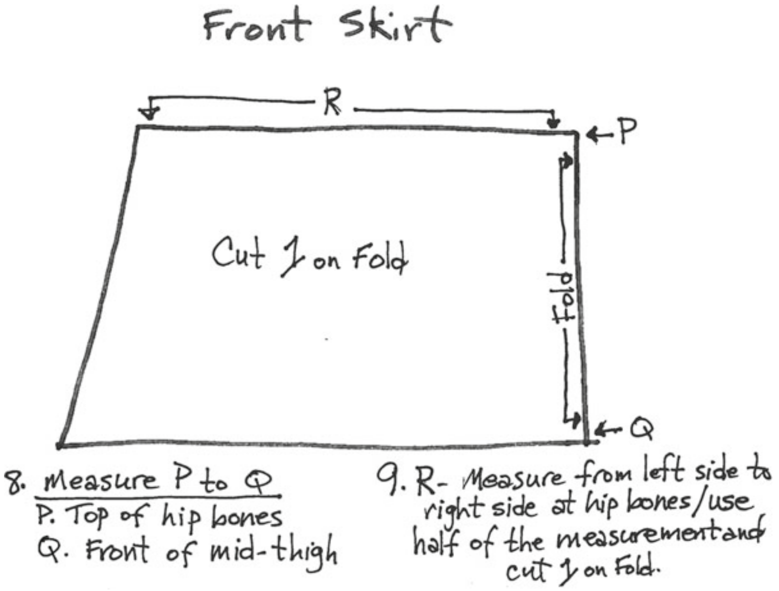

Front Skirt
R
P
Cut 1 on Fold
fold
Q
8. Measure P to Q
P. Top of hip bones
Q. Front of mid-thigh
9. R- Measure from left side to right side at hip bones/use half of the measurement and cut 1 on Fold.

Figure 11-10. *Pattern piece for skirt (black, will be under blue pocket band)*

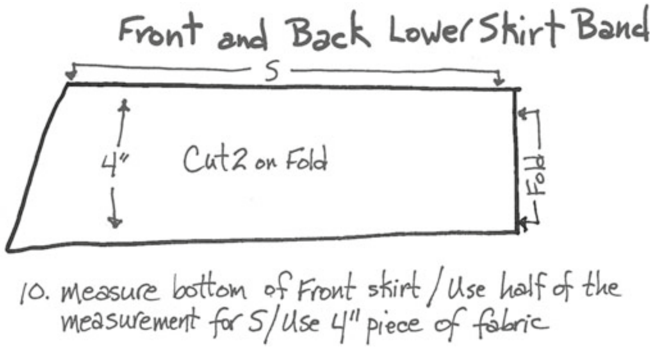

Front and Back Lower Skirt Band
S
4"
Cut 2 on Fold
Fold
10. measure bottom of Front skirt / Use half of the measurement for S / Use 4" piece of fabric

Figure 11-11. *Pattern piece for lower skirt band (yellow)*

A few special notes are in order for measuring the back. In addition to the measures shown in Figure 11-12, also measure from side seam to side seam at mid back and across hips. Add 1 to 2 inches at the bust level for side seams and about 3 inches for seam and a bit of room at the hips. The back is cut on the fold so there is no back seam.

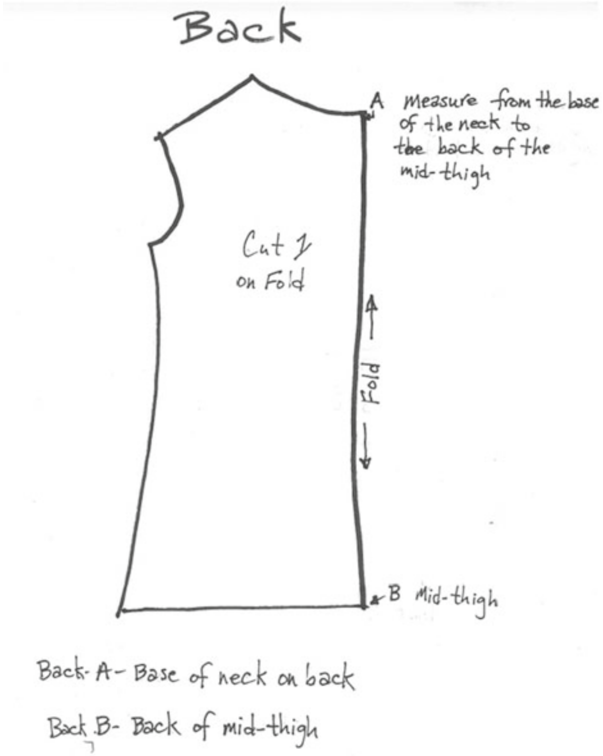

Back
A measure from the base of the neck to the back of the mid-thigh
Cut 1 on Fold
Fold
B Mid-thigh
Back- A- Base of neck on back
Back B- Back of mid-thigh

Figure 11-12. *Pattern piece for the back of the dress (black). Note that the A and B labels here are different from those in the front*

Making the Casings for the EL Ribbon and Wire

The dress needs three casings for the three strips of EL wire. However, it is easiest to measure those later, when the dress is nearly assembled, but make sure you buy enough fabric for all three. You may also want to create a conduit for the wire that will run from the upper EL ribbon band to its battery/inverter in the blue band pocket—or you may just want to use some other means of managing that wire, depending on how often and how long you plan to wear it with the EL ribbon there. Other alternatives for the black casing might be just some loops to keep it from moving around too much (which is what I did in this demo dress), but a conduit would be more secure.

Organizing the Pieces

At this point, you should have all the pieces for the dress and the EL wire casings. Figure 11-13 shows the major pieces and one of the casings in a test layout. Note that the blue band near the bottom of the skirt will be a patch pocket on top of the black band. In other words, the black band is sewn to the yellow band on the bottom and the red and black sections above it. The blue will ride as an additional double thickness on top of that, as a pocket to hold the batteries. You may want to review Chapters 3 and 4 if any of the steps here seem unfamiliar.

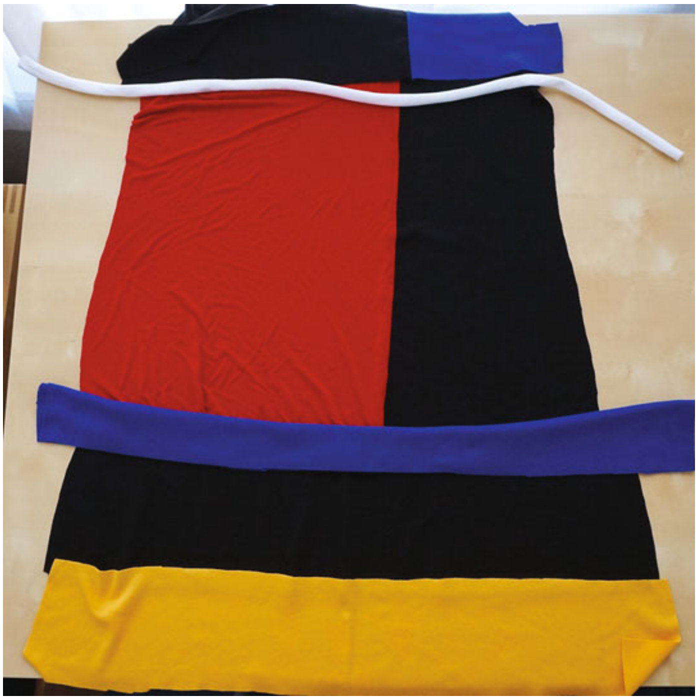


Figure 11-13. *Laying out the front of the dress*

Sewing the Front

For the upper chest (yoke) pieces, pin the two pieces together along the front seam and sew them together using a straight stitch. Trim the seam and press it open. Next, pin and sew the mid-torso sections together along the front seam. Trim that seam and press it open.

Match the vertical seam on the yoke and the vertical seam on the mid-torso piece and pin them together along the horizontal seam (with right sides together). Check that the sides of your pieces match when the vertical seams are lined up. If you need to trim a bit, that's fine. But if your side seams don't line up because one is smaller than the other, you may need to take one of the vertical seams out and redo it, or take one of them in a bit more—whichever way will make the garment fit is the right choice.

Hold the pieces up to you in front of a mirror to check for appropriate fit. Once you are happy with it, pin and sew the yoke section to the mid-torso along the horizontal seam. Trim the seam and press it open. To finish the front, pin and sew the front skirt section to the mid-torso section along the horizontal seamline, and as always trim the seam and press it open. Pin and sew the front to the back at the shoulder seams. Trim and press the seams open.

■ **Note** The neckline shown in the pattern is quite a bit higher than that in the dress demonstration we show here. You can start with this one and always trim a little if you want a lower neckline. We recommend the higher one for dress stability.

Adding the Back

Pin and sew the front to the back at the side seams. You may have to stretch the fabric slightly to make the sides fit together all along the side seamlines. That is normal and to be expected when you are creating and cutting out your own pattern. Be careful not to overstretch the fabric because it will distort the shape of the dress. Figure 11-14 shows an inside-out view of this stage.

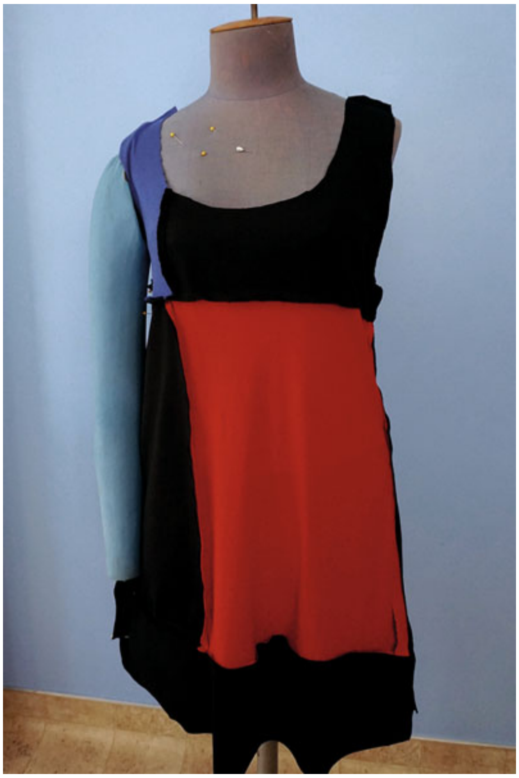


Figure 11-14. *Right after sewing the back to the front (dress is inside-out)*

Pin and sew the bottom (yellow) band to create a circular piece. Trim seams and press them open. Fold the piece in half and mark the center-front, which will be at the opposite side of the seam you have just sewn. In other words, the seam of the yellow band will be at center back. Pin the piece to the bottom of the skirt, starting with the seam at the center-back of the skirt and the center-front mark at the center-front of the skirt. Ease and stretch the fabric a bit as you pin the rest of the band to the skirt. Make sure it looks even, not too stretched in one place or too bunched up in another. Sew this seam, trim, and press open. Figure 11-15 shows where we are now, right-side-out.

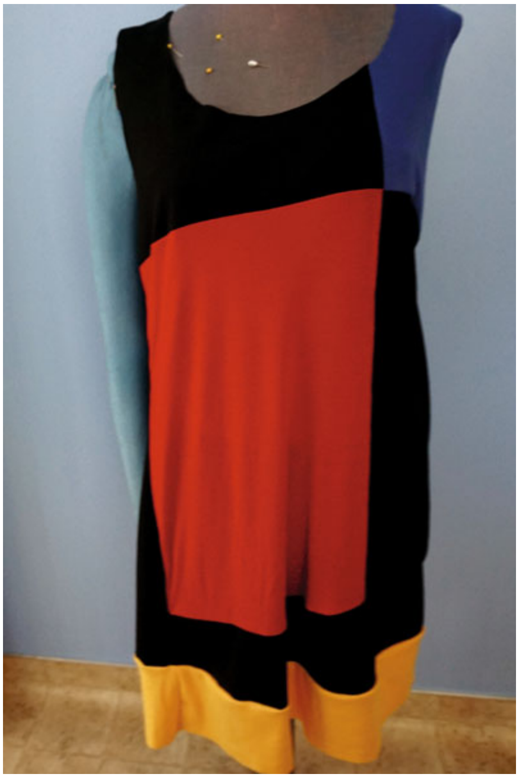


Figure 11-15. *Assembly after adding the yellow band*

The Dropped-Waistband Pockets

Fold the dropped waistband piece (blue) in half with right sides together and sew the short seam to make a loop of fabric (show in Figure 11-16). Turn the fabric right-side-out and fold the piece in half lengthwise. Mark the center-front the same way you did for the lower skirt band. Pin the fold of the band to the skirt at the seam line of the mid-torso and skirt sections. Start with the center-back of the dress and the seam of the band together, then pin the center-front of the skirt and the band together.

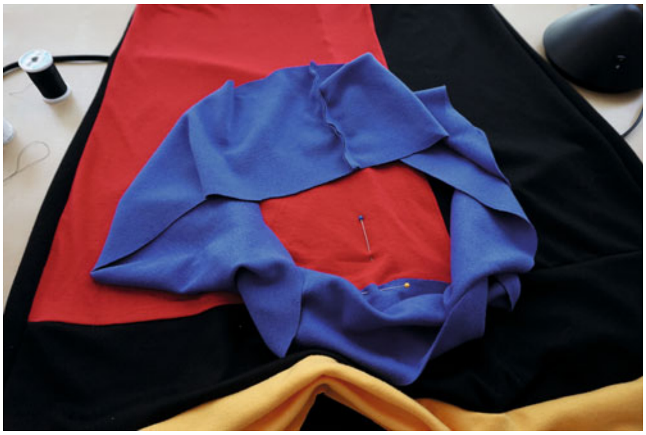


Figure 11-16. *The blue band that will be the electronics pocket pinned to the top of the black skirt band*

Continue pinning the blue band evenly all the way around the dress. Pin the bottom of the band in place all the way around the skirt, stretching and fitting it evenly. Sew the bottom of the band to the skirt, making sure it doesn't bunch up or overstretch. Mark the top of the band on the front where you want the openings for the two pockets, where you will insert each of the battery/inverter packs. These two areas will not be sewn closed. Sew all the way around the top of the band, leaving the pocket sections open (Figure 11-17). Check that the inverter/battery pack you have will fit into the pockets (Figure 11-18).

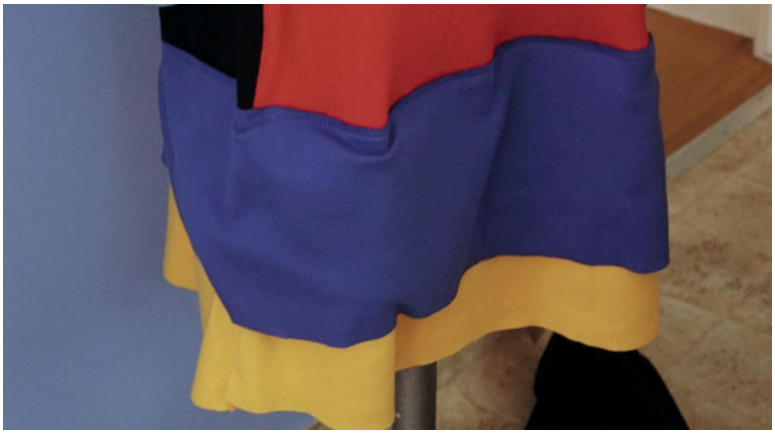


Figure 11-17. *Attaching the blue band, leaving open space for pockets*

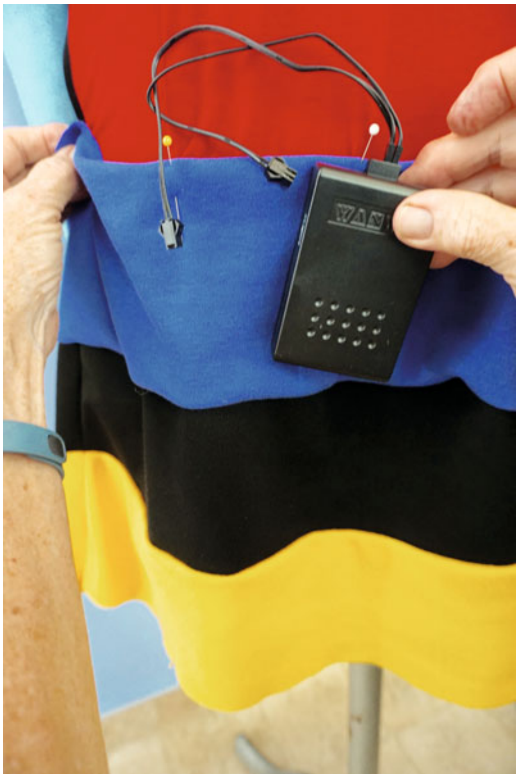


Figure 11-18. *Checking that the battery pack/inverter will fit into its pocket*

The EL Ribbon Casings

Measure the horizontal seams on the front of the dress (where the yoke and torso attach and where the torso and skirt attach. Cut two strips of white fabric 3 inches wide and 2 inches longer than each horizontal seam.

Next, measure the vertical seam from the neckline to the top of the dropped waistband. Cut one piece of white fabric 3 inches wide and 2 inches longer than this vertical seam measurement. If you want a conduit for the wire from the upper strip of EL ribbon down to the pocket with the inverter, cut another piece of black fabric the same length as the vertical white one.

For each EL ribbon casing, fold and pin the white fabric in half lengthwise and stitch this seam about 1/4 inch from the edge. Use a safety pin at one end to turn the casings right-side-out (Figures 11-19 and 11-20).

Pin and sew the two horizontal casings to the front of the dress in the correct locations. Sew very close to each side of each white strip, leaving the ends open. This will allow you to insert and remove the EL ribbon as needed.

Pin and sew the vertical casing to the front, making sure to stop sewing at the top of each horizontal white strip and start sewing again at the bottom of the horizontal strips. You will be able to tack the vertical piece in place with a few stitches of hand sewing if it is too large a gap.

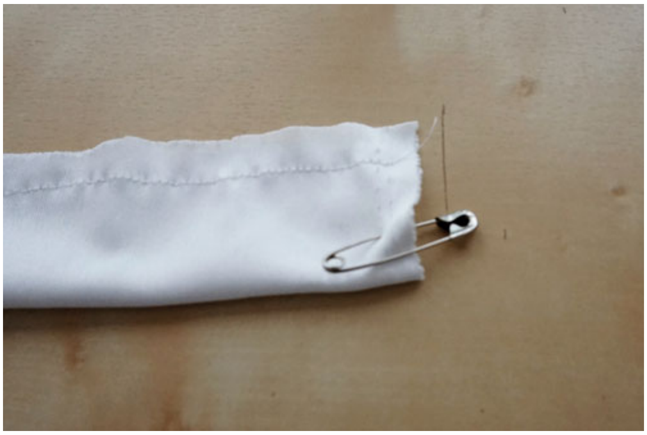


Figure 11-19. *Creating the EL ribbon casing, with a pin to turn it right-side out*

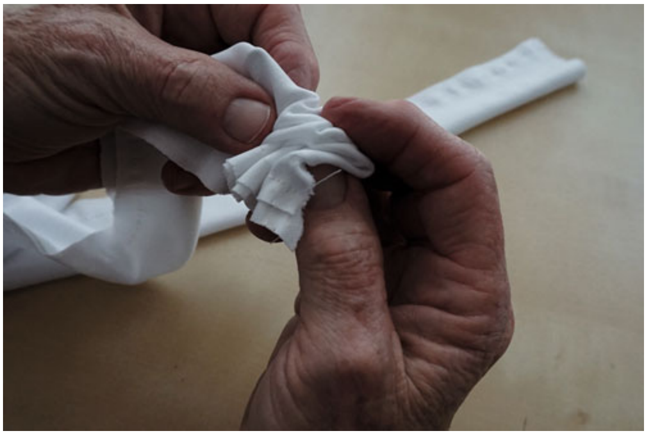


Figure 11-20. *Turning the EL ribbon casing right-side out*

Pin the ribbon casings to the dress and sew them down. Leave one end of each open to insert the wires. Be sure that you leave open the places where the casings cross each other. You may fiddle with the wires and how you want to arrange the EL wire ends a little—when I photographed Figure 11-21 I thought we'd have the wires on one side, but changed my mind. You should probably assemble the whole dress before deciding where your black wire conduit will be. In this case, I discovered that the red fabric was stretchier than the other side, so I wanted to have most of the battery weight on the less-stretchy side.

■ **Caution** One issue with this dress of stretchy fabric is that it is hard to keep the color blocks straight. Check carefully at each step that your seams are straight.

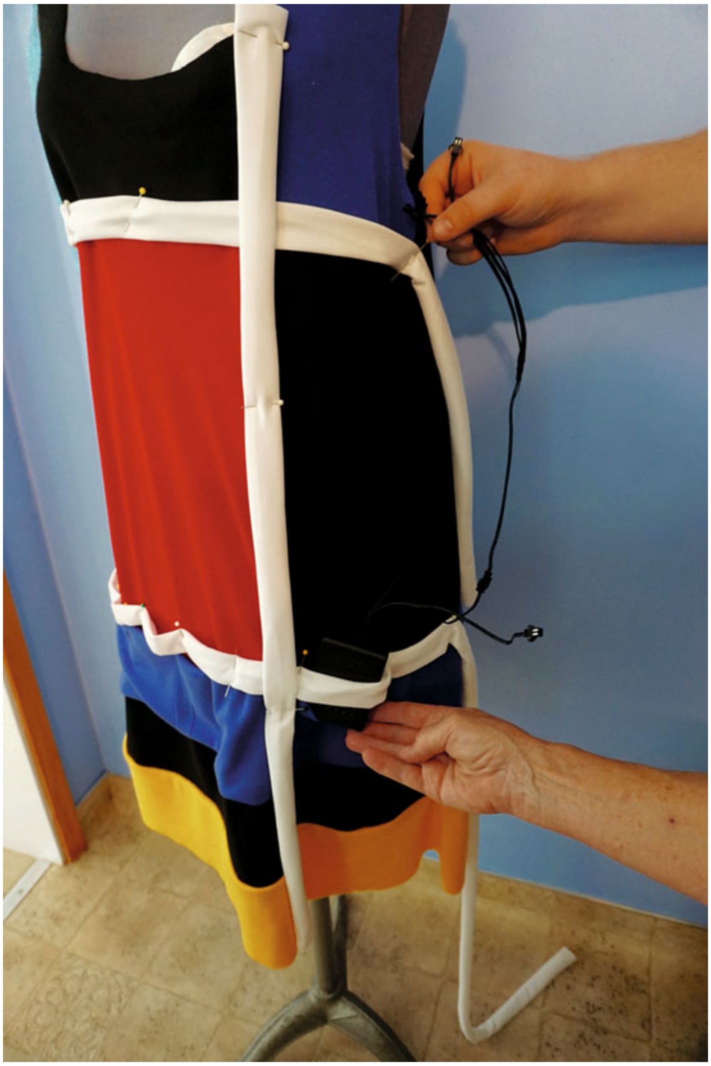


Figure 11-21. *Placing the ribbon casings on the dress and experimenting with wire routing (ultimately I routed from the other side)*

Try the dress on and mark where you want the hem, or just turn it under if it is not too long. Sew a hem with the machine or by hand. If it is a fabric that will not fray, you can just cut it at the hemline. (If you have folded the yellow band in half lengthwise and sewn it on, as in the instructions, you will not need to hem it.)

Turn under the neckline and armholes 1/4 inch (or finish them off with facing, if you want to create those pattern pieces as well—see Chapter 4). Sew on a black channel or other means of holding down the power wire that runs to the upper horizontal EL ribbon—see the "Finishing Up" section for other options that you may want to look at first.

EL Ribbon and Wiring the Dress

Now we will prepare the EL ribbon. You can cut EL ribbon with scissors, but you need to end up with a connector on each piece or it will not work. In the case of the ribbon in Figure 11-22, I cut a 1-meter piece into two pieces (one slightly longer than the other), and that was about right for the two horizontal bands.

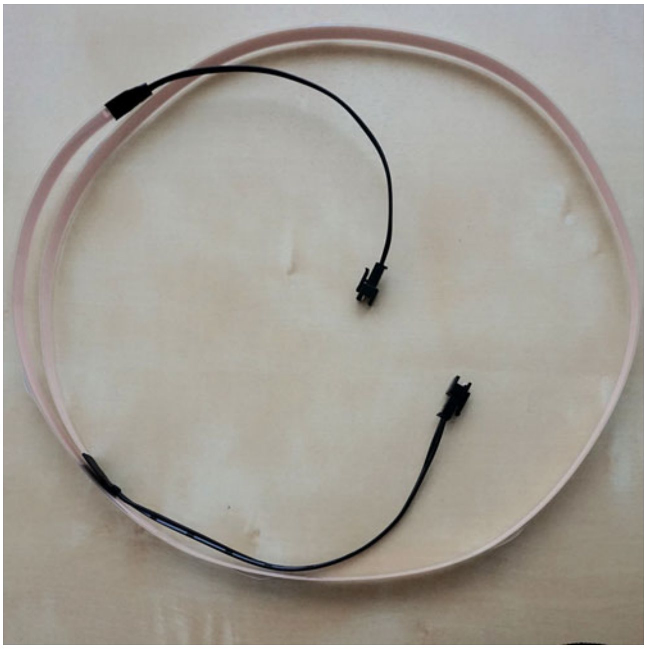


Figure 11-22. *EL ribbon before cutting*

EL ribbon requires an inverter and a battery pack to power it. You can run two short strips like this on one of the bigger (4-battery) packs shown in Figure 11-23. You run two from one pack with a Y-shaped connector cable. I used two of these for the dress.

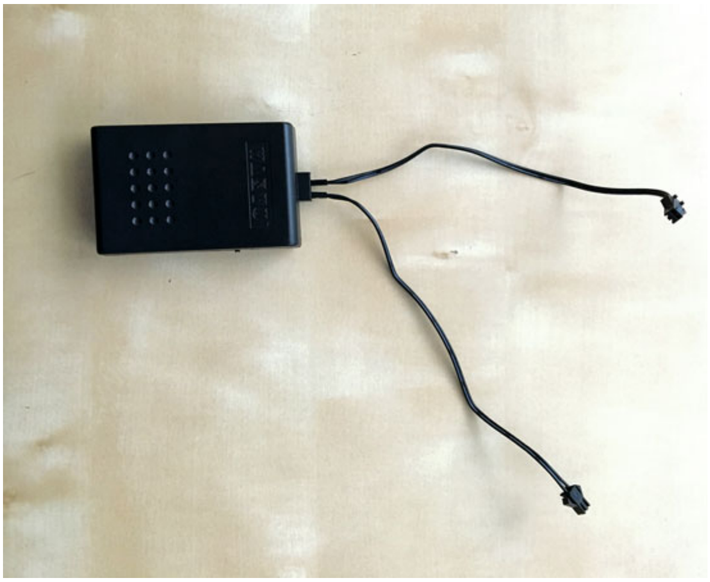


Figure 11-23. *Inverter/battery packs and cabling you will need (with two connectors)*

The EL ribbon comes with a paper backing that covers a sticky layer, with the idea being that you might want to just stick it onto whatever you are illuminating. However, this backing is not very secure, so peel it off and replace it with a layer of regular adhesive tape. Trim the end of the wire to be a bit rounded so it will go into the casing easily (Figure 11-24). Finally, insert the EL ribbon into its casings (Figure 11-25).

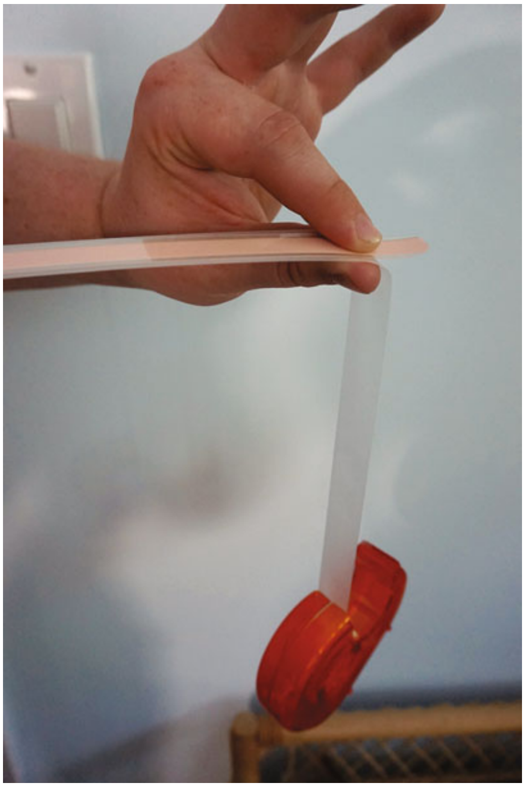


Figure 11-24. *Covering the sticky side of the ribbon with tape*

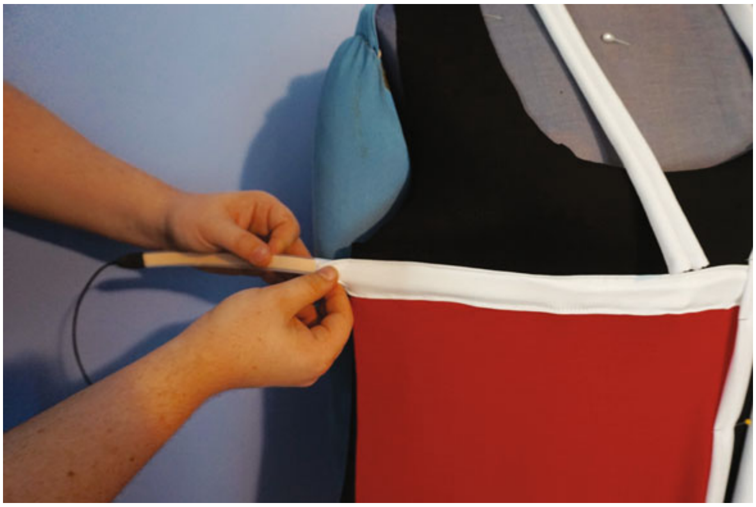


Figure 11-25. *Inserting the EL ribbon into its casing*

Insert the lower horizontal EL ribbon into its casing, and finally the vertical one. Be careful not to sew shut any of the casings where they cross. Tuck the inverters into their pockets on the blue section of the dress (Figures 11-26 and 11-27). Try on the dress and see how things hang.

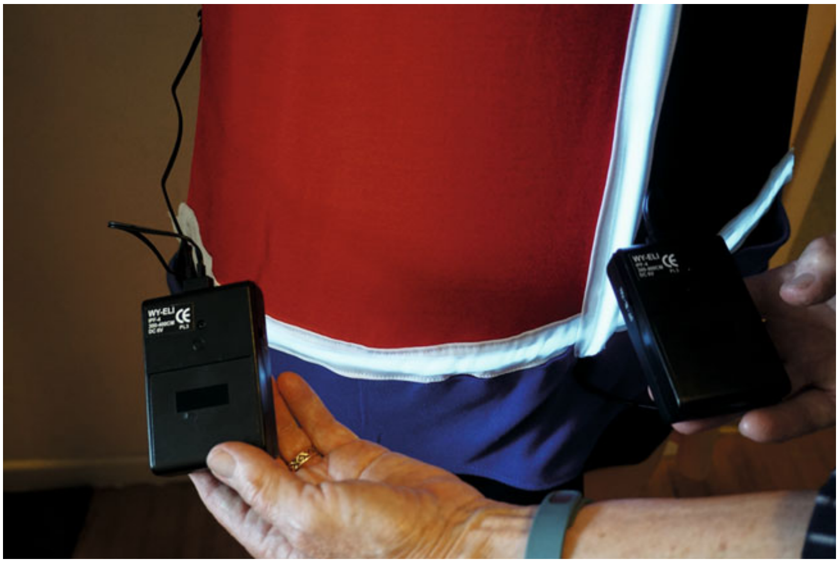


Figure 11-26. *Attaching one inverter to the two horizontal EL ribbons, and the other (on your right) to the vertical one*

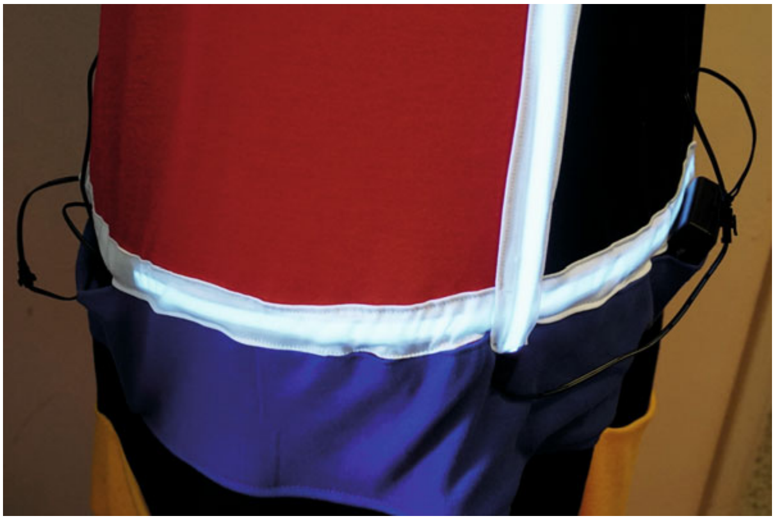


Figure 11-27. *The inverters before they were fully tucked in*

Finishing Up

At this point, try on the dress and gently move around in it. The EL ribbon wants to slide out of the casing, so you may need to sew on a few loops to hold it in place. You may want to sew on a channel for the power wires if you are going to be moving around a lot. Once you are happy with fine adjustments, you can remove the EL wire and add a wire conduit. Or come up with a wire-management solution that works for you, bearing in mind that the wire will not stretch, but the dress will. Be careful sitting down!

If the costume is just going to be worn theatrically once, you might just put a few loops for the wire. Have the power off when you do this and remember that you might need to cut off some of this to wash the garment. Remove the EL ribbon and power sources before washing the garment.

You can add snaps or other closures to keep the pockets containing the inverters from hanging open.

Pillbox Hat Construction

This is my take on the standard 60s pillbox hat, designed to match and compliment the dress. This hat as constructed is 6 1/2 inches in diameter (20 inches in circumference) and 2 inches high, which seems to be pretty stable if you have short hair. You will need the following:

- Fabric (use the leftovers from your dress, or buy additional fabric if needed). I used 4 1/2 inches by 16 inches for the black, and 4 1/2 inches by 7 for the yellow, and I needed a circle 8 1/2 inches in diameter to cover the crown (with enough left over to turn over to tape or glue the edges down).
- A circle of medium weight cardboard 6 1/2 inches in diameter for the crown.
- One piece of buckram that is 21 inches by 2 inches, or one 21-inch by 8-inch piece of heavy duty interfacing fabric (folded into a 21-inch by 2-inch strip). Note that buckram is the stiff material used to make hats. Find it or the heavy-duty interfacing online or at a fabric store.
- Tape, textile glue, or hot glue gun and glue sticks.
- Sewing machine.
- Hand sewing needle and thread.
- Heavy-duty scissors.
- Ruler and measuring tape.
- Pencil or marker.

Cutting the Pieces

I used a 6 1/2-inch diameter circle for the crown of the hat. Draw the circle for the crown onto the cardboard and cut it out with scissors. Cut a circle of the fabric 2 inches larger than the crown circle. To keep the fabric neatly in place, make small cuts in several places around the circle of fabric and overlap them on the underside of the base material circle (Figure 11-28). Glue or tape in place and allow to dry (if glued). If you used buckram, you could sew it instead. Do not try to sew cardboard.

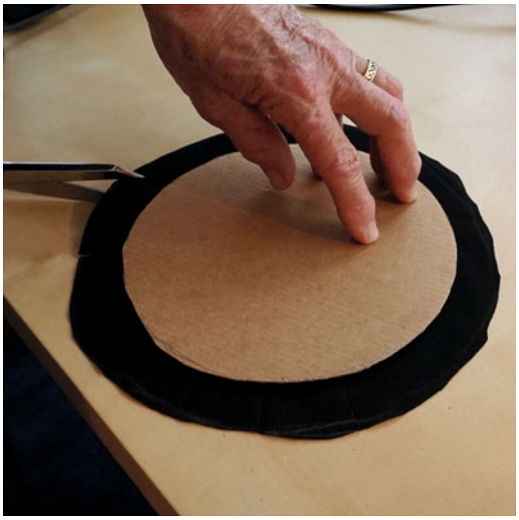


Figure 11-28. *Clipping the edges of the top fabric for the hat*

Cut a strip of buckram 21 inches by 2 inches to give a 2-inch-high hat. I will need fabric strips 4 1/2 inches by 16 inches for the black, and 4 1/2 inches by 7 inches for the yellow. This will make a quarter of the hat yellow and make a sharp fashion statement that ties the dress and hat together with the yellow accents.

Assembling the Hat

Sew the yellow and black portions of the hat side together and press the seam down. Take the strip of buckram, or pin the interfacing into a strip as shown in Figure 11-29, fold the fabric over the strip, and pin. If using interfacing, remove the pins from it as you pin the fabric over it (Figure 11-30). Hand sew the seam along the inside of the hat.

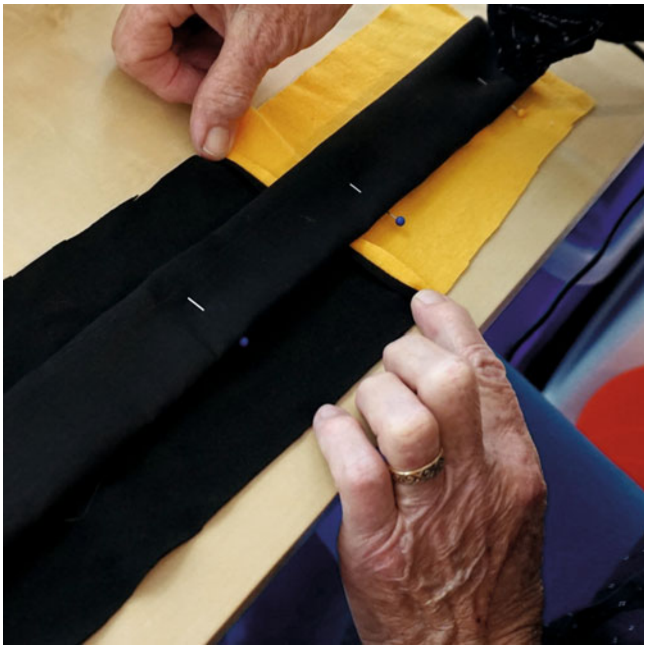


Figure 11-29. *Assembling the side of the hat*

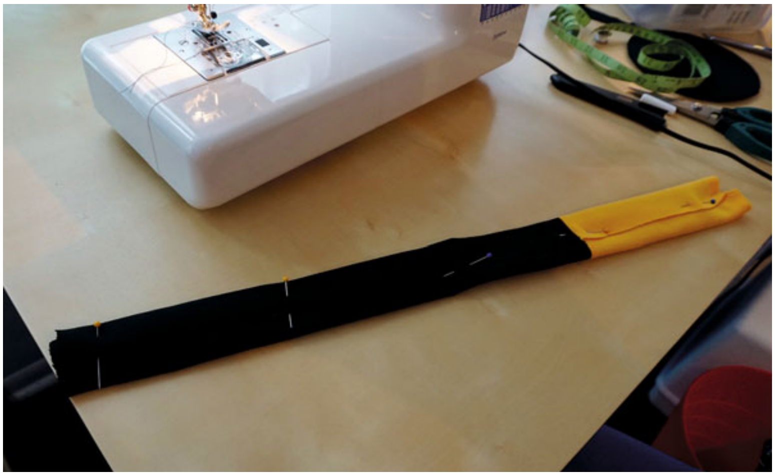


Figure 11-30. *Assembling the side of the hat, continued*

Finally, hand sew the top of the hat to the sides and sew the side seam (Figure 11-31). You now have a pillbox hat (Figure 11-32)!

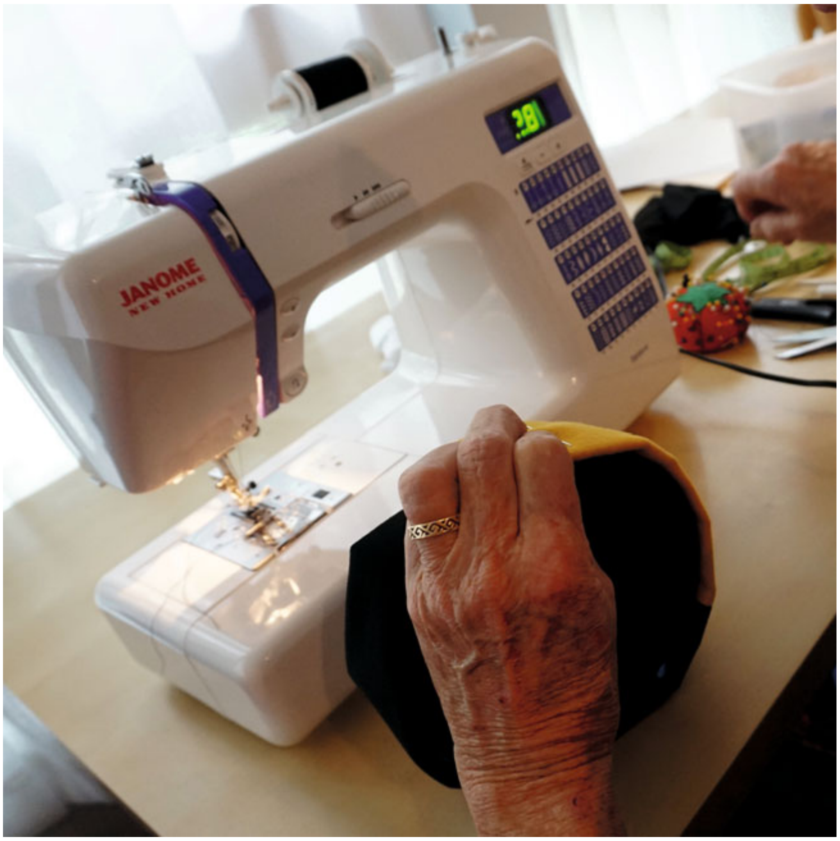


Figure 11-31. *Connecting the top and sides of the hat*

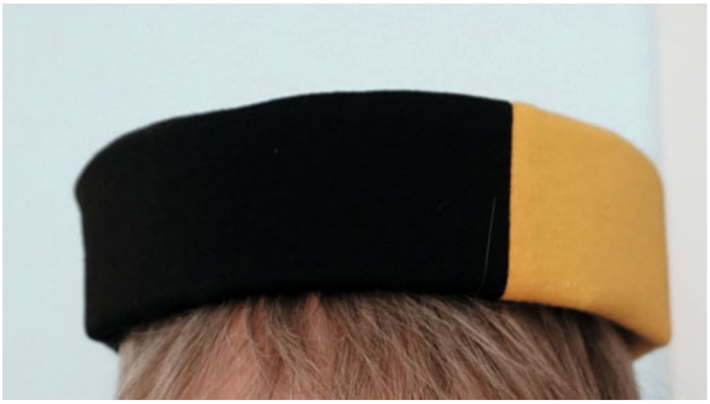


Figure 11-32. *The completed hat*

Options

The completed outfit is shown in Figure 11-33. You can see that the dress sags a bit with the weight of the electronics. You might want to create an alternate design that carries the batteries in a mod-style belt popular in the 1960s, or possibly reinforce the dress a little with a few strategic strips of buckram from the shoulder to the blue dropped waist band to prevent sagging. You could also make the front one piece and sew the colored panes on top. Wire management is a challenge with this design, and you should experiment a bit with the best way to get a good balance between the drape of the dress, the stiffness of the EL ribbon, and the weight of the battery packs.

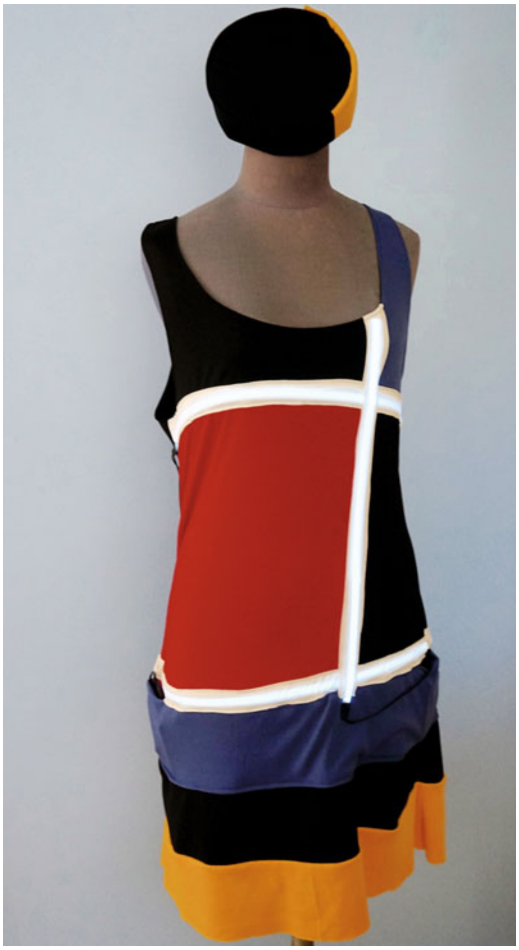


Figure 11-33. *The full effect*

Summary

This chapter covered two very different final projects: one that used a pre-existing hat and added some electronics, and another that did not require any programming but did require some fancy sewing. The intent was that these two projects between them cover the application of what you have learned in the book about construction techniques. The final chapters will review other technologies you might consider looking into and give some speculations on what will come in the future.

PART IV

Where to Go From Here

In this Part we take a broader view than the previous Parts. In Chapter 12, we show you some other technologies, like laser cutting, foam armor creation, and vacuum forming, that are frequenly used by cosplayers to replicate things that are not supposed to be made of cloth. Chapter 13 concludes the main part of the book by looking at where all of this can lead, with examples of high-end fashion designers that are using these technologies now and some speculation as to how these techniques may enter the mainstream sooner than you might think.

We have also included two appendices that will help you explore the topics in this book some more. In Appendix A, we give you the benefit of our experience with designing a high school pilot class which had fashion tech components (both what worked well, and what we would do differently). Appendix B captures all the links in the book in one ready reference.

CHAPTER 12

Other Technologies

One of the challenges of writing a book like this is knowing when to stop. If you go to a large public library to look at their section of sewing, fashion, and similar craft books, you most likely will find shelf after shelf of books on those topics. This book focuses on the overlap of sewing and electronics, with a bit of an introduction to 3D printing thrown in.

This chapter gives a brief introduction to a few more tools and techniques that are becoming common, particularly in the cosplay community. Some of these involve tools that are not really available for home use yet, but that are probably available in a makerspace or community college if you have access to resources like those.

Cutting Tools

In this first section we talk about ways of cutting things. Chapter 9 introduces 3D printing, which creates objects by building them up a layer at a time. Laser cutters and CNC (computer numerically controlled) tools both work more traditionally, by removing material from an initial piece of raw material. Neither machine is particularly suitable for home use, although smaller CNC machines are now on the market.

Laser Cutting

A laser cutter uses a laser to burn through (or melt) a piece to cut it into desired shapes. Laser cutting can be used to make very intricate things, as a search online for "laser cut art" will quickly reveal. The minus is that laser cutters are not really home machines, for safety, cost, and logistical (for example, size, noise, and air quality) reasons.

When a laser cuts something it creates both smoke and some fumes from whatever is burning away, so laser cutters need to be vented, either to the outside or to an air scrubber. Laser cutters can produce very toxic fumes if you cut the wrong thing, which can damage both the laser cutter and the person running it. The cutter has to be monitored closely to make sure that the object being cut does not catch fire, and regular maintenance is needed. Given all that, most facilities will either do the job for you or make you take some training, so we will not get into details here beyond covering some concepts about what is possible and how to create a file for laser cutting.

Cutting Fabric

Laser cutters can cut many kinds of cloth. Some fabrics will melt around the edges, which can either mean you have a nice sealed edge or your fabric melted away, depending on the skill of the operator that sets it up. Cotton cuts cleanly but can discolor a little from the cutting (burning) process. Figure 12-1 shows a laser cutter set up to cut fabric, and Figure 12-2 displays some examples of laser cut test pieces. If you can, try a very small test first in case it does not work out well. That way you can still fall back on using scissors without ruining your fabric.

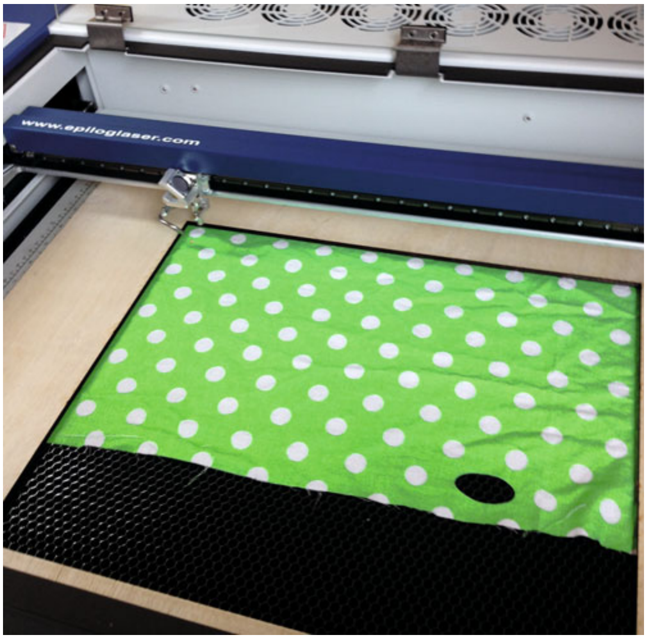


Figure 12-1. *Laser cutter setup after cutting small oval of fabric*

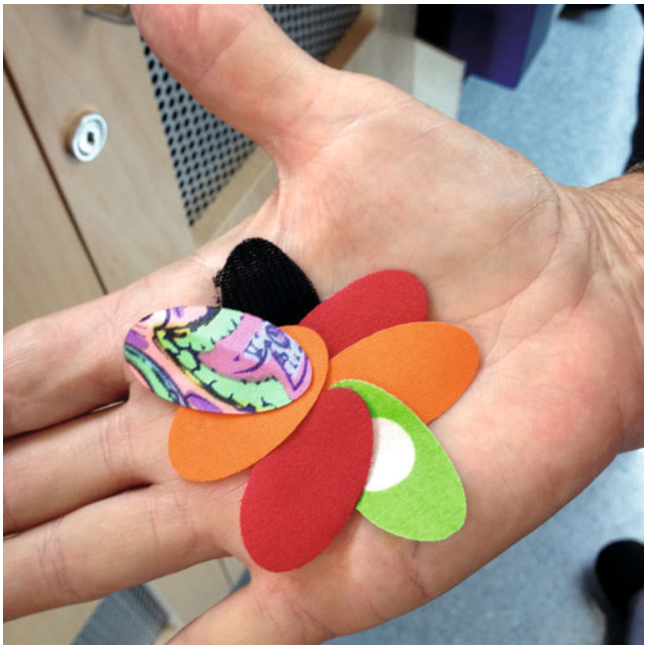


Figure 12-2. *Test pieces of various fabrics after laser cutting (the green one was cut in the setup of Figure 12-1)*

Designing for a Laser Cutter

Like a 3D printer (see Chapter 9), a laser cutter only works from a computer file—you cannot "draw" with it. Unlike a 3D printer, a laser cutter works from a two-dimensional drawing. Usually this is a file in Scalable Vector Graphics (SVG) format that you create in a drawing program such as Adobe Illustrator or even Tinkercad (see Chapter 9). You can specify whether you want to have a cut go all the way through or just engrave a piece. Engraving light fabric probably will not succeed, because cutting through it takes so little energy, but you can engrave leather, denim, acrylics, and similar heftier materials.

Vector graphics are different from photographs and other raster images. A *raster* image (sometimes called *bitmap*) is made up of a series of pixels that are all the same size and laid out in a grid pattern. In other words, raster images are like most digital images you see. The only information the image contains about each pixel is its color. If you scale these images up, those pixels just get bigger, and the image looks blocky because there's no information about what's between the pixels.

Vector images, on the other hand, have lines, arcs, and curves that follow mathematical rules so that a computer can always tell what the features should look like when you look closer. The complex shading of a photo would be nearly impossible to represent this way without simplifying it to create a stylized effect (Figure 12-3 shows an example), but things like letters, geometric shapes, and line drawings can be represented with nearly infinite precision in a vector image.

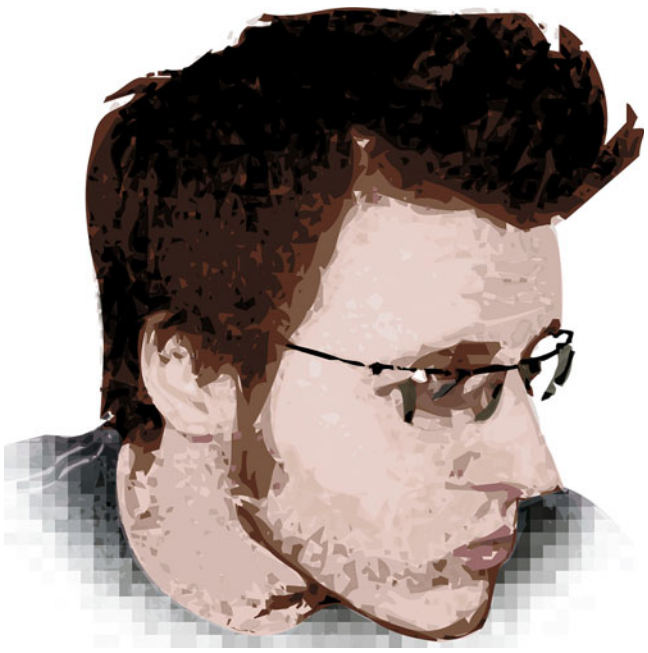


Figure 12-3. *A self-portrait of Rich done in a vector format. Notice the stylized appearance created by the vector shapes*

Cutting with a laser cutter requires a file with vector lines for the laser to follow. You can make these lines in a program like Adobe Illustrator, either by drawing them directly or by tracing lines in a raster image (Illustrator comes with automated tools that can help with this).

As mentioned, you can also use a laser cutter to etch or engrave the surface of a material, and you can do this with vector or raster images. In these cases, the laser will darken (or, depending on the material, possibly lighten) the surface where the laser hits it, allowing you to make a monochrome image. Etching may also remove the top layer of a surface (such as a layer of paint) to reveal a different color below it.

You're not stuck with Illustrator. You can also create a vector file for cutting with a CAD program. TinkerCAD has an option to download an SVG for laser cutting (found in the Download for 3D Printing menu) that will create lines wherever your design is intersecting the workplane. Other CAD programs commonly let you create 2D drawings that you can export in DXF or DWG format. You will need to find out what formats the laser cutter's software accepts, but all programs that run a laser cutter should work with SVG. Inkscape (`https://inkscape.org/en/`) is a free and open source alternative to Adobe Illustrator, and although it is not as full-featured, you may at least find it useful for converting one vector format to another.

■ **Tip** Laser cutting is a great way to make multiple copies of a cutout pattern. Lyn had students laser cut cloth fish for a production, making the fish all consistent while allowing for a lot of variation of color and details.

CNC Milling

A CNC mill is a computer-controlled cutting machine. It moves a cutting tool around in three dimensions to remove material, similar to the way a 3D printer adds material. CNC mills can work with more materials than laser cutters or 3D printers, with possibilities ranging from EVA foam to some metals, but it is a much messier process that involves throwing tiny pieces of cut material out of the way. Even so, these machines may be more amenable to a home workshop than a laser cutter because rather than release smoke and fumes, they create the kind of mess than can be cleaned up with a broom or a wet-dry shop vacuum cleaner.

CNC milling may need to be very slow when working with hard materials like metal, but with softer materials like foam or machining wax, it may be faster than 3D printing for large pieces. CNC mill prices also generally start lower than a laser cutter of comparable size, but can range pretty high upward for industrial machines designed to cut intricate shapes out of metal. Othermill, Inventables, and Carbide 3D are popular brands for hobbyist-level machines.

Construction Techniques

You can use a variety of materials to build up prop pieces or costume armor—complex shapes that might be far too large to 3D print. In this section we introduce you to some common construction techniques practiced in the cosplay community. Obviously there are many craft materials available, from wood to aluminum foil and duct tape. We just touch on some of the more "techie" ones here.

Foam Armor

Cosplayers often want to create make-believe armor or weaponry. A common way to do that is to cut and bend sheets of ethylene vinyl acetate (EVA) foam, commonly used for floor mats. One of the best-known practitioners of the art is Bill Doran, who founded Punished Props (`www.punishedprops.com`) in 2012 with his wife, Brittany. They have extensive resources including a YouTube channel (`www.youtube.com/user/punishedprops`) and the *Foamsmith* series of books available on their site.

Punished Props started out doing commission work, and now Doran mostly trains people to use these techniques. His personal favorite cosplay outfit that he and his wife made for themselves was from the video game Skyrim. They were Drauger Deathlords, with old rusty armor, skeleton faces, and glowing eyes. He enthuses that they looked "inhuman."

Doran says that the key to a good cosplay costume is to "pick a character that [you] really love, and make sure it is something you are obsessed with." He also says that it can take months to make a full set of foam armor, so you really do need to want to wear these costumes!

The process involves starting with EVA foam (either in the form of floor mats or bought in large rolls) and cutting it out to desired shapes (see Figure 12-4) using patterns similar to those used for sewing. You then use a variety of tools to bend and finish it. EVA foam cuts easily with a razor knife (though Bill warns that it dulls the blade quickly), a band saw, or even a laser cutter. You can do the detail work with a rotary tool, such as those made by Dremel.

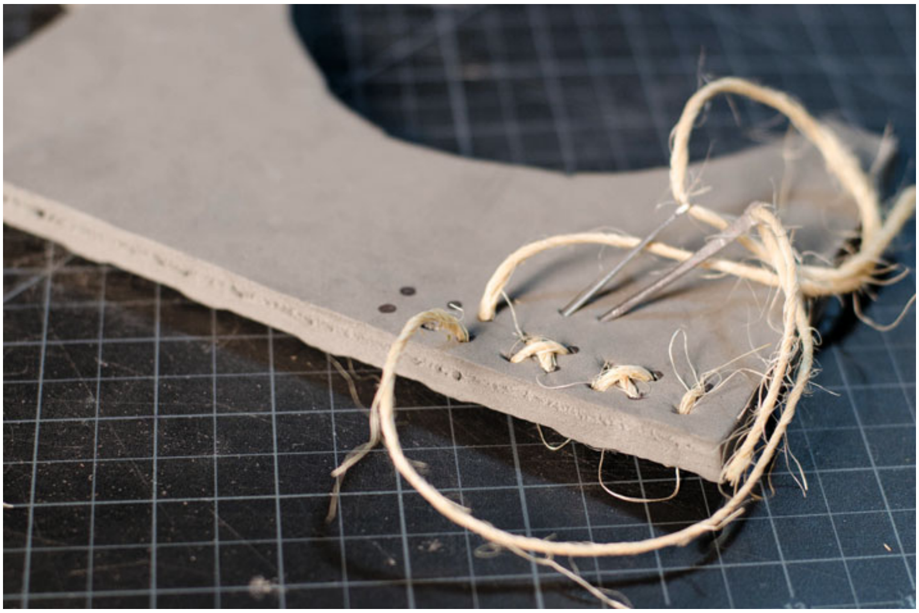


Figure 12-4. *EVA foam costume piece, early in the process (photo courtesy of Bill Doran/ Punished Props)*

Bill says a lot of the techniques involve ways of strapping the pieces to the wearer's body, knowing what to wear under it, and figuring out how to glue pieces together. The finished and painted prop pieces can be quite spectacular (see Figures 12-5 and 12-6).

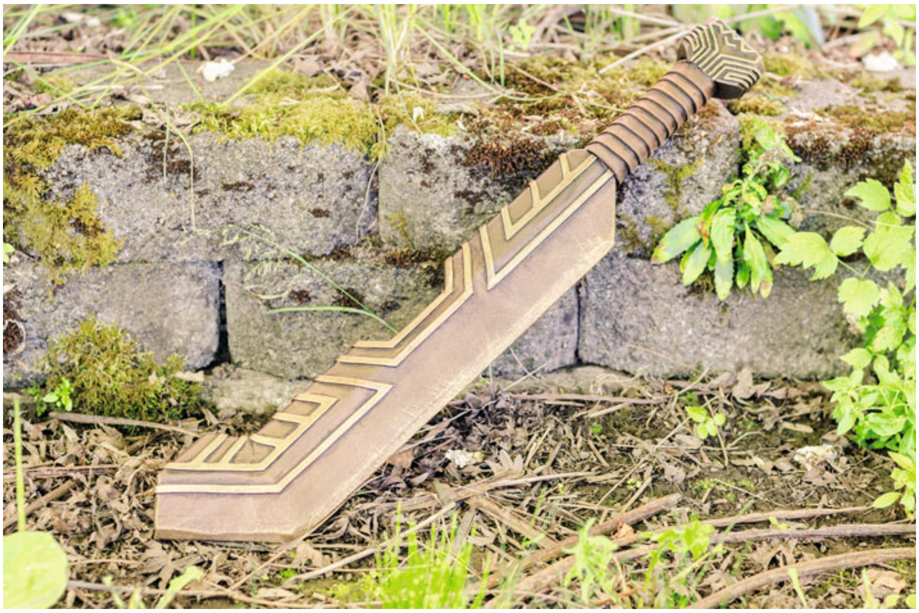


Figure 12-5. *A finished, painted EVA sword (photo courtesy of Bill Doran/Punished Props)*

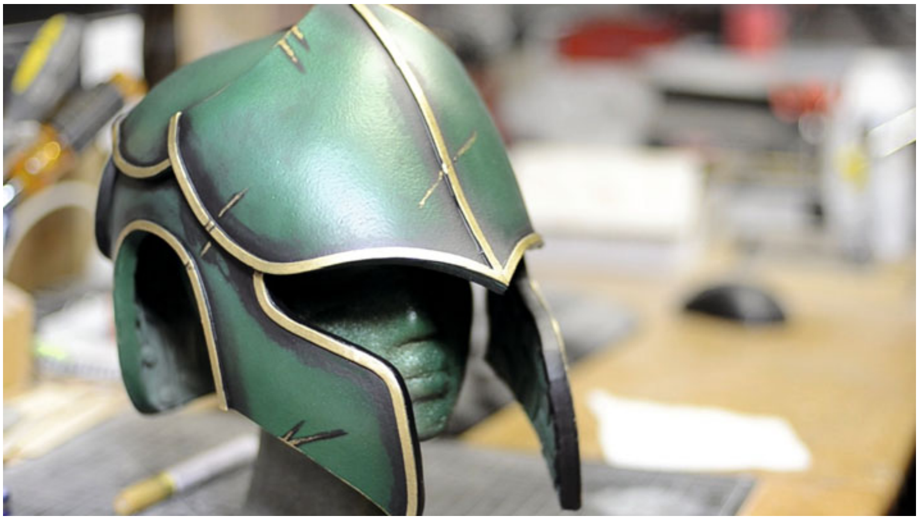


Figure 12-6. *A finished, painted EVA helmet (photo courtesy of Bill Doran/Punished Props)*

Vacuum Forming

Thermoforming is the process of heating thin sheets of plastic so that they can be stretched and formed to the shape of a mold. A common form of thermoforming is *vacuum forming*, in which the heated plastic is first stretched over a mold or form and then sealed down to a vacuum table with a frame so that the air can be sucked out, forcing the plastic against the surface before it re-hardens. This a common way to make things like masks, and is how almost all Star Wars stormtrooper cosplay armor is made.

Clear face or eye covers can be made as part of a larger costume by vacuum forming a clear plastic. Many hobbyists build their own vacuum forming tables by making a box with holes in the top and connecting a wet-dry vacuum to the side to suck the air out. More professional vacuum formers include a heating element for the plastic, but others may require you to heat the framed plastic in an oven before transferring it to the vacuum former.

Worbla

There are also plastics that are designed to be formed without a mold. Products like Worbla, Sintra, Wonderflex, and Terraflex can be made soft with a heat gun and then formed by hand into complex shapes. These can be used to make props and armor, or just to make detail pieces to be attached to a larger piece. There are several books (including *The Book of Cosplay Armor Making with Worbla and Wonderflex* by Svetlana Quindt) listed at `www.worbla.com/?cat=33`, and the author has a YouTube channel at `www.youtube.com/user/Mogrymillian`.

■ **Caution** As always, be sure to follow the manufacturer's directions on ventilation and other protective gear when you are working with the materials in this chapter.

Other Ideas

There are numberless craft materials and tools out there for experimentation. Here are a couple of other techniques and tools that might be valuable for you to know about:

- *Pepakura* is a technique for creating solid objects by making a cutout paper pattern that can be folded to create a surface. (This is a little different from *origami*, which creates more delicate creations starting from a square of paper.) Various software packages can do this, such as Autodesk 123D Make (`www.123dapp.com/make`). Items made this way are not very strong, but can be interesting shapes. A laser cutter is a good way to cut out a pepakura pattern.
- A *vinyl cutter* is what it sounds like: a computer-controlled knife that cuts out shapes from a thin vinyl roll. It is useful if you want to have a thin-surface, colored, cut-out design on something, like fancy lettering.

Painting

Once you have made something out of plastic or foam, it will probably still look like it's made of plastic or foam. Assuming that is not what you're going for, it is now time to paint. Making your creation look scratched and dirty (known as *weathering*) can make it look even more real.

Paints for these applications are usually sprayed on, either with an aerosol paint can (known as a *rattle can*) or with an airbrush, to create an even look without brush strokes. You will usually want to use a primer first and then apply the color of your base layer. Depending on the effect you want, you may lay down many layers of paint, and you can mask off different areas while spraying to get sharp boundaries between sprayed and un-sprayed areas.

Weathering can involve techniques like brushing on paint and then removing it, intentionally spilling coffee on your shiny new thing, and painting the edges with metallic paints so that it looks like paint has chipped off of those areas.

Once it is all done and dry, you'll probably want to put a clear coat over everything so that your fake damage does not fall prey to real damage. For more on painting and weathering, we recommend *Painting and Weathering for Props & Replicas* by Harrison Krix of Volpin Props, which you can find at `www.volpinprops.com/product/painting-and-weathering-for-props-and-replicas-ebook/`.

■ **Tip** To paint 3D-printed pieces (other than weathering them), acrylic paint works on PLA or ABS. Nylon can be dyed with appropriate (nylon) fabric dye.

Summary

This chapter covered other tools and techniques that are not talked about in depth in other chapters. First we reviewed cutting tools like laser cutters and CNC machines. Next, we explored structural techniques like making armor from foam, vacuum forming, forming with Worbla, and vacuum forming. Finally, we introduced pepakura and vinyl cutting, and wound things up with some thoughts abo ut painting your creations.

CHAPTER 13

A Look Ahead

In this book so far the three of us have presented projects that we feel are within reach of beginners, or for those who may be a bit more advanced in just one aspect of the triad of sewing, electronics, and programming. In this final chapter we wind up the book with some more aspirational projects, showing what professional designers and artists can do with some of these technologies. We also speculate a little on where the technology may go from here, featuring a few examples from the research lab.

3D-Printed High Fashion

Fashion items using 3D printing have become commonplace at the haute couture end of the spectrum. Industry website 3Ders.org recently compiled a list of their top 15 choices at `www.3ders.org/articles/20160225-3ders-top-15-list-of-our-favorite-3d-printed-dresses.html`. You can see the wide range of styles by clicking through their list. Or, if you search online for the phrase "3D printed dress," you will find things both beautiful and cringeworthy.

The Dita von Teese Dress

In 2013, one of the first examples of an entirely 3D-printed dress was designed by Michael Schmidt and Francis Bitoni for Dita von Teese (`www.michaelschmidtstudios.com/dita-von-teese.html`). Schmidt describes himself on his website as a "wardrobing and jewelry designer" with a roster of celebrity clients. The von Teese dress was created of tiny, articulated, interconnected nylon parts, sort of like chain mail, that were printed on nylon powder printers and then dyed black. There is a good "making of" video on the Shapeways website: `www.shapeways.com/blog/archives/1952-revealing-dita-von-teese-in-a-fully-articulated-3d-printed-gown.html`.

Nervous System

Jessica Rosenkrantz and Jesse Louis-Rosenberg cofounded Nervous System (`http://nervo.us`), a "generative studio," in 2007. Their vision is: "Drawing inspiration from natural phenomena, we write computer programs based on processes and patterns found in nature and use those programs to create unique and affordable art, jewelry, and housewares." The implementation of that vision is spectacular.

J. Horvath et al., *Practical Fashion Tech*, DOI 10.1007/978-1-4842-1662-0_13

Their Kinematics dresses are printed by a nylon powder printer, like the Von Teese dress. Their key innovation, though, is that Nervous System developed software that creates their designs custom for a particular wearer. Figure 13-1 shows their Kinematics Dress 2, worn by Rosenkrantz.

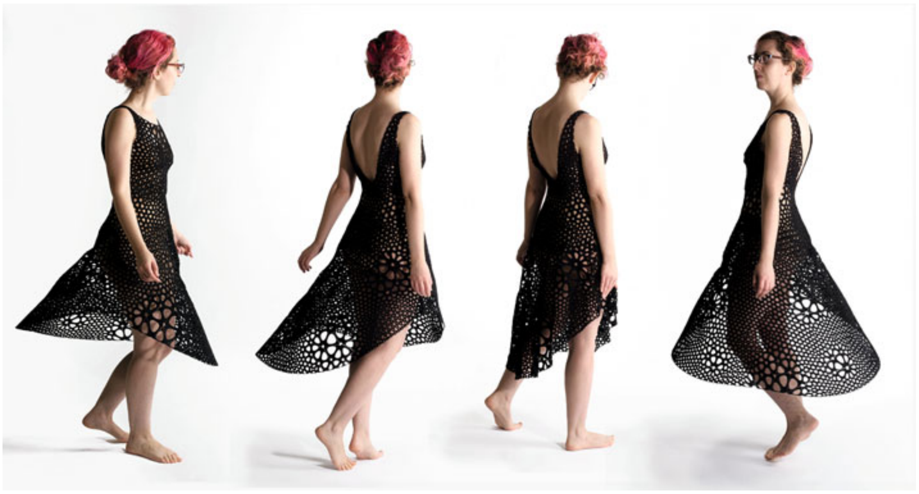


Figure 13-1. *The Kinematics Dress 2 worn by its designer. Image courtesy of Nervous System.*

The dresses are made from structures (like those in Figure 13-2) that can articulate freely. They come out of the 3D printer already assembled and (after blowing off the excess powder) ready to wear.

Another Nervous System innovation was the software used to create a 3D-printable model of the dress. This model is tightly folded and printed in one piece in a *selective laser sintering* (SLS, powder-based) 3D printer. Figure 13-3 shows what one of the Kinematics line of dresses looked like when pulled from the printer before the excess powder was removed. By the way, if you were wondering, Nervous System says that Kinematics pieces can be hand-washed using mild soap and water and a soft-bristled brush.

Pieces like this cannot be printed on filament-based 3D printers like those we talk about in Chapter 9; the shapes are so complex and delicate that the support needed for filament printing would be prohibitive to remove. SLS printers have the virtue that the powder acts as a support that can just be blown away at the end. The drawback is that the powder is very fine and hard to manage. SLS printers are still industrial machines at the moment, and the prints are expensive as of this writing, but who knows how technologies will evolve.

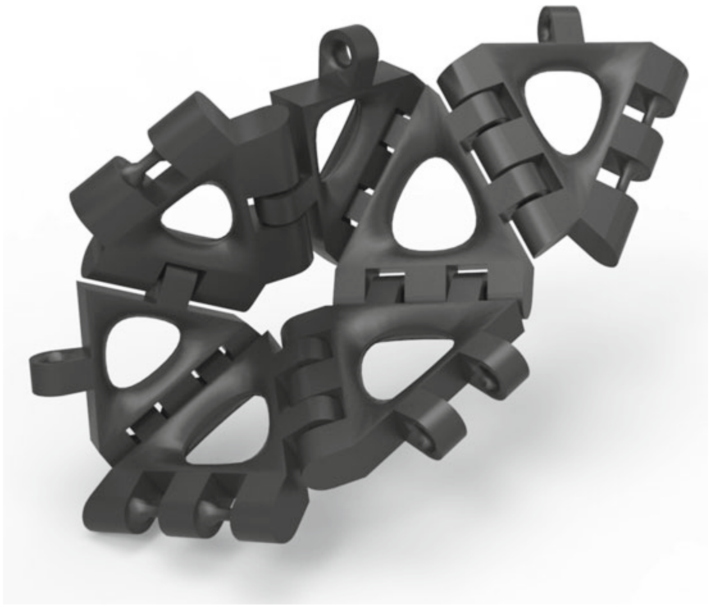


Figure 13-2. *Swatch of Kinematics structure showing hinge details. Image courtesy of Nervous System.*

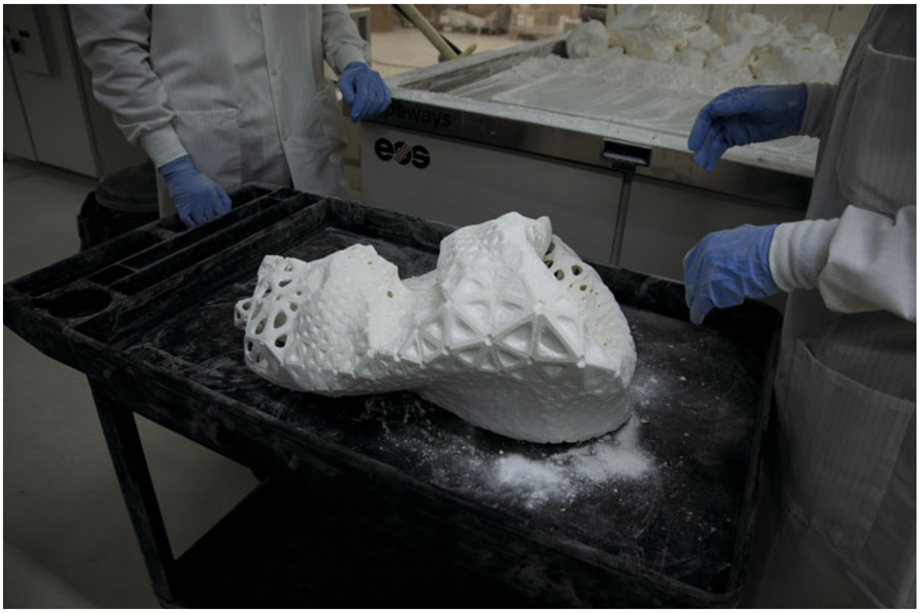


Figure 13-3. *The Kinematics Dress 1 before all the excess powder was removed. Image courtesy of Nervous System.*

The pair has continued to innovate, and in 2016 they created the Kinematic Petals Dress (Figure 13-4), commissioned by the Museum of Fine Arts in Boston. Of this dress they say, “Inspired by petals, feathers, and scales, we developed a new textile language for Kinematics where the interconnected elements are articulated as imbricating shells. Like our previous garments, this dress can be customized to the wearer’s body through a 3D scan, and additionally, each element is now individually customizable: varying in direction, length, and shape.” They note that they had to develop new software to allow for the overlapping petals when the design is folded for 3D printing. Spend some time on the Nervous System website if you want to be truly intimidated by what world-class designers can do with this technology.

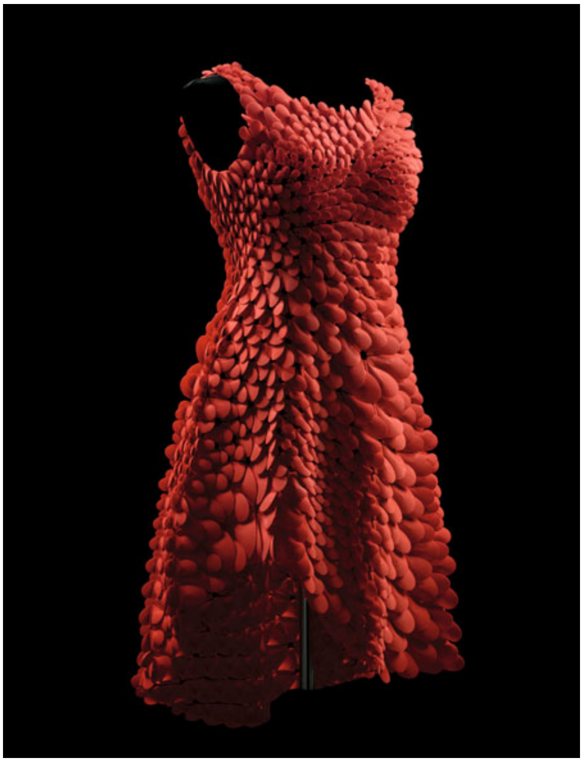


Figure 13-4. *The Kinematics Petals Dress. Image courtesy of Nervous System.*

Electronic Fashion

In another era, gowns were encrusted with jewels. Now LEDs, sensors, and mechanisms seem to be taking their place. Some of these creations have been developed as technology demonstrations, but others are becoming mainstream red-carpet attire.

Anouk Wipprecht

Dutch designer Anouk Wiprecht (http://anoukwipprecht.nl) is a pioneer in the use of 3D-printed parts and materials in her creations. Her recent Spider Dress 2.0 was partly a demonstration of the Intel Edison chip, which was used to control it. The dress involved sensors that could tell when others were too close, in which case robotic arms moved menacingly to fend off the interloper. Wipprecht's earlier work included the Faraday Dress, which allowed the wearer to interact with the high voltage of a Tesla coil onstage, and a variety of other creations that can be viewed on her website.

The 2016 Met Gala

The Metropolitan Museum of Art's 2016 Met Gala was themed to align with the Costume Institute's exhibit at the museum, *Manus x Machina* (www.metmuseum.org/exhibitions/listings/2016/manus-x-machina). This annual celebration was filled with dazzling gowns and suits showing what spectacular effects can be obtained by using large numbers of components in one dress. Claire Danes wore a stunningly beautiful light blue gown made from organza and fiber optics that lit up in the dark. It was lovely in the light and a magical delight in the dark. It was a one-of-a-kind, hand sewn confection designed by Zac Posen. Lyn would love to know how many people and hours it took to create!

Another piece in this exhibition was created by Marchesa designers Georgina Chapman and Keren Craig in collaboration with the IBM Watson supercomputer. It was worn by model Karolina Kurkova and was meant to be a "compassionate dress." Its 150 LED lights would change color depending on fan input from Twitter. We suggest you spend some time perusing the Met's website noted in the previous paragraph to get your own inspiration.

Textile Technologies

For truly radical change, innovation may need to move down to the textile level in the form of *active fabrics*. That term can mean many different things though. One could embed tiny *actuators* that have the ability to make something move within the fabric, like microscopic versions of the servos we talk about in Chapter 8. Or perhaps an active fabric will just have circuits woven into it from the beginning. Here are a few examples of these farther-out innovations.

bioLogic

Suppose one could have *living* fabric? MIT's Tangible Media Lab's bioLogic team (http://tangible.media.mit.edu/project/biologic/) under the direction of Professor Hiroshi Ishii, has discovered that the bacterium *Bacillus subtilis natto* expand and contract relative to atmospheric moisture. This bacterium has been used for a thousand years in Japan to make the soy dish *natto*, so it is not an exotic new organism.

The team is using this behavior to create fabric that responds to moisture. Living cells are harvested and then bio-printed into a "synthetic bio-skin." Fabric that has been designed with this bio-skin (Figure 13-5) reacts to heat and moisture by opening cooling vents. So if someone is wearing performance clothing made of this material, it automatically opens up the vents when needed. The website says the team is collaborating with sports clothing maker New Balance.

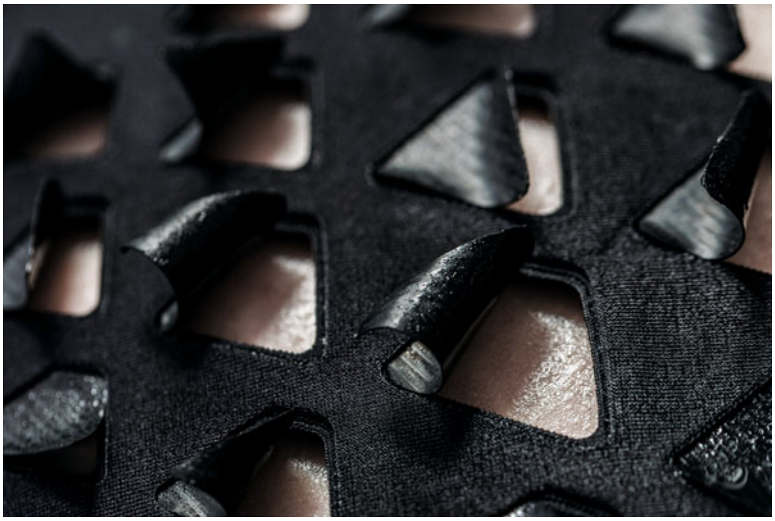


Figure 13-5. *The bioLogic fabric. Photo courtesy of Tangible Media Group, MIT Media Lab.*

The group has also created more whimsical applications, like fabric flowers that are folded up and pale when dry (Figure 13-6) and change both color and shape when sprayed with water (Figure 13-7).

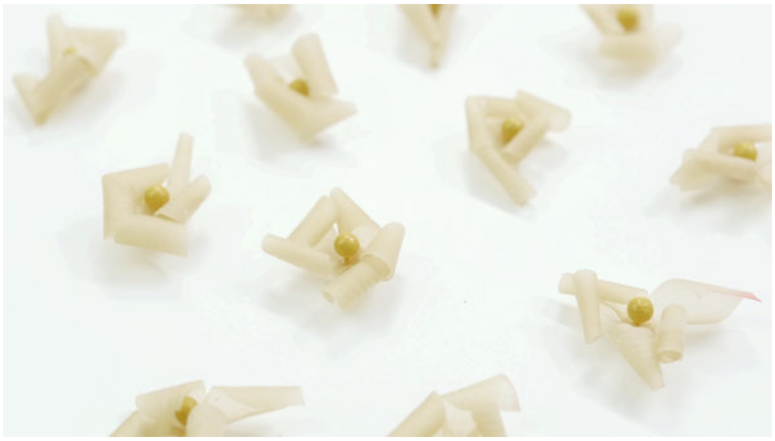


Figure 13-6. *Bio-hybrid flower, dry. Photo courtesy Tangible Media Group, MIT Media Lab.*

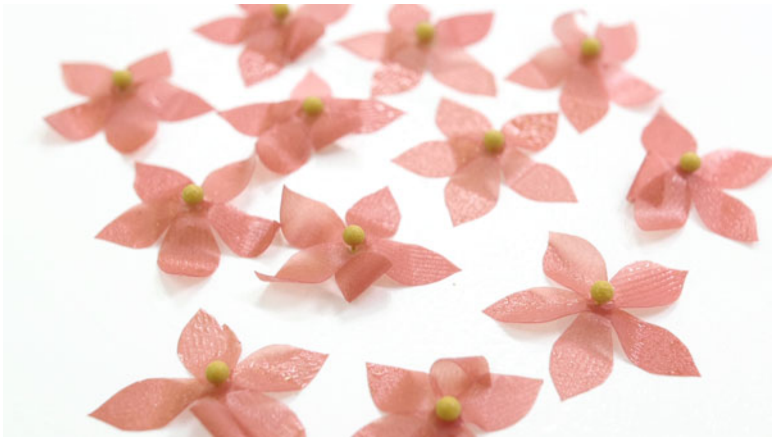


Figure 13-7. *After they are sprayed with water, the flowers in Figure 13-6 change both shape and color. Photo courtesy Tangible Media Group, MIT Media Lab.*

Project Jacquard

Google's Project Jacquard (`https://atap.google.com/jacquard/`) is turning fabric and apparel into touchscreens. Headed by Ivan Poupyrev, the project is creating a new kind of braided, conductive thread. It comes in many colors and can be used in existing industrial looms and sewing machines to mass-produce clothing. The conductive thread is connected to a small Bluetooth controller, which uses a standard watch battery and can be kept in a pocket. This gives the fabric or garment the ability to work with all kinds of gadgets, including touchscreens, smartphones, other media devices, and house lights or thermostats. Touch sensor grids can be woven directly into clothing, so that a user can swipe a sleeve to send a signal, and LEDs and be controlled to get the wearer's attention.

Clothing Meets the Internet of Things

The idea of using technology in clothing and even embedded in the human body has been around for a long time in science fiction. Neal Stephenson's book *The Diamond Age:or, A Young Lady's Illustrated Primer* (Bantam Spectra, 1995) featured "matter compilers" to make clothes (or anything else). There are many more examples in books by Arthur C. Clarke, Frank Herbert, Ray Bradbury, and many others.

In the movie *Back to the Future Part II,* Marty McFly has self-tying shoes and a self-drying jacket. Nike finally caught up to the sci-fi genre this year with its HyperAdapt self-tying shoes (`http://news.nike.com/news/hyperadapt-adaptive-lacing`).

Things can get even more complicated if the automated clothing is controlled (or sensed) remotely. There are many definitions of the so-called Internet of Things (IoT). We will say here that the IoT involves sensors or actuators on something in the world (like a shoe, dress, refrigerator, or thermostat) that is then connected to the Internet to either take action or report data at a distance.

The future of wearables and the IoT is not just in the areas of fitness, entertainment, and fashion, but also in the health and medical fields. Innovations in communication and interface development are improving patient's abilities to monitor health conditions. Fitness trackers are the early examples of this, but more advanced devices are starting to be approved for medical use, like continuous glucose monitors that share a diabetic's status wirelessly with the wearer and, if desired, caregivers (`www.dexcom.com/products`). Google has developed contact lenses with embedded chips to sense glucose levels.

Devices that monitor and communicate continuously raise some privacy issues, but also may lead to safer, more independent living for those with certain medical conditions. Wearable sensors for the independent elderly seems to us to be an area of particular promise.

A Few Last Words

As adoption of new technologies becomes faster and faster, one of the biggest challenges in all areas is making and keeping the technology useful, efficient, and desirable. As electronic components become smaller and smarter, it is easier to use them in everyday garments and accessories. It is important, though, to keep the person wearing your garment or accessory in mind.

One of the things we have tried to emphasize in this book is to think through ahead of time how the garment will be used and whether the technology you are thinking about will work in the environment where it will be used.

We started this book with some philosophy about wearable tech and what makes a good costume. Then we moved on to teaching basic skills in the areas you need for wearable tech, concluding in our apron project. Next we took you through sensors, 3D printing, and a cautionary tale about what could go wrong if your first project was overly ambitious. In the latter chapters we laid out more ambitious projects, discussed other technologies you can use, and offered a glimpse at where researchers may lead us next.

Even after all that, though, some basic principles from the first chapters still apply. Whether you are making a stuffed animal that lights up to make a friend laugh or a complex art piece, think about the following points as you lay out your design:

- Is it as simple as it can be? There is a temptation to make things more complicated than they need to be.
- Walk through how someone will move in the garment in detail and think about what might catch, break, wear out, or short. The projects in this book were designed for indoor use; consider the overall environment and how you can remove electronics and batteries for weather and laundry.
- Do you know how to finish your project, or are there things you need to learn? If there are things in your project that are just an abyss to you now, you may want to learn some of those things first and then do the familiar things. Often people do not finish something because they do the part they know about first and discover they should have done that part differently to accommodate the new component. Then the whole thing lies in a corner for years.
- Try to work in teams. It is hard to be awesome at sewing, electronics, and coding. Some people are, but most people need a few helpers to get by. If you are in a city with a makerspace doing this sort of stuff, you might consider joining it. The three of us were able to create projects that none of us could have considered alone.
- Share your ideas! Most of the technologies we talk about in this book are primarily based on the work of open source communities. We link to those communities in the book where we encounter them. The best way to thank people who have done all that work in the past is to build on it and share when you do.

We hope you have enjoyed this book and feel that you have learned a lot. Now go out there and make something cool—and then teach someone else how to make it, too.

Summary

In this chapter we looked at professional fashion applications of technologies discussed elsewhere in this book. We also introduced new textile-level technologies that could make embedding actuators or sensors literally seamless. Finally, we ended with a few ideas about good design practices, looking back at the path the book has taken.

APPENDIX A

Teaching Fashion Tech

For those who want to teach a fashion tech class that covers much of the material in this book, this appendix lays out some of the things you might want to think about. We also have laid out some sample course details based on the projects in this book.

The material in this appendix is based on lessons learned by Lyn over many years of teaching a variety of classes at the high school level, and on Joan's experience teaching undergrads and grad students. We are also drawing on anecdotal evidence the three authors have of the results of various colleagues' experiments, as well as on a pilot version of a fashion tech costuming class that Lyn taught with Joan and Rich as consultants.

Course Objectives and Grading

A year-long class at the high school level or a two-semester college elective might have a course description, learning objectives, and grading criteria something like that covered in this section. A good way to structure a class like this is to teach through progressively more difficult projects. Lecture in isolation does not work all that well. Teaching project-based curricula also allows students to move at their own pace and enables more-experienced students to do harder projects (or coach less-experienced students).

Course Description

Participants create a series of projects, theoretical and practical, to help understand the processes involved in designing and creating costumes. Students will develop and improve skills in conceptualizing and rendering that will facilitate ways of thinking about costuming and how it relates to a dramatic production. They will also be required to learn the basics of circuits and coding, including implementing at least one fashion tech project. Each project will pass through several iterations and presentations for discussion and critique by faculty and peers, perhaps supplemented by external reviewers.

Learning Objectives

In a fashion tech class, students will accomplish some or all of the following learning objectives. For a college class, one might go heavier on the principles, and in a high school class, lean more on the hands-on assembly.

- Learn the fundamentals of hand and machine sewing.
- Understand elements of circuit design and programming for creating wearables using Arduino-compatible electronics.
- Learn the process of designing and 3D printing custom objects.
- Appreciate the process of costume design and its historical context.
- Learn what techniques are suitable for the beginning designer/maker and how to acquire skills not necessarily explicitly taught in the class.
- Learn how to approach and solve design challenges and problems, and how to interpret and apply the principles and elements of good design and construction to creating costumes and other projects.

Grading Criteria

It is always a challenge to grade how well students have met learning objectives like the ones just mentioned. Quizzes or skill demonstration could work for the skill sections, but by and large projects seem to us to be a better way of seeing how well skills have been learned. In addition to regular, formal evaluation, a lot of learning happens during critique and sharing of constructive criticism on each iteration of a project, by both the faculty member and the students.

With subjective course objectives like these, it is challenging to come up with any one-size-fits-all rubrics other than the general criteria noted. A faculty member can also evaluate a student's effort based on class participation, attendance, and the level of craftsmanship and improvement exhibited in a student's completed projects.

■ **Tip** "Success" may not mean a perfect project. Students start at different places, so they need to feel successful if they have grown over the year, without being compared to other students who are at a more advanced level.

Logistics Issues

Project-based classes—particularly ones that might span multiple departments—often pose logistical challenges. In this section we suggest some things to think about when you are planning a fashion tech or similar class.

Scheduling

The class schedule has to be somewhat flexible, since the same project may take different students very different amounts of time to complete. One student may start a project in week 3 and finish it in week 7, whereas another may finish in week 5 and start something else that will finish in week 10. A faculty member will need to be comfortable with students learning both from her and from each other, lecturing infrequently but more often introducing material one-on-one as a project needs it.

The class can also be built around just-in-time learning—the first student who needs to learn something for his or her project later passes it on to others, or others stop working and watch when someone else learns something. This style works well in a small pilot class, but scaling it up may require some thought. Thus, rather than a week-by-week specific schedule, we describe a suggested mix of projects. A large class might require more structure.

This class is difficult to run in hour-long chunks, since it takes a lot of that time just to pull out the materials of the day and then put them away. If possible we suggest scheduling it as a back-to-back double period or a scheduled class with required after-school/Saturday time intensively for a few weeks of the year (for example, to support building costumes for the school theater productions).

Sharing Materials

For a large class, many people will need to share resources like sewing machines, cutting tables, soldering stations, and so on. Phasing projects so that people are working on different things helps with this sharing. Think through how you will use your room when people start to get out of phase or (perhaps more problematic) if they all get to the same point at the same time.

Also think about where to store works-in-progress. If more than one section is sharing materials, it may be frustrating or impractical to tear down and dismantle circuits at the end of each class, so you may need a separate set of materials for each class.

Between classes, though, that material needs to live somewhere, and in our experience student projects always seem to expand to fill all the available space plus an additional 10% that you wish you had. Also, consider how to display (and how to select for display) exemplary projects.

Classroom Environment

Providing facilities for a fashion tech class can be a challenge. Lab benches with stools that work well for electronics might not work well for sewing, since sewing machines have foot pedals that might not be reachable from a lab stool. If the class will be in an electronics room or a sewing room, actually try out the activity that does not normally happen in that room. For example, try sewing in the electronics room before the students arrive.

It is difficult to show how to sew or assemble electronics so that students in the whole room can actually see what you are doing. Consider emailing students photos ahead of time or otherwise share them electronically. We have experimented with a webcam and document cam, but one's fingers tend to get in the way.

■ **Tip** Be sure you have good task lighting for both electronics and sewing.

Budget

Materials for this class can get pricey, depending on whether you have every student build a project or have them work in teams, and whether students can take their projects home or have to undo their sewn circuits to recover the electronics. Your capital equipment budget (for sewing machines and 3D printers) might be outstripped by the cumulative cost of small electronics and fabric, notions, training, and machine maintenance.

Course Resources

Many students have trouble with open-ended design and invention. The resources offered in this section may help. Define a project start point that is neither too broad nor too narrow. Suggesting that students create a simple prop or costume piece for an upcoming show, for instance, might be focused enough to get students started.

Teaching others to design requires a delicate balance between direction and encouragement of self-expression. There are many good books on this topic from the design education literature, but most are aimed at undergraduates for in-class use. For college students, or for a high school instructor's background and preparation, we suggest:

- Donald Norman's classic *The Design of Everyday Things* (Basic Books, 2002)
- Henry Petroski's books, notably *To Engineer is Human* (Vintage Books, 1992)
- Peter Forbes's *The Gecko's Foot* (Norton, 2005) for biomimetic ideas (a good source of possible projects)
- Horvath and Cameron's *The New Shop Class* (Apress, 2015)

Learning to sew and create/use patterns (theater oriented):

- Barbara and Cletus Anderson's book *Costume Design*, 2nd Edition (Harcourt Brace College Publishers, 1999)
- Shirley Dearing's *Elegant Frugal Costumes* (Meriwether Publishing, Ltd., 1992)
- Sheila Jackson's *Costumes for the Stage*, 2nd Edition (New Amsterdam Books, 2001)
- Nancy Bradfield's *Costume in Detail 1730–1930* (Costume and Fashion Press, an imprint of Quite Specific Media Group, Ltd., 1997)
- `http://sewing.craftgossip.com` and `www.wikihow.com/sew-using-patterns`.

Electronics background:

- Becky Stern and Tyler Cooper, *Getting Started with Adafruit Flora* (Maker Media, 2015)
- Kate Hartman, *Wearable Electronics* (Maker Media, 2014)
- Tutorials at `https://learn.adafruit.com`, `https://learn.sparkfun.com/tutorials`, and `www.instructables.com`.

Course Content Outline

This section suggests a list of projects to be completed by students in a one-year class, tied to items in this book. This class takes significant prep time; walk through exactly what the students are building just before they do, using the equipment they will use. It is easy to lose time to software not being available, or to a small component being in short supply, and so on.

This outline assumes that the students know nothing about sewing, coding, and electronics. Particularly in a larger class, this may not be the case, and students with more background can be recruited to help less-experienced ones. Students with some background may skip some of the suggested projects and either tutor their fellow students during that time or move on to a more advanced project.

■ **Tip** Time allowed will vary a lot based on student experience and class size. Obviously the major projects could be quarter or semester projects instead. Allow time to fix problems and deal with the unexpected, particularly the first time you teach it.

Projects

The first project should be a simple one that does not require any significant amount of fitting. It could either be the apron in Chapter 7 or just a small stuffed animal with a light-up eye. Or you can start with an existing garment (like a hat) and add a NeoPixel circuit to it. Either way, the pattern development should be minimal.

The project in the second major time period (weeks 10–15) can be a garment that requires significant fitting and adapting of a pattern, plus more complex assembly. The vest from Chapter 4 can be a simple version of this, or you can embellish it.

After those two projects are under the students' belts, the remaining time (or another semester, if you have 13 or 15 week periods to work with) can be used to incorporate 3D printing (Chapter 9), other techniques, if the school has the tools and facilities (Chapter 12), or a project incorporating sensors (Chapters 8 and 11).

If you want to get really fancy, you can think about some biomimetic projects—creating fabric and electronics projects that mimic nature in some way, or an art piece with sensors that react to touch (something a step up from the apron's saucepan in Chapter 7).

Schedule

A notional schedule for a year-long (or two-semester) course using this book as a text might be something like this:

- *Weeks 1–9*: Simple project with a light-up component. This project should not require students to take extensive measurements. One choice might be the Chapter 7's apron project. Faculty should read Chapter 10 for tips on what not to do and create a few demonstration projects to show students what is coming.
    - *Weeks 1–2*: Review good design practices. Design, sketch, create pattern. Read Chapters 1–4 and 7.
    - *Week 3*: Select fabric and cut out pattern.
    - *Weeks 4-5*: Machine sew garment seams, hand sew details.
    - *Week 6*: Introduce Fritzing, lay out circuit with alligator clips. Read Chapter 5.
    - *Weeks 7–8*: Sew on circuit and run code. Read Chapter 6.
    - *Week 9*: Finish up and document the project.

- *Weeks 10–15*: A next project could involve a garment that is simple but still requires some amount of fitting for the wearer. An example might be the vest in Chapter 4. Historical aspects might be incorporated here.
    - *Week 10*: Select store-bought pattern for the garment (or create one), add design components, fit to wearer.
    - *Week 11–12*: Select fabric, cut out pattern.
    - *Week 13–14*: Machine sew and hand finish, and document the project.
- *Weeks 16 through end of year*
    - More-complex projects, such as costumes for school production or other projects. Read Chapters 9–13.
    - Research historical background of social influences on costuming, or more depth in the area of principles of fashion design.
    - Group critiques.
    - Field trips, outside speakers.

The first nine weeks have a bit of everything, so if you wanted to instead run a "sampler" one-quarter or one-semester class, you could sensibly stop there. The second six-week segment looks in more depth at how to make a garment that requires some fitting that could incorporate some aspects of proper fit and other clothing design issues. Those first 15 weeks could be a semester class. Any remaining time would be dedicated to student projects.

APPENDIX B

Links

This appendix aggregates all the links in the book in one place for convenient reference. If a link appears in more than one chapter, it is listed here under the chapter in which it first appears.

About the Authors

Nonscriptum LLC: www.nonscriptum.com

Chapter 1. Fashion Tech

Stereoscopic animation of the Amoskeag (New Hampshire) Gingham Mill weaving room: http://stereo.nypl.org/view/14480

History of textile technologies: www.cs.arizona.edu/patterns/weaving/index.html

Jacquard Machine Analyzed and Explained (Posselt, 1893): http://hdl.handle.net/2027/gri.ark:/13960/t26b0d33d

stereoscopic image of the Amoskeag (New Hampshire) Gingham Mill weaving room: http://digitalcollections.nypl.org/items/510d47e1-7c18-a3d9-e040-e00a18064a99

Babbage's Analytical Engine: https://en.wikipedia.org/wiki/Analytical_Engine

Renaissance Faires: www.renfaire.com

Chapter 2. Practical Costume Design

Make Believe costume shop: www.makebelieveinccostumes.com

Burbank Makerspace: www.burbankmakerspace.com

Chapter 3. How to Sew

History of sewing: https://en.wikipedia.org/wiki/Sewing

How to sew: www.instructables.com/id/How-to-Sew./ and www.wikihow.com/sew

Hand sewing basics: www.youtube.com/watch?v=B2mfJweh8aO

Hand Sewing: Basic Slip Stitch (Blind Stitch) on Instructables: www.instructables.com/id/Hand-Sewing-Basic-Slip-Stitch-Blind-Stitch/

Chapter 4. Making and Using Sewing Patterns

Sewing pattern companies: www.mccall.com and www.sewingpatterns.com

Crafts: www.craftsy.com

Online craft marketplace: www.etsy.com

Joann Fabrics: www.joann.com

How-to-sew sites: http://sewing.craftgossip.com and www.wikihow.com/sew-using-patterns

Adobe Illustrator: www.adobe.com/products/illustrator.html

Chapter 5. Wearable Tech Electronics

Arduino: http://arduino.cc

Instructables: www.instructables.com

Adafruit resources and tutorials: http://learn.adafruit.com

Sparkfun resources and tutorials: http://learn.sparkfun.com

Sparkfun electronics: www.sparkfun.com

Adafruit: www.adafruit.com

Resistor color code: https://en.wikipedia.org/wiki/Electronic_color_code

LED colors and materials: https://en.wikipedia.org/wiki/Light-emitting_diode#Colors_and_materials

Fritzing: www.fritzing.org

Getting started with Fritzing: http://fritzing.org/learning/

Adafruit Fritzing library: https://learn.adafruit.com/using-the-adafruit-library-with-fritzing/download-the-fritzing-library-from-github

Sparkfun Fritzing library: https://learn.sparkfun.com/tutorials/make-your-own-fritzing-parts

Chapter 6. Programming Wearables

Programming books at Apress: www.apress.com/programming/c-c?p=1

Processing development environment: www.processing.org

Arduino language reference: www.arduino.cc/en/Reference/HomePage

Download the Arduino IDE: www.arduino.cc/en/Main/Software

Arduino libraries: www.arduino.cc/en/Guide/Libraries

Chapter 7. Your First Project

How to use Neopixels: https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library

Chapter 8. Sensors and Other Hardware

Soldering tutorial: https://learn.sparkfun.com/tutorials/how-to-solder---through-hole-soldering

Thermistors: http://playground.arduino.cc/ComponentLib/Thermistor2

Thermistor theory: https://en.wikipedia.org/wiki/Thermistor#B_or_.CE.B2_parameter_equation

Arduino hardware libraries: http://playground.arduino.cc/Main/interfacingWithHardware

Arduino mechanical libraries: http://playground.arduino.cc/Main/InterfacingWithHardware#Physical_Mechanical

Chapter 9. 3D Printing

Things the same width as a 3D printer layer: https://en.wikipedia.org/wiki/100_micrometres

The REPlicating RAPid prototypes project: www.reprap.org

Thingiverse 3D model repository: www.thingiverse.com

Youmagine 3D model repository: www.youmagine.com

Autodesk 123D software suite: www.123dapp.com

TinkerCAD 3D design software: www.tinkercad.com

Star and Saturn pendant on Tinkercad: https://tinkercad.com/things/kQD9mpZ1BLf

OpenSCAD 3D design software: www.openscad.org

OpenSCAD documentation: www.openscad.org/documentation.html

Blender 3D design software: www.blender.org

Maya 3D design software: www.autodesk.com/products/maya

Onshape 3D design software: www.onshape.com

MatterControl 3D printing software: www.mattercontrol.com

Aleene's textile glues: www.ilovetocreate.com

3D printing on fabric: http://makezine.com/projects/how-to-3d-print-on-tulle-net-or-lace-fabrics/

3D printing shoes: https://3dprint.com/1331/nike-awarded-two-major-3d-printing-footwear-patents/

Chapter 10. The Importance of Planning

Lisia Trubat's E-traces project: http://cargocollective.com/lesiatrubat/E-TRACES-memories-of-dance

Fiber Optic Jellyfish Skirt by Linawassong: www.instructables.com/id/Jellyfish-Skirt/

Chapter 11. Two Bigger Projects

LSM9DS0 sensor: https://learn.adafruit.com/adafruit-lsm9ds0-accelerometer-gyro-magnetometer-9-dof-breakouts/pinouts

Electroluminescent wire: https://en.wikipedia.org/wiki/Electroluminescent_wire

Mod 60s Mondrian Dress Pattern:
https://betsyvintage.com/index.php?main_page=product_info&products_id=120

Chapter 12. Other Technologies

Inkscape: https://inkscape.org/en/

Punished Props: www.punishedprops.com

Punished Props YouTube channel: www.youtube.com/user/punishedprops

Worbla books: www.worbla.com/?cat=33

Svetlana Quindt's cosplay YouTube channel: www.youtube.com/user/Mogrymillian

Autodesk 123D Make: www.123dapp.com/make

Painting and Weathering for Props and Replicas:
www.volpinprops.com/product/painting-and-weathering-for-props-and-replicas-ebook/

Chapter 13. A Look Ahead

3ders.org's top 15 3D-printed dresses: www.3ders.org/articles/20160225-3ders-top-15-list-of-our-favorite-3d-printed-dresses.html

Dita von Teese's 3D printed dress: www.michaelschmidtstudios.com/dita-von-teese.html

Making of the Dita von Teese dress: www.shapeways.com/blog/archives/1952-revealing-dita-von-teese-in-a-fully-articulated-3d-printed-gown.html

Nervous System: http://nervo.us

Anouk Wiprecht: http://anoukwipprecht.nl

Manus x Machina exhibit: www.metmuseum.org/exhibitions/listings/2016/manus-x-machina

MIT's Tangible Media Lab's bioLogic team: http://tangible.media.mit.edu/project/biologic/

Google's Project Jacquard: https://atap.google.com/jacquard/

HyperAdapt self-tying shoes: http://news.nike.com/news/hyperadapt-adaptive-lacing

Continuous glucose monitors: www.dexcom.com/products

Index