Arduino Wearable Projects

This book provides you with the skills and understanding to create your own wearable projects. You will look at prototyping boards that are compatible with Arduino and are suitable for creating wearables.

You begin your journey by understanding electronic components, including LEDs and sensors, to get yourself up to scratch and comfortable with different components. You will then gain hands-on experience by creating your very first wearable project, a pair of interactive bike gloves that help you cycle at night. This is followed by a project in which you make your own funny LED glasses and a cool GPS watch. You’ll also delve into other projects including creating your own keyless door lock, wearable NFC tags, a fitness-tracking device, and a Wi-Fi-enabled spark board. The final project is a compilation of the previous concepts where you make your own smart watch with fitness tracking, Internet-based notifications, GPS, and, of course, the ability to tell the time.

Who this book is written for

This book is intended for readers who are familiar with the Arduino platform and want to learn more about creating wearable projects. No previous experience in wearables is expected, although a basic knowledge of Arduino will help.

What you will learn from this book

- Develop a basic understanding of wearable computing
- Learn about Arduino and its compatible prototyping platforms suitable for creating wearables
- Understand the design process surrounding the creation of wearable objects
- Gain insight into the materials suitable for developing wearable projects
- Design and create projects including interactive bike gloves, a GPS locator watch, and more by using various kinds of electronic components
- Discover programming for interactivity
- Learn how to connect and interface wearables with Bluetooth and Wi-Fi
- Get your hands dirty with your own personalized designs

Tony Olsson

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 4 'LED Glasses'
- A synopsis of the book’s content
- More information on Arduino Wearable Projects
**Tony Olsson** works as a lecturer at the University of Malmö, where he teaches multiple design fields with the core being physical prototyping and wearable computing. His research includes haptic interactions and telehaptic communication. Olsson has a background in philosophy and traditional arts, but later shifted his focus to interaction design and computer science. He is also involved in running the IOIO laboratory at Malmö University.

Besides his work at the university, he also works as a freelance artist/designer and author. Prior to this publication, Olsson published two books based on wearable computing and prototyping with Arduino and Arduino-based platforms.
Almost 10 years have passed since I picked up my first Arduino board. At the time, I was an interaction design student at Malmö University. At the front of the classroom that day, there was a bearded Spaniard talking, rather claiming, that he could teach us all about electronics and how to do programming for microprocessors, all in 1 week. Of course, since I knew nothing about electronics and never thought I would learn anything about it, I did not believe him.

The Spaniard had a completely new approach to teaching, which I had never encountered before. He wanted to teach us, not by books and lectures, but by doing things. One of my classmates pointed out that most of us did not know anything about electronics, so how are we supposed to do anything with it? The Spaniard replied that it does not matter, you can do things without knowing what you are doing, and by doing them, you will learn.

After 15 minutes, we all had connected a small lamp to our Arduino boards, and we had managed to program the lamp so that it would turn itself on and off. What baffled me was not only what we had achieved in such little time, but also that parts of what was going on actually made sense. We were learning by doing.

The bearded Spaniard was actually David Cuartielles, who together with Massimo Banzi, just 1 year before, invented the Arduino board. Soon after they invented it, Tom Igoe and David Mellis joined the team, and as they say, the rest is history. But I still remember that day, as if it was yesterday, when I looked down at my blinking light and something sparked inside me. I wanted to learn and do more. Then David gave me the second valuable lesson, that the best way to learn more is to share your knowledge with others, and he put me in a position where I was able to do so. Again I was skeptical, since I had no knowledge to speak of, but again the lesson followed, even if you only know a little, it is enough to help those that know nothing yet.
Soon after, I found out about a field called wearable computing. The idea was to design and apply a technology to the human body in different ways, and it all sounded as wonderfully crazy as the idea that you could learn electronics and programming without any prior knowledge of how to do so. With inspiration from Arduino and its team members, I leaped headfirst into the field. In this new field, I found new inspiration in the works of Steve Mann and Leah Buechley. Mann, now a professor at the University of Toronto, developed his own wearable computer in the 80s and had mostly done so on his own. Buechley, also a professor at MIT, had taken the Arduino board and developed a new prototyping platform, which is specialized for a wearable context. Both seemed to have done this against all the odds. Again, I was inspired, and started to develop my own wearable devices, teaching others how to do the same. Eventually, I collected enough know-how on things that I started to write them down. When I started to share my writing, I found out how truly amazing the Arduino community is a world-wide group of people that share a love for making things with electronics.

It's safe to say that if it had not been for all these people, I probably would never have written any of my books, so I would like to extend my thanks to all. I would also like to thank you for picking up this book. You might be a novice or an expert, but I do hope it will not matter. This book is based on the idea that anyone can learn anything by the simple principle of actually "doing." If you are already an expert, then you know there is always something to learn from "doing" things in a new way.

So, I hope you will gain some new knowledge and inspiration from the projects we created in this book, and I wish you all the best in your creating endeavors.

Do check out "Soldering with David Cuartielles" on my YouTube channel at https://www.youtube.com/watch?v=Mg01HFjsn6k.

**What this book covers**

*Chapter 1, First Look and Blinking Lights*, covers the basic steps of installing the development environment and how to get started with coding. We also take a look at how to create our first circuit and control an LED.

*Chapter 2, Working with Sensors*, teaches about interfacing with sensors and extracting data from them. The chapter also introduces digital and analog sensors ranging from simple to complex sensors.

*Chapter 3, Bike Gloves*, introduces the reader to the first project of the book, where the goal is to create a pair of bike gloves. In this chapter, we introduce the use of LEDs and how to control them, as well as how to use sensors for some simple gesture recognition.
Chapter 4, *LED Glasses*, teaches you to create a pair of programmable LED glasses. These glasses will be covered by LEDs in the front, which will be programmable to display different patterns and shapes. The reader will also be introduced to the construction of a pair of sunglasses.

Chapter 5, *Where in the World Am I?*, focuses on the making of a wrist-worn GPS tracking device. The information will be displayed on a small LCD screen. This chapter also includes instructions and tips on how to create a casing containing the components so that the device can be worn on the wrist.

Chapter 6, *Hands-on with NFC*, deals with NFC technology and servomotors and how they can be combined into a smart door lock. This chapter also includes how to design around NFC tags and make wearable jewelry that will work as a key for the lock.

Chapter 7, *Hands-on BLE*, deals with low-powered Bluetooth technology and how it can be implemented into wearable projects. This chapter introduces the Blend Micro board and how it can be used to create projects that connect to your mobile phone.

Chapter 8, *On the Wi-fly*, introduces you to the Wi-Fi Particle Core board and its web IDE. This chapter also talks about how to connect to online services.

Chapter 9, *Time to Get Smart*, focuses on the creation of a smart watch, which connects to the Internet and uses online services to create custom notifications to be displayed on a small OLED screen.

The online chapter (Chapter 10), *Interactive Name Tag*, expands upon *Chapter 7, Hands-on BLE*, which deals with small screens, and shows you how to interact with them over Bluetooth in order to make an interactive name tag. This chapter is available at [https://www.packtpub.com/sites/default/files/downloads/ArduinoWearableProjects_OnlineChapter.pdf](https://www.packtpub.com/sites/default/files/downloads/ArduinoWearableProjects_OnlineChapter.pdf).
In this chapter, we will look at how to create a pair of LED glasses. In essence, these glasses are an LED matrix. A matrix is an arrangement of LEDs in columns and rows, where we take advantage of the polarity of the LEDs so we can control 30 LEDs separately using only 15 pins on the Arduino board. The LED matrix has been around for many years and is still used today in some screen technologies. For example, digital bus signs are usually made by implementing an LED matrix, where each individual LED acts like a pixel.

In this chapter you will also learn a little bit about how to create your own glasses and soldering techniques. We will also have a look at some more advanced programming, where we will be implementing some simple animation for you to build upon.

The materials needed are:

- 30 LEDs (5 mm) in any color
- A trinket board or an Arduino micro board
- An FTDI to USB converter
- Wires
- A soldering iron
- A 3-5V lithium battery (the smaller the better)
- A JST female connector
- 3 mm MDF
- 220Ω resistors
Making the glasses

You can make your glasses out of anything you like and it might even be possible to modify a pair of existing sunglasses if they are big enough. My good friend Roger Persson was nice enough to design a pair of glasses for this book. In Figure 3.1 you will find the design of the glasses with measurements. Remember that these measurements might need to be modified so that the glasses fit your particular head size.

![Figure 3.1: The glasses design](image)

The design of the glasses is very simple yet fits the purpose. The design consists of three main pieces, the front and two side frames. The two smaller pieces are used for added support when connecting the frames to the front panel. The frames can be simply glued to the corners of the front panel, while the two small pieces are glued to the inside corners on each end, as shown in Figure 3.2. The design template can be used to cut the pieces out from any material you like, but I recommend a sturdy material of at least 3 mm thickness. Plastic materials would also work for this project, or you could even make the frames out of cardboard, which can easily be cut by hand.
The glasses presented in this chapter were cut from 3 mm MDF using a laser cutter. I don't expect most readers to have their own laser cutter, but many bigger cities today have something called hacker spaces or Fab Labs, and some even have them in their city's libraries. The concept of Fab Labs was developed by the Massachusetts Institute of Technology, building on the idea of hacker spaces, which are locations that offer access to different fabrication tools and equipment where different people come to work on projects. Hacker spaces are usually a place where people interested in electronics and programming meet to share ideas and work on projects. Fab Labs are more oriented toward fabrication, not just digital entity, and are open to the public. If you haven't checked already, I suggest you investigate whether there is a hacker space or Fab Labs close by since you now have the perfect excuse to head over for a visit.

Figure 3.2: The laser-cut pieces

If you choose to modify the design, remember to keep the size of the holes at 5 mm since this is the size of the LEDs we will be using. You can swap these LEDs for the smaller 3 mm ones if you like, but I would not recommend LEDs bigger than 5 mm since these might complicate the design.
**Entering the matrix**

An LED matrix is also known as a **diode matrix**, referring to the LED’s one-directional polarity. An LED matrix is a two-dimensional grid with an LED connected to each intersection where a row crosses a column. The columns and rows are isolated from one another in order for the matrix to work. In electronics this is also known as **multiplexing**.

*Figure 3.3* illustrates the entire schematic of all the connections. To the right, you will find the matrix layout. All the negative sides of the LEDs are connected in rows and all the positive sides of the LEDs are connected in columns. When power is applied to one of the columns, and a ground connection is opened up on the negative rows, only one LED will light up. As you might have noticed, we have connected part of the matrix to the analog pins. Since there are not enough digital pins, we will use some of the analog pins instead. The analog pins can be operated as digital pins and their numbering continues on from pin 13. In other words, analog pin A1 is the same as digital pin 14, and so on.

As I said before, the current can only pass through an LED in one direction, a fact we are using to our benefit while creating the LED matrix, giving us the possibility of controlling many LEDs with fewer pins. If we did not connect everything in a matrix, we would need 30 pins in order to control all the LEDs separately. The downside of using a matrix configuration is that we can only control one LED at a time.

However, we will take advantage of another phenomenon called **POV (persistence of vision)**. POV is a term used to describe an optical illusion where multiple images blend into one image in the mind of the beholder. The brain can only interpret about 25 discreet images per second; any more than that and the images start to blend together.

The following *Figure 3.3* illustrates the entire schematic of all the connections:

![Figure 3.3: The matrix schematic](image-url)
Chapter 4

Arduino is fast, even faster than the human eye, so we will use this speed to our advantage in order to give the impression of lighting up many LEDs at the same time. As I said, we can’t technically light up more than one LED at once in the matrix but we can switch between them so fast that the human eye will perceive it as more than one LED being on. But before we get to this part, we need to connect everything, and this means it is time to turn on the soldering iron.

Before we start soldering, we need to place the LEDs in the right order. A good idea is to check that all your LEDs work before soldering them. By connecting the LEDs one by one to a 3.3V coin cell battery, or using the power that goes from your Arduino to a breadboard, you can easily do this. If you are using a breadboard, don’t forget to add a 220Ω resistor.

If you cut 5 mm holes, the LEDs should fit nicely. If they are a bit loose don’t worry, as once they are soldered together everything will be held in place. To create the matrix, we need to solder the LEDs into rows and columns. In Figure 3.4 you can see how I have prepared the LEDs for soldering by bending the legs of the LEDs into the desired rows and columns. All the negative legs (the shorter ones) will be soldered in horizontal lines, and then the positive legs (the longer ones) will be soldered in vertical lines. Make sure that you bend the positive lines over the negative lines so they do not come into contact with one another. If they do, the matrix will not work as it is supposed to. If you want, you can cover up your lines using some tape needed. This is done by placing a small piece of tape in between the legs of the LEDs so they do not touch one another.
Once you are done, we can move on and add the wiring and the resistors that will connect to our Arduino board. Figure 3.5 shows a close-up of the resistors connected straight to the positive column in the glasses, and the wires connected to the other side of the resistors. The idea is to place the Arduino board on the inside of either the left or right frame. Before you cut your wires, measure the distance between the row and the location of the Arduino board where it will be placed on the inside of the side frame. Make sure you add some extra length before you cut them because it is better to have wires that are too long than too short.

Figure 3.5: A close-up of the matrix
Now is also a good time to put the JST female connector in place as shown in Figure 3.6. JST connectors are fairly standard connector for batteries, and in this project we will be using a very small 3.7V battery with a male JST connector. You can place the JST connector anywhere you like, but I found a good spot where the front panel meets the frame just under the supporting piece of MDF. Make sure you leave enough space on the back side of the connector to fit the power cable that connects to the Arduino board. To keep the JST connector in place, use some glue:

Figure 3.6: The JST female connector in place

When you have added the wires to all the positive columns, you can add three cables for the negative rows, again ensuring you make them long enough to reach the Arduino board. You don't need resistors on these lines since these will act as our GND channels.

Once you have all the LEDs, resistors, and wires in place, it is time to connect everything to the Arduino board. In this chapter, we are using the Trinket board from Adafruit, but you could also use an Arduino micro board, which is very similar in size. These boards are some of the smallest Arduino boards that offer most of the functionality of a standard Arduino board.
Soldering all the wires in place might be tricky. I started by gluing the board to the inside of the frame and then soldering the wires one by one. I would suggest that you place them where they fit best. You can always switch the layout in the code later on. In Figure 3.7 you will see what it might look like once all the wires are connected:

![Image of the Trinket board in place](image)

Take your time soldering all the wires in place. I admit that even someone with good soldering skills might find this project a bit tricky since it requires some unconventional soldering. I call this type of solder action "soldering" since you usually end up with something that looks like it came from a movie. Usually, you solder components on a flat surface, but with wearable projects like this one you need to be a bit creative when it comes to soldering things together. Eventually you will end up with an inside that is as impressive as the outside.
Next, we will move on to the programming side, and this is where we get to see the glasses in action. For the programming part, we will power the glasses via the USB cable, and once we are done we will add a battery, then you will be ready to head out into the night to impress everyone.

**Programming the glasses**

In order to make the Trinket board so small, the serial to USB conversion is left out from the design. On a regular Arduino board, this conversion is handled by another Atmega chip, and on older versions this was done by an FTDI (Future Technology Devices International) chip. The FTDI chips are still around and you can buy these as standalone breakout boards as shown in Figure 3.8 to the left of the Trinket board:

![Figure 3.8: The FTDI serial to USB converter and the Trinket board](image)

Normally, you solder male pins to the end of the Trinket board that connects to the FTDI converter, but in this case we want to keep the Trinket board as flat as possible and we don’t want sharp pins on the inside of the glasses that might hurt your eyes. So the trick is to just attach the male pin headers to the FTDI converter and hold it in place while programming the Trinket board. Once in a while there will be glitches in the connection and the upload will fail. This is probably due to the FTDI not connecting properly to the Trinket board. However, this is not a big problem since you can just start the upload over again while making sure the pins have a good connection.
Now let's make a sketch that checks that all of the LEDs light up. In order to do so, we will loop through the LEDs one by one to see that everything works as it is supposed to. The Trinket is programmed as a normal Arduino Uno board, so make sure you select this type in the board menu. Upload the following code and check the LEDs in front of the glasses:

```cpp
/*Collect all the positive columns pins in one array. You need to make sure that these pins correspond to the direction you have placed the columns in the glasses*/
int powerPin[]={
  3,4,5,6,8,9,14,16,17,18,19};

/*Collect all the negative row pins in one array. Again make sure they are added in the same order corresponding to the glasses*/
int gndPins[]={
  10,11,12};

void setup(){
  /*In order for the matrix to work we need to be able to control our gnd lines in the matrix. The trick is to use all pins as output. When we turn the gnd pins HIGH we will be able to block the ground connection*/
  for(int i=0; i<20;i++){
    pinMode(i,OUTPUT);
  }
  //Turn all the gnd pins HIGH in order to keep all LEDs off
  for(int j=0;j<3;j++){
    digitalWrite(gndPins[j],HIGH);
  }
}

void loop(){
  //Run the function
  looper();
}

void looper(){
  /*In this function we run through all the LEDs using two for loops starting by opening a gnd connection*/
  for(int i=0; i<11;i++){d
digitalWrite(powerPin[i],HIGH);
  //Once a gnd pin is accessible we turn on one of the LEDs
  for(int j=0;j<3;j++){
    digitalWrite(gndPins[j],LOW);
    delay(50);
    digitalWrite(gndPins[j],HIGH);
    delay(50);
```
In this example sketch, we are implementing a trick in order to be able to control the ground connections. If we connected the negative rows of the matrix straight to GND we would not be able to control the separate LEDs. The trick is to use normal pins as outputs. When a pin is LOW, it connects to ground, which we can use to light up our LEDs in the matrix. But once it turns to HIGH, we block the connection to the ground. So now we can control each LED individually by turning one of the positive columns HIGH and one of the negatives rows LOW. You will need to make sure that your pin declarations line up with the actual physical layout in your glasses or else looping through them could get very hard. As you can see in the schematic in Figure 3.3, the columns are connected after one another to the digital pins.

### Making a pattern

In the next code example, we will implement some pattern designs. These patterns are stored in arrays that correspond to the layout of the LEDs in the glasses. We can draw our patterns in code and later loop through the array and activate the LEDs. When the code is formatted as it is in the next sketch, we get a visual repetition of the pattern. A 0 in the array represents a turned off LED in the same position in the matrix and a 1 represent an LED that is turned HIGH:

```c
/*Collect all the positive columns pins in one array. You need to make sure that these pins correspond to the direction you have placed the columns in the glasses*/
int powerPin[]={
19,18,17,16,14,9,8,6,5,4,3};
/*Collect all the negative row pins in one array. Again make sure they are added in the same order corresponding to the glasses*/
int gndPins[]={
```
12,11,10};
//This is a two dimensional array that holds the pattern
int pattern[3][11] = {
    {1,1,1,1,0,0,0,1,1,1,1  },
    {0,1,1,0,0,0,0,1,0,1,1 },
    {0,1,1,0,0,0,0,1,1,1,1  }
};
//Variable to store the refresh rate on the led display
int refreshRate=200;

void setup(){
    //Declare all pins as outputs
    Serial.begin(9600);
    for(int i=0; i<20;i++){
        pinMode(i,OUTPUT);
    }
    //turn all the gnd ports High to keep them blocked
    for(int j=0;j<3;j++){
        digitalWrite(gndPins[j],HIGH);
    }
}
void loop(){
    //Run the pattern function
    displayPattern();
}

/*Function that runs through all the positions in the pattern array*/
void displayPattern()
{
    for (byte x=0; x<3; x++) {
        for (byte y=0; y<11; y++) {
            int data =pattern[x][y];
            //If the data stored in the array is 1 turn on the led
            if (data==1) {
                digitalWrite(powerPin[y],HIGH);
                digitalWrite(gndPins[x],LOW);
                delayMicroseconds(refreshRate);
                digitalWrite(powerPin[y],LOW);
                digitalWrite(gndPins[x],HIGH);
            }
            //If it is something else turn the led off
        else {
            digitalWrite(powerPin[y],LOW);
            digitalWrite(gndPins[x],HIGH);
            delayMicroseconds(refreshRate);
            digitalWrite(powerPin[y],HIGH);
        }
    }
}
This sketch implements a two-dimensional array, which is the same as placing an array into an array. As you can see, we have three arrays, and inside each of those arrays we have 11 positions in the first two and eight in the last one, which corresponds to the layout of the matrix. Using the two-dimensional array, we can now fetch the positions of the LEDs similar to $x$ and $y$ coordinates, which is much easier than storing everything in a normal array. If the values are stored in a normal array, we would need to define where each row ends on our own. This could be done using `if` sentences to check where the row begins, but using a two-dimensional array makes things much easier and makes for better-looking code.

Then the arrays run through the display pattern function, which loops through all the positions in the array. Every time it finds a 1 in the array it turns on the LED corresponding to the position in the actual glasses. It only turns it on for a brief time based on the refresh rate before it turns it off, since we can only have one LED on at a time in the LED matrix. Again, this is where we use the POV phenomenon, looping through all the LEDs very fast so that when we look at the glasses it looks like multiple LEDs are on, though in fact there is only one LED on at a time.

In order to get a better understanding of the code, I would suggest you modify the pattern array by changing which LEDs light up. If you look closely at the array, you might make out that I have tried to spell my initials TO with 1s in the code, which corresponds to LEDs turned on. Try switching the letters for your own initials and upload the code to your glasses.

**Finishing the glasses Knight Rider style**

For the last code example, we will have a look at how to create an animation. An animation is a simulation of motion, and in one sense the first code example in this chapter is a form of animation. We will build on the same principle in this section. Once you get the hang of the basic concepts, you can start building your own animations, combining the knowledge from the pattern example with the knowledge in this sketch.
In my beginner Arduino classes, the Knight Rider example is a classic. This example is inspired by the 80s hit show *Knight Rider* with David Hasselhoff. To be more precise, the example is inspired by the robotic car featured in the show, which is called Kit. In the front of Kit there is a small LED display that shows a bouncing light effect. This is the effect we will recreate on the front of the glasses.

The code example is fairly simple and does the same thing as the test sketch, but instead of lighting up only one LED at a time we will light up an entire column. We will move the light column from left to right and once we hit the end of the glasses, we will move the column back again:

```cpp
int powerPin[]={
  19,18,17,16,14,9,8,6,5,4,3};

int gndPins[]={
  12,11,10};

int refreshRate=200;

void setup(){
  for(int i=0; i<20;i++){
    pinMode(i,OUTPUT);
  }
  //
  for(int j=0;j<3;j++){
    digitalWrite(gndPins[j],HIGH);
  }
}

void loop(){
  nightrider();

}

//
void nightrider(){
  /*Instead of starting to loop through the columns we loop through
the row*/
  for(int i=0; i<11; i++){
    /*Then we loop through the column*/
    for(int j=0; j<3; j++){
      /*In order to perceive that the column is lit we need to loop it a few
times*/
      for(int k=0; k<50; k++){
        /*Then we light the column*/
      }
    }
  }
  /*Then we move the light column back again*/
}
```

digitalWrite(powerPin[i],HIGH);
digitalWrite(gndPins[j],LOW);
delayMicroseconds(refreshRate);
digitalWrite(powerPin[i],LOW);
digitalWrite(gndPins[j],HIGH);
}
}
/*/Once we have reached the end of the glasses we do the same thing backward*/
for(int i=11; i>0; i--){
  for(int j=3; j>0; j--){
    for(int k=0; k<50; k++){
      digitalWrite(powerPin[i],HIGH);
digitalWrite(gndPins[j],LOW);
delayMicroseconds(refreshRate);
digitalWrite(powerPin[i],LOW);
digitalWrite(gndPins[j],HIGH);
    }
  }
}

Once your code is done, upload it to the board and enjoy. In order to show off your new glasses, we need to finish up the circuit by attaching the battery to the glasses so you can walk around with them (since being forced to be connected to a USB port at all times might not look as cool). Any battery between 3-5V will work for this project, and the more amperes the battery has, the longer the glasses will stay on. However, large ampere batteries are also bigger. The smallest lithium battery I have found is the 3.7V 150mAh battery, which I recommend. There are smaller ones with less amperes but this type will still fit your glasses and give you enough power to keep the glasses on long enough for you to impress a few people.

Before you connect the battery, you need to solder wires from the JST connector to the board. The pins for external power on the Trinket board are marked BAT+ for the positive connection. Next to this pin there is a pin marked with only a G for the ground connection. This connection is connected to the negative side of the JST connector. In order to figure out which pin is which on the JST connector, check with the battery cables. Usually, these cables are red and black; the red one is the positive and the black one is the ground. The design of the glasses should leave enough space on the inside of the frames to place a small battery. Figure 3.9 shows what it might look like. I recommend gluing the battery in place using hot glue, but be careful not to heat the battery for too long or it might explode. This might sound scary, but you would need to heat it for some time before it explodes. However, I always recommend caution when combining heat and batteries.
LED Glasses

As you might have noticed by now, visibility is limited in these glasses and it has to be fairly dark for the light to be visible, so be careful when using them. You can still look cool while keeping safe.

![Battery connection](image)

**Figure 3.9: Battery connection**

In *Figure 3.10* you will find my co-worker Johannes at the university showing off the finished glasses. Unfortunately, pictures do not show the full effect of animating the LEDs, but hopefully you will be close to finishing your pair by the time you read this.
Summary

In this chapter, you learned how to create an LED matrix in a pair of custom-made LED glasses. The principles behind the LED matrix are the same as in any matrix that you might use in another project. On the code side, we had a look at some pattern designs and how to generate animations. This chapter introduces the basic concepts, but as you progress you can build upon this knowledge and develop your own patterns and animations.

In this chapter, we also had a look at what the custom casings looks like when cut using a laser cutter. If you do not have access to a laser cutter don’t worry, as you can still achieve the same results using cutting tools and materials such as plastic, wood, or cardboard if you like. The only difference is that it might take a bit longer to cut. The material for the frame also gives you a lot of personalization options, so I recommend you test a few materials before you decide.

The remaining projects will also implement laser-cut designs, but remember that the casings for all the projects in this book are just examples of how it can be done. I encourage you to develop your own designs since it is more fun to personalize your creations, and let the designs in this book act as inspiration.
Where to buy this book

You can buy Arduino Wearable Projects from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.