Welcome!

Welcome to the SpinWheel Field Guide! In this guide, you will find a few lessons and adventures that you can follow along with to begin learning to program your SpinWheel. This book represents just a small fraction of the interactive tools, lessons and adventures on our website, so we recommend checking out our online content in parallel to your explorations with this book! Like a true field guide, we encourage you to keep referring back to this book as you move to the online materials and begin to write your own programs for the SpinWheel.

We encourage you to begin by completing the online Color Coding page mentioned on the separate welcome sheet before following the Initial Setup Guide to install the software necessary to program the SpinWheel for yourself. Then learn how to upload the code you wrote in Color Coding onto your SpinWheel using Customizing the SpinWheel's Display. If you are curious about how to make all the colors of the rainbow with the SpinWheel's LEDs, then read Mixing Color with Light next (and check out the Biology of Sight adventure online after you finish).

To learn more about the software you are using to control the SpinWheel and how it works, jump into Arduino 101. Learning a programming language is just like learning any language - you'll need to work on your vocabulary and you might not understand everything at first! With practice, you will become more comfortable with writing your own programs and recognizing what different lines of code do.

Creating animations with the SpinWheel introduces how to design dynamic patterns on the SpinWheel's display, an idea that is expanded in the online adventures. Coding Building Blocks dives more deeply into the key elements of writing code. Finally, the SpinWheel Functions Resource is a summary of some of the key functions

necessary for controlling your SpinWheel's LEDs.

We hope you enjoy exploring physics, math, and art with your SpinWheel through this guide and through our online resources!

SpinWheel Initial Setup

We're so excited that you are ready to use your SpinWheel! This page begins the process of setting you up to create new and exciting programs on your SpinWheel! If you haven't already done so, please complete the **Quick Start Page** that came with your device.

Installing the Arduino Software

Much of the joy of the SpinWheel comes from your ability to change it and make it do whatever you wish! The rest of the chapter will walk you through adding a new animation to your SpinWheel. Do not worry if you find this part challenging. Learning new things can be confusing at first. If you get stuck, check out the troubleshooting guide online at *spinwearables.com/troubleshoot* and don't be afraid to experiment. While feeling confused is normal, it will get easier as you go!

In order to write new animations for the SpinWheel, you

will need a way to reprogram its onboard computer. We use the Arduino software to communicate with the SpinWheel.

You can download the Arduino software onto your computer for free from *arduino.cc/en/Main/Software#download*. For step by step help, Arduino has provided instructions for each operating system at *arduino.cc/en/Guide*.

Configuring the Arduino Software

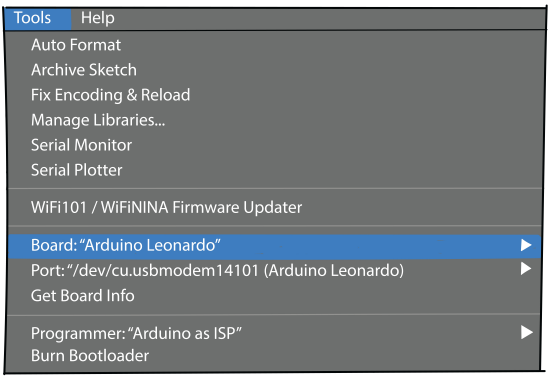
Once the software is installed, we have to configure it to communicate with the SpinWheel.

1. Flip the switch to the position labeled "USB" and then plug your SpinWheel into your computer with the provided micro USB cable.

2. Open the Arduino software.

3. Open the **Tools** menu and go to **Port**. You will see a list of serial ports on your computer; select the port that corresponds to the SpinWheel.

If there are multiple ports and you are unsure which one to use, simply unplug the SpinWheel and see which serial port disappears when you do so. This port corresponds to



Use **Tools** → **Port** and **Tools** → **Board** to pick the port corresponding to the SpinWheel and the "Leonardo" board type.

your SpinWheel's serial port. If you do not see a port appear/disappear, make sure you are using the micro USB cable that came with your SpinWheel as others may not have the needed functionality.

4. Go back to the **Tools** menu and select **Tools → Board**. Select Arduino Leonardo as the board (a.k.a. processor), so that the software knows which "dialect" to use to talk to the SpinWheel. Computer languages have dialects just like human languages!

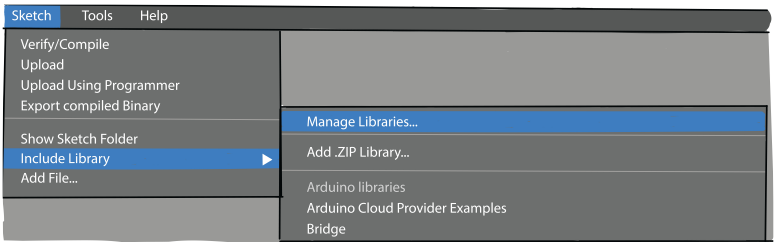
Properly selecting the board and port are essential for the Arduino software to communicate with the SpinWheel. If you are unable to upload code to the SpinWheel in the next section, double check that the switch is set to "USB" and that you have the correct board and port selected.

Installing the SpinWheel libraries

1. To get the first set of example programs you can run on the SpinWheel, download our SpinWearables Arduino Library using **Sketch → Include Library → Manage Libraries...**

2. In the search bar of the Library Manager, search for **SpinWearables** and then click **Install**.

3. You will be automatically prompted to install two other required libraries (NeoPixel for controlling the LEDs and ICM 20948 for reading the motion sensor). You will need to install both of these to use the SpinWheel.



Installing Arduino libraries

Running a program on the SpinWheel

To test that your SpinWheel is working properly, you can install a new program, or sketch, from the example files to animate your SpinWheel.

1. Choose a file to install by opening **File** → **Examples** → **SpinWearables** and picking one of the examples. For instance, pick **BlinkingFirmware**. This will open a new window with the code.

2. Upload the code to your SpinWheel by pressing the upload button (the arrow at the top).

Upload programs to your SpinWheel using the **Upload** button (highlighted in white).

Now your SpinWheel will have the new colorful blinking pattern (from **BlinkingFirmware**) you just uploaded. If you get an error here, then check out the troubleshooting guide online for help on some common problems. This will help you learn how to figure out what to fix from the error messages (which can seem overwhelming at first).

Feel free to open any of the other SpinWheel sketches and upload them onto the device. Do not worry about understanding what the code does, you will learn more about this language in future lessons. We encourage you to experiment with these examples! If you want to save any changes, you will be prompted to save the sketch in a new location (can be anywhere on your computer). The original file will always be available to open again.

Uploading a new sketch to your SpinWheel will overwrite the preloaded sketch that came on it. If you

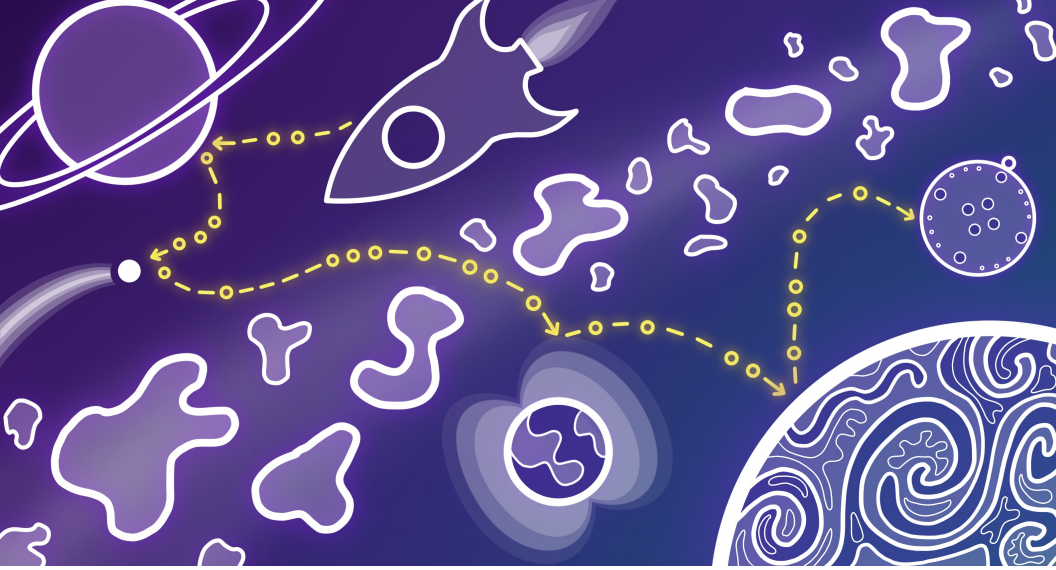
want your SpinWheel to have the sketch that it came with, simply open the **SpinWheelStockFirmware** example and upload it.

In future SpinWheel activities, you will be writing new sketches to animate the SpinWheel. To transfer a sketch from your computer to your SpinWheel, simply connect your SpinWheel to your computer, change the switch to "USB", open the code of your new sketch in the Arduino software and press the upload button.

Congratulations!

You are now ready to continue with the rest of the SpinWheel activities!

This SpinWheel Field Guide contains some hands-on adventures and reference material. This material is found in expanded versions online, along with many more activities for you to enjoy. We highly recommend that you make use of both resources. In addition, the online version includes virtual SpinWheels that allow you to test and experiment with your code and see what the result might look like before uploading to your real SpinWheel!



Customizing the SpinWheel's Display

A program is a set of written commands for your computer to follow. Here you will create your own design on your SpinWheel. Add more text here to explain a little bit more

Now that you have successfully added a sketch to your SpinWheel and have had a chance to practice controlling the SpinWheel's LEDs using the virtual SpinWheel on our website, let's learn how to create a simple program for the SpinWheel itself.

Approach this chapter the way you would approach the first few lines of a foreign language you want to learn. Try to pick out words that make sense, without worrying about the overall structure, correct syntax, or proper grammar. As time passes and you have learned new things, come back to this page and see whether you can understand a bit more of it.

Computers follow instructions. They do not solve

problems on their own. So, when writing a program for the SpinWheel's onboard computer, you need to be very explicit in the instructions that you write! The particular language we are using requires our programs to have a certain structure. In the Arduino software, if you navigate to `File -> New`, you can see the basic structure for an Arduino program.

To produce a program capable of sending instructions to the hardware of the SpinWheel (e.g. the LEDs and motion sensor), we need to add a few more lines:

```
#include "SpinWearables.h"
using namespace SpinWearables;

void setup() {
  SpinWheel.begin();
}

void loop() {

}
```

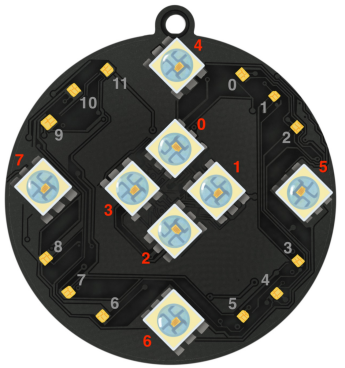
If you are curious about why you need these other lines, then flip through to Arduino 101.

Now we can add something to the loop() section in our sketch to have the SpinWheel light up. This is similar to what you did in the Color Coding page with the virtual SpinWheel. To start with, let's turn on an LED by simply adding two commands to the empty line in the loop section of code we started above.

```
SpinWheel.setLargeLED(0, 255, 0, 0);
SpinWheel.drawFrame();
```

If you don't want to have to type this into the Arduino software, you can also open the code from Examples -> ?? in the Arduino software and upload it to your SpinWheel. You may remember from the Color Coding page that this command will change the color of a specific large LED based on the values given in the command. These values follow the format of `SpinWheel.setLargeLED(LED_you_want_to_change, amount_of_red, amount_of_green, amount_of_blue)`. In

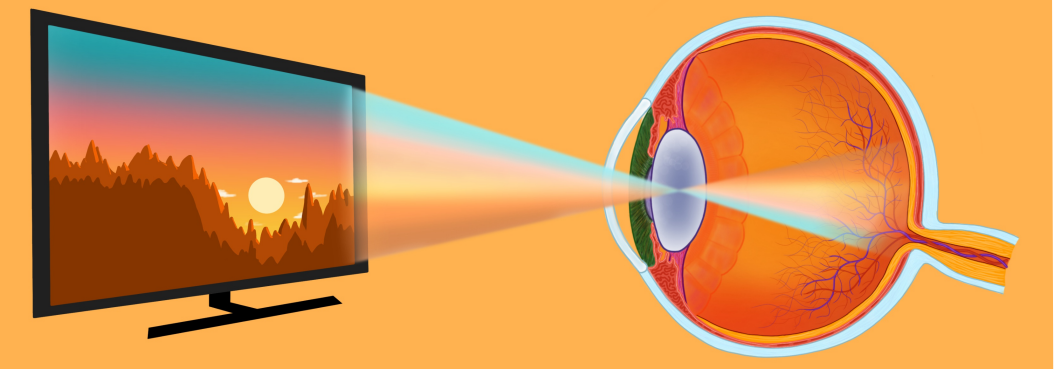
this case, it will light up the first LED in red (the programming language used by Arduino starts counting at 0). In the next chapter, we'll explore more about how we create a rainbow of colors using just red, green, and blue light. Feel free to check it out now if you are curious.



SpinWheel LED numbering

If your new sketch fails to upload, then reread the Initial Setup Guide and check out the troubleshooting guide on our website. A few things to be sure to check are that you have the correct `port` and `board` selected and that you have copied the header information from above. We have an example sketch you can compare yours to under `File -> Examples -> Book`.

If you want to do something more complicated in this section, then try replacing `SpinWheel.setLargeLED()` with your code from the Color Coding page (spinwearables.com/intro). These commands are also listed in the back of this guide in the "SpinWheel Commands Reference". Keep experimenting like you did on the virtual SpinWheel to create beautiful designs from your imagination on the SpinWheel's interface!



Mixing Color with Light

Human perception of light and color has many curious features rooted in biology and physics. In this chapter, you will learn how to trick your eyes into perceiving a rainbow of colors using only red, green, and blue LEDs. If you want to learn more, check out the Biology of Sight Adventure on our website.

When light comes from the Sun (or most other sources of illumination), we perceive it as lacking a hue (or, white light). In reality, white light is made up of many colors. You can use a prism to separate the components of the mixture. A prism works by bending, or "refracting", light at different angles depending on its color, thereby allowing us to see all of the colors that make up white light. This is why if you let sunlight shine through a prism, you can see a rainbow.

The light-sensing tissue in the back of our eyes, the retina, has small cells that respond to some of these colors. They are called "cone cells" and are classified into three separate groups by the color that they sense the best: red, green, or blue. Each of these cells responds to one of these three colors, but not the others. For instance, the blue-sensing cones respond to blue light, but they do not respond to red light, and vice versa.

White light being split into colors by a prism. The white light shines on the prism from the bottom left, and a big part of it is refracted and split as it passes through the prism.

If our eyes can sense only red, green, and blue, how can we see yellow? Our eyes and brains have evolved so that our red- and green-sensing cones both respond slightly to yellow. If our brain detects that both groups of cones are activated, it knows to interpret the color as yellow. We can see other colors this way too. For instance, purple activates both red- and blue-sensing cones.

We can exploit this imperfection of our eyes to make rich colorful electronic displays while using only three

An artistic rendering of a close-up of the back of the eye illustrating the rods (black) and cones (triangles colored by type).

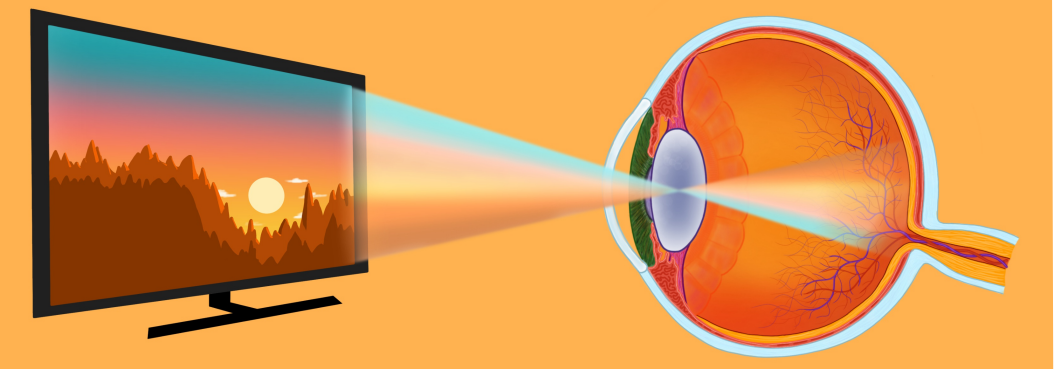
colors. For instance, since our eyes cannot distinguish between true purple and a mixture of blue and red, we don't need a purple light source, only blue and red lights (and green for the other color combinations).

The SpinWheel's colorful display takes advantage of this. If you look closely at a LED light on the SpinWheel, you can see that it contains 3 small light sources placed very close together: one red, one green, and one blue. Combining these lights in different intensities allows for a wide variety of colors to be displayed on the LEDs.

To better see the components of an LED on the Spinwheel, open Examples -> ?? in the Arduino software, and upload it to your SpinWheel. Look closely at the red LED on the SpinWheel; you should see 1 red light in the LED. Likewise, the blue and green LEDs will also have 1 light in them. If you look instead at the white LED, you should see 1 red, 1 green and 1 blue light inside it. When each light is turned on, the colors combine to make white light.

Concusion paragraph goes here

An up-close picture of an LED, showing the red, green, and blue subpixels.



Arduino 101

The SpinWheel relies on the Arduino software to be programmed. This guide expands on topics introduced in earlier chapters and will be useful to keep referring back to as you finish reading this book and start the online adventures.

Many simple computer chips for DIY projects make use of the Arduino software, a platform for writing and uploading code onto a physical device, like the SpinWheel. The Arduino software uses a simplified version of C++, a programming language that we will begin to introduce below.

The Skeleton of an Arduino Program

As we discussed in Customizing the SpinWheel Display, the Aruduino software requires our programs to have a certain structure. The most basic program looks like this:

```
void setup() {

}

void loop() {

}
```

This program does absolutely nothing. It contains two blocks (or sections) of code which start and end with the brackets: `{` and `}`. For our program to do anything, we

need to fill these blocks with instructions. The first block is called `setup` and runs only once, immediately after the device is powered up. This block is used for setting up any initial conditions we might require.

Next there is the `loop` block. This block is executed repeatedly "in a loop", starting immediately after `setup` is done. The loop repeats itself until the power is turned off. Most of our instructions will be written in this block. They will frequently involve measuring time or motion and then producing a colorful pattern based on the SpinWheel's motion senso'sr measurement.

When you turn on the SpinWheel, `setup()` is run once and then the `loop()` block is run repeatedly until the SpinWheel is turned off.

Extra elements for a SpinWheel program

To produce a program capable of sending instructions to the hardware of the SpinWheel (e.g. the LEDs and motion sensor), our program requires a few more lines:

```
// Include extra resources and commands
// specific to the SpinWheel.
#include "SpinWearables.h"
using namespace SpinWearables;

void setup() {
  // Instruct the specific SpinWheel hardware
  // to be ready to receive instructions.
  SpinWheel.begin();
}

void loop() {

}
```

Adding `#include "SpinWearables.h"` and `using namespace SpinWearables;` before `setup()` ensures that the rest of the program has access to the extra resources and commands specific for programming your SpinWheel. It is also necessary to add `SpinWheel.begin();` in the `setup` block. This line makes sure that the SpinWheel hardware is ready for the instructions that we will add in the `loop` block of code. When you start writing other

programs for the SpinWheel, these extra lines will be essential.

The `loop` block is still empty and this program still will not do anything interesting. However, our `setup` section is complete: it prepares the SpinWheel to receive instructions. Through our activities, we will rarely need anything more sophisticated in `setup`.

Finally, let us turn on an LED by adding a single command (or function, like we introduced in the last chapter) in the loop section of code we started above.

```
SpinWheel.setLargeLED(0, 255, 0, 0);
SpinWheel.drawFrame();
```

Open the code from Examples -> ?? in the Arduino software and upload it to your SpinWheel. It should cause one large LED to turn on in bright red.

Let's discuss each line we added to the code:

The first line, `SpinWheel.setLargeLED(0, 255, 0, 0);` tells the SpinWheel to get ready to set one LED to the color specified. The first item (also called a "parameter" or "arguement") in the parentheses identifies the affected LED and should be between 0 and 7. The other three numbers are the red, green, and blue components of the desired color. They have to be numbers between 0 (color is off) and 255 (color is on at full brightness). Together, this line of code looks something like: `SpinWheel.setLargeLED(LED_you_want_to_change, amount_of_red, amount_of_green, amount_of_blue)`.

::: further-reading

0-255 is the range used by the SpinWheel for adjusting the LEDs. While these numbers look strange to a human, who would probably prefer the range 0-100, computers use binary numbers and it is more efficient to use ranges that are powers of 2, as in $256=2^8$.

:::

The line `SpinWheel.drawFrame();` signals to the

SpinWheel that we are done specifying actions to take. It tells the SpinWheel to “draw” all of the commands that were listed above it. Without this line, the LED specified in `SpinWheel.setLargeLED` won’t get lit up.

While looking at the code in the Arduino software, you may have noticed a few more things about the style of this programming language:

- We tend to have only one "command" per line. This makes the code more readable.

- Each command is followed by a semicolon `;`. That makes it easier for the computer to separate different commands.

- Commands tend to be some name followed by parentheses `()`.

- Inside these parentheses we frequently put some extra information: this information can control how a command performs. For instance, in `setLargeLED` we have one parameter that selects the LED we want to modify and three parameters for the color of that LED.

- There are other ways in which LED colors can be modified and motion be detected. We will be discussing many such tools in later pages.

Receiving communication from your device

It can be very useful to have a way to receive messages from the computer chip you are programming. For instance, you might want to be able to see the values that the motion sensors are recording. Having this information can be very important when debugging - or attempting to find errors in - code.

Different computer languages provide different ways of doing this. If you are working with Arduino, then you can use the `Serial Monitor` to see these messages being sent back over the micro USB cable. To access the `Serial

Monitor`, navigate to `Tools ! Serial Monitor`.

Using the Serial Monitor we can check the output of the SpinWheel's motion sensors. The SpinWheel is capable of detecting a magnetic field, acceleration, and rotation. To start with, we will look at the rotation along the x-axis. In the "Step Counter" and "Dancing with Color" online adventures, we use this output in more exciting ways. Check them out, when you want to do more.

Here is the code to use the Serial Monitor to record the rotation of the SpinWheel in three demensions (corresponding to the axes in the above diagram). In the `setup` block, we can tell the device to send messages at the right connection speed using `Serial.begin(9600)` and print the message you want using the `Serial.println()` function. When you upload this sketch, you may notice that a small LED flashes. We programmed the SpinWheel's software to do this as an indicator that you are sending information to the Serial Monitor. Notice how the value in Serial Monitor changes as you spin the SpinWheel.

```
#include "SpinWearables.h"
using namespace SpinWearables;

void setup() {
  SpinWheel.begin();
  // The next line ensures that the communication hardware
  // on our device is ready to send messages.
  // The name "Serial" is such for historical reasons
  // (it is the name for this type of communication).
  Serial.begin(9600); // The 9600 is the speed of the connection.
}

void loop() {
  // This line gets the information from the SpinWheel's
  // motion sensor.
  SpinWheel.readIMU();
  // Send a message to the connected computer.
  // The message will just be the value of the SpinWheel's
  // rotation around the x-axis.
  Serial.println(SpinWheel.gx);
  // Wait for 500 milliseconds (half a second) before you
  // start the loop function again.
  delay(500);
}
```

You may have noticed that we use the `delay()` function here. As `loop` repeats many times in a second, we use the delay function so that the speed that counter is displayed through the Serial monitor is slow enough to follow. In this case, we tell it to pause for 500 milliseconds or half a second (there are 1000 milliseconds in a second). Try removing this line (or adjusting the time) and see how the Serial Monitor's output changes.

If you are interested in seeing this output as a plot, you can also navigate to `Tools -> Serial Plotter` to see a plot of the value.

This chapter expands on some concepts that you've already seen and will be useful as you begin to code the SpinWheel in more complicated ways. We will refer back to this guide and hope that you will use it as a reference.

::: further-reading

The Arduino community has very detailed resources on the programming language that we are using. You can start with their tutorial at https://www.arduino.cc/en/Tutorial/Foundations for instance.

:::