

BeagleBone Black Weather Visualizer

by Mark Neuburger

Introduction	2
Getting started with the BeagleBone Black	2
Update system software and configure WiFi	3
Set up LEDscape on your BeagleBone Black	7
Get familiar with the Cloud9 IDE and control your pixels from JavaScript	10
Finish the proof of concept	10
Laser cut the front and back pieces	11
Paint the front board	13
Wire the electronics	19
Solder the LED Strand	20
Add final code	26
Tape LEDs to board	27
Set up legend LEDs	27
Next steps	31
Attribution	31

Introduction

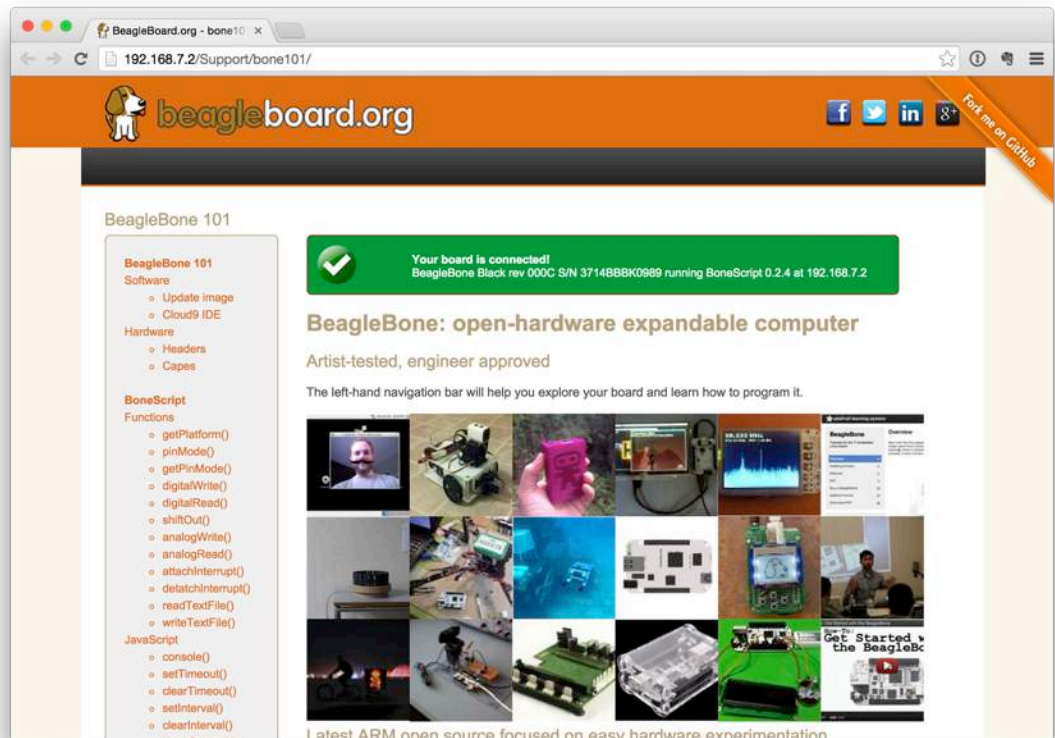
The BeagleBoard is an open-source computer that packs a lot of processing power and functionality onto a circuit board the size of a credit card. The latest BeagleBoard, BeagleBone Black rev. C (hereafter, “BBB”), boasts a 1 GHz ARM processor, 512 MB DDR RAM, HDMI video, USB port, and 4 GB of flash storage with Debian Linux pre-installed. These specifications place it on the low end of desktop-class performance. You might compare its computing muscle with that of a midrange laptop from a few years ago: it can handling running modern software, browsing the web, and playing high-definition video, but it won’t be able to handle more demanding tasks like running modern games or transcoding high-definition videos in real time. In addition to its small size and low price, the BBB consumes only 1-2 W of power, making it economical to run continuously. Contrast that with a typical desktop tower’s usage of 60-250 W (or think of a 60-100W lightbulb)! The BBB really shines when also considering its microcontroller-like ability to interact with physical inputs like sensors, buttons, and switches and outputs such as motors, lights, and relays through its GPIO (general-purpose input/output) pins. This capability makes the BeagleBone a strong choice for controlling an Internet-connected device; it offers the flexibility and power of high-level computer programming and software, and it has the versatility of a microcontroller in interacting with the physical world around it. In this project, we’ll build a poster-sized map of the USA that shows a live-updating LED visualization of the temperature at each of the fifty state capitals.

Getting started with the BeagleBone Black

1. Take your BBB out of the box and admire its small size.
2. Plug the BBB into an available USB port on your computer using the provided microUSB to USB A cable. Your computer will provide 5V power over USB and it will automatically boot up.

Note: avoid suddenly removing power from your BBB (i.e. unplugging it while it’s on). Always shut down by either pressing the PWR button on the board or issuing a shutdown command via software, then waiting for all of the lights to go out before unplugging the power source.

3. Install drivers for your operating system as indicated in the chart at <http://beagleboard.org/static/beaglebone/latest/README.htm#step2>
4. By default, the BBB comes with Debian Linux installed, running an instance of Apache web server that hosts a few interactive web pages. Browse to the web server on your board by navigating your web browser to <http://192.168.7.2/>. You should see a green box with the text “Your board is connected.”



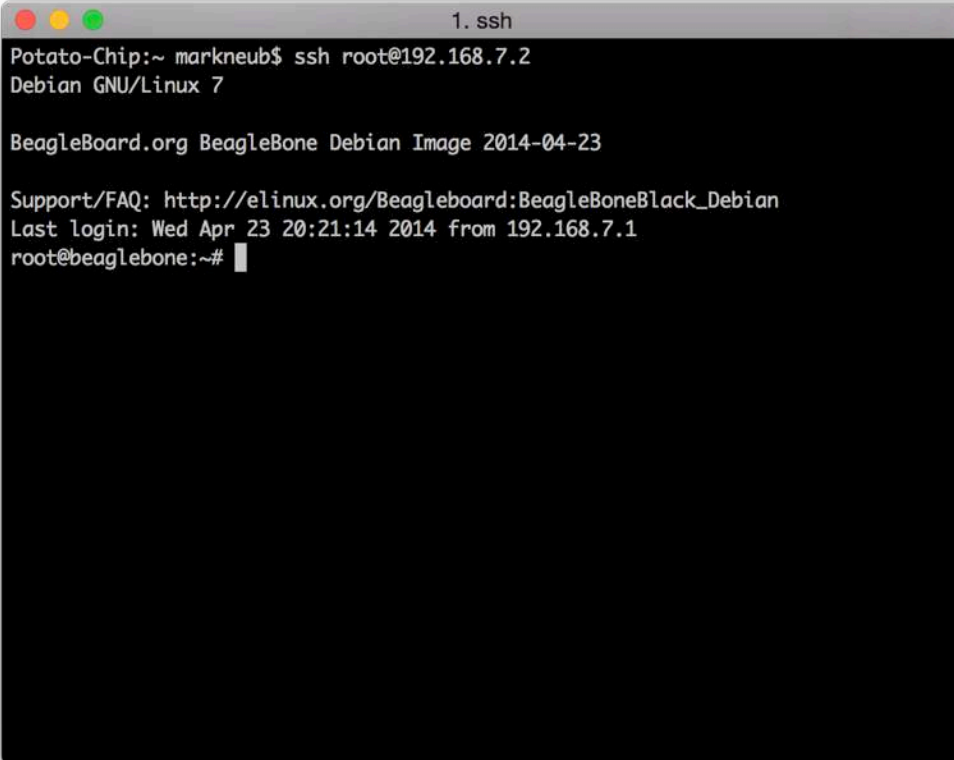
5. If you'd like, check out some of the other content on this page, such as the BoneScript interactive guide. Click "run" next to the example code to make all of the small blue LEDs on your board light up for two seconds.

Update system software and configure WiFi

We've successfully powered up your BBB, but before we can use it to fetch data from the web, we'll need to configure its Internet access. To cut down on cable clutter, we'll attach a WiFi dongle to your BBB's USB port and configure it to automatically connect to your local WiFi network when your BBB starts up. If you don't mind permanently hardwiring your BBB to the Internet over Ethernet, you can just do that and skip the rest of this section.

1. First, we'll update the system software on your board. Attach an Ethernet cable from your BBB to your router in addition to the USB cable already connected to your computer. If you're at NextFab, you can connect your BBB using any of the blue Ethernet cables around the second floor co-working area.
2. Open a Terminal window on your computer. If you're on a Mac or Linux machine, you can use the Terminal app provided with your system. On Windows, I recommend the popular program [PuTTY](#). Connect to your BBB via SSH by entering the following command into your terminal application and pressing enter:
`ssh root@192.168.7.2`
(In PuTTY, you'll need to enter the Host Name of 192.168.7.2 and select the SSH connection type, then click Open. You'll enter the username of root after you connect.)

3. You should see a command prompt like the one below. You're now connected to your BBB over SSH.

A terminal window titled "1. ssh" showing an SSH session. The user "markneub" on "Potato-Chip" connects to "root@192.168.7.2". The terminal displays the Debian GNU/Linux 7 banner, the BeagleBoard.org BeagleBone Debian Image version (2014-04-23), support/FAQ URL, and the last login time. The prompt is "root@beaglebone:~#".

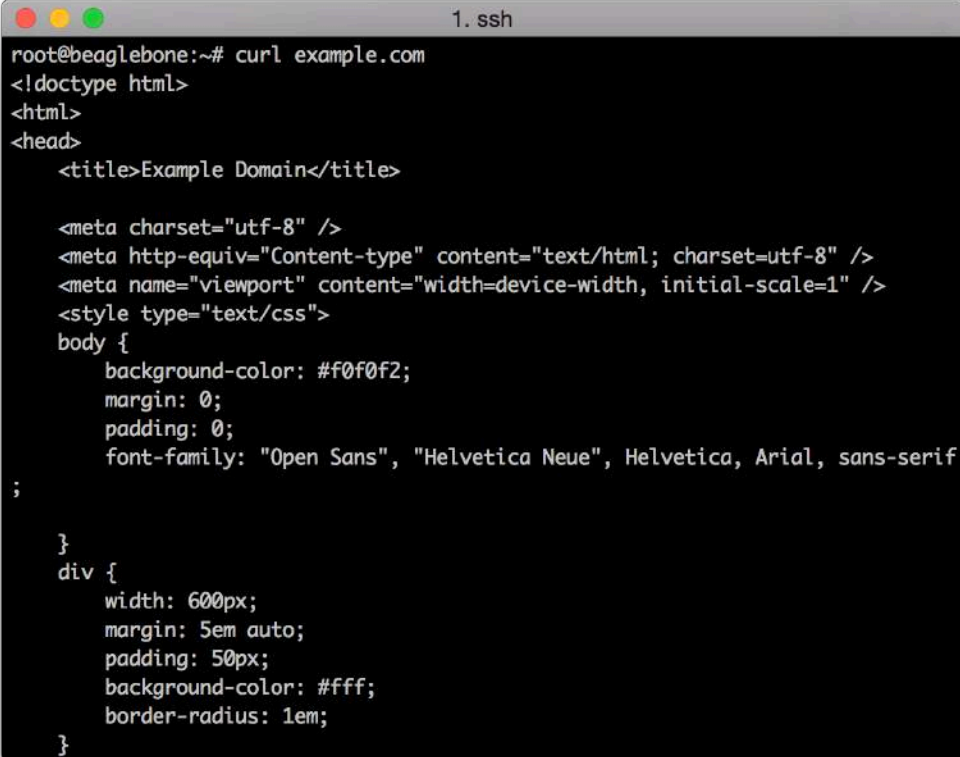
```
1. ssh
Potato-Chip:~ markneub$ ssh root@192.168.7.2
Debian GNU/Linux 7

BeagleBoard.org BeagleBone Debian Image 2014-04-23

Support/FAQ: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian
Last login: Wed Apr 23 20:21:14 2014 from 192.168.7.1
root@beaglebone:~#
```

4. Your BeagleBone should have automatically connected to the Internet via the Ethernet connection to your router, assuming that you run a typical setup. Confirm that your Internet access is functional on the board by downloading a web page using the program `curl`. You should see the HTML code for the webpage if you were successful, or the text "could not resolve host" if you were not. Use the following terminal command:

```
curl www.example.com
```

A terminal window titled "1. ssh" showing the output of the command "curl example.com". The output is an HTML document with a title "Example Domain" and some CSS styling. The terminal has a dark background and standard window controls at the top.

```
root@beaglebone:~# curl example.com
<!doctype html>
<html>
<head>
  <title>Example Domain</title>

  <meta charset="utf-8" />
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <style type="text/css">
    body {
      background-color: #f0f0f2;
      margin: 0;
      padding: 0;
      font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif
    ;

  }
  div {
    width: 600px;
    margin: 5em auto;
    padding: 50px;
    background-color: #fff;
    border-radius: 1em;
  }
}
```

If you're unable to connect, try the following commands in sequence to reset the Ethernet interface on the BBB. You should also double-check both ends of the Ethernet connection to make sure they're properly seated.

```
ifdown eth0
ifup eth0
```

5. Unlike most computers, the BBB does not come with a real-time clock (RTC), and thus cannot remember the date and time between boots. Why should we care about the time being correct? Any time we need to interface with an external web server over a secure connection (HTTPS), we use a certificate stored on the BBB's flash memory. If the system date is too far off the actual value, those certificates will not work. Type the following command to see what your BBB believes the date is:

```
date
```

We could add our own RTC, but we don't really need it since we can just get the time from the Internet every time the board boots up, since we're going to add WiFi. For now, let's set the system clock using the `ntpdate` command. Enter the following command, and then check the date again to confirm that it's correct (note that it will probably be expressed as UTC time – that's fine for our purposes).

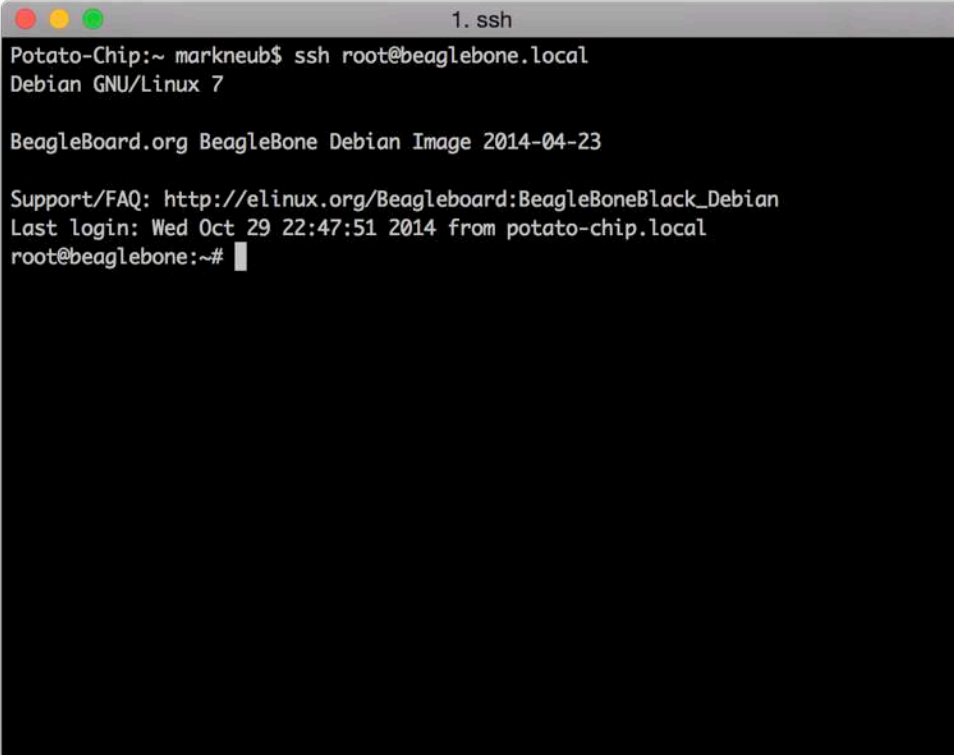
```
ntpdate pool.ntp.org
```

6. Update your software packages using the Linux software package manager `apt-get` by issuing the following commands in order. The first will make sure your local package lists are up to date. The second will download and apply any software updates to software packages that are out of

date. It will take a few minutes, but all the text scrolling by looks pretty cool.

```
apt-get update  
apt-get -y upgrade
```

7. Follow all of the instructions on this page to set up your WiFi adapter:
<https://learn.adafruit.com/setting-up-wifi-with-beaglebone-black/configuration>
 - a. *Note:* Be aware that when you get to the “WiFi Configuration” stage, you must make sure to attach the 5V 3A power supply to the barrel jack on the board rather than only supplying power over USB as we have been so far. The WiFi dongle draws more power than can be stably supplied over USB. You can have both the power supply attached for power and the USB attached to your computer for communication (and you’ll still have Ethernet attached at this point as well). Be aware that when you attach a power supply, the board will automatically boot up, so attach the WiFi dongle before attaching the power.
8. Once you have WiFi working, power off the board and remove the USB cable so the 5V adapter alone powers the board. Restart the board. The BBB will broadcast its hostname (beaglebone.local) to your network so you can connect to it without needing to know its IP address. This requires your computer to be on the same network as the BBB . You should be able to now ssh into the BeagleBone using
`ssh root@beaglebone.local`
without any physical connection. This is how you will connect to the BeagleBone for the remainder of the project.



```
1. ssh  
Potato-Chip:~ markneub$ ssh root@beaglebone.local  
Debian GNU/Linux 7  
  
BeagleBoard.org BeagleBone Debian Image 2014-04-23  
  
Support/FAQ: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian  
Last login: Wed Oct 29 22:47:51 2014 from potato-chip.local  
root@beaglebone:~#
```

Set up LEDscape on your BeagleBone Black

LEDscape is low-level driver software that runs on your BBB and allows you to write software to interface with your NeoPixels. Trammell Hudson at NYC Resistor originally developed LEDscape for the WS2811, an older version of the LED drivers we're using, but for this project we'll be using Yona Appletree's updated fork that supports the WS2812B drivers now used in Adafruit's NeoPixels: <https://github.com/Yona-Appletree/LEDscape>

1. Make sure you're SSH'ed into your BBB, then enter these commands in order, one line at a time, to download LEDscape and prepare your BBB for installation:

```
git clone git://github.com/Yona-Appletree/LEDscape
cd LEDscape
cp /boot/uboot/dtbs/am335x-boneblack.dtb{,.preledscape_bk}
cp am335x-boneblack.dtb /boot/uboot/dtbs/
modprobe uio_pruss
sudo sed -i 's/#optargs=capemgr.disable_partno=BB-BONELT-HDMI,BB-BONELT-HDMIN/optargs=capemgr.disable_partno=BB-BONELT-HDMI,BB-BONELT-HDMIN' /g
/boot/uboot/uEnv.txt
sudo sed -i 's/optargs=capemgr.disable_partno=BB-BONELT-HDMI,BB-BONELT-HDMIN,BB-BONE-EMMC-2G/#optargs=capemgr.disable_partno=BB-BONELT-HDMI,BB-BONELT-HDMIN,BB-BONE-EMMC-2G' /g /boot/uboot/uEnv.txt
reboot
```

2. After your BBB reboots, SSH back in and run these commands to compile the LEDscape source code:

```
cd LEDscape
make
```
3. When the code finishing compiling, you'll add LEDscape to run as a system service when your BBB turns on. Enter the following commands in order:

```
systemctl enable /root/LEDscape/ledscape.service
systemctl start ledscape.service
```
4. Modify the default LEDscape configuration file to tell the board to power one strip of 57 LEDs. We're only going to start with two, but eventually we'll have 57 of them when we have all 50 states and 7 additional ones for the temperature-to-color legend.
 - a. Enter the following command to modify the run-ledscape configuration file using the included text editor nano:

```
nano run-ledscape
```
 - b. Use your computer's arrow keys to go down to the last line and change it to read as follows.

```
./opc-server -p 7890 -c 57 -s 1
```



```
markneub - ssh - 96x26
GNU nano 2.2.6 File: run-ledscape
#
/bin/sh
DIRNAME="$(dirname "$0")"
cd "$DIRNAME"

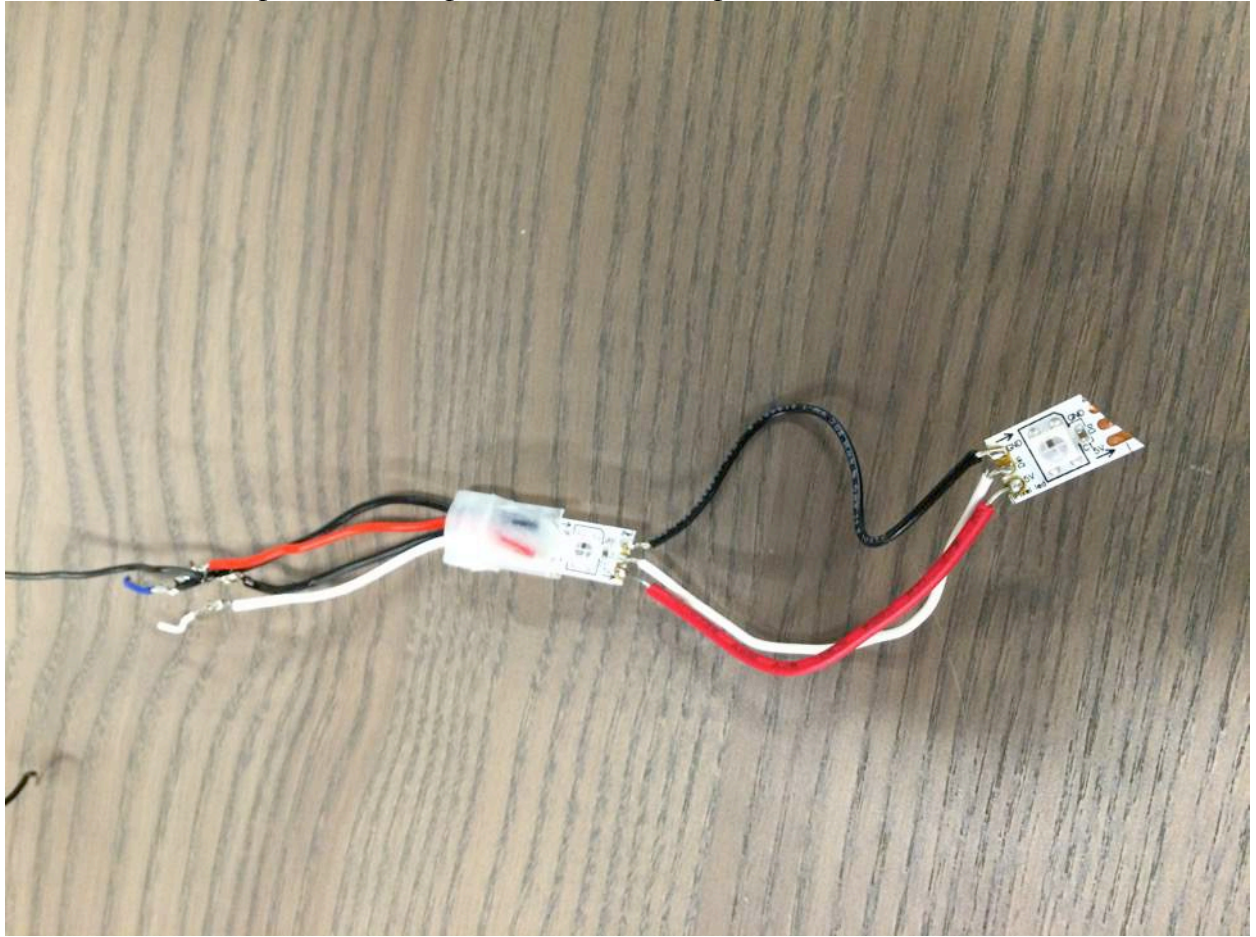
# Ensure that SSH is running. It really sucks when you (or systemd) screw up it doesn't start. $
# it's REALLY important that SSHd be running, since you have no other easy option to connect.
(sleep 30; /usr/bin/sshd -f /etc/ssh/sshd_config) &

modprobe uio_pruss
./opc-server -p 7890 -c 57 -s 1

[ Read 10 lines ]
^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is      ^V Next Page     ^U UnCut Text    ^T To Spell
```

- c. Type Ctrl-O to save and then press return to confirm overwriting the file, then type Ctrl-X to exit the nano text editor.
- d. This new configuration will take effect next time the service is restarted, which in our case will be the next time we boot the BBB
5. Power down the board by either pressing the PWR button on the board, or entering the following command into your SSH session:
`poweroff`
6. Now, let's prepare some NeoPixels for test! Remove two pixels by carefully cutting through the cutline that bisects the copper pads. Cut off and discard the plastic weatherproof sheathing. I went from testing with one NeoPixel to two, so I'd already cut the copper trace between the two and had to solder some wires between them, but you can just remove two NeoPixels from your reel. Note the arrows indicating the direction of data flow. Also, note the terrible job I did of cleaning bisecting the copper

trace on the second pixel (use sharp scissors or wire snips to do better!).



7. Connect a jumper wire from your first NeoPixel's data input to P9_22 on your BBB, NeoPixel PWR to BBB 3.3V, and GND to GND. Refer to the [pin diagram for the BBB](#). *Make sure you solder to the correct end of your NeoPixels!* There are little arrows on the NeoPixel strip that indicate the direction that data "flows." It doesn't matter which end the NeoPixels receive power and ground from, but it's easiest to attach them the same way you do your data.
 - a. Power up the BBB
 - b. By default, LEDscape runs a color wipe pattern on your pixels. You should see this happening now on your short NeoPixel string.
 - c. The BeagleBone outputs 3.3V logic signals so we power it with 3.3V as well – if the voltage difference between the control signal and the power signal are too great, the NeoPixels will not perform properly. Though the NeoPixels expect 5V, they'll still work at 3.3V, albeit a bit dimmer, and we probably wouldn't be able to connect too many pixels at 3.3V. Another reason not to connect more than two NeoPixels: even though we've already configured LEDscape to control a strip of 57 NeoPixels, connecting the entire strip would drawn an unsafe amount of current through the power regulator on the board (it's unsafe for the BBB, not unsafe for you). Later, we'll divide the power for the BBB and the power for the NeoPixels before it gets to the board, and we'll

use a logic level shifter to convert the 3.3V logic signal from the BBB to a 5V one.

Get familiar with the Cloud9 IDE and control your pixels from JavaScript

Cloud9 is a web-based integrated development environment (IDE) that comes pre-installed on your BBB. In non-programmer's terms: it's a fancy text editor that lets you get started with writing programs for your BBB with minimal effort.

1. Access Cloud9 by visiting <http://beaglebone.local:3000> in your web browser.
2. Make a new folder in the `cloud9` directory by clicking on the `cloud9` directory on the left side file browser, then going to the File menu in the IDE and selecting New Folder. Name the new folder `pixeltesting`.
3. Select the `pixeltesting` folder and create a new file by selecting File > New File. Name the file `pixeltest.js`. We are going to write our application in the Node.js environment, so our source code will be JavaScript files with a `.js` extension.
4. Double click on `pixeltest.js` to open the file for editing in the IDE.
5. We are going to install a Node.js library called `open-pixel-control` to make it easier for our Node.js app to communicate with the LEDscape driver we installed in the last section. You might think of `open-pixel-control` and LEDscape as bridges between your code and the NeoPixels. In programming parlance, these bridges are known as abstractions. Click on the tab in the bottom pane of the IDE titled 'bash – "beaglebone"' to activate it. This is a command line just like the one we've been using over SSH. It's not quite as responsive as a real SSH session but it's very convenient for small tasks we may need to do while writing our app. Type `cd pixeltesting` into the prompt and press enter to change the working directory to the folder we just created.
6. Type
`npm install open-pixel-control`
to install the `open-pixel-control` library into this folder. NPM is the node package manager, a program that makes it easy to install Node.js libraries from a central database or repository.
7. I've uploaded the code for our first testing program to an online text hosting service called pastebin. Visit this page and paste the program from RAW Paste Data into your `pixeltest.js` file in the IDE, then save the file in the IDE: <http://pastebin.com/zY3ENB9n>
8. To run the program, enter the following onto the command line tab in the IDE and press enter
`node pixeltest.js`
After a moment, your NeoPixels should stop presenting the random color wipe and you should see a solid white and a blue pixel. When you're ready to halt the program, type Ctrl-C to stop it, and it will go back to the color wipe test pattern. Our test program expects colors to be represented as an RGB triplet containing three values from 0-255 for the amount of red, green, and blue to emit, respectively. Try modifying the program with different RGB values for the two pixels, and then re-run it.

Finish the proof of concept

Now, let's change the color of our test pixels to reflect real world weather data. For this stage of the project, we're going to register a developer account with the Forecast.io API, which will allow us to programmatically download weather data. APIs, or application-programmer interfaces, are typically the preferred way to interface software with external data sources over the web. (An alternative is scraping, usually used when there is no API available, and involves

writing a program to download webpages and parse out the relevant information. That method is slow, tedious, prone to break as soon as the webpage changes, and is often against the site's terms of service. Avoid writing programs that scrape data from web pages.)

1. Register for a Forecast.io developer account at <https://developer.forecast.io/> (click "Register" at the top).
2. Install the Forecast.io Node.js library to make working with it dead simple
 - a. Go back to Cloud9 and enter the following into the bash prompt pane at the bottom:
`npm install forecast`
 - b. In my case, it took a few minutes to download and install the Forecast library and all of its dependencies, but you can continue while this runs in the background. Our next program will not run until npm finishes installing the library, though.
3. Create a new file in your `pixeltesting` folder called `weatherpixeltest.js` and open it in the editor by double clicking on it
4. Paste in this code: `http://pastebin.com/qcfZCCNJ`
5. Run the code again!
 - a. Try changing the GPS coordinates to visualize the weather at other locations. The GPS format that the Forecast.io API understands is WGS-84 decimal. An easy way to get coordinates in this format is to look up a location on Wikipedia, find its location (often labeled as "Coordinates" in the sidebar), click on that, and then copy the second line underneath WGS-84. It should be two decimal numbers separated by a space and a comma.

Laser cut the front and back pieces

Let's take a break from the electronics for a little while and turn our attention to building the map. For this part of the project, we'll be using a laser cutter and acrylic paint to create a map of the USA. I used the Trotec Flexx 300 at NextFab, which has a 75W CO₂ laser. If your laser setup is different, you'll need to adjust the power that you're cutting at. Either way, I strongly recommend running some tests on scrap acrylic of the same thickness to come up with suitable power levels for both cutting and engraving. The engraving does not need to be particularly deep (1/16" or so is fine; being exact doesn't matter), but I recommend you err a little bit on the side of going deeper because irregularities in your material or laser bed can cause the laser go in out of focus, decreasing the engraving depth.

1. Leaving on the contact paper that came on the 7328 white acrylic (we're going to use it as a mask for painting in the next section), load the sheet into your laser cutter and follow your typical process for initializing the machine. Load the vector file "map-file-pluslegend.ai" and send it to your laser cutter's software, such as Trotec JobControl. Here's what my laser settings looked like. Black is the engraving for the line art, red is a slightly deeper engraving that countersinks the screwheads on the front to help hide them, and the two blues are cut lines. Remember – test for your own setup! Even if you're using the exact same machine that I did, I recommend running some quick tests first.

Lenses get dirty, parts get replaced, etc., all changing the effective power of the laser.

Color	Process		Power	Speed	PPI/Hz	
1	Engrave CO2	▼	100.00	35.00	1000	PPI
2	Engrave CO2	▼	100.00	10.00	1000	PPI
3	Cut CO2	▼	100.00	0.30	1000	Hz
4	Cut CO2	▼	100.00	0.30	1000	Hz
5	Skip	▼	---	---	---	---
6	Skip	▼	---	---	---	---

- When the job is complete, you should have something that looks like this. Note that my photo is missing the temperature legend on the right side, because I added that later, but yours should have it.



- Now cut the back board from the clear acrylic sheet. This file is “backboard-final.ai.” For this file, cut out the red and blue lines with the same settings that you used for your cutlines above, and optionally use black to indicate engraving areas (use the larger of the

two engraving speeds you used before – 35 is the speed I used for general engraving). If you decide to engrave, make sure to flip your art or text horizontally as you can see in the provided file, to make sure it appears facing the correct direction in the final product. Here's how my backboard looked after I took it out of the laser cutter. Note that while I cut my backboard with the clear protective contact paper still attached, it shouldn't matter if you do that this time since we're not using it as a painting mask.



Paint the front board

Now let's paint the lines on the front board.

1. Using the black acrylic paint and a Q-tip (or your finger), fill in all of the etched areas where the contact paper was burned away by the laser. Make sure to work the paint in a little bit so that the lines are completely filled in (you don't want the paint to be sitting above the grooves as it will look patchy when you remove the mask). These photos should give you a good idea of what the process looks like.

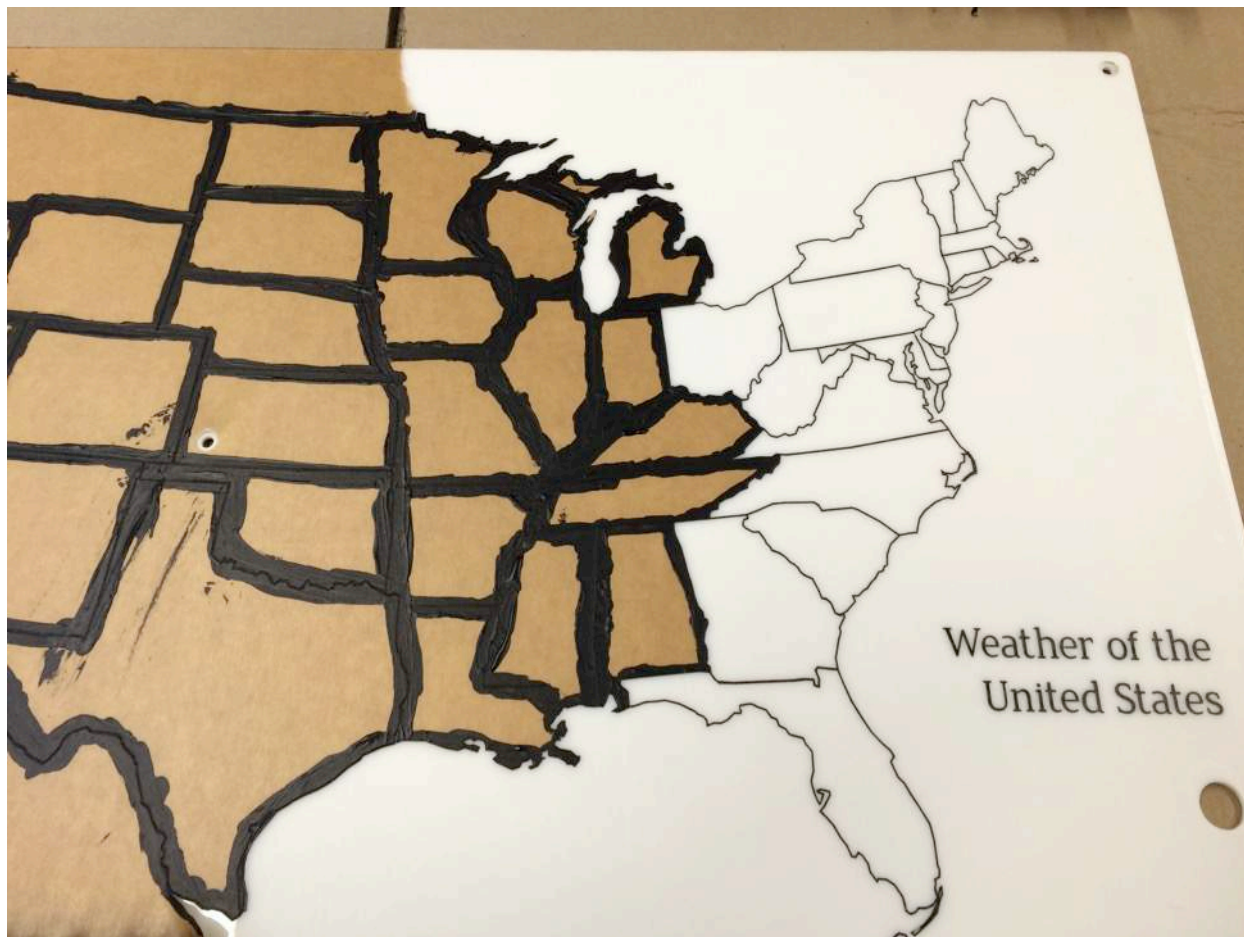


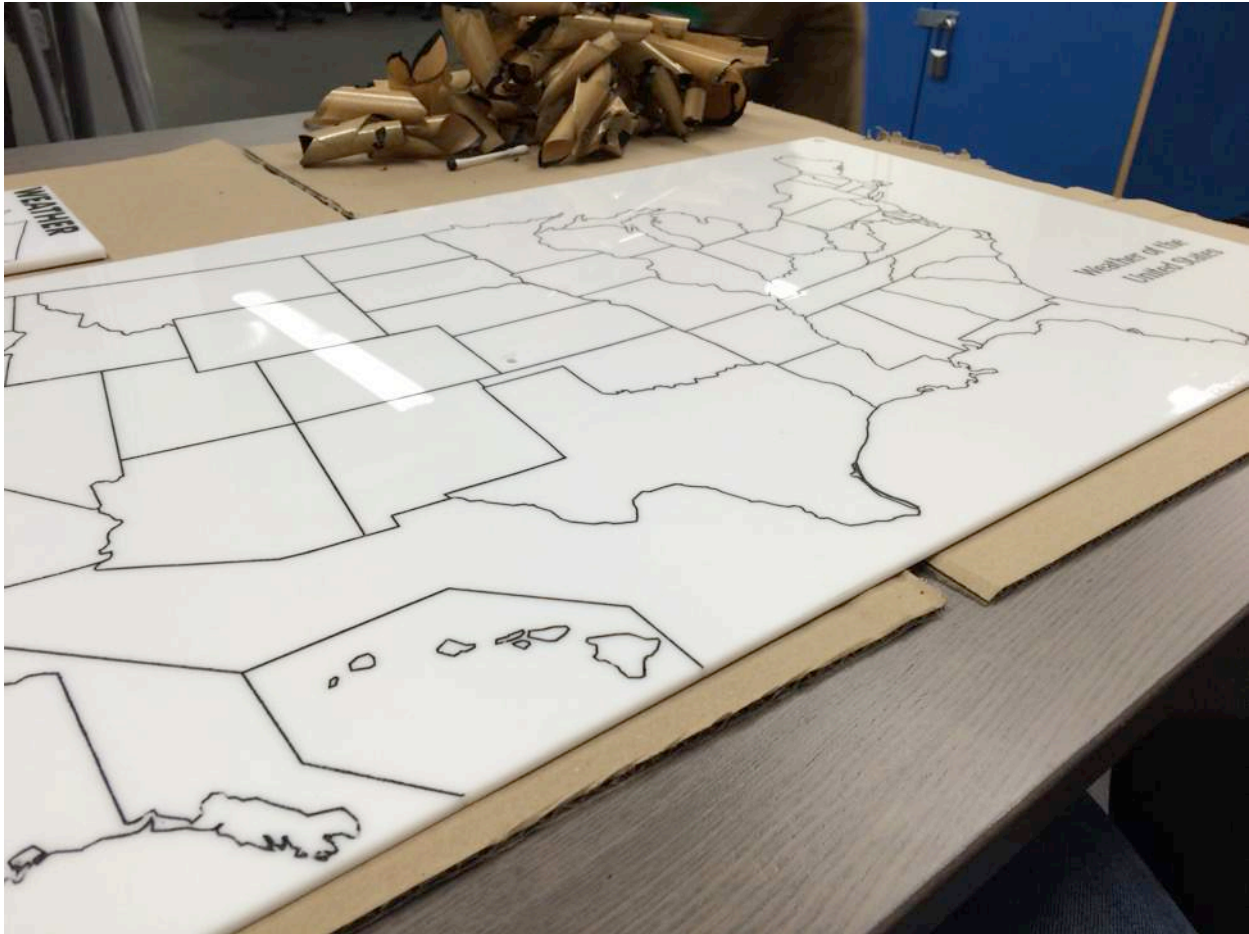




2. Once you're done filling in the lines, give the paint at least an hour to dry, then peel away all of the contact paper. If there are any areas that are not properly filled in, you can use a black fine-tip permanent marker to touch them up a little bit.







Wire the electronics

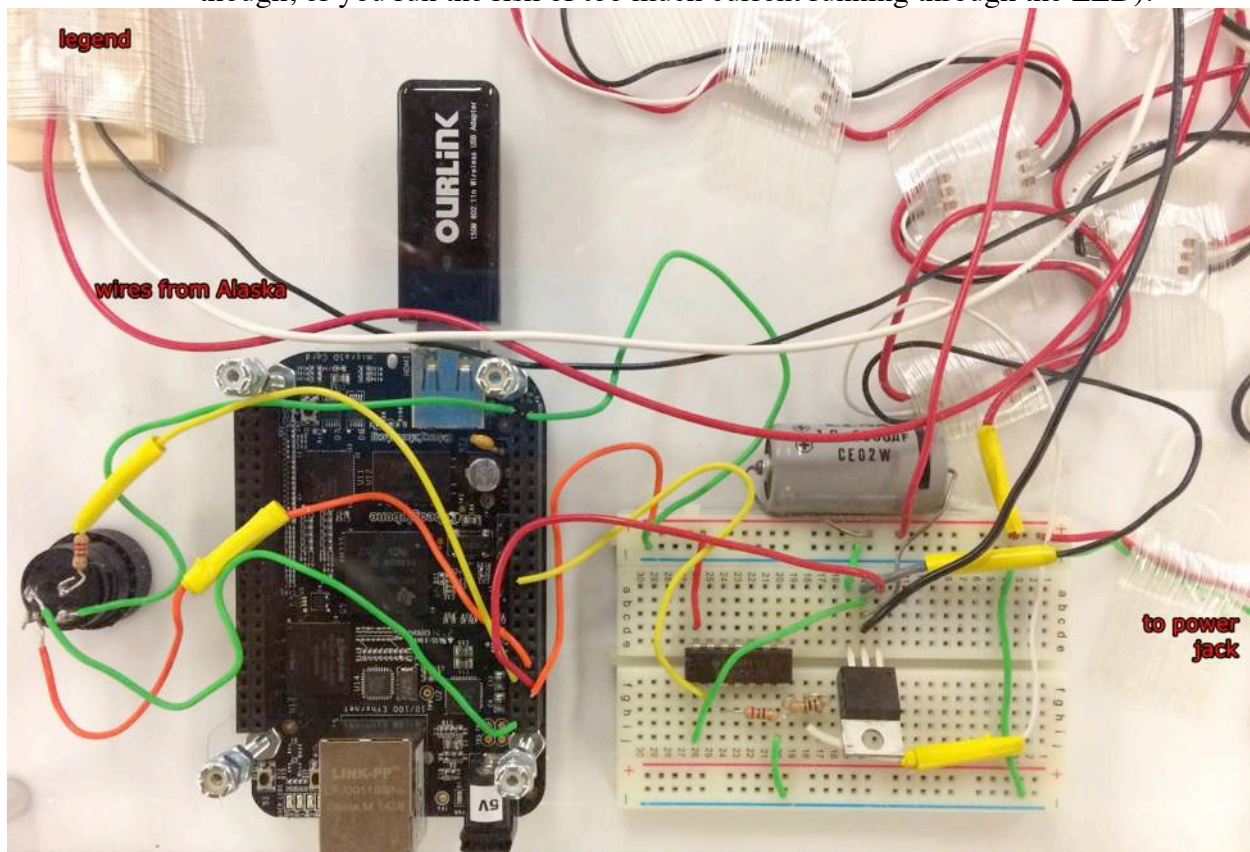
Before we solder the LED strand together, let's get the rest of our electronics sorted out. This way, we can power up the NeoPixel strip periodically during the construction of the NeoPixel strip to make sure that we don't have any broken connections. The basic idea for the electronics is that the 5V power adapter will provide current-separated power directly to the BBB and to the NeoPixels. The BBB provides 3.3V logic from its output pins, which won't be quite "loud" enough to consistently control the 5V-powered NeoPixels, so we'll use a logic level shifter to step up the 3.3V data signal from the BBB to 5V, and then run it into the first NeoPixel.

1. Wire up your project as per the schematic attached (beagleweatherschematic.fzz or beagleweatherschematic.png) and the photograph of my setup. To open the .fzz version of the the schematic (the source file), you'll need the open source software [Fritzing](#).

When wiring, pay special attention to the following items:

- a. There are four leads on your light-up power button. The LED leads have a + and a - symbol next to them (the jumper wires are yellow for + and green for - in my photo). Do not connect power to the leads without having a resistor on the line or you'll blow out your LED, like I did with my first one. The other two leads are for the power button and do not require any resistors.
- b. I recommend placing a drop of hot glue under your electrolytic capacitor to keep it from shifting around. You can use the adhesive backing on your breadboard to keep it from shifting.

- c. Unlike in my photo or the schematic, you won't actually be attaching your NeoPixels as a strand of lights connected by wires at this point, since you haven't created your strand yet (we'll do that in the next section). To test your wiring setup, you can attach the two test pixels you already made. Alternatively, you can add jumper wires to the "input" end of your reel and attach the remaining 58 pixels just to try it out. We can safely do this now because we're dividing the power before it goes to the BBB, so we won't burn out the power regulator on the board. We're also using the logic level shifter to move the 3.3V output from the BBB up to the 5V expected by the NeoPixels, which wasn't really necessary with just two NeoPixels. (The reason for this is that there's a small voltage drop created by each NeoPixel, which doesn't matter too much with only two, but with a high number of NeoPixels, we might see the voltage fall below the threshold necessary for consistent performance.)
- d. You might notice that I have two resistors where the schematic only shows one 330Ω resistor – mine add up to 330Ω in series. That one actually can actually be any value between 300 and 500Ω. The 220Ω resistor wired to the LED on the power button can actually also be between 220Ω and 1000Ω (not less than 220Ω though, or you run the risk of too much current running through the LED).



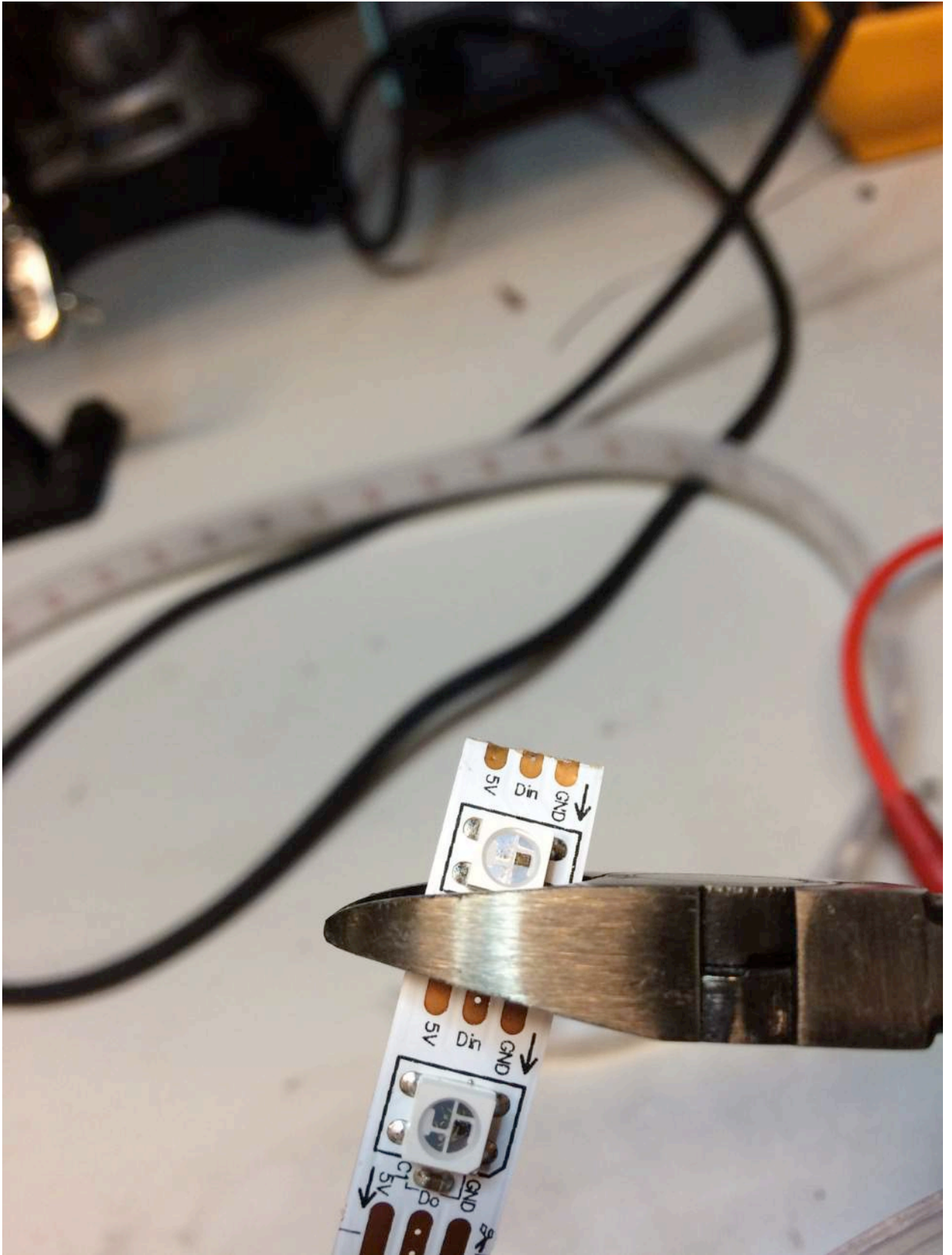
Solder the LED Strand

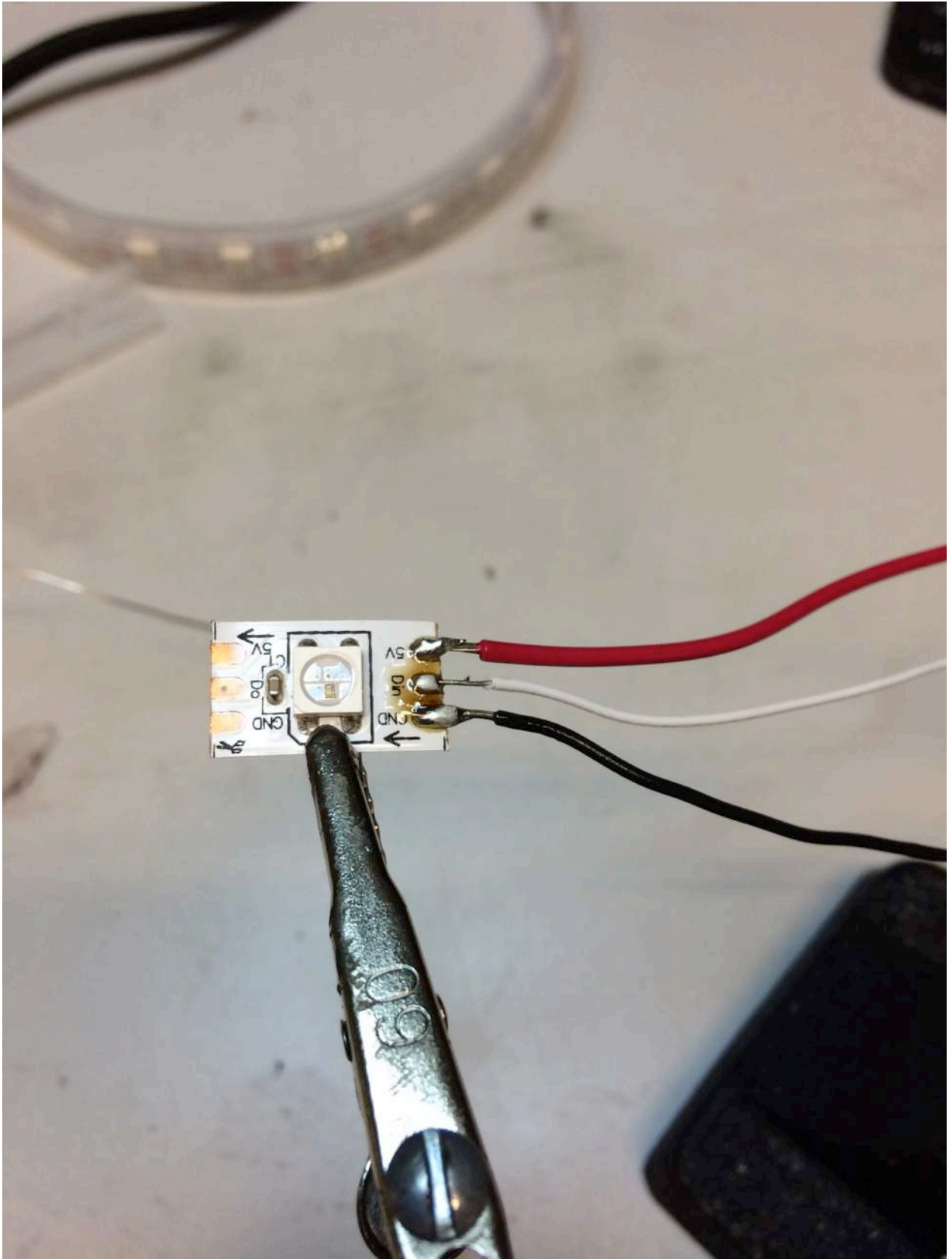
The next step is to separate the individual NeoPixels from their reel to create a strand of NeoPixels that will be run around the back of the board. This part requires a little bit of tedious work, but it's good practice in soldering and in learning US state capitals. You don't have to run your pixels in the same order I did, but you'll have to make a small update to the final code if

you don't. If you want to run your pixels in the same order that I did, use the following order (I hope you remember your state abbreviations!):

```
["FL", "GA", "SC", "NC", "VA", "MD", "DE", "NJ", "CT", "RI", "MA", "NH", "ME",  
"VT", "NY", "PA", "WV", "KY", "TN", "AL", "MS", "LA", "AR", "MO", "IA", "IL", "IN",  
"OH", "MI", "WI", "MN", "ND", "SD", "NE", "KS", "OK", "TX", "NM", "CO", "WY", "MT",  
"WA", "OR", "ID", "UT", "NV", "CA", "AZ", "HI", "AK"]
```

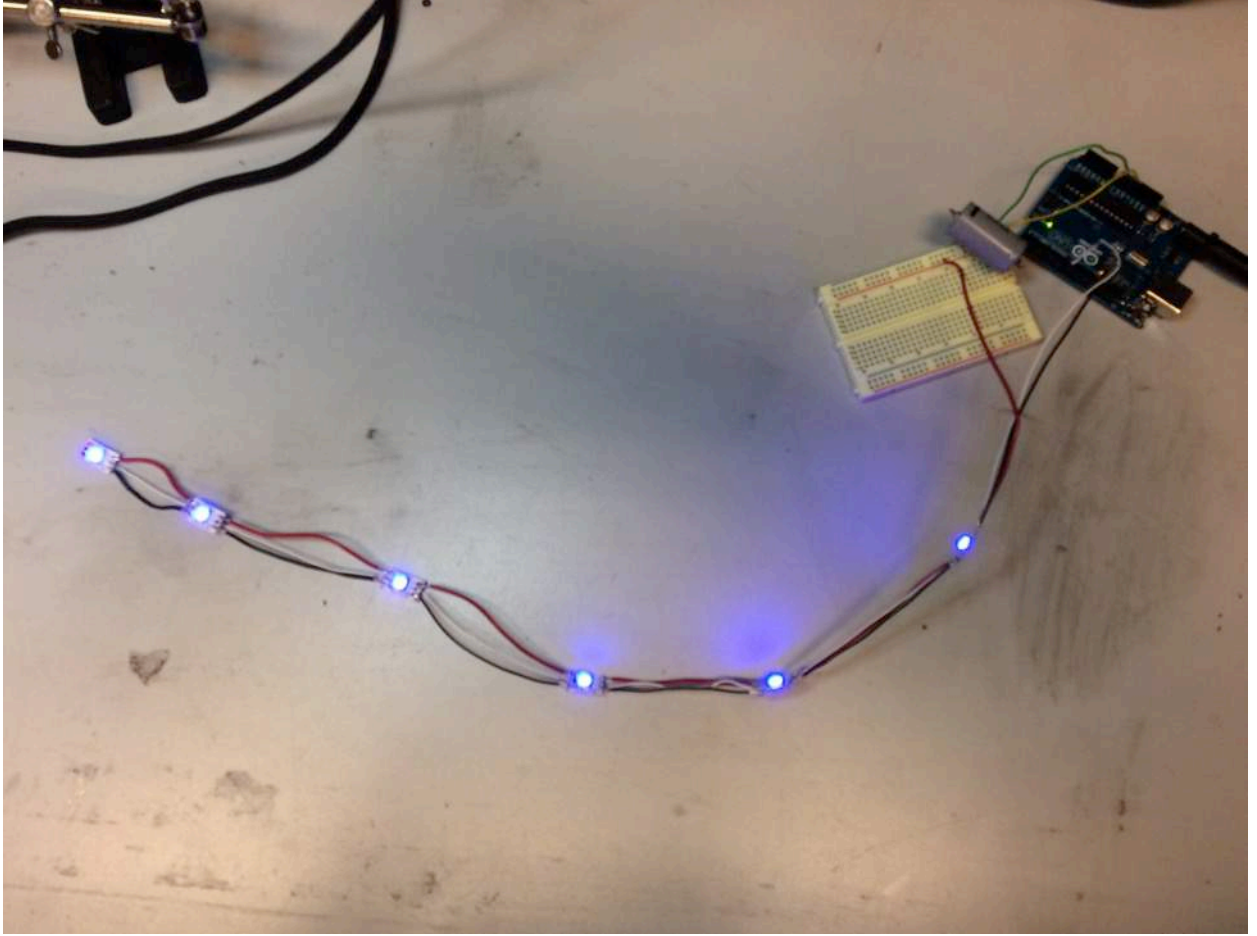
1. Snip off and solder 50 NeoPixels individually together with wire jumpers, making sure to be careful to run the data in the correct direction as indicated by the arrows on the NeoPixel strip. The distance of each wire connection should be at least as long as the distance on the map between the state capitals of the states that you're connecting. For example, the first two states are Florida and Georgia, so you'll need to figure out where Tallahassee and Atlanta are on the map and make sure that the wires will reach from one city to the next. When in doubt, go a little longer. I used red wire for power, black wire for ground, and white wire for data for consistency. If you have any questions about the best way to solder to the pads, it's best to seek out someone experienced with soldering and ask for advice, or you can look for help online.





You can periodically connect the strip to your BBB as per the schematic and power it up

to make sure you don't have any broken or flaky connections, using the default color wipe pattern provided by LEDScape (make sure to cleanly power your board on and off each time). Alternatively, if you have an Arduino handy, you can use Adafruit's NeoPixel library to easily test your NeoPixels. Finally, you can use an ohmmeter to check the resistance down from one end to the other of each of the lines (5V, data, and GND) to make sure there are no breaks. The line resistance should be only a few ohms on each, not infinite.





Add final code

Before we attach the NeoPixels to the back of the map, let's load our final version of the program code onto the BBB.

1. Log into your board's Cloud9 instance at <http://beaglebone.local:3000/>
2. Open the "autorun" folder in the file browser on the left side. BBB provides a built-in feature where any executable code placed in this directory will automatically be run when the board is powered up. It's a very handy feature for building an installation where the BBB will be run "headless" (without a monitor).
3. Using the bash prompt at the bottom of the screen, switch the working directory to autorun by typing `cd autorun`
4. Install the required NodeJS modules for the final code by using the following commands in order:

```
npm install forecast  
npm install winston  
npm install open-pixel-control
```
5. Copy and paste this linked code into a new file called `weathermap.js` and save it. If your NeoPixels are connected at this point, you should see them lighting up after a moment to show the current weather! Of course, things will be a little clearer when they're attached to the back of the map. <http://pastebin.com/xs2VHSfi>

6. Explaining all of the Node.JS code in the final program is a little bit outside of the scope of this guide, but I've done my best keep the code simple and well commented. Take a look around it and try to figure out how it works! Note that we've added a new Node.JS module called Winston, which will handle logging to a file. When we autorun the program, we have no way of seeing the debug messages in real time like we could with the `console.dir` statements, so instead we'll log to a file in the same directory called `weathermap.log`, which you can check after the fact for purposes of troubleshooting.
 - a. *Important note:* Take a look in the code at the variable `updateMinutes` (line 36), which is currently set to 20. This means that the colors on the map will update every 20 minutes in response to updated data provided by the Forecast.io API. Forecast.io provides 1000 free API calls per day. Since we're updating values for 50 state capitals, we can run this update 40 times, or for a little over 13 hours. If you attempt to make more than 1000 API calls in a day, your access is simply shut off and the board won't update any more. You can either make sure to run the board less than 13 hours per day (which is probably fine unless you want to leave it on all the time), change the `updateMinutes` variable to something larger (like 40), or enter a credit card and a small amount per day for the extra API calls. Forecast.io's pricing information is available on their developer website at <https://developer.forecast.io>.

Tape LEDs to board

Once you've created your NeoPixel strip, it's time to attach it to the board. I used a strong, clear tape I found in the electronics area at NextFab. Because the front board is slightly translucent, I recommend using clear tape, as black tape will be visible through the board if it's backlit. Clear tape will also obscure less of the wires and NeoPixels, which looks better when checking it out from behind. (I originally used black tape and then took it all off and replaced it with clear tape.)

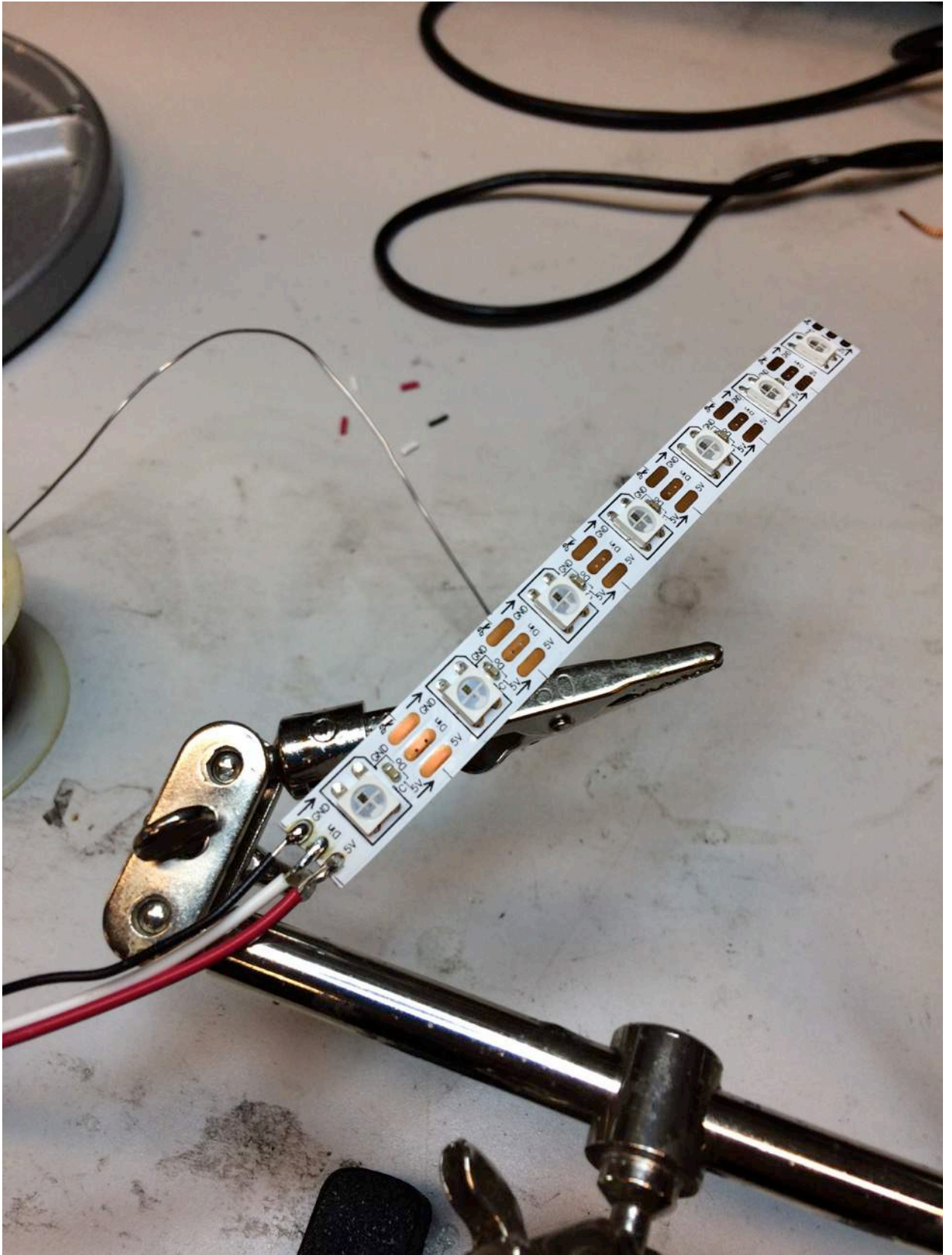
1. Locate the first capital (Tallahassee in my ordering) and tape the first pixel face down so that the light shines through the map where Tallahassee should be on the map. When I did this, I had my laptop open to Google Maps to see where the proper capital locations were. Of course, you'll need to either be running the software or the test pattern to make sure the pixel is lit up.
2. Repeat for the rest of the country.

Set up legend LEDs

You may have noticed that there are a few numbers running down the right side of the front of the map, above the power button. These are the labels for a temperature/color legend, so that viewers of your map can see what temperature each color displayed on your map represents.

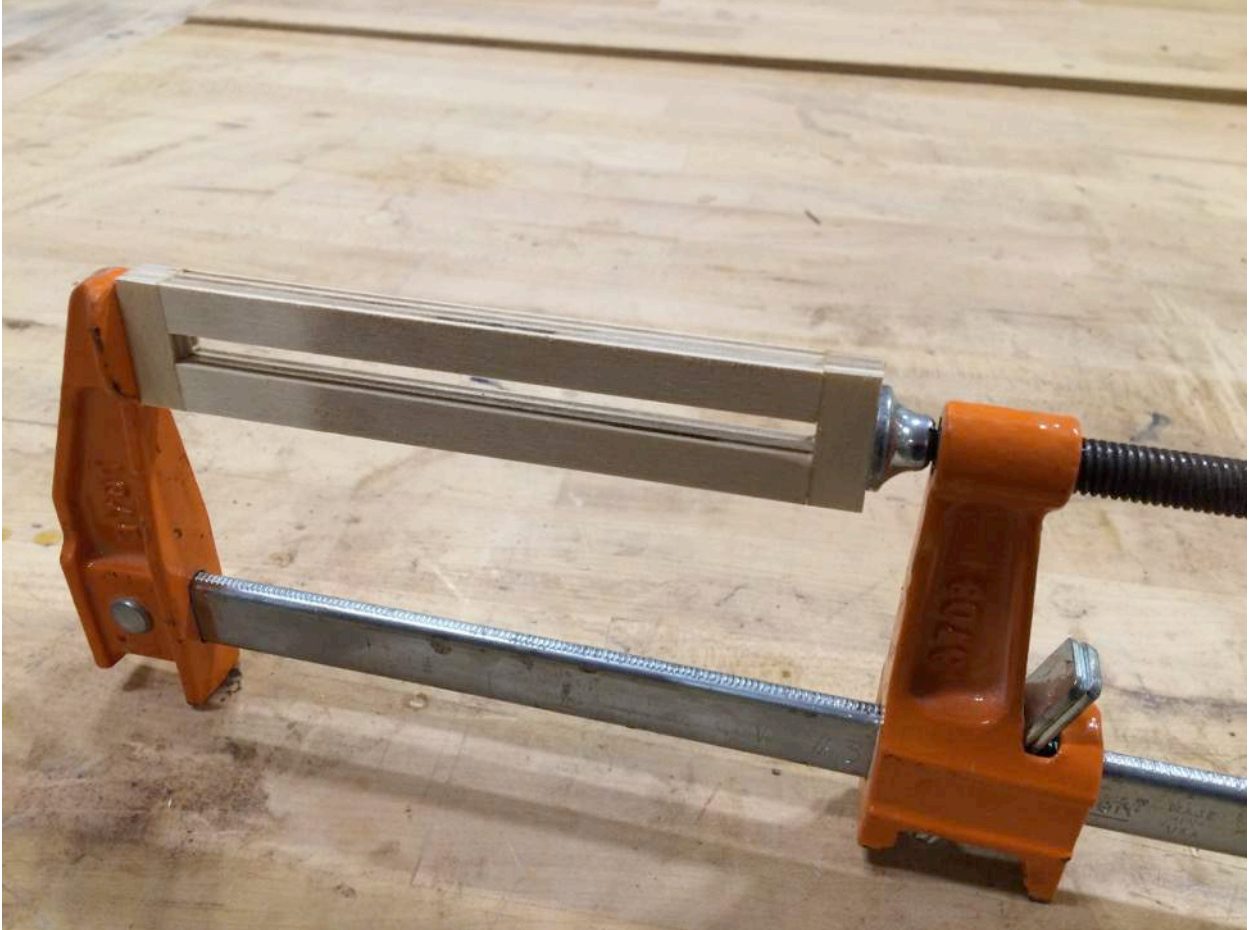
1. The legend is designed to be backlit by a strip of seven NeoPixels that are cut together from your reel, in contrast to snipping each one off the reel individually and soldering them together with jumper wires. Remove 7 LEDs from your reel.
2. Solder the "input" end of your 7 LED strip to the "output" of your last NeoPixel (Alaska, if you followed my ordering). Use long enough data, power, and ground wires to reach all the way back across the board from Alaska to where the legend goes.
 - a. *Note:* In case you're wondering why the legend pixels are awkwardly positioned after the eastern-most state, it's because I designed and added the legend after wiring all of the NeoPixels for the states.

- b. *Note 2:* I recommend that you also connect the power and ground lines from your legend pixels (not data) to the point on your breadboard where the power and ground go into the “input” of your first NeoPixel. The NeoPixels don’t care which end they get power and ground from, and if they’re too far from a power source they’ll start to look a little brown due to voltage drop. ([Read this article on Adafruit for more info](#)).



c.

3. Rather than taping the legend NeoPixels directly to the backboard as we did for each state, it's better to elevate them slightly off the backboard. This will reduce the individual pinpoints of light that was handy when denoting specific cities, and instead cause the lights to blur together a little bit and create a continuum effect, which is not only more visually pleasing but also more accurate to the way our software creates color gradients. I made a small wooden frame that holds the NeoPixel strip and elevates it a few millimeters off the back of the map – I recommend you make something similar.



4. When you restart your BBB, the legend strip should light up too since all of the code is there for it. With the lights on, position and tape down the legend over the numbers so that red is over 95+, yellow is over 75, blue is over 55, and white is over 32-. (This was

frustratingly hard to get a in-focus photograph of.)



Next steps

There are no more steps. Congratulations! But if you were going to keep going, what else can you think of to extend this project? I hope that in this tutorial I've given you a useful technique for downloading public data from the web, processing it, and visualizing it in the physical world. Using this basic technique, there's a lot of room for derivative and improved designs. For example, you could use this technique to focus on the weather for a specific state, or expand it to show the weather for the entire world. If you're showing the entire world, perhaps you could also vary the brightness of the LEDs to correspond with the current amount of daylight. You could use a flat projection (like we did in this project), or mount LEDs inside of a globe. Maybe you don't want to use weather at all; you could find an API to visualize other public data, like crime, public transit, voting results, etc.

Attribution

- Black US map (modified for laser cut-ability) used under license from Wikimedia Commons: http://commons.wikimedia.org/wiki/File:Blank_US_Map.svg
- Weather forecast data provided by Forecast.io
- Sponsored by NextFab