
Build Your Own Weather Station!

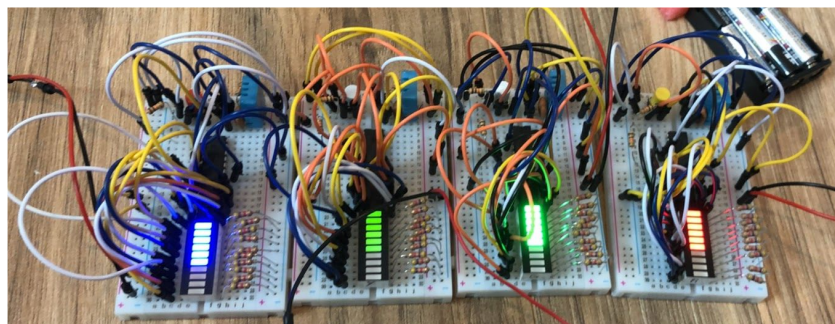
{The "Hello World" of Hardware}



Presented by Dahlia Dry

Table of Contents

1. Introduction	2
2. Background	3
a. Making Connections 101	3
b. Analog vs Digital	5
c. Polarized vs Unpolarized.....	5
3. Materials	5
4. Instructions	8
5. I'm Done! What Next?	17
a. Mode 1: Default	18
b. Mode 2: Temperature	18
c. Mode 3: Humidity	19
d. Exploratory Questions..	20
6. Built-In "Mini-Projects".....	20
a. Introduction	20
b. Project 1: Light as an Input	21
c. Project 2: Music = Science	23
7. Upgrading Your Weather Station	25
a. Introduction.....	25
b. Supplementary Materials.....	25
c. Instructions.....	26
d. How The System Works.....	28
e. Exploratory Questions.....	29
8. Want to Learn More?	30
9. Appendix A: Troubleshooting	30



1.Introduction

So, I've done a lot of talking and maybe? hopefully? haven't lost your interest just yet. We've learned about how a computational approach is revolutionizing the way we do science, and we've seen some cool examples of how we can use data analysis, simulation, and inference to solve real-world problems. Regrettably, I don't have the time to teach you the fundamentals of coding and prototyping and software-hardware integration. But we do have the time to explore the very tip of the iceberg and try our hand at the "Hello World" of hardware projects- the weather station. This was essentially the project that led me on my first science fair journey, and it's a great way to get exposure not only to coding, but also to sensor interfacing, circuit design, and methodology for data collection and analysis. The microcontrollers we'll be using are the same ones used in Arduinos, which are immensely useful and tiny computers often available online for less than \$30. I've pre-programmed our microcontrollers with code to communicate with the temperature/humidity sensor and send output signals to the LED strip so that the final product is a "bar graph" of sorts which displays the current temperature in increments of 5 degrees and humidity in increments of 10% . All the code I've written for today is available on my GitHub repository, <https://github.com/Dahlia-Dry/STEMinist-Movement-2018> , and is thoroughly commented out so that if you'd like to understand it for yourself, you should be able to learn quite a bit about how the hardware-software interface works in this particular design.

The same program that communicates with our weather sensor also allows you to repurpose it into two entirely different projects using the included "extra stuff" bag [see section 6] or go buy some stuff online and visualize your data on an LCD screen [see section 7] . Our goal here is to get you as excited about STEM as we are, so if you think tinkering with circuits and using computers to unlock the secrets of the universe might be for you, here you'll find plenty of ways to get started!

Questions? Comments? (Hopefully no concerns)? Contact me at dahlia.dry24@gmail.com

2. Background

Making Connections 101

You have in front of you a breadboard. It's the flat, white piece of plastic with a bunch of holes in it. It is your gateway to endless possibility. But don't get too excited- we must first learn how to wield this mighty tool, for with great power comes great responsibility.

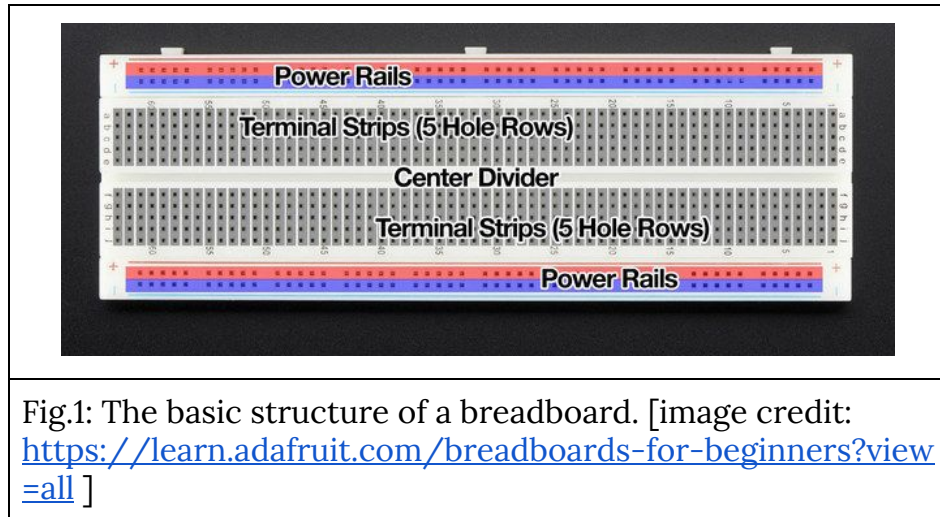


Fig.1: The basic structure of a breadboard. [image credit: <https://learn.adafruit.com/breadboards-for-beginners?view=all>]

A breadboard is composed of rows of metal clips that hold your parts and connect them to your circuit. Just as in math you use the x-y coordinate system to identify where things are, here we use the a-j rows and numbered columns (shown in fig. 1) to identify individual pin clips on the breadboard. As is shown in the image above, you connect your components to the terminal strips, and your components connect to your power source via the power rails. As it is referred to in the rest of this manual, the “negative terminal” references the blue strip highlighted in the image above, and the “positive terminal” references the red strip above. All the pins along the blue strips are connected, and all the pins along the red strips are connected, as is shown in the image below. The red strip will ultimately connect to the red wire of our AAA battery pack, and the blue strip will connect to the black wire of our AAA battery pack, providing power to the entire circuit. The two terminal strips are separated by a center divider, which means that the top and bottom sides of the breadboard are entirely disconnected from each other.

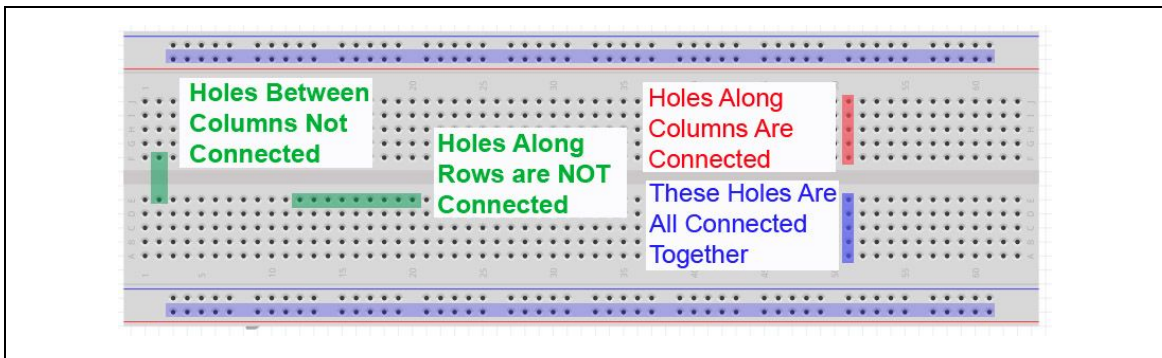


Fig. 2: What's connected and what's not. [image credit: <https://core-electronics.com.au/tutorials/how-to-use-breadboards.html>]

So, how do connections work? The image above shows what's connected in a breadboard and what's not. But why does this matter? Because, for example, if I'm trying to connect my Atmega to an LED, I can't place it like this.. (see fig. 3, left)- that won't do much of anything. Instead, I have to place it like this..(see fig.3 right)- the green shows what holes are affected by the placement of the LED, resistor, and atmega. In fig.3 right, I use the vertical connection of the terminal strips to connect a pin of the Atmega to a pin of the LED, and then use another vertical connection on a terminal strip to connect the other pin of the LED to a resistor, which then goes to the negative terminal. In fig.3 left, none of the parts are at all connected. For our purposes today, you don't have to have a mastery of circuit building on breadboards to do this project, but it does help to have some knowledge of what's going on.

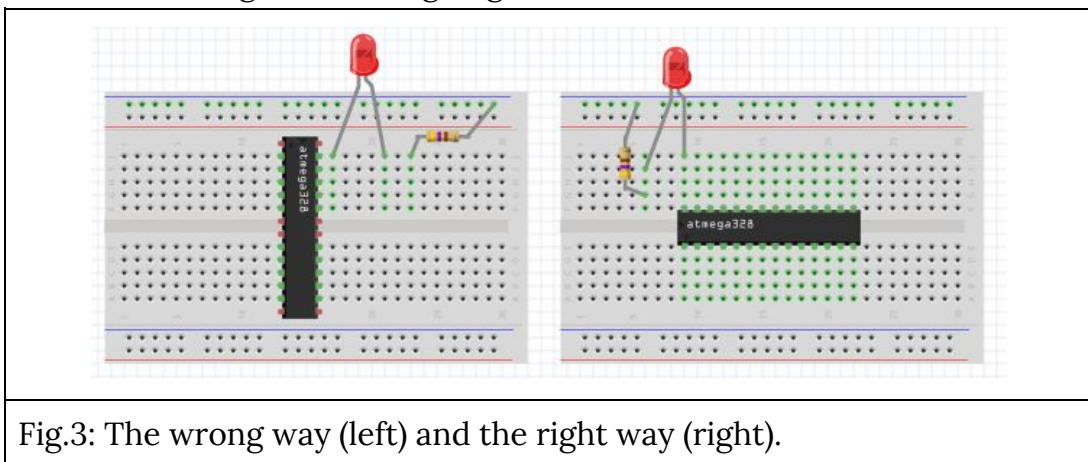


Fig.3: The wrong way (left) and the right way (right).

Analog vs Digital

Throughout this manual I'll talk about analog and digital inputs and outputs, since on the Atmega we have digital pins and analog pins for connecting to input/output devices. The difference between analog and digital comes down to voltage- digital pins can have a value of 0V or 5V, while analog pins can have a value ranging anywhere between 0V and 5V. If you're a math person, you'll call analog values continuous and digital values discrete. Digital input devices send the Atmega one of two values (HIGH or LOW), while analog input devices send the Atmega any value between 1 and 1023 (digital values converted from analog using an analog-digital converter).

Polarized vs Unpolarized

You'll also see references to things that are "polarized" and "unpolarized" throughout this manual. If something is polarized, current can only flow through it in one direction, and so it has a positive end and negative end indicating which direction the current is supposed to flow. For our purposes, that means that you have to make sure that you stick it in the breadboard a certain way, otherwise it won't do what it's supposed to do. The only thing that is polarized in our project is the LED "bar graph", so for everything else- resistors, jumper wires, the pushbutton, and the 16 MHz oscillator- you don't have to worry about which direction you plug it into the breadboard. Other things, like the DHT11 sensor and the Atmega, aren't "polarized" per se, but they do have specific pins with specific functions, so orientation matters for them, too.

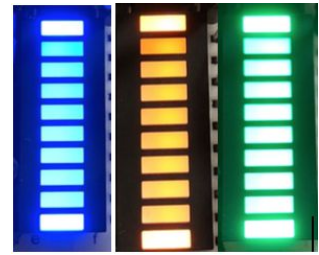
3. Materials

- **1: Half-Size breadboard** [fig.4]: Your blank canvas. The humble foundation on which to craft your masterpiece. Essentially, a bunch of metal clips to hold your wires. How to use it is explained in section 2.
- **1: Atmega 328P Microcontroller** [fig. 5]: The brains. It is your CPU (central processing unit) (where all calculations are made), your RAM (where all memory is stored), and your input/output handler. It runs one program and (hopefully) runs it well. This is essentially the definition of a microcontroller- it is a tiny computer specialized to perform a specific task. Our particular microcontroller is the same one used on the arduino, the hobbyist's favorite

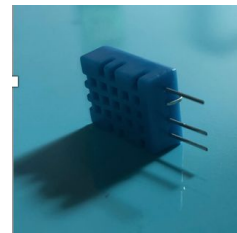


tiny computer, and it can perform a wide variety of functions, from controlling robots to running weather stations.

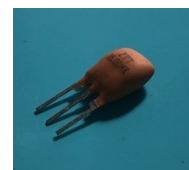
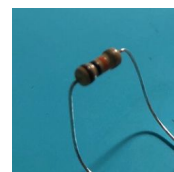
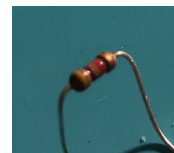
- **1: LED “Bar Graph”** [fig.6]: The primary output device. It’s essentially 10 individual LEDs stuck together. All LEDs are polarized, which, as you learned above, means they have a positive end and a negative end. LED stands for light emitting diode- in other words, an LED is a device that manipulates electrons to produce an effect called electroluminescence, which is the emission of photons (particles of light) under an applied current.



- **1: DHT11 Temperature/Humidity Sensor** [fig.7]: It’s all in the name- this device measures the temperature and humidity of the air around it. Temperature is measured using a type of a material called a semiconductor which kind-of conducts electricity and kind-of doesn’t. This kind-of-not-really property is useful because as temperature increases, the semiconductor becomes more electricity-friendly (in other words, its resistance decreases). Using the Steinhart-hart equation, we can calculate temperature from the resistance value of the semiconductor, which when used in this application is called a Negative Temperature Coefficient (NTC) thermistor. The DHT11 measures humidity in a similar way, instead using a moisture-dependent rather than temperature-dependent resistor. This moisture-dependent resistor consists of a moisture-holding material sandwiched between two thin pieces of metal (electrodes). When the air is more humid, more moisture is held in the moisture-holding material of the resistor, and so its resistance value changes. We can track this change to calculate the humidity of the air.

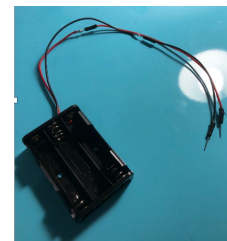


- **10: 4.7 kΩ Resistors** (band pattern = yellow-purple-red-gold) [fig.8]: A resistor, as touched on above, is any material through which electricity cannot easily pass. It is used to moderate the voltage and current in a circuit, through the relationship $Voltage = (current) \times (resistance)$, Ohm’s Law. Resistance is measured in Ohms (Ω), so 4.7 kΩ is the equivalent of 4700 ohms.
- **3: 10 kΩ Resistors** (band pattern = brown-black-orange-gold) [fig.9]: Explained above. 10 kΩ is the equivalent of 10000 ohms.
- **1: 16 MHz ceramic oscillator** [fig.10]: Also referred to as a ceramic resonator. It consists of a piece of ceramic material connected to three pieces of metal (referred to as electrodes).



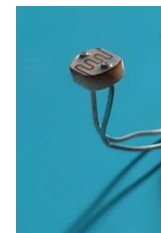
When an electric current is applied to it, it vibrates to produce a signal at a specific frequency (in our case, 16 MHz), which can be used to help a computer such as the Atmega tell time.

- **1: Pushbutton** [fig.11]: Essentially a switch. It can connect and disconnect parts of the circuit, in our case sending a digital signal to the Atmega that tells it what mode to run in.
- **25: Jumper wires** [fig. 12]: You only need 24, but there's an extra in case you lose one. Used to build circuits by connecting different parts through the breadboard. Like a brick wall without cement, your circuit is just a bunch of separate parts without jumper wires.
- **3: AAA Batteries** [fig. 13]: You've seen these before. But do you know how they work? Batteries supply voltage to the circuit through essentially using chemical reactions to make electrons build up in the anode (negative end). Electrons all have the same charge, so they repel each other and are attracted to the cathode of the battery (positive end). The electrolyte (fluid) in the middle of the battery stops them from going straight from the anode to the cathode, so the electrons travel around the circuit to get to the cathode instead, creating a flow of electricity. We are using 3 AAA batteries, each of which is capable of producing 1.5 volts of electricity. Therefore, the voltage of our circuit is $3 \times 1.5 = 4.5$ volts, although it's more like 4.7 volts when the batteries are fully charged. The Atmega runs on 5 volts, so this is close enough to power our circuit.
- **1: AAA Battery holder** [fig. 14]: This connects our three AAA batteries in series, meaning that they are connected anode to cathode so that their voltage adds, and we end up with 4.5 volts of electricity rather than 1.5 volts of electricity.

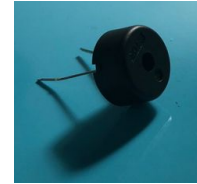


Extra Stuff (See Section 6):

- **1: Photoresistor** [fig. 15]: This is a resistor (see above) whose resistance value changes based on the amount of photons (light particles) that hit it at any given point in time. It consists of a semiconductor (explained above) which, when it absorbs more photons, “releases” electrons, making it more “electricity-friendly” and thus lowering the resistance of the resistor itself.



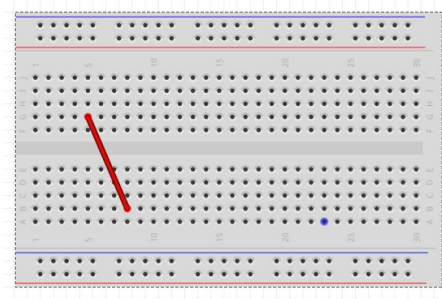
- **1: Piezoelectric Buzzer** [fig. 16]: A thing that produces sound waves of varying frequencies. Piezoelectricity is the ability of a material to either produce electric current when it vibrates or vibrate when subjected to electric current. Our ceramic oscillator vibrates when subjected to electric current, and so does the piezoelectric buzzer, except in the audible range instead of at 16 MHz. By applying varying voltage to the piezoelectric buzzer, you can make it vibrate at varying frequencies, producing sound waves at different pitches.
- **2: Jumper Wires** [fig. 12]: Explained above.
- **2: 10 k Ω Resistors** (band pattern = brown-black-orange-gold)[fig. 9]: Explained above.



4. Instructions

1. Run a quick inventory of your supplies. Something listed in the materials list that you don't have? Let someone know so you can get started. Briefly familiarize yourself with the name of each part and its function as explained in the materials list. Even if you choose not to read every part description in section 3, it's important that you at least skim section 2 and understand some of the idea behind making connections using the breadboard. If you're still a bit confused, ask myself or a volunteer and we'll be happy to explain further.
- 2.

Practice round

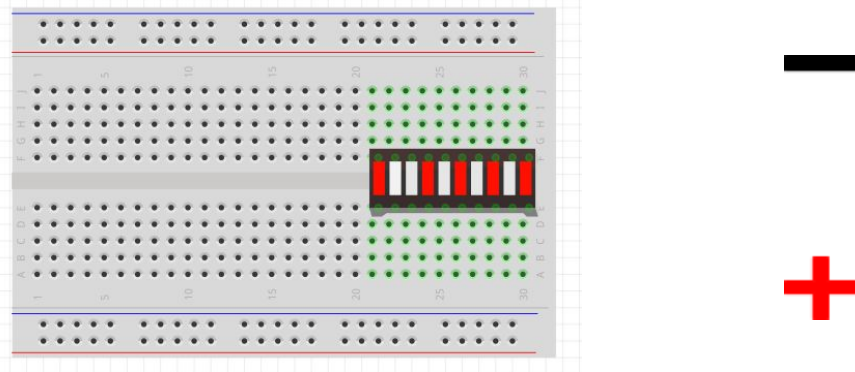


Let's look at the breadboard more closely. Notice, as explained in section 2, that rows are labeled a-j and columns are labeled 1-30. This will be important when we're placing and connecting all the parts. You can choose to place your jumper wires according to the diagrams included with each step or using the given coordinates of pin slots- for example, try locating slot b8 on the breadboard. What if I ask you to make the following connection using a jumper wire: {b8->g5}? You should have something

looking like the diagram above (don't leave this wire in, we haven't started quite yet!). As explained in the background section, you can use jumper wires any way you like, since they have no specified orientation, and color doesn't matter. The colors chosen in the diagrams below are just for clarity, your wires aren't supposed to match them.

3.

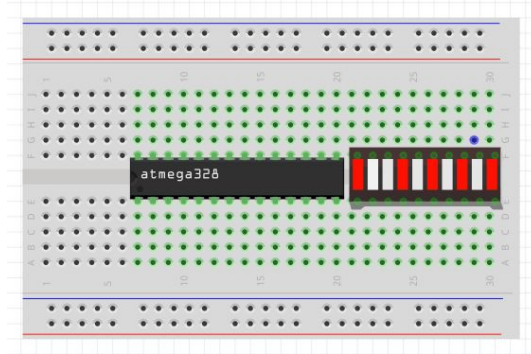
LED "Bar Graph": e,f 21-30, "+" side facing you.



Now that we understand the basics of making connections, we can begin building. The most challenging aspect of this project is the limited space that we're working on- pretty much every slot on this breadboard is going to be occupied by something. As the board gets more crowded, it'll get tricky to see rows and columns and place wires, so keep this in mind as you go. A good practice is to double check that you've placed each jumper wire in the right place after you place it, not after you've made all your connections and realize something's wrong. To start, take your LED "bar graph" and place it at the bottom of the board in the center. Be careful- it looks the same from either end but, as you read, LED's are polarized, meaning they have a positive side and a negative side. There should be a "+" and "-" drawn on the left and right sides of the LED strip. Place it so that, as in the image above, the "-" faces "up" and the "+" faces "down". It should take up columns e and f, rows 21-30.

4.

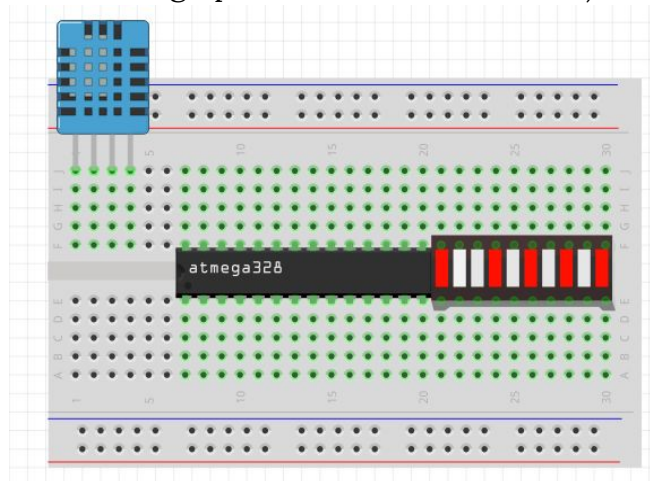
Atmega: e,f 7-20, semicircle indent to the left



Next, place the Atmega. Be careful- It has a little semicircle indentation on one end. Place the Atmega in columns e-f, rows 7-20, so that the semicircle is at the “left” in row 7, as shown above.

5.

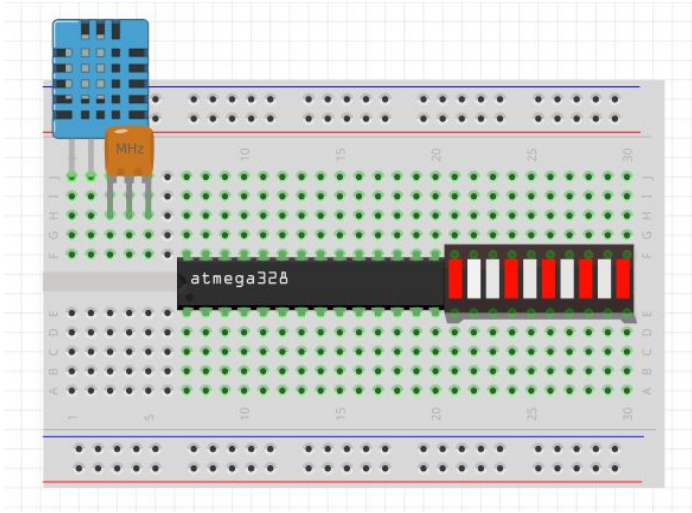
DHT11: j1, j2, j4 (3rd pin should be bent up and away from breadboard, although picture doesn't show this)



Place the DHT11 temperature/humidity sensor. The textured side should face the Atmega, and it should occupy column j, rows 1,2, and 4. Notice that I skipped row 3- the 3rd pin on the DHT11 is unused, so it should be bent up and NOT inserted into j3. The third pin of the DHT11 is connected in the image above but should already be bent upward and away from the board on your sensor.

6.

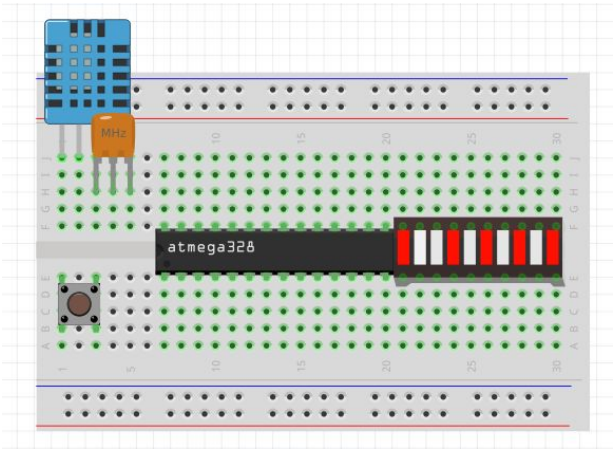
16 MHz Oscillator: h3,h4,h5



Place the 16 MHz ceramic oscillator. As explained in Section 2, the oscillator is not polarized- you can insert it into h3,h4, and h5 facing either direction.

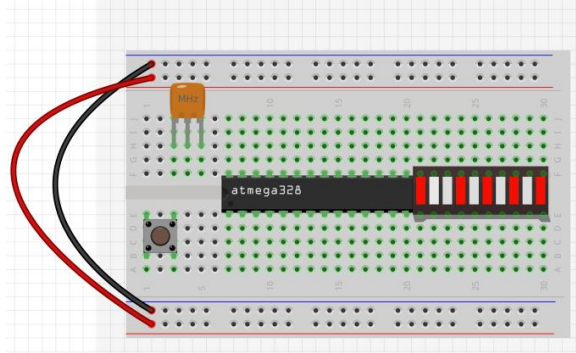
7.

Pushbutton: b,b3,e1,e3



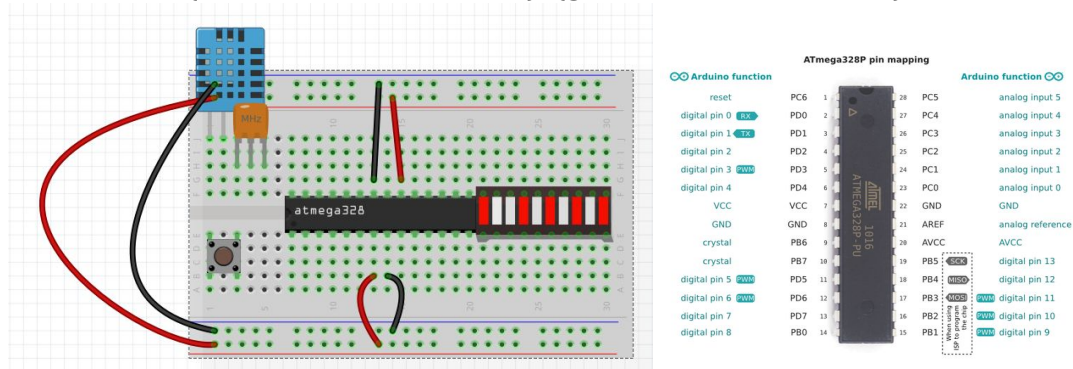
Place the pushbutton. This is also unpolarized, but it only fits into the breadboard one way. See how its prongs point kind of inward? Orient the pushbutton so that, facing you, the prongs point forward and backward instead of left and right, then insert it into b1, e1, b3, and e3. You should also be able to tell which way it goes by trying it in both orientations and seeing which way makes the prongs line up with the holes in the breadboard.

Jumper Wires: {negative ->negative}, {positive ->positive}



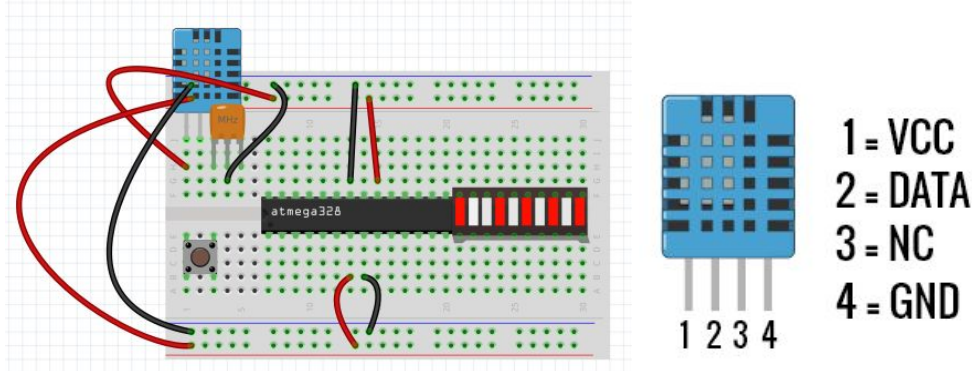
Now that we've placed all the major components, we're going to start wiring up the Atmega so that it gets the power it needs and can run the program using the 16MHz oscillator as its clock. Good practice when working with breadboards is to start by connecting the two power rails so that both sides of the board share the power supply. Simply run one wire from blue end to blue end and one wire from red end to red end. The DHT11 has been removed so that you can see the power rail behind it. From this point forward, black wires will show connections to the negative (blue) terminal, and red wires will show connections to the positive (red) terminal.

Jumper wires: {b14->negative terminal}, {g13->negative terminal},
{b13->positive terminal}, {g15->positive terminal}



10.

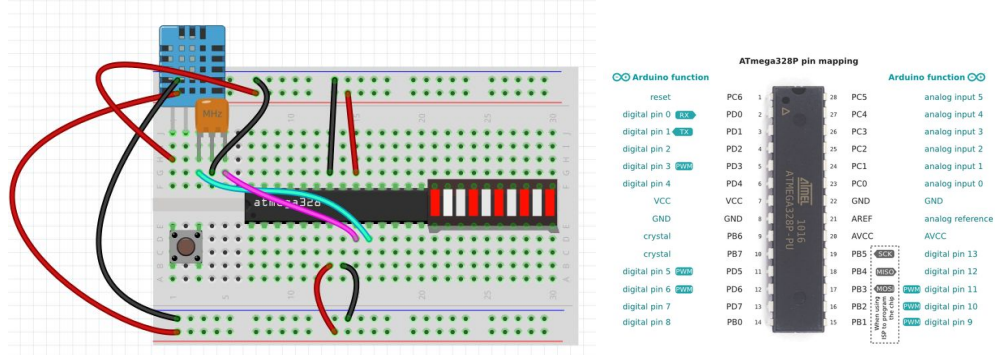
Jumper wires: {h1 ->positive terminal}, {f4 ->negative terminal}



Next, we'll connect the DHT11 sensor to the power rails. The 16 MHz oscillator shares row 4 with the DHT11, so both will share a connection to the negative terminal. As is shown above, pin 1 of the sensor goes to VCC (positive terminal), and pin 4 will connect both the sensor and the 16 MHz crystal to GND (negative terminal).

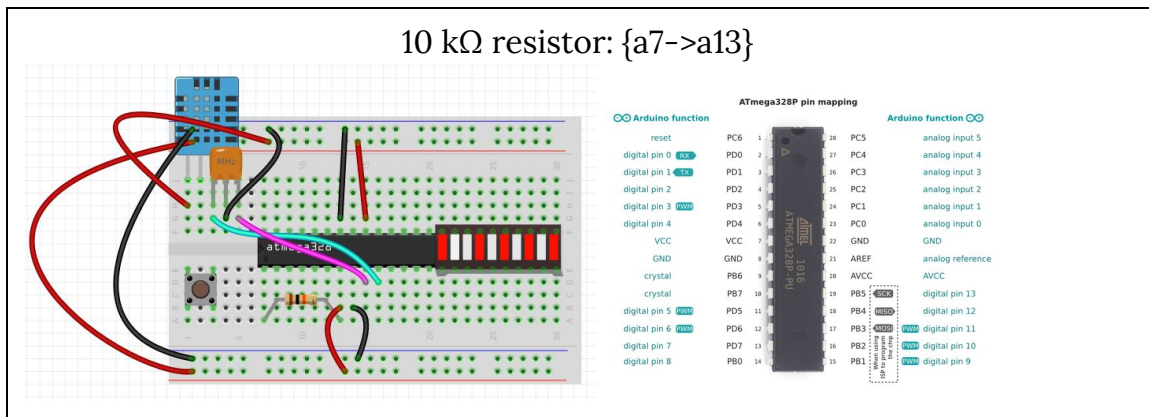
11.

Jumper Wires: {g3 ->d16}, {g5 ->d15}



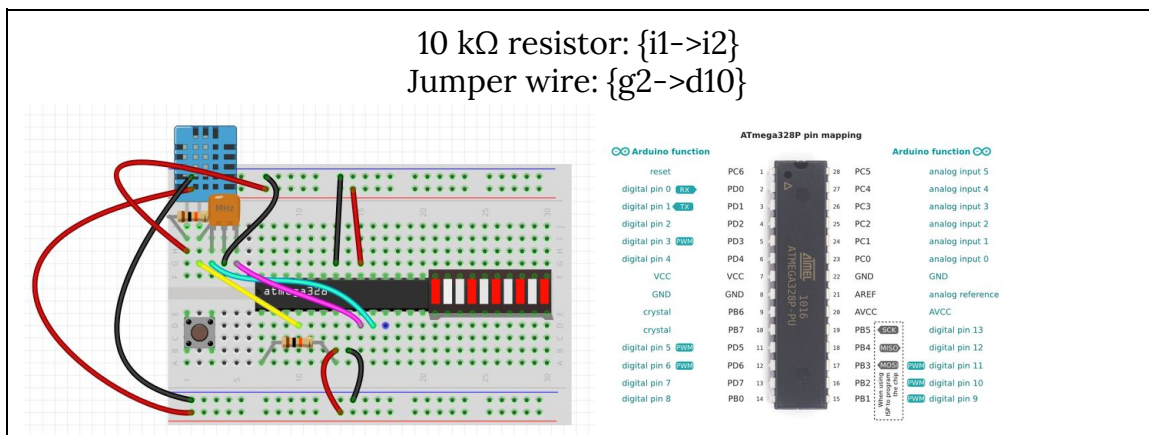
Now, connect the 16 MHz oscillator to the Atmega. Without the oscillator, the Atmega has no way to tell time. As is shown in the Atmega pin diagram, pins 9 and 10 go to “crystal”, which is another way of saying “time-telling device”, AKA our oscillator. Use jumper wires to make the connections shown above.

12.



To finish setting up the Atmega, we need to control its “reset” button. As shown in the pinout, pin 1 is the “reset” pin of the Atmega, which triggers a complete reboot if no voltage is supplied. In order to prevent the Atmega from resetting during normal operation, we need to use a 10 kΩ resistor to connect the “reset” pin to the positive terminal, or VCC. This means we need to connect pin 1 and pin 7 of the Atmega. On the breadboard, use the resistor to make the connection shown above.

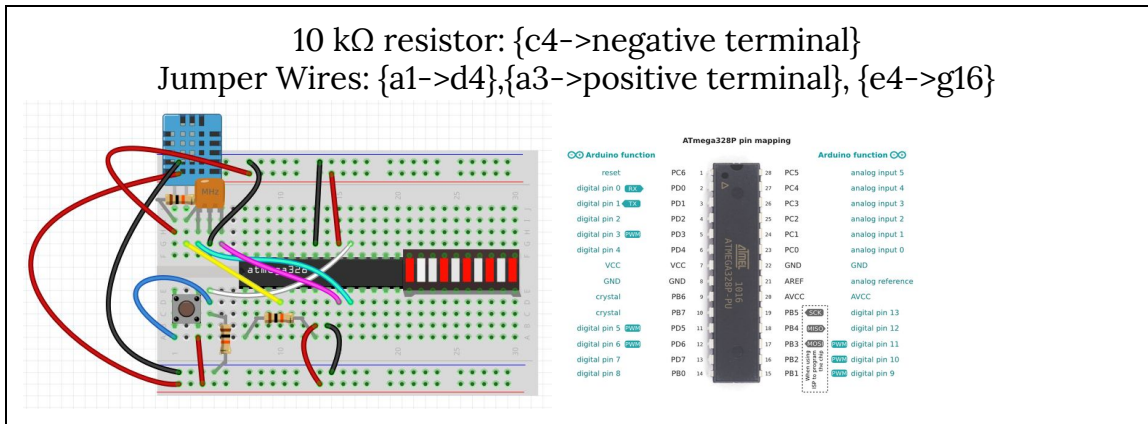
13.



Now that the Atmega is ready to execute code, we need to connect it to our inputs and outputs. Our inputs are the pushbutton and the DHT11, and our output is the LED “bar graph”. The pushbutton is used to tell the Atmega what mode to execute, and the DHT11 feeds the Atmega temperature and humidity data, all of which is expressed on the LED “bar graph”. First, we’ll connect the Atmega to the DHT11 using a 10 kΩ resistor (brown-black-orange-gold) and jumper wire. The DHT11 provides digital input rather than analog input (see Section 3), so we connect it to one of the Atmega’s digital input pins- here, we will use digital pin 2, which is pin 4 on the Atmega itself. Use the resistor and jumper wire to make the

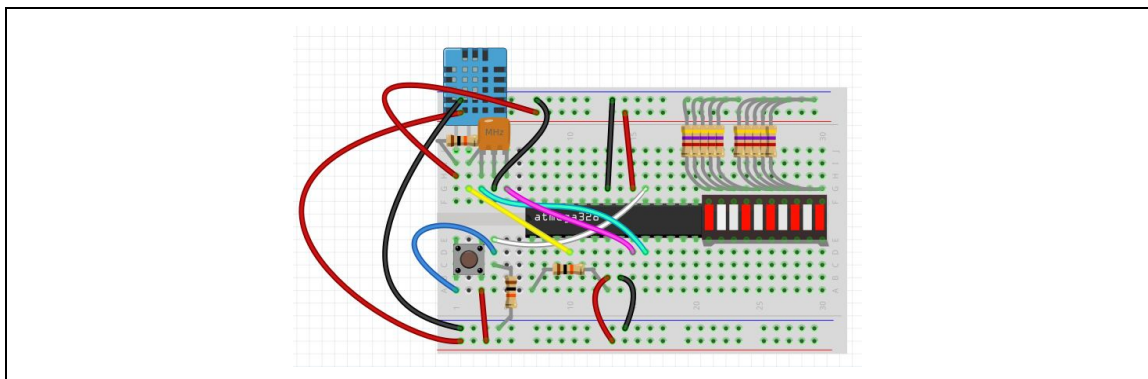
connections shown above.

14.



Next, let's connect our other input- the pushbutton. Like the DHT11, the pushbutton also provides digital input, so we'll connect it to pin 19 of the Atmega, AKA digital pin 13 (we're saving digital pins 3-12 for the LED strip). The pushbutton also needs to connect to power, so make the connections shown above.

15.



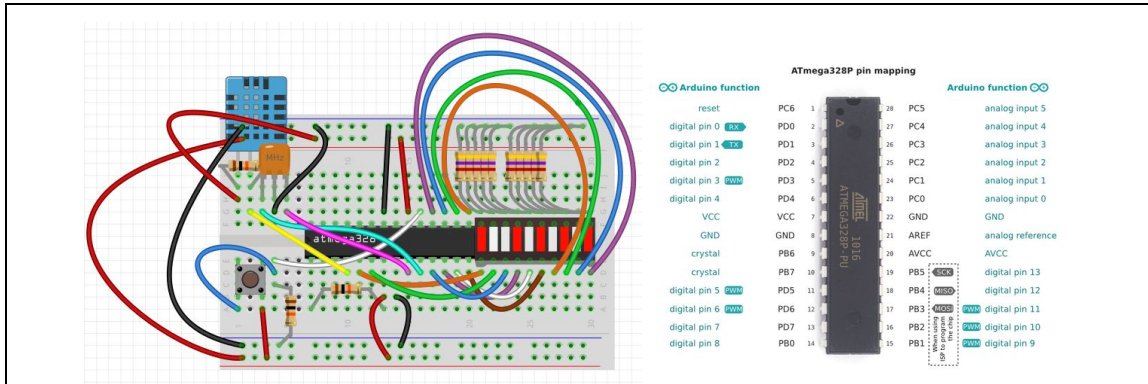
Almost done! We've got the Atmega ready to go, and now it can communicate with its inputs to execute code. All that remains is to connect the output- the LED "bar graph". The positive side of the LED strip will connect to digital pins 3-13, which correspond to pins 5,6, 11-18 on the Atmega. The negative side of the LED strip will be connected to the negative terminal using 4.7 k Ω resistors (yellow-purple-red-gold). Things are going to get crowded quick, so take your time. First, we'll connect the negative side of the LED strip to the negative terminal on the breadboard. Use the resistors to make the following connections.

4.7 k Ω resistors:

{g21->negative terminal} {g22->negative terminal}
 {g23->negative terminal} {g24->negative terminal}
 {g25->negative terminal} {g26->negative terminal}

{g27->negative terminal} {g28->negative terminal}
{g29->negative terminal} {g30->negative terminal}

16.

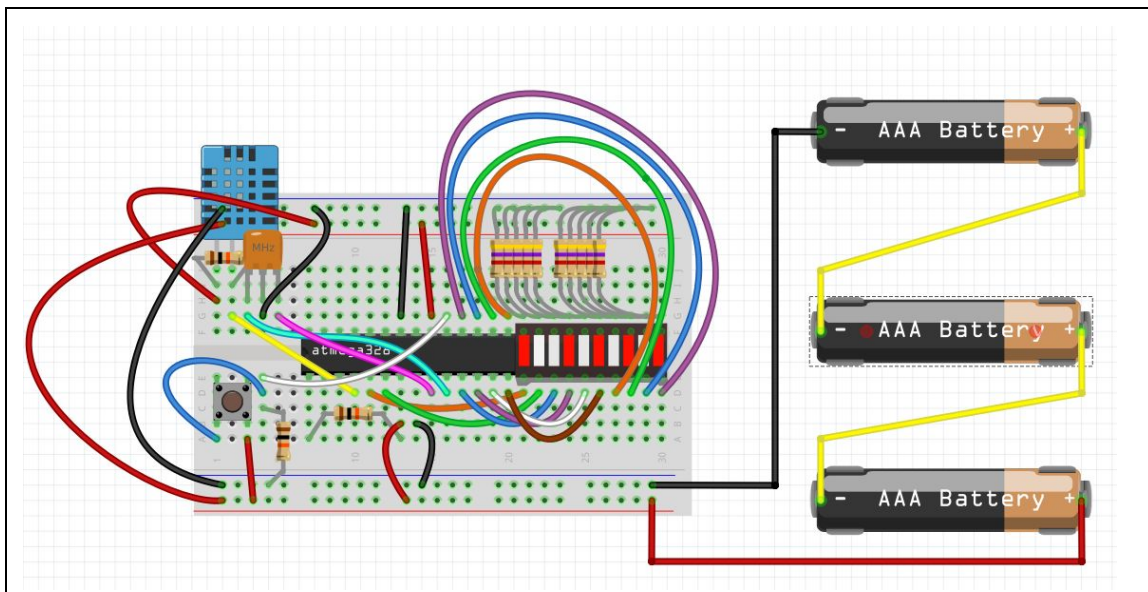


It's the final stretch! Finish by connecting the positive side of the LED strip to the Atmega. As stated above, the positive side of the LED strip will connect to digital pins 3-13, which correspond to pins 5,6, 11-18 on the Atmega. Use jumper wires to make the following connections (be careful and take your time!) .

Jumper wires:

{d11->d21} {d12->d22} {d17->d23} {d18->d24} {d19->d25}
{d20->d26} {g20->d27} {g19->d28} {g18->d29} {g17->d30}

17.



Assuming you've done everything right, all that remains is to connect the batteries. Insert the 3 AAA batteries into the AAA battery holder. Connect the black wire of the battery holder to the negative terminal of the

breadboard, and connect the red wire of the battery holder to the positive terminal of the breadboard. If all your connections are right, you should see the LED bar graph light up!

18. Not seeing anything light up? First, disconnect your battery pack from the breadboard. Then, start double checking your work and ask myself or a volunteer to come help. Most likely, it's a simple error that we'll be able to find in no time. Common issues I've encountered are outlined in appendix

19. Note: The battery packs do not have switches, meaning that they're always on. **When you're not using yours, pop one of the batteries out of the socket to avoid any potential short-circuiting. Repeat, IT IS PROBABLY NOT A GOOD IDEA to keep all of your AAA batteries connected 24/7.** If the positive and negative wires rest touching each other, you create a "short circuit" (a circuit where there is practically no resistance) that will carry a lot of current and cause the batteries to heat up quickly. It's fine as long as you make sure this doesn't happen, but we'd all rather you be safe than sorry.

5. I'm Done! What Next?

If you're seeing blinking lights, it's time to figure out how to use your weather station! The software has 5 modes: default (the travelling blinking light you're seeing right now), temperature, humidity, and mini projects 1 and 2. Modes 4 and 5, corresponding to mini-projects 1 and 2, won't be available unless you add one of the mini-projects to your breadboard. With the basic configuration, you can toggle through modes 1,2, and 3. You can jump between modes by pressing and holding the pushbutton- you'll know when you enter a new mode because the lights on the bar graph will flash quickly in succession and a light denoting the mode number will shine for a few seconds [fig.16].



Fig. 16: The indicator lights for modes 1, 2, 3, 4, and 5, respectively.

Mode 1: Default

This is what you should see every time you start up your weather station. It's a nice way to tell whether your Atmega is doing what it's supposed to be doing. If the cadence of the moving bars is irregular or seems too slow/fast, you'll know there's a problem with your ceramic oscillator- they're very sensitive- so what seems to work is double checking all the connections and making sure no wires are loose (see Appendix A for more details).

Mode 2: Temperature

In mode 2, the the temperature in Fahrenheit is displayed using the scale in fig. 17. If the temperature is greater than or equal to the threshold value for each bar given in fig. 17, then the Atmega tells that bar to light up. Since we live in Florida and temperate weather is the norm, I made the scale span from 50 to 100 degrees, although this is modifiable if you get an arduino and edit the programming of the Atmega (discussed in section 7). The DHT11 is fairly accurate in most circumstances, although being a relatively low end sensor it is vulnerable to some discrepancy- perhaps you'll see one fewer or one more bars than your weather app would indicate.

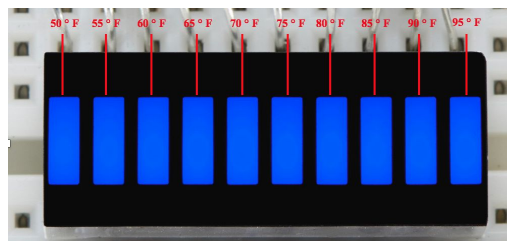


Fig.17: Temperature reading scale

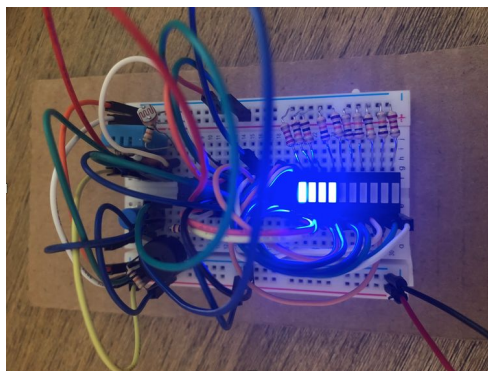


Fig.18: Translating LED output to temperature reading (example). 4 bars are illuminated, meaning that the temperature is probably somewhere between 65 and 70 degrees Fahrenheit.

Mode 3: Humidity

In mode 3, the percent humidity is displayed using the scale in fig.19. If the humidity is greater than or equal to the threshold value for each bar given in fig. 19, then the Atmega tells that bar to light up. It might appear slightly misleading when you notice that 1 bar illuminated corresponds to 0% humidity, but remember that this actually means that the humidity is between 0 and 10%.

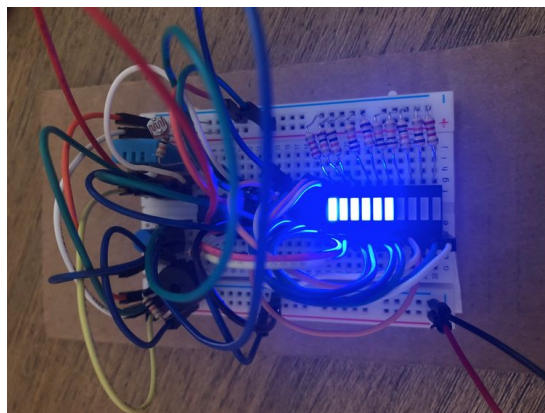
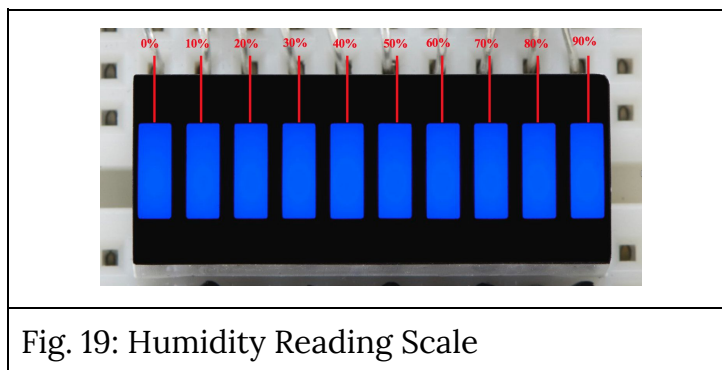


Fig.20: Translating LED output to humidity reading (example). 6 bars are illuminated, indicating that the humidity is probably somewhere between 50% and 60%.

Exploratory Questions

- Can you create a formula that relates temperature (y) to number of bars illuminated (x)? (hint: $y=mx+b$)
 - $y=$
- Can you create a formula that relates % humidity (y) to number of bars illuminated (x)? (hint: $y=mx-b$)
 - $y=$
- What happens if you breathe on the sensor?
- The heat index (HI) is the “feels like” temperature- the higher the humidity, the hotter it feels. Heat Index can be calculated from temperature and humidity using the following formula (simplified):
 - $HI = -42.4+2(T)+10(H)-0.2(T)(H)-0.007(T^2)-0.05(H^2)$
 - What is the heat index if the temperature is 90 degrees and the humidity is 95%? Check you answer using the heat index table:
<https://www.weather.gov/safety/heat-index>
 - Looking at the current output on your weather station, what is the heat index?
 - Temperature =
 - Humidity =
 - Heat Index =

6. Built-In Mini-Projects

Introduction

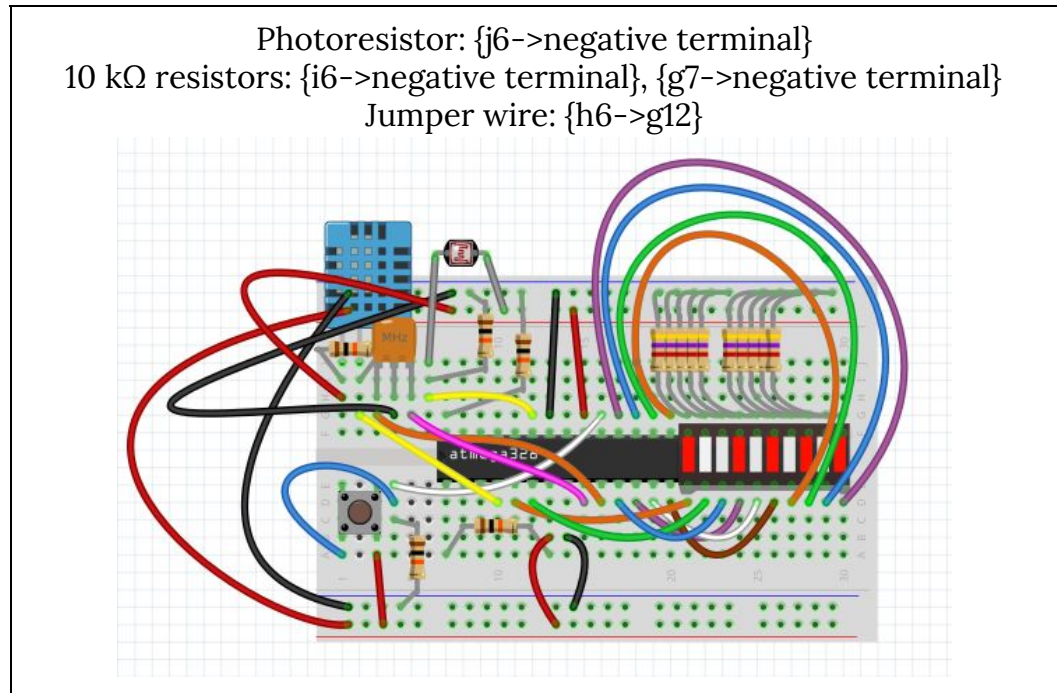
Maybe you’ve finished your weather station and are thankful you’ll never have to touch another breadboard for as long as you can help it. But perhaps you’re curious about what’s left to explore. Maybe you’d like to see just a bit more of what the Atmega can do. If this describes you, you’re in luck- for some reason I enjoy writing code and debugging and messing up circuits in my free time, so I’ve got tons of random projects I can share with you! The two I’ve included below are designed to show you more varieties of inputs and outputs that can be used to perform a desired task, and hopefully get you thinking about how some of these concepts can apply to engineering questions in the real world. So, are you hooked on hardware? Have you

discovered the beauty of breadboarding? If yes, it doesn't have to end just yet- grab that "extra stuff" bag and get to work.

Project 1: Light as an Input

- Instructions

-



Add a photoresistor to the existing circuit. You can remove the DHT11 and its connections first if you'd like more room, but for instructional purposes we'll keep the DHT11 on the breadboard (sorry, I know it's crowded already!). We need to connect the photoresistor to the Atmega (analog pin 0), the negative terminal, and the positive terminal. To indicate that we want the Atmega to run the "extended option" of the program, we can make analog pin 5 read a constant value of 0 by connecting it to the negative terminal. To do all this, make the connections shown above.

- That's it! If you were starting from scratch, you'd have a lot more work ahead of you, but you've already hooked up the Atmega and wired it to the LED output, so there's nothing left to do!
 - What now?
 - Connect the AAA battery pack and use the pushbutton to navigate to mode 4. The photoresistor now controls the number of bars that light

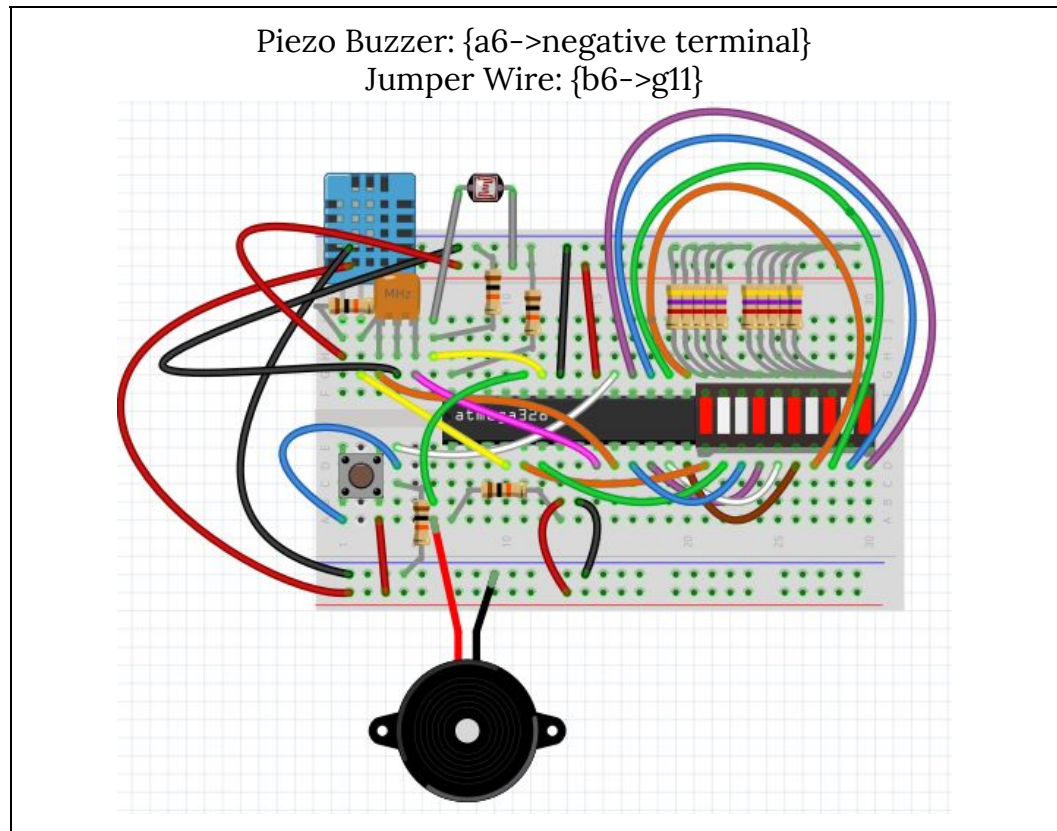
up. Instead of the DHT11 sensor feeding temperature and humidity data to the Atmega, the photoresistor now sends data about the amount of photons (particles of light) that are hitting it at a given time. The difference? The DHT11 sends digital input, while the photoresistor sends analog input. As explained in the background, this means that while the DHT11 sends information encoded in binary (0V = 0, 5V = 1), the photoresistor will send the Atmega a value usually ranging between 0 and 700 that corresponds to a higher or lower level of resistance depending on the amount of light it is exposed to. While the DHT11 works solely in 1s and 0s, the photoresistor navigates all possible values in between.

- Without doing anything, you might notice that the number of bars illuminated in mode 4 tends to fluctuate a bit on its own. Let's experiment. Try the following and see what happens:
 - Swipe your hand over the breadboard
 - Slowly cover the breadboard with your hand
 - Rest your thumb on the photoresistor itself
 - Get out your phone and shine your flashlight on the breadboard
 - Make a "u" motion with your wrist while holding the flashlight over the breadboard
- Exploratory Questions
 - The value read from the photoresistor by the Atmega typically ranges between 0 and 700. If the LED display works similarly to the temperature and humidity modes by mapping the 10 LEDs to this 0-700 scale (more LEDs illuminated = higher value), does more light make the value read from the photoresistor lower or higher?
 - Given what you've learned about photoresistors, can you think of a way that they can be applied to solve a real world problem?
 - Can you think of a piece of current technology that probably uses some type of photoresistor?

Project 2: Music = Science

- Instructions

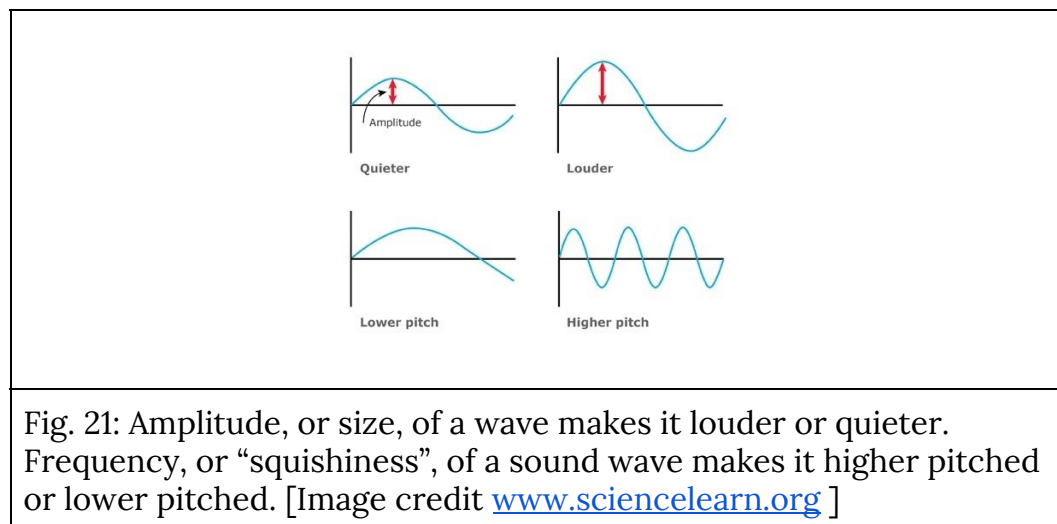
-



Add a piezo buzzer to the existing circuit. Again, you can remove the DHT11 or the photoresistor if you'd like, but for the diagram I kept it intact. If you do remove the photoresistor, make sure you keep the 10 kilo ohm resistor connected to the analog 5 pin on the Atmega (g7 on the breadboard) to let the Atmega know that you're still in extension mode. This breadboard is so crowded you probably think there isn't possibly another space we could shove something, but there is (There are exactly two spaces left to put things, and we'll only use one of them, so how you use that last space is up to you-the possibilities are [not quite] endless)! Anyway, the buzzer just needs to connect to the negative terminal and the Atmega, since connecting to the Atmega effectively gives it access to the positive terminal as well, and we don't need to limit the voltage it draws using a resistor (we could, but it's not entirely necessary here and space is limited as is). Stick one end of the buzzer (either end, it's unpolarized) into the negative terminal and the other end in a6. Then, use a jumper wire to connect b6 to g11, analog pin 1 on the Atmega.

- You're done!

- What now?
 - Connect the AAA battery pack and use the pushbutton to navigate to mode 5. Hear anything familiar?
 - Although we connected the piezo buzzer to analog pin 1 on the Atmega, it is not an analog input- it's a digital output. The only reason we used the analog pin is because we ran out of digital pins, and analog pins can be used for digital purposes! So, in this case the Atmega has two outputs: the piezo buzzer and the LED strip. The input is a list of frequencies corresponding to notes that are read into a command telling the piezo buzzer to buzz. The frequency also determines which bar of the LED strip will light up, creating a “piano” effect.
 - But how do you hear music? Sound is a longitudinal wave, or a disturbance in air that follows a certain pattern- it has a frequency and period which determine whether we hear a low or high pitch. As shown in fig.21, higher frequency means higher pitch. If I tell the Atmega that middle C has a frequency of 262 Hz, the Atmega can tell the piezo buzzer to vibrate for some duration at 262 Hz, causing you to hear a middle C note.



- Exploratory Questions
 - Observe the LED strip as the song is playing. When a note gets higher, does the illuminated bar move left or right?
 - Given this, you decide to program your Atmega with alien music that consists of notes β and ζ . If, while your Atmega is playing an alien song using the piezo buzzer, the bar on the bar graph moves to the left when a ζ is played after a β , which has a higher frequency- ζ or β ?

7. Upgrading Your Weather Station

Introduction

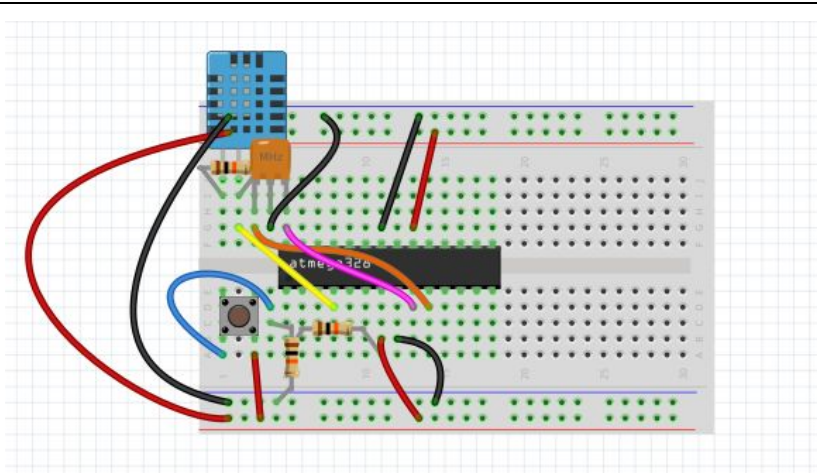
If you're still interested in learning more, or maybe want to try coding for yourself, the best way to do so is to jump right in. To that end, I've created an "upgraded" version of our weather station that allows you to visualize temperature and humidity in real time. It uses the same inputs- the DHT11 and pushbutton- but a much more sophisticated output- an LCD screen. To do this project, you'll need to learn how to use an arduino, how to solder, and, by playing with the code behind it, you'll learn how to interface with the LCD screen using java programming. Learning to code using a computer alone is great, but when you add hardware to it, it makes the whole experience so much more hands on- you'll see what consequences your bugs have on a tangible object, and you'll get to experience the amazing feeling of knowing that your code does something cool in the real world.

Supplementary Materials

- Arduino/Genuino Uno: You won't need this to run your new and improved weather station, but you will need it to transfer the code from my github (<https://github.com/Dahlia-Dry/STEMinist-Movement-2018>) to your Atmega from this workshop. If you're looking to get into coding and engineering, it's just about the best learning tool there is. I usually purchase electronics equipment from www.adafruit.com, and they have some excellent arduino starter kits that include more project tutorials to get you started.
- 1.8" Color TFT LCD display with MicroSD Card Breakout - ST7735R: Available for purchase at www.adafruit.com. This is an LCD screen that we'll use to display and graph the data from the DHT sensor. You can also do this with different sized LCD screens (or even a touchscreen!), but the wiring shown here is specifically for the 1.8" model.
- DHT22 Temperature/Humidity Sensor (optional): As great as the DHT11 is, you might have noticed that it's not exactly the most accurate sensor in the world. It's usually off by anywhere from 5 to 10 degrees, and this is no fault of our own- it's simply just about the cheapest weather sensor you can buy, and the price tag comes with some sacrifices. At \$10 on the website mentioned above (double the price of the DHT11), the DHT22 is much more reliable in terms of accuracy. But if you're like me and just enjoy making cool graphs, the DHT11 works perfectly fine.

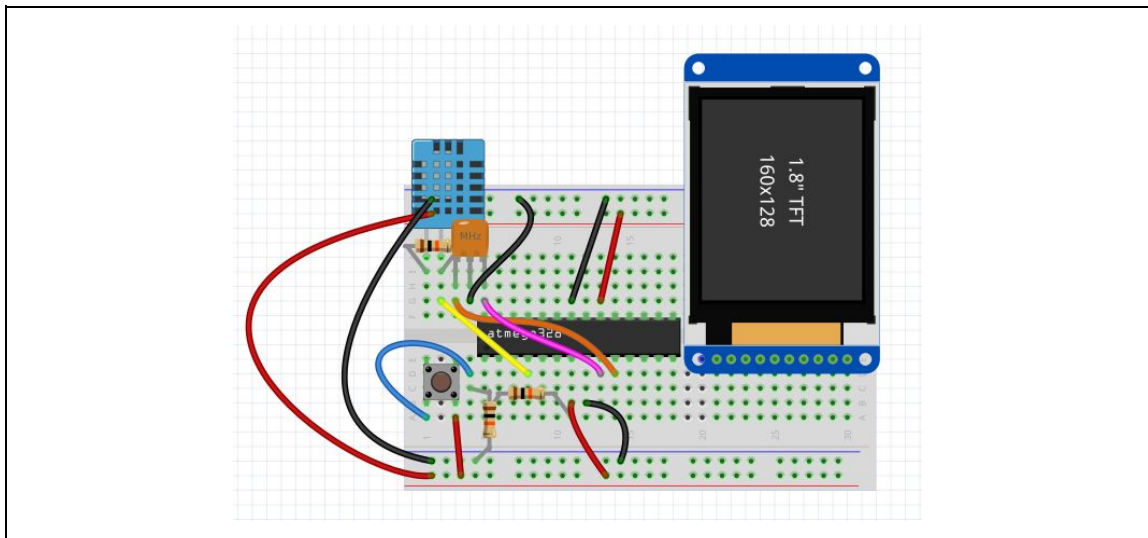
Instructions

1. Using the arduino uno, transfer the code from the above github link to your Atmega chip. Instructions are included on the Github and are readily available on the internet as well. Make sure you select “Duemilanove” as your board even though you’re programming on the Uno- the Atmega itself uses the Duemilanove bootloader. Keep in mind, once you reprogram the Atmega, your old weather station code will be lost. Don’t worry, it’s also on the Github if you want to go back to the other version.
2. Solder the included pins into the holes on the LCD screen. Without secure connection to the pins, you won’t be able to place your LCD screen into the breadboard. Learning to solder is frustrating at first, but it can be fun once you know what you’re doing. If you don’t have a soldering iron at home, you can find one at your local hardware store, along with rosin-core solder. Make sure you’re supervised by an adult and do the proper internet research before attempting to solder the pins to your LCD screen.
- 3.



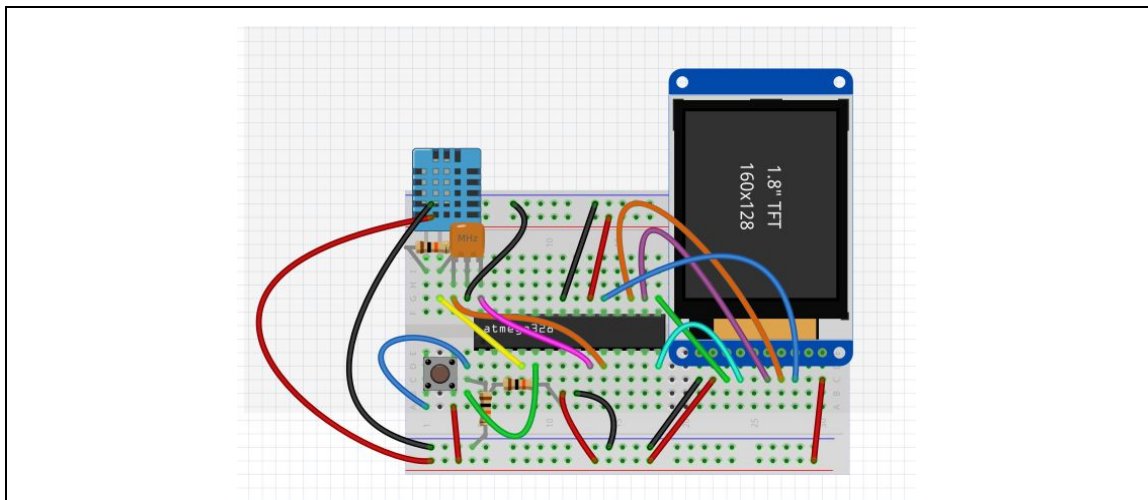
Now, assuming your old weather station is still intact, we’ll need to make some changes to it. The LED “bar graph” will be removed and we’ll shift the Atmega setup over a few spaces on the breadboard to make room for the LCD screen. The DHT11, ceramic oscillator, and pushbutton will remain in the same place, but the Atmega needs to move over so that its end pins are at e5, f5, e18, and f18. Remove the wire connecting the button to the Atmega [the white wire in previous instructions]. Reconfigure your board so that it matches the image above. Steps to do this are explained in more detail in the initial instructions, although keep in mind you’ll have different coordinates this time.

4.



Place the LCD screen on the breadboard. Its pins should go into slots e21-e30.

5.



Connect the LCD screen to the Atmega. You can follow the picture, or use jumper wires to make the following connections:

- {c21->negative terminal}
- {c22->positive terminal}
- {c23->g18} [digital pin 9 on atmega]
- {c24-> d18} [digital pin 8 on atmega]
- {c26->g17} [digital pin 10 on atmega]
- {c27->g16} [digital pin 11 on atmega]
- {c28->g14} [digital pin 13 on atmega]
- {c30->positive terminal}

6. Connect your AAA battery pack, and that's it!

How the System Works

Assuming you've done everything right, you should now see a black screen displaying temperature and humidity data in white text. If you don't see this, there's a few things that could have gone wrong- refer to appendix A for more details.

Mode 1: Raw Data

This is the mode that the Atmega will boot into and is effectively the default. Temperature (Fahrenheit and Celsius), Humidity, and Heat Index (Fahrenheit and Celsius) are displayed and updated every 2 seconds. Illustrated in fig. 22. You can transition between modes the same way as with the "Bar Graph"-based weather station- just press and hold the pushbutton until the screen changes.

Mode 2: Live Temperature Graph

This mode displays the current temperature (in Fahrenheit) at the top of the screen, and shows a line graph of temperature (y) vs time (x) that adds a new data point every two seconds.

Mode 3: Live Humidity Graph

This mode displays the current humidity (percent) at the top of the screen, and shows a line graph of % humidity (y) vs time (x) that adds a new data point every two seconds.

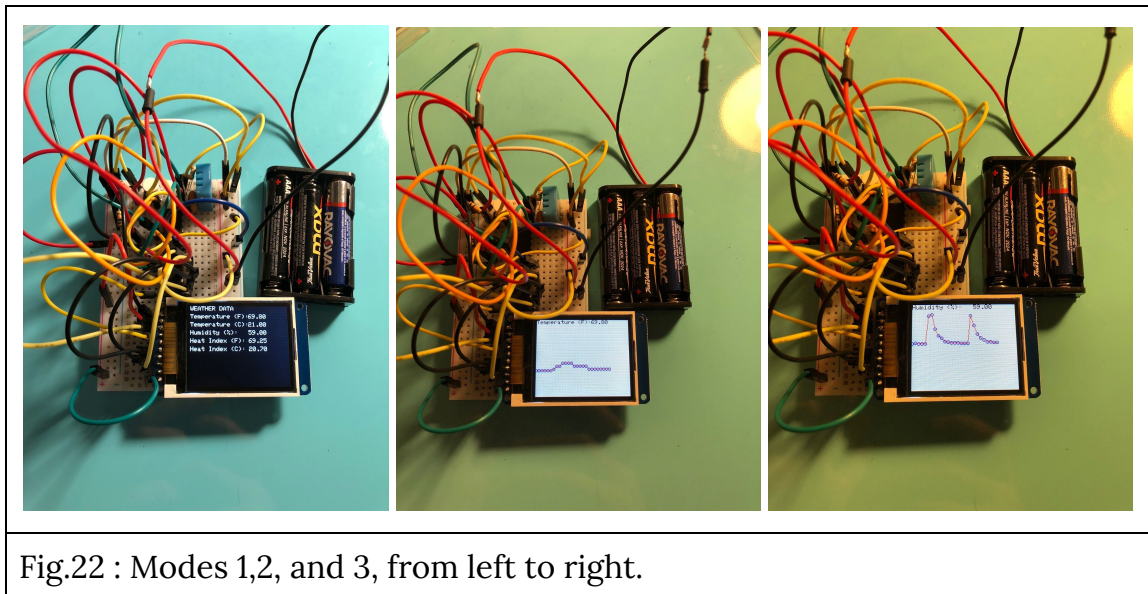


Fig.22 : Modes 1,2, and 3, from left to right.

Exploratory Questions

- Use the graphing modes to experiment with changing the temperature and humidity. Try a few of the following and observe how the graphs change.
 - Walk from inside to outside
 - Walk from outside to inside
 - Breathe on the sensor
 - Stick the weather station in the refrigerator
 - Put the weather station in front of a fan
 - Take the weather station outside while it's raining (but don't get it wet!)
- Can you figure out how to transfer this setup to your arduino? (Hint: It's pretty straightforward, if you pay attention to what pins on your Atmega are connected to what)
- If you've successfully accomplished the above, try using the Serial Monitor and Serial.println() to print the data to your computer instead of the LCD screen.
- Try making cool plots of your data using code!

Still Want to Make It Better? Check Out These Links!

Are you determined to create the best weather station ever? Here's some of the best arduino-based weather stations that I could find on the internet...

- Wifi Connected Weather Station:
<https://learn.adafruit.com/wifi-weather-station-arduino-cc3000?view=all>
- Another Cool Wireless Weather Station:
<https://www.instructables.com/id/Arduino-Wireless-Weather-Station/>
- VERY fancy Weather Station with Remote Sensing Unit:
https://create.arduino.cc/projecthub/antiElectron/arduino-based-weather-station-with-remote-sensor-unit-671f09?ref=tag&ref_id=weather&offset=19

8. Want to Learn More?

It's hard to find your footing when you first start exploring computer science- everything's so new and unfamiliar that it's challenging to figure out how to even get started. If you're interested in learning how to code, I've compiled some good resources below.

- Learning Python The Hard Way, by Zed Shaw. Book and website. <https://learnpythonthehardway.org/>
- Codecademy. Like Khan Academy, but for learning to code. Free and packed with cool lessons. <https://www.codecademy.com/>
- Arduino Project Hub. A great place to look for inspiration, if you're interested in coding with hardware. <https://create.arduino.cc/projecthub>
- Raspberry Pi. A more sophisticated version of the arduino, built on python instead of java/C. <https://www.raspberrypi.org/>
- The Github for this project. I've included some fun miscellaneous simulations and an interactive jupyter notebook that can teach you the fundamentals of plotting data in python, as well as the code applicable to today's project. <https://github.com/Dahlia-Dry/STEMinist-Movement-2018>

Appendix A. Troubleshooting

The Basic Weather Station

- Nothing happens when you connect to power
 - There could be a variety of things wrong, but good practice is to try the easiest potential fix first and save the hard stuff for if it's absolutely necessary. Try the following fixes in order.
 - Did you put the LED bar graph in wrong? Is the "+" side facing "down", as in the diagrams? If you're not sure, flip it around and see if anything happens.
 - Is the Atmega oriented correctly? Check the diagrams, but if you're not sure, just try flipping it.
 - Check all your connections and make sure they match the diagram exactly.
 - Check that no wires are loose (they might appear to be connected but are actually only loosely brushing the hole they are supposed to go in)
 - If all else fails, sometimes all you can do is rip out everything and try again (I know it's frustrating, but it's true). Often you'll have skipped

over a mistake hiding in plain sight when you check your wiring, and if you redo the whole thing you'll find that the problem has gone away.

- The LEDs blinked briefly, then went out. The LED's flashed really crazily, then went out. The default mode functioned briefly and irregularly, then the LEDs went out.
 - Based on personal experience, this seems to be an issue with the sensitivity of the oscillator.
 - Double check the wires connecting the oscillator to the Atmega, making sure they're not loose.
 - Remove the oscillator and flip it around, then place it back in the breadboard, making sure it's secure (there is no scientific reason why flipping it should work but it did on one occasion, I think the main thing is that you make sure the oscillator is securely connected to the breadboard).
- Can't use the pushbutton to change between modes. Pressing it does nothing.
 - Check that a 10 kilo ohm resistor connects the pushbutton to the negative terminal
 - Check that the pushbutton is connected to power
 - Check that the other end of the pushbutton is connected to pin 13 on the Atmega.

"Mini" Projects

- Can't access modes 4 and 5 using the pushbutton- it just loops through modes 1,2, and 3.
 - The 10 kilo ohm resistor connecting analog pin 5 on the Atmega to ground is either not in place or not connected correctly.
 - Check that the pushbutton is connected to power and the Atmega.
- Mode 4 doesn't seem to be responsive to light changes.
 - The photoresistor is connected incorrectly. Check to see that you put it in the same number row as the jumper wire connecting it to analog pin 0 on the Atmega. Check to see that a 10 kilo ohm resistor is in the right place according to the diagram and that one end of the photoresistor is connected to the negative terminal.
- No sound is heard on mode 5.
 - The piezo buzzer is connected incorrectly. Check the diagram and make sure you ran a jumper wire from one end of the buzzer to analog pin 1 on the Atmega and that you put the other end of the buzzer in the negative terminal.

Upgraded Weather Station

- The LCD screen won't turn on- there isn't any backlight or anything.
 - This is a tricky issue, as it could be caused by a variety of things.
 - Hopefully, your soldering is good- check to see that none of your solder joints are fused together or touch the ribbon cable running from the back of the chip to the screen itself. If the problem is with the soldering, sometimes there isn't much you can do besides buy a new LCD screen and try again.
 - Check that the wiring is done correctly, specifically the wires connecting the LCD screen to the positive and negative terminals. If you're not getting any backlight at all, it's likely the problem is with one of those three wires.
- The LCD screen turns on, but the screen is blank white.
 - There is most likely a problem with your wiring. Double and triple check. Your wires connecting the LCD screen to the positive and negative terminals are probably right.

