M.A.D.E.

Intermediate E-textiles Guides

for Music | Art | Design | Experiences

Mapping

Please cite this work as Fields, D. A., Amely, J., Jayathirtha, G., Lindberg, L., Lui, D. & Kafai, Y. B. (2019). *M.A.D.E. Intermediate E-Textiles Guide: Mapping.* Available online at http://exploringcs.org/e-textiles/modules.

Emma Eccles Jones College of Education and Human Services, Utah State University

Graduate School of Education, Penn GSE

ECS Exploring Computer Science

What you need to know first:

This module assumes that users have basic experience with making lighting patterns in Arduino (i.e., with digitalWrite() and delay() commands). For more introductory material, see the Exploring Computer Science e-textiles curriculum unit, Buechley & Lui Sew Electric, or other introductory Arduino guides. Other ECS E-Textiles modules as well as supporting code samples that may be referenced in this text can be found through http://exploringcs.org/e-textiles/modules.

Table of Contents

---

1. What is Mapping?

Mapping in Arduino is used whenever you want one set of numbers to translate to a different set of numbers. When we read a sensor it gives us a range of numbers from **`0`** to **`1023`**. We want to link the information the sensor processes to an LED to make that information visible to us. To do that, we have to evenly smoosh all of the possible readings of **`0`** to **`1023`** down to **`0`** to **`255`**. Why **`255`**? Because those are all of the possible programmable steps of an analog LED. To "map" a smaller range of numbers to a larger range of numbers, we would evenly stretch that small range out.

2. Mapping Code Breakdown

This is the example code that Arduino gives us:

```
map(variable, fromLow, fromHigh, toLow, toHigh);
```

To use this function, you need to have these 6 parts:

1. The **`variable`** number that you just got from reading the sensor
2. **`fromLow`**, the lowest number that your sensor can give
3. **`fromHigh`**, the highest number your sensor can give
4. **`toLow`**, the lowest number you want
5. **`toHigh`**, the highest number you want
6. And you have to store it all within a **`variable`** to hold all your newly mapped numbers (see **`mappedValue`** below)

For deeper mapping knowledge, dive into this link:
https://www.arduino.cc/reference/en/language/functions/math/map/

3. Code Ingredients for Mapping

To work the **`map()`** function in your code, you will need:

- A variable like **`lightSensor`** for the sensor you want to read
- A variable to store the sensor reading, such as **`lightValue`**
- **`analogRead()`** function to read the sensor
- Serial Monitor functions: **`Serial.begin()`**, **`Serial.println()`**, and **`Serial.print()`**. If you need to brush up on the Serial Monitor, check out the

Exploring Computer Science E-Textiles Unit PDF (page E45) or this video tutorial on how to use it at ProgrammingElectronics.com.

- A short `delay()` so that you don't crash the computer by making it read too fast

And then add to that…

- A variable to stick the mapped reading into, like **`mappedValue`**
- `map()` function so that you can squoosh readings to a smaller range or explode them to a larger range
- The lowest and highest of numbers you want to use from your sensor
- The lowest and highest of numbers that you want in your output

4. EXAMPLE Using the `map()` Function

Download this sketch code: simpleMapping (found at http://www.exploringcs.org/e-textiles/modules/supporting-code).

Take a walk through each line of this code to understand what happens here.

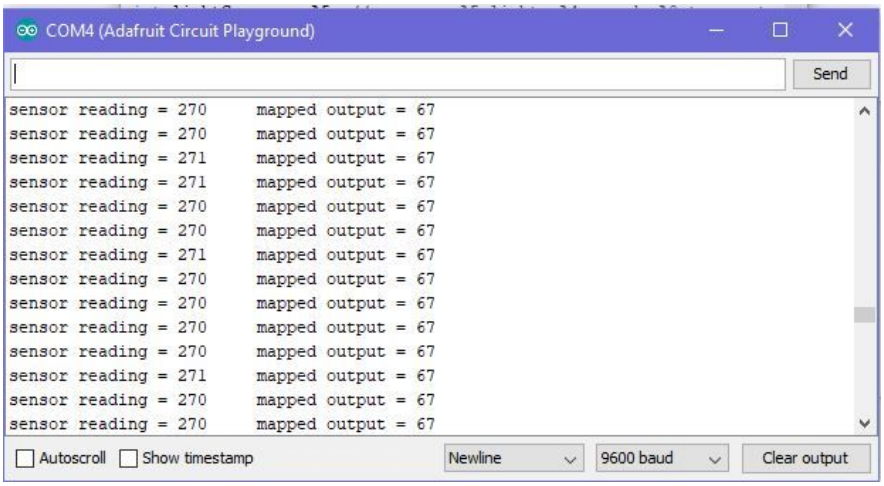
simpleMapping | Arduino	

`int lightSensor = A5;`
`int lightValue;`
`int mappedValue;`
	NAMING
	With the sensor pin, sensor reading variable, and the mapped sensor variable declared.

`void setup(){`
`  pinMode(lightSensor, INPUT_PULLUP);`
`  Serial.begin(9600);`
`}`
	SETUP
	We want what is coming IN to the sensor that we are reading and we want the internal resistor to be used so we use INPUT_PULLUP. Also start reading the sensor at 9600 bps, aka baud rate.

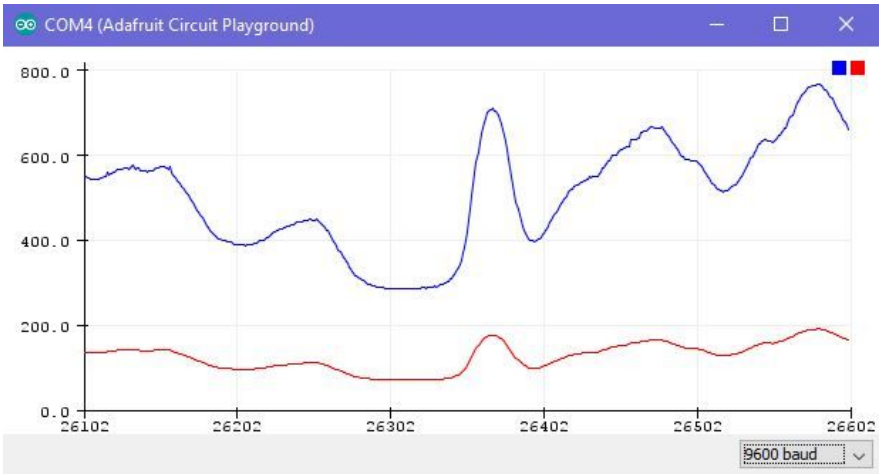
`void loop(){`
`lightValue = analogRead(lightSensor);`
`mappedValue = map(lightValue,0,1023,0,255);`
 	
`Serial.print("sensor reading = ");`
`Serial.print(lightValue);`
`Serial.print("\t mapped output = ");`
`Serial.println(mappedValue);`
`delay(2);`
`}`
	MAIN BODY
	Read the sensor and store the number. Map the number to desired low/high text to type in Serial Monitor, then variable numbers.
	
	Short delay so it doesn't crash the computer.

- Load this program onto your Circuit Playground.
- Open up the Serial Monitor to see what all of this extra text in the Serial monitor looks like.



Here we take the large range of **`0`** to **`1023`** and smoosh it down to the range of an analog LED, that of **`0`** to **`255`**. See how the mapped output is a much smaller range than what the sensor reading actually is?

- Close the Serial Monitor and open up the Serial Plotter.



Hey, the text we added doesn't even show up, but the plotted number ranges do! Now you can really visualize how mapping changes your sensor values. The blue line is what the sensor is reading, and the red line is the new mapped value that we can send to an analog LED to make it dim or brighten according to what the sensor senses.

4.1 CHALLENGE WALKTHROUGH | Link an LED to a Sensor

Download the sketch to start: simpleMappedSensorToLED (found at http://www.exploringcs.org/e-textiles/modules/supporting-code).

To link your input sensor to an LED, you will need:

- The **analogWrite()** function
- A variable for the analog LED pin, like **mappedLED = 13;**
- The **mappedValue** variable that you created above

To combine these pieces together into one line of code that goes into your **void loop()**, this very important bit of code links your LED pin to the range of numbers:

```
analogWrite(mappedLED, mappedValue);
```

Your LED will now get brighter or darker depending on the values that the light sensor is reading! For low values it will be darker, and for high values it will be brighter. You might have to tweak your numbers a bit to get full dark and full bright.

simpleMappedSensor | Arduino	

`int lightSensor = A5;`
`int mappedLED = 13;`
 	
`int lightValue;`
`int mappedValue;`
	NAMING

`void setup(){`
`  pinMode(lightSensor, INPUT_PULLUP);`
`  pinMode(mappedLED, OUTPUT);`
`}`
	SETUP

`void loop(){`
`  lightValue = analogRead(lightSensor);`
`  mappedValue = map(lightValue, 250, 975, 0, 255);`
 	
`  analogWrite(mappedLED, mappedValue);`
`}`
	MAIN BODY
	Read the sensor. Map the value to a range that the LED can use. This example shows that this light sensor ranges from 280 at the lowest to 950 at the highest. If add a little more for either direction, it stops the occasional bright/dark flash when you get readings outside of the sensor range.
	
	Link them together so you can see an interaction with the sensor by how the LED brightens or fades.

4.2. CHALLENGES for Mapping

Code to start with: Simple Mapped Sensor To LED and, if you need it, Simple Serial to only read the light sensor (both found at http://www.exploringcs.org/e-textiles/modules/supporting-code).

- ❏ Explore the current light sensor ranges.
  - ❏ **EASY** What did you do to get the lowest reading?
  - ❏ **EASY** What did you do to get the highest reading?
  - ❏ **MEDIUM** Can you map the ambient light levels to get a full range on the Serial Monitor?
- ❏ Explore the current sound sensor readings.
  - ❏ **EASY** What did you do to get the lowest reading?
  - ❏ **EASY** What did you do to get the highest reading?
  - ❏ **EASY** What is your normal conversation volume reading?
  - ❏ **EASY** What is the smallest thing you can do to get the sensor reading for conversation to rise a little bit?
  - ❏ **MEDIUM** Can you map your conversation levels to the whole range of an analog LED?
- ❏ **HARD** Make an LED get fade off or fade on linked to your light sensor's lows and highs. (See section 4.1 for a guide.)
- ❏ **HARD** Do the opposite! Make the LED get brighter when the sensor is low
- ❏ **HARDER** Map the sound sensor to your LED to get the full range of the LED, even during normal conversation.
- ❏ **HARDEST** Use the delay variable to change how fast the LED blinks. Map the input of your light sensor to the length of *time* between LED blinks. HINT: You will need to put a mapped variable in your delay code (i.e., **`delay (timelength)`** —be sure to adjust as needed in the naming section!) mapped from the range of your sensor's highs and lows, to the range of 100–1000 (milliseconds for the **`delay()`**).
- ❏ **HARDEST** Using the SimplemappedInteraction code as an example (found at http://www.exploringcs.org/e-textiles/modules/supporting-code), split up your mapped light sensor readings to have two LED patterns. For 0 to 500 have a slow blinking LED. For 501 to 1023, have a fast blinking LED. Play with and change this to whatever patterns and ranges you need.

If you are stuck, try these:

> Light: cover the sensor—wave your hand over it—shine a light into it.

Sound: try talking—snapping your fingers—yelling—tapping—blowing—whispering—dragging something over the sensor.

Code: You may have to adjust the highest and lowest numbers you read *from* the sensor to exaggerate the results that you want.