

Flame Controlled MIDI Controller

by [bergerab](#) on February 23, 2015

Table of Contents

Flame Controlled MIDI Controller	1
Intro: Flame Controlled MIDI Controller	2
Step 1: The Concept	3
Step 2: Get the Parts!	4
Step 3: 3D Print the Case	5
File Downloads	6
Step 4: Touch Up the Case	6
Step 5: Tap Screw Holes	7
Step 6: Add Rubber Feet	7
Step 7: Print Labels	8
Step 8: Destroy a Breadboard!	8
Step 9: Add Arduino to Breadboard	9
Step 10: Add Male Headers	10
Step 11: The Schematic	10
Step 12: Add Potentiometers	11
Step 13: Add ON/OFF Switch	12
Step 14: Connect LED	12
Step 15: Add Fire Sensor	13
Step 16: Hot Glue Everything Together	14
Step 17: Put it All Together	14
Step 18: Close Up the Case	15
Step 19: Add Bracket (Optional)	16
Step 20: Add Potentiometer Caps	16
Step 21: Upload Code	17
File Downloads	17
Step 22: Install Hairless MIDI	17
Step 23: Setup MIDI Yoke	18
Step 24: Assign CC in your DAW	18
Step 25: Conclusion	19
Related Instructables	20
Advertisements	20
Comments	20



Author: [bergerab](#) [author's website](#)

I'm a student at UW-Milwaukee studying computer science with a passion for electronics. I'm always working on a project or thinking of new ideas. If you have any questions about any of my instructables, or would like to collaborate on a project, don't hesitate to send me a private message!

Intro: Flame Controlled MIDI Controller

One huge issue in the world of digital music production is keeping that analog warmth (that resonated from reel-to-reel systems and tubes) in modern day digital music. Many swear that analog systems have a sound that can never be replicated by bits, and hope is lost for digital music to match that analog quality. Virtual Studio Technologies (VSTs) have tried to replicate the authentic analog sound, but they (being entirely digital) cannot give you the true sound. **In this instructable, I'll share with you how we can bridge the gap between digital and analog music production by creating a Flame Controlled MIDI Controller using an Arduino micro-controller.**

Fire is awesome. Flames sway, crackle, and waver which makes them a perfect medium to capture a room's atmosphere, and ultimately to create a great analog signal. These characteristics are optimal because even when the signal is converted into digital MIDI signals, it will keep that warmth and still paints a picture of the feeling in the room.

I wanted this Flame Controlled MIDI Controller to be available to anyone, even if they have no experience with electronics and soldering. This is why I have structured my design to not even use a soldering iron. By only using the simplest components, I have made this project viable for anyone to make, and be very inexpensive. Science and logic are important fields for anyone to learn from and I hope I can get you hooked in.

To see the effect this controller can have on a track check out this short cover of "Almost is Never Enough" by Ariana Grande I recorded with and without the controller (the candle was controlling the "pan" on the track to simulate an old rotary speaker):

Without controller (dry)

With controller (pan) (The effect is far more noticeable on headphones or stereo speakers)

With controller (tremolo/volume) (An extreme example of the another way the effect can be used)

In the instance of the pan effect, the result is pretty subtle, but it accomplishes exactly what this controller was design to do. To bring aspects of analog into modern digital sound.

Rather see a video? No problem I recorded myself playing the song, captured the screen (so you can see all the automation from the controller), recorded the candle's movements, and matched all of that to the audio of the automated pan effect in the following video!

If you're ready to learn more about how this candle works, and exactly how you can build your very own, please proceed to the next step: The Concept!





Step 1: The Concept

General:

Once completed, this controller is very simple to use. All you have to do is plug it in via USB, pull up one program on your computer (plus your DAW), light the candle, wait about 5 seconds (for the Arduino to take some measurements), and you're all ready to go. There are three important components of this controller that the user will interact with:

The LED:

To keep you updated, there is an indicator LED on the front panel

One of the coolest features of this controller is how the indicator LED (when it is sending MIDI commands) mimics the exact movements of the candle. There are actually three different messages the indicator LED will tell the user:

1. Two blinks means it does not sense a fire near enough and you must light the candle.
2. If it grows from dim to bright, that means the Arduino is taking its measurements.
3. If it flickers with the light from the candle, it is currently showing you roughly what is being outputted via MIDI.

The Knobs:

There are two knobs that alter the output of the MIDI controller. I named them "Sensitivity" and "Intensity". Granted that standard MIDI CC receives data from 0 - 127. These knobs effect how the candle uses those 7 bits of information.

The sensitivity knob controls how much a single waver of the candle effects the output. For example, at a high sensitivity, the smallest movement of the candle will change the output by just several values. As opposed to the lowest sensitivity, where any movement from the candle will send the candle from 0 to 127.

The intensity knob controllers the default value of the output. So the default could be at 0, and the candle work as a linear knob (only going up from 0). If the knob is in the middle position, the default output would be 64, any movement from the candle (depending on the sensitivity) would be more of a sinusoidal ouput. The knob goes anywhere from 0 - 127.

The Flame Sensor:

The Fire Dependent MIDI Controller functions using an analog flame sensor. Basically it is just an infrared LED which changes the signal based on how much fire is present. For simplicity, I've used a flame sensor module (can be found on eBay or most online electronics stores). The only issue with the sensor is that it was not designed for close measurement. Values read from the sensor typically vary between 3 and 8 (but can change depending on the candle). Therefore, I wrote the code to help give the device a features to give the controller more resolution for flame detection. If possible, it would be best to find a more sensitive close range fire sensor. The design of the project is loosely coupled (the Arduino is programmed to work with any flame sensor), so if you replace the flame sensor (as long as you use an analog output) it will work just fine (or maybe even better!).

Well those were the basics. If you REALLY want to know how it works you've gotta build it. So let's get some parts in the next step!

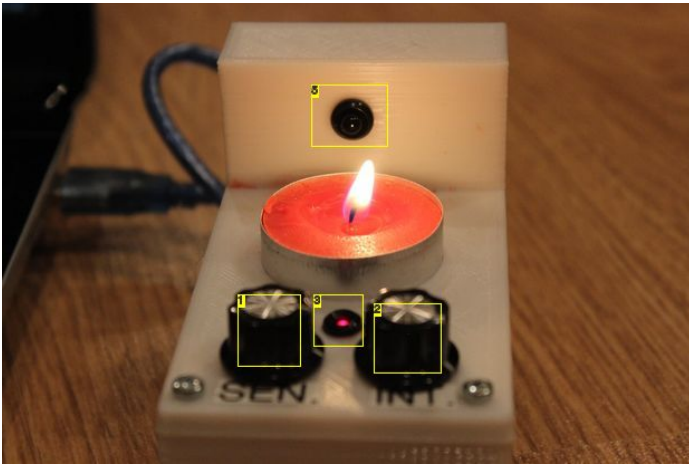
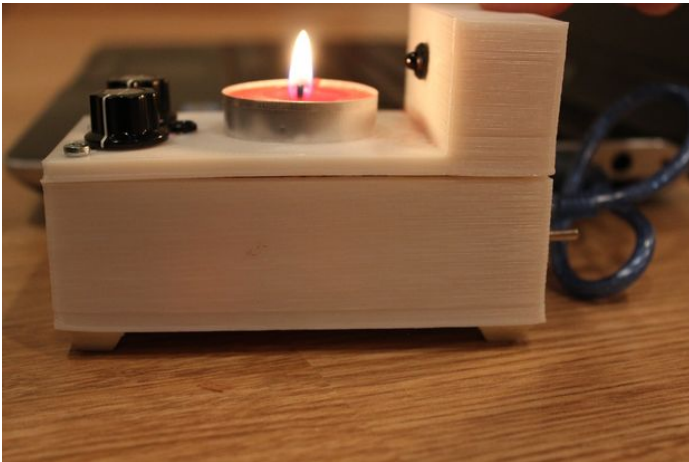


Image Notes

1. Sensitivity Knob
2. Intensity Knob
3. Indicator LED
4. Flame Sensor LED
5. Flame Sensor IR LED



Step 2: Get the Parts!

You'll need the following parts:

- Female to female Dupont wire
- Breadboard
- Arduino Nano
- SPST Toggle Switch
- Keyes Flame Sensor
- Two 10K Potentiometers
- 3D Printed Case
- 5mm and 3mm LED Bezel
- 3mm Red LED
- Male to male jumper wires
- 2K Ohm Resistor (Anything close is OK)
- Male header pins
- Two M3 10mm - 15mm screws
- One tea candle

You'll need the following tools:

- Power Drill

<http://www.instructables.com/id/Flame-Controlled-MIDI-Controller/>

- Dremel (Optional)
- 3D Printer (with max print volume of about 6 cubic inches)
- Screw driver

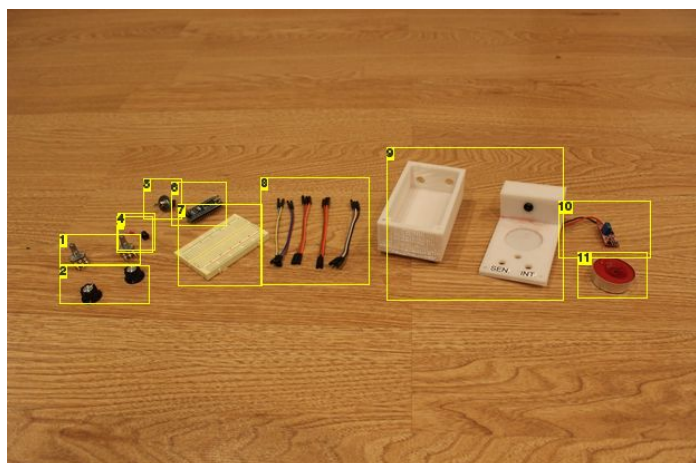


Image Notes

1. 10K Potentiometers
2. Potentiometer Caps
3. 3mm LED (with bezel)
4. 3mm LED (with bezel), and 2K ohm resistor
5. SPST Switch
6. Arduino Nano
7. Breadboard
8. Female Dupont Wires
9. 3D Printed Case
10. Keyes Fire Sensor
11. Tea Candle

Step 3: 3D Print the Case

This project really wouldn't work without a functional case. Thanks to the availability of rapid prototyping machines, almost anyone can use the STL I made to print their own case. If you don't own a printer it is quite easy to access them.

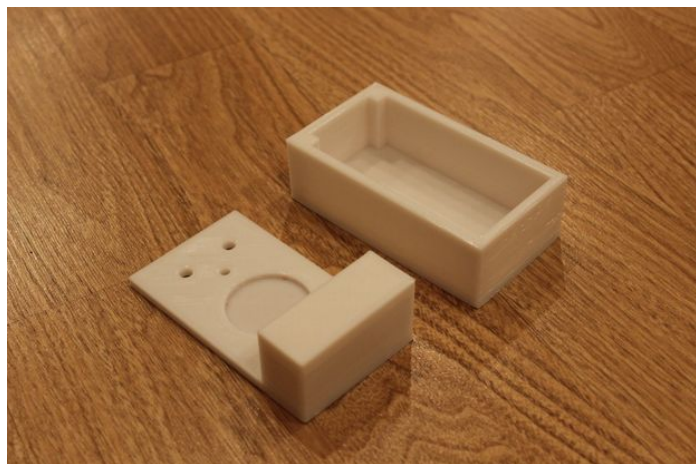
You can find a printer to use at:

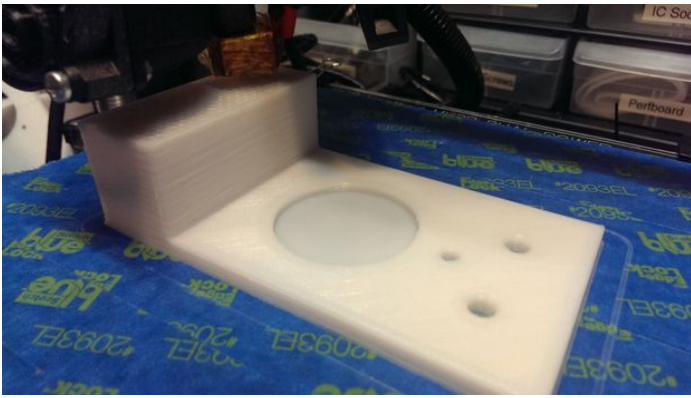
- A university or high school
- Your local MakerSpace
- Most robotics teams or clubs

If you have regular access to a 3D printer, printing the case should be simple. I have created the STL with the budget printer in mind. I only use simple geometric shapes to make the case so there is an insignificant amount of overhang. The only overhang you'll need to print is the box that encases the fire sensor, which is just a simple bridge.

Looking for some advice printing this case? Check out my post on Thingiverse for additional tips from the Thingiverse community!

If you have any issues printing, please either message me personally, or post in the comments!





File Downloads



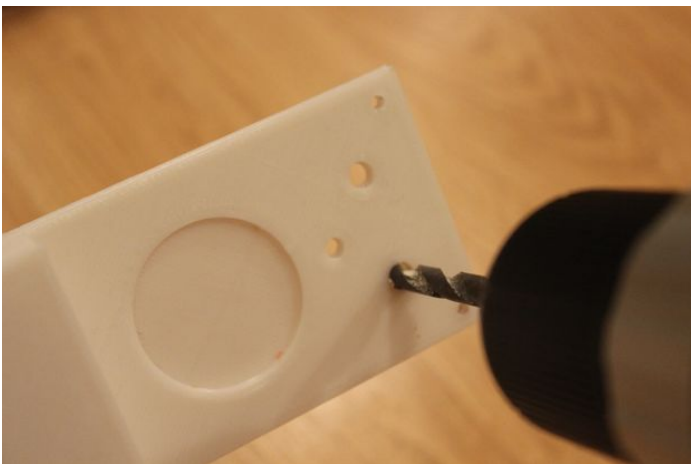
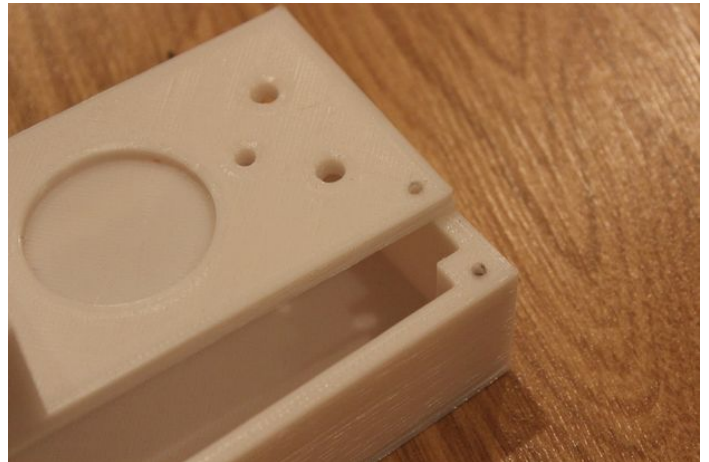
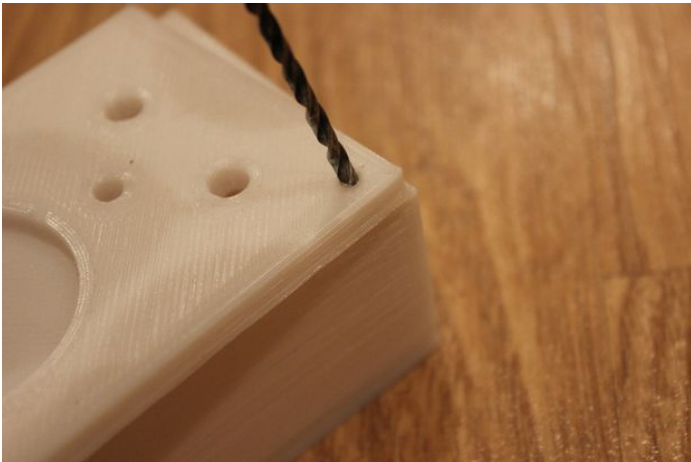
Fire_MIDI.stl (476 KB)

[NOTE: When saving, if you see .tmp as the file ext, rename it to 'Fire_MIDI.stl']

Step 4: Touch Up the Case

3D prints almost never come out exactly how you want them. This is why we will need to do a few touch ups before we put this together. I have excluded several features on the case so you all can personalize it based on how you want to make this controller. If you want to complete the project exactly like mine, you'll have to drill ~2.5mm holes for the screws on the front panel, going into the back of the case (just about 10mm - 15mm deep). Also you will have to drill a 5mm hole on the fire sensor housing box (close to the center). Finally, drill two holes in the rear of the bottom case one for the USB output (from the arduino nano) and another for your SPST switch.

Once you can push several components through the appropriate holes in the case, we can finish up working with the case!

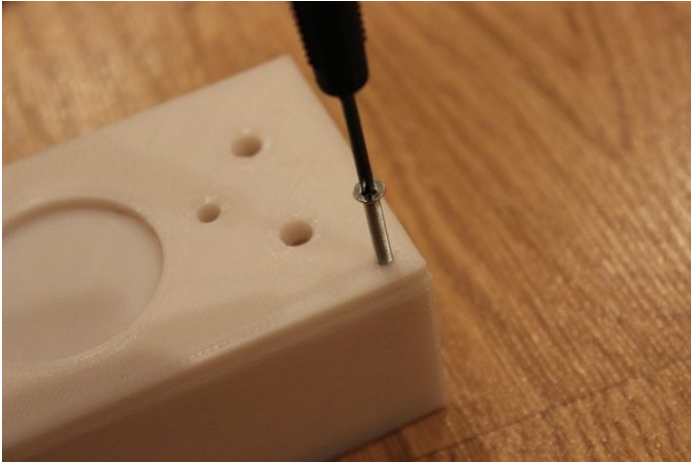


Step 5: Tap Screw Holes

Since no printer can actually print precise threads for screws, we will have to tap them ourselves. As I suggested in the previous step, make sure you drill out the screw holes at about 2.5mm in diameter. This is so that we fill the M3 screw's roots and crests entirely with plastic for the highest amount of friction.

To tap the screw holes you could use an actual screw tap, or just a screw. You may simply use a screw because the PLA is far less hard than the screw, so it will not do any damage.

You'll know you've successfully tapped the screw hole once a screw can easily be screwed through the front panel into the back of the case. Make sure you test it out before moving on!



Step 6: Add Rubber Feet

To finish up the case, and to make it as safe as possible, we will add four adhesive rubber feet on each corner of the case.

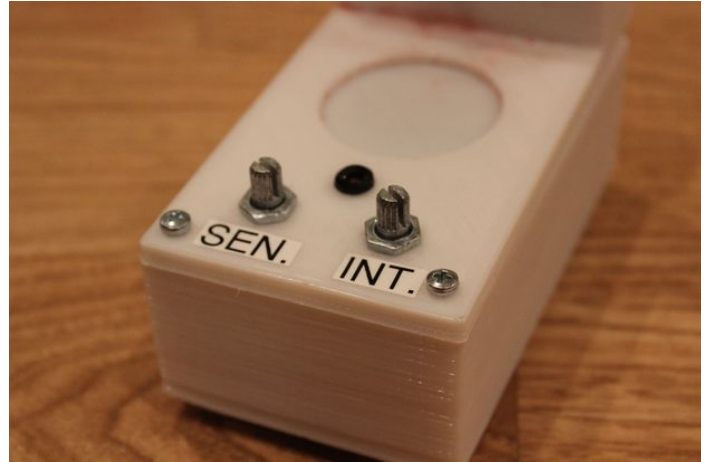
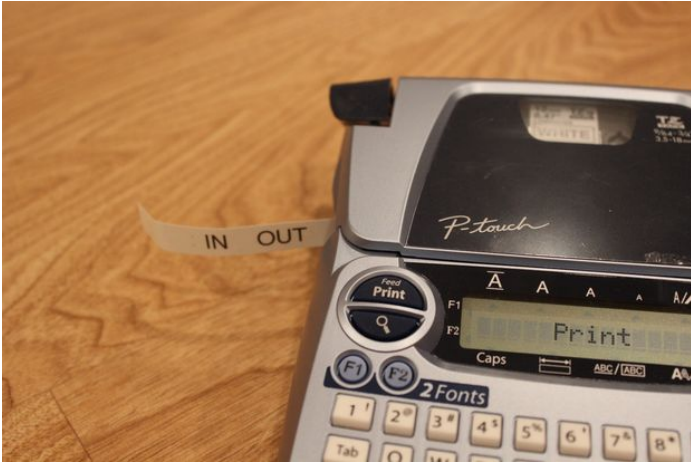
For this build, rubber feet are crucial since you certainly do not want the tea candle falling over anywhere in your house!



Step 7: Print Labels

For any controller or music pedal its a good idea to label your knobs and inputs/outputs. This ensures that there is no confusion between the knobs (we will have one sensitivity and one intensity knob). To keep them ordered, I printed out one label named "SEN" and on for "INT".

OK so maybe this step was my inner perfectionist speaking, but I think labels are necessary for a functional design. If you don't have a label printer, just marking the knobs with sharpie/paint will work too!



Step 8: Destroy a Breadboard!

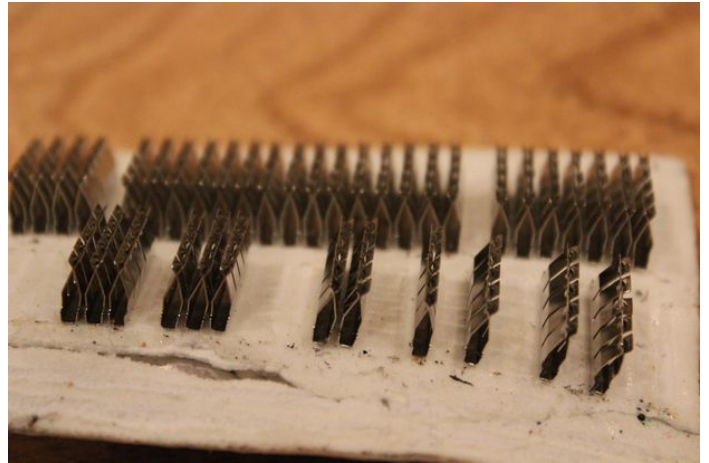
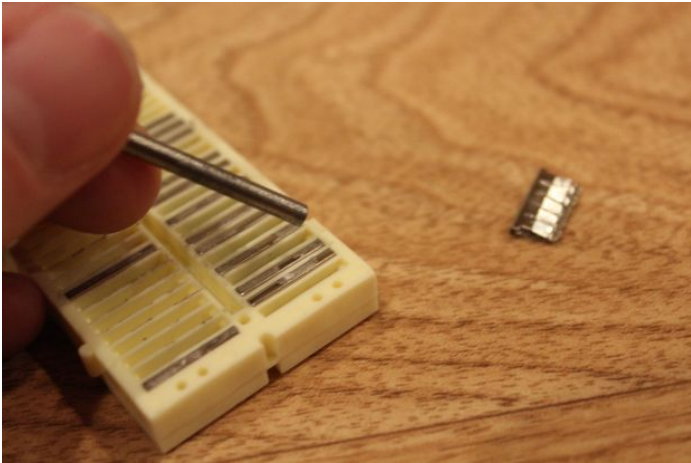
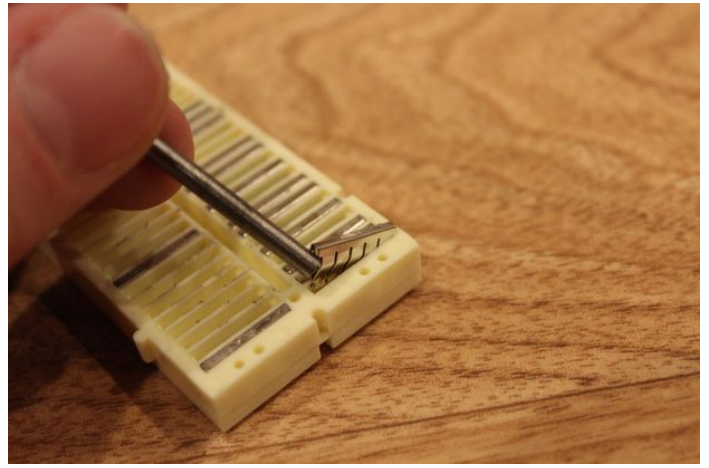
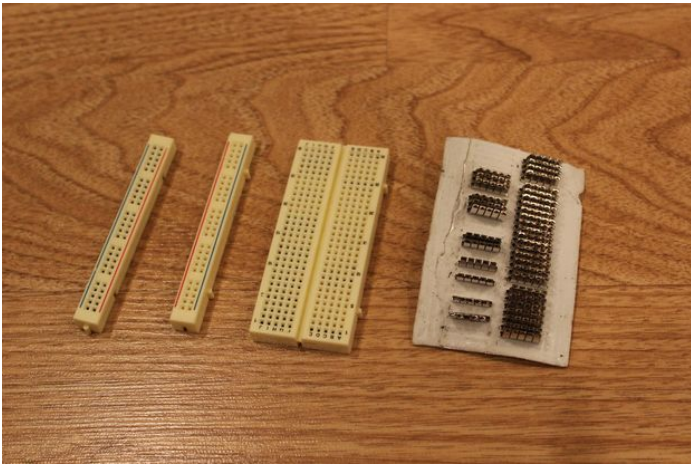
A breadboard is a great way to prototype your circuits before you build them. They are perfect for beginners and advanced hobbyist (and professionals). In this project, my goal is to allow anyone to build their own fire controller without the use of a soldering iron. This breadboard will be our ticket to a solder-less project! But.. first we are going to have to destroy it.

First of all, you will only need to take apart your breadboard if you are using an Arduino Nano and are mounting the way seen in the following steps. If you find another way to mount the Arduino, feel free to try it and show how you did it! But I think its a great idea to take apart a breadboard to see how it works, and to help mount that arduino.

Using a pliers or flat-head screw driver, separate the adhesive bottom from the plastic on the breadboard. Soon you'll see the metal connectors that actually make the board work. Continue to separate the two layers until the plastic is completely separated.

Take a few moments to check out the metal connectors and visualize how the breadboard works.



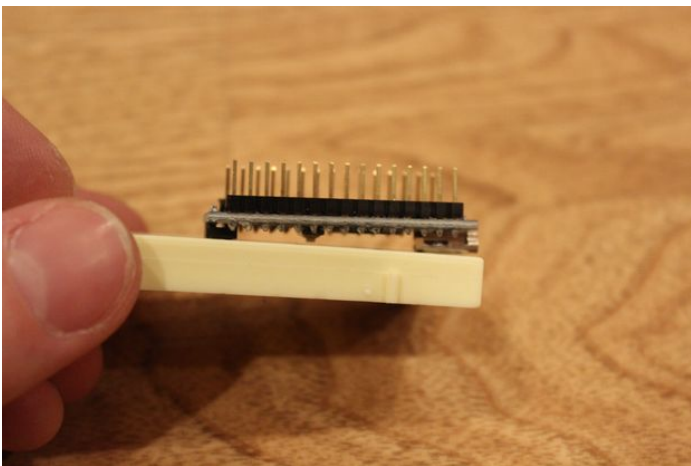


Step 9: Add Arduino to Breadboard

The brain of this project will be the Arduino Nano. Since I wanted to build this project without using a soldering iron, we will actually be mounting the Arduino directly to our destroyed breadboard via its SPI headers (located on the front of the board).

We can start by pushing the Arduino upside down on the breadboard. ***Make sure to place it in a position on the breadboard where the metal connectors are missing! Otherwise you may destroy your Arduino!***

For now the Arduino can just sit on the breadboard upside down. In later steps we will secure it with hot glue. So let's move on!

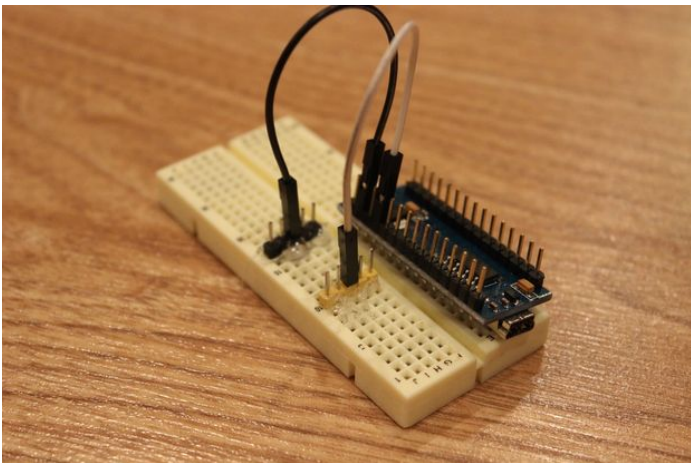
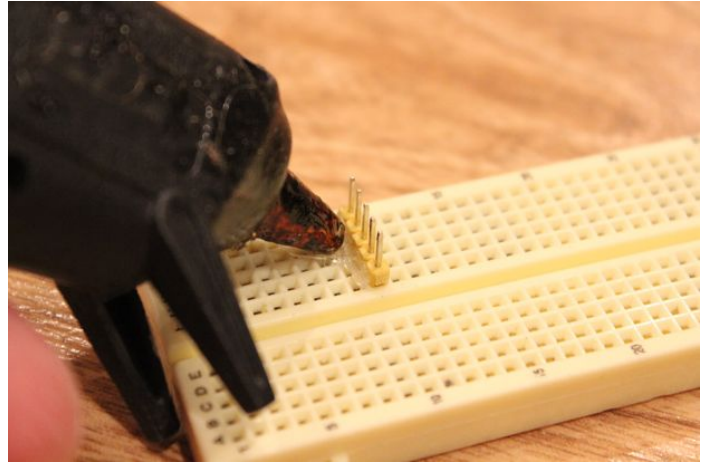
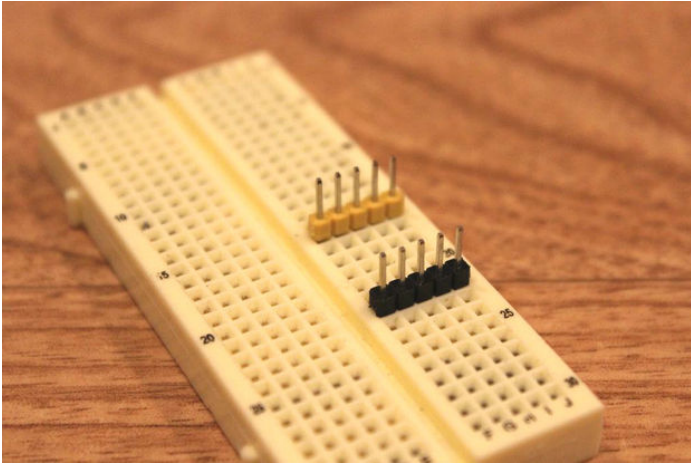


Step 10: Add Male Headers

Since we will be making multiple connections to +5V and GND, we should add some extra headers into the breadboard to connect to. I found that having five headers for each +5V and GND works well for this project.

To connect the headers, simply push two rows of five headers (one row for each) through the breadboard (make sure goes into the metal connectors). Then connect one row to the Arduino's GND, and the other row to +5V. Make sure you secure these headers with hot glue before you move on since if they fall out, the whole project would stop working.

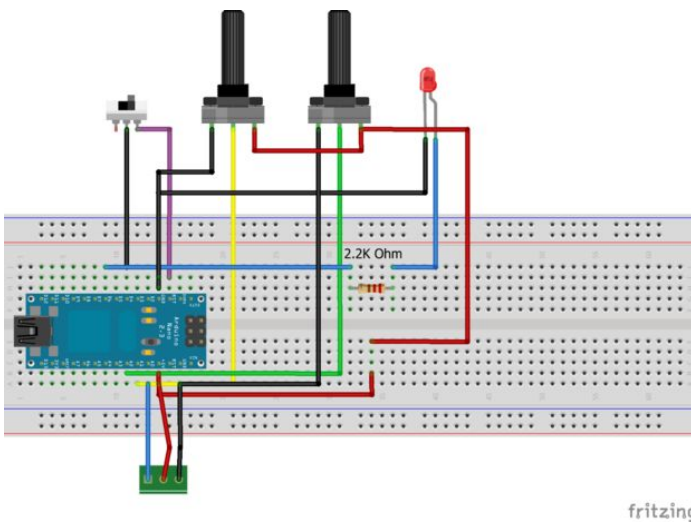
Once the headers are secured into the breadboard, we can move on to the next step, and actually use these headers!



Step 11: The Schematic

A schematic is a "picture" of the components in a circuit that describes how everything is connected together. The picture above is more of a psuedo-schematic since it doesn't use any of the conventional symbols that are typically used. However, it makes for a great reference for you while you go about putting together this project (since the project itself is on a breadboard).

So keep this image in mind (or print it!) while you work on the project. It'll help to show you the larger picture of each step you do!

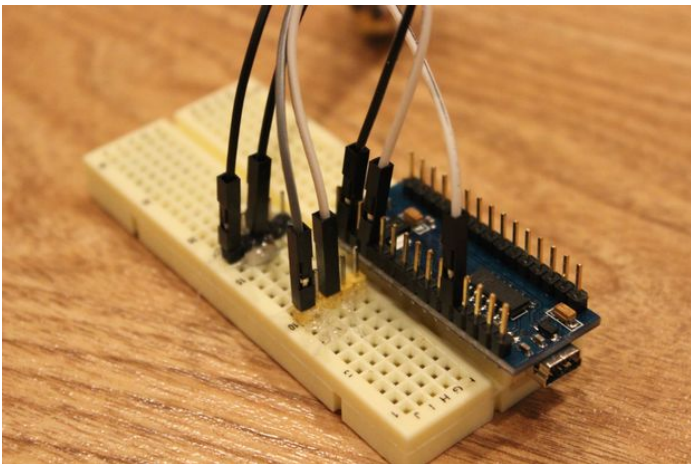
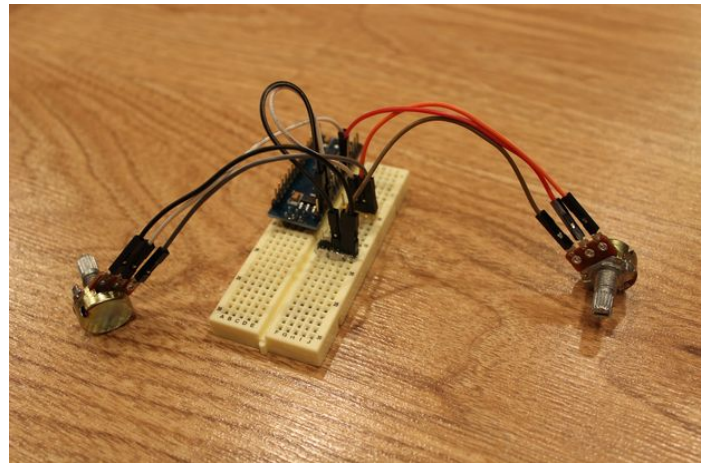
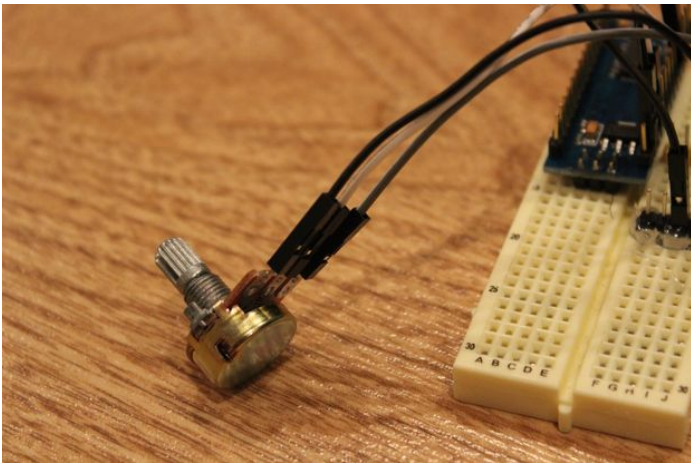


Step 12: Add Potentiometers

To enable the user to input values to the Arduino, we will be use two 10K potentiometers. Both will be wired the same way to the Arduino (except they will be on different analog inputs).

Connect the left terminal of one potentiometer to GND, the right terminal to +5V, and finally the middle terminal to A1. Then connect the other the same way except to A2, instead of A1.

As I will remind you throughout the entire project, make sure you test each component you add to the Arduino. You can test the potentiometers with the sample potentiometer sketch from Arduino's website!

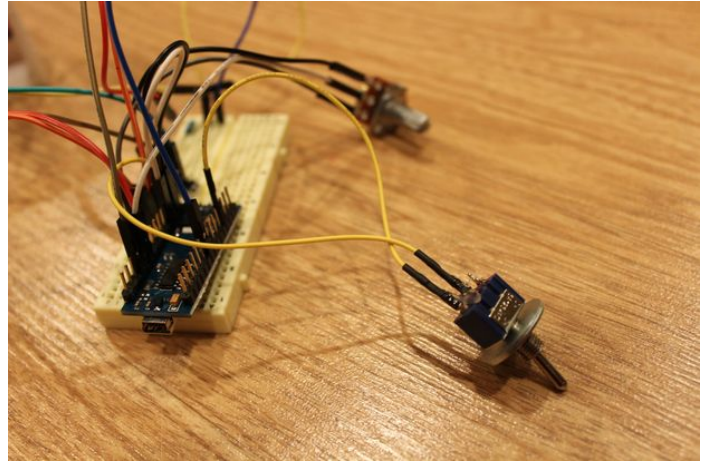
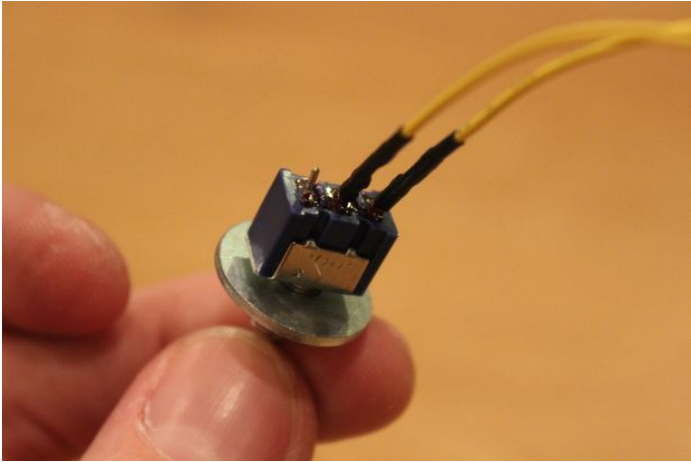


Step 13: Add ON/OFF Switch

Sometimes you will need to re-boot the controller (for re-calibration or troubleshooting purposes), for that reason I've added a switch which restarts the Arduino's program when its toggled. It works by connecting the Reset pin on the Arduino to GND when it is flipped ON and GND is connected to nothing when it is OFF.

To connect the switch, connect the middle terminal to GND via a female jumper cable, and the left (or right) side to the Reset pin.

When starting out, the pins on switches can be a little confusing. So try to test out the switch before moving on! Upload a program to the Arduino (such as the [sample blink sketch](#)) and see if the switch actually restarts the program.



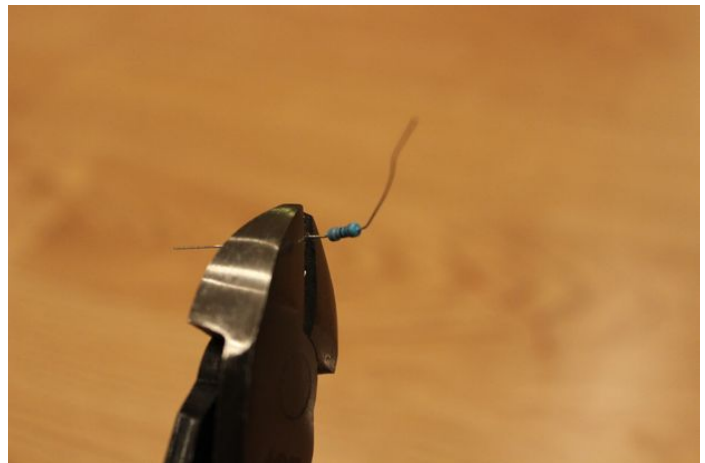
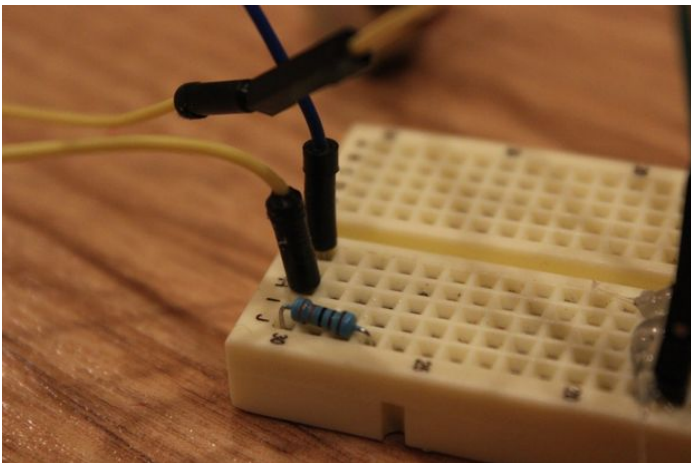
Step 14: Connect LED

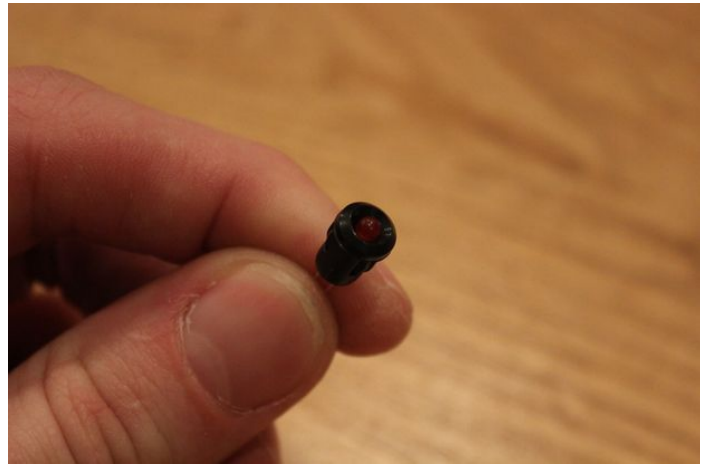
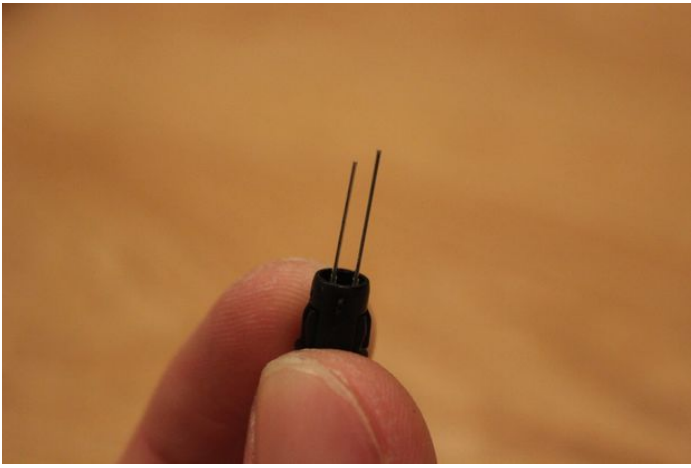
Our controller needs some way to communicate to its user output data, and alert when something is required from the user. For this, I decided to use a 3mm LED. It is a simple solution and it looks very sleek.

To connect the LED to the Arduino, connect the positive lead of the LED (longest lead) to a 2K ohm (or something close to that) to +5V and the negative lead (shortest lead) to ground.

We use a resistor to limit the current so the LED doesn't burn out. You can calculate the most optimal resistor value by using Ohm's Law ($V=IR$). But using a 2K ohm should be safe for just about any LED you use.

Before moving on apply some 5V current from the Arduino to the LED to make sure it is working properly!



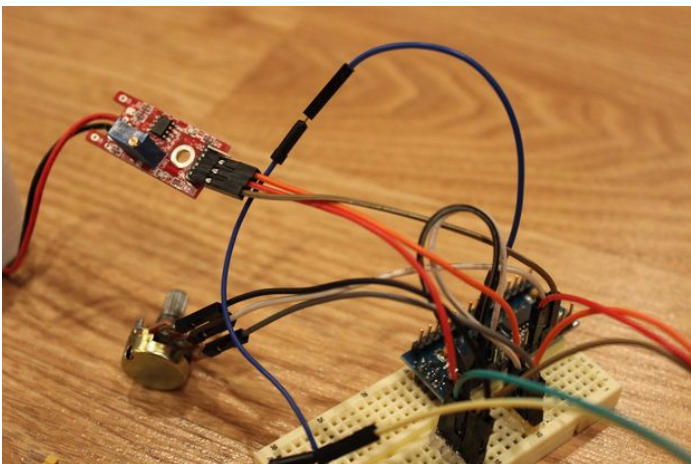
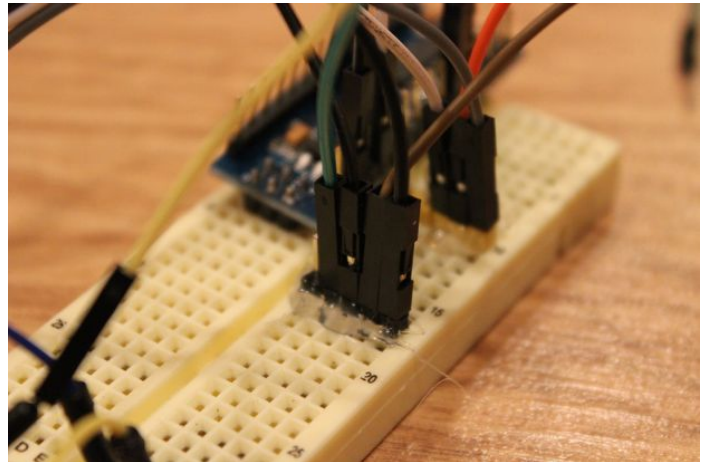
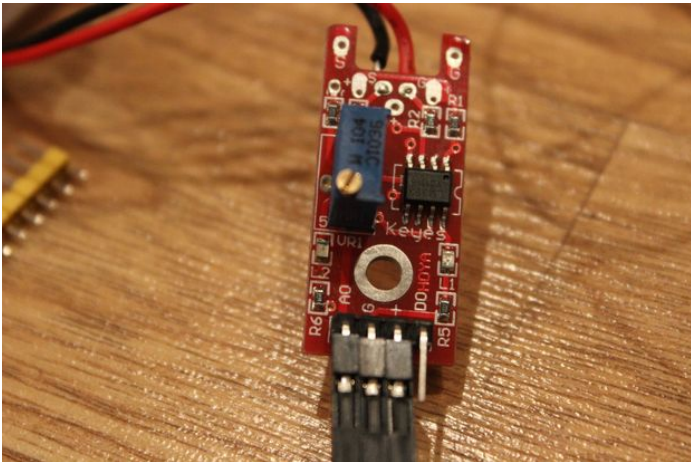


Step 15: Add Fire Sensor

Time to add the heart of our project: the flame sensor. Luckily all the hard work with the module is done for us so all we have to do is attach it to the arduino via female dupont wires. Keep in mind you may have to modify the PCB on your flame sensor to make it compatible with my project box. If you have a sensor with short LED leads, you will have to use a soldering iron to extend the leads. This is why I suggest you use a module that already has long leads.

To connect the sensor, simply attach the module's A0 output to the Arduino's A0, 5V to 5V and GND to GND.

That's it! Everything should be connected and able to communicate with the Arduino. Before moving on, make sure you double check all your connections are correct!

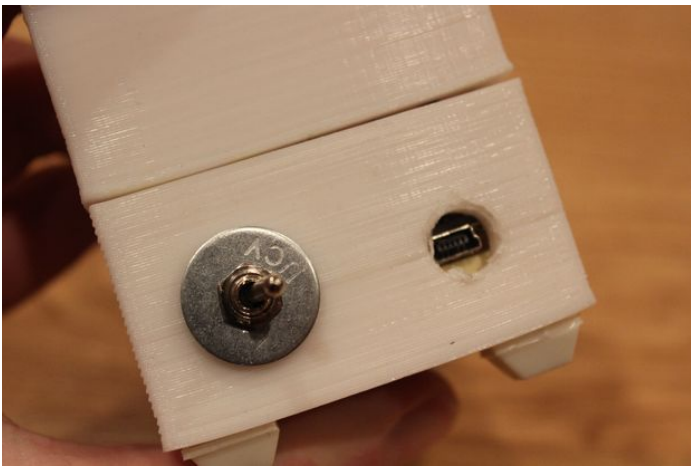
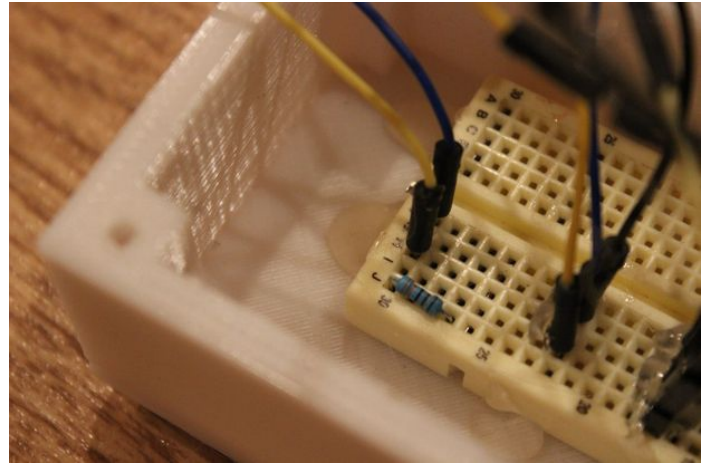
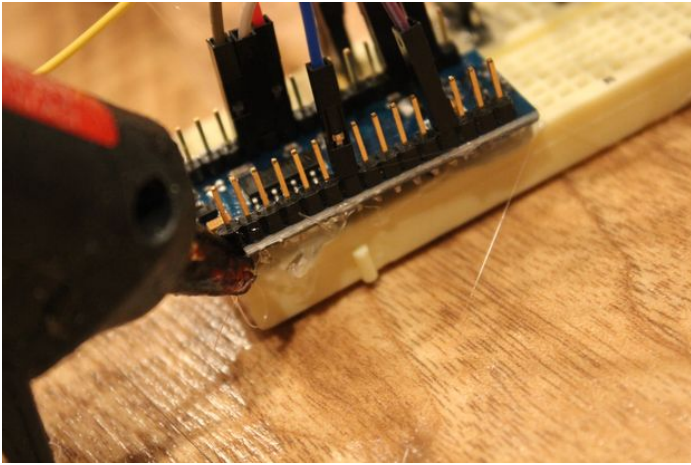


Step 16: Hot Glue Everything Together

Hot glue is awesome. It enables you to connect two things together, but still have the option of taking it apart. Since the bond is so strong it makes a great adhesive for any project, and it can easily be taken off with a knife. This is why I chose to mount the breadboard, and arduino with hot glue instead of screws. Not to mention how easy it is to hot glue something.

So fire up that hot glue gun and secure anything to the case that could come loose. This includes any jumper wires, arduino, breadboard, etc... Add a generous amount, as I have designed the project box to be thick so that the heat of hot glue does no damage to the plastic. (In my experience, hot glue can sometimes melt PLA).

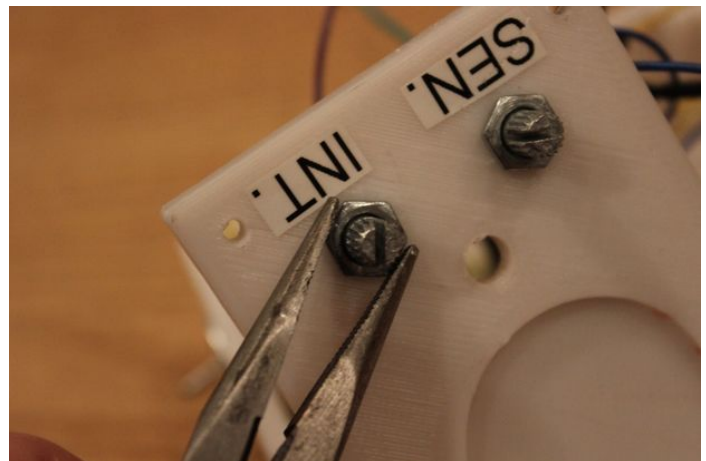
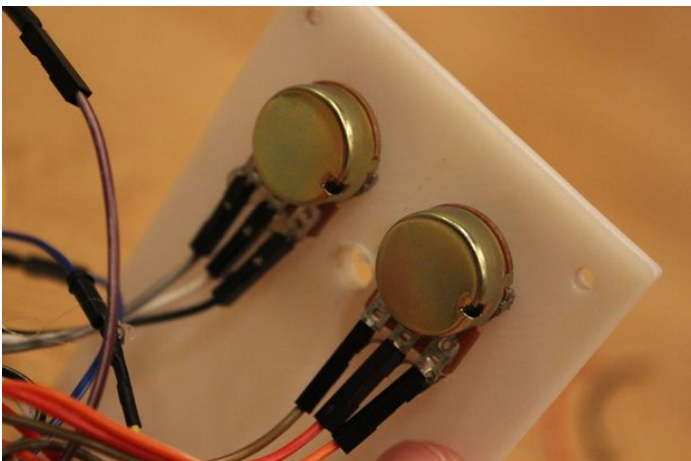
Make sure everything is secure and future proof, since soon we will be closing the box. If glued correctly, you should never have to perform maintenance on the internals.

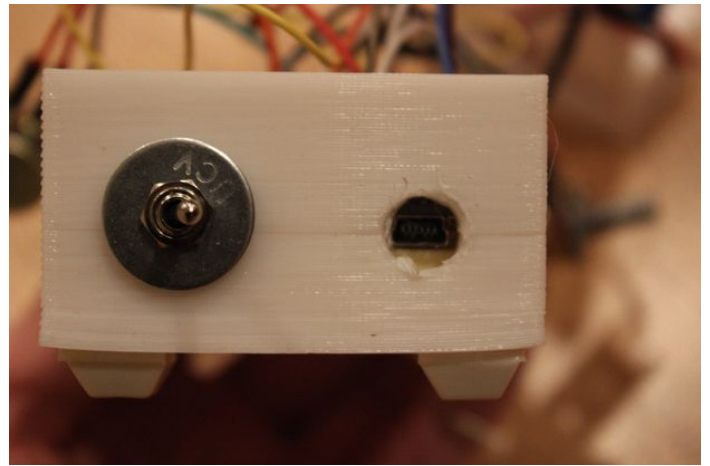
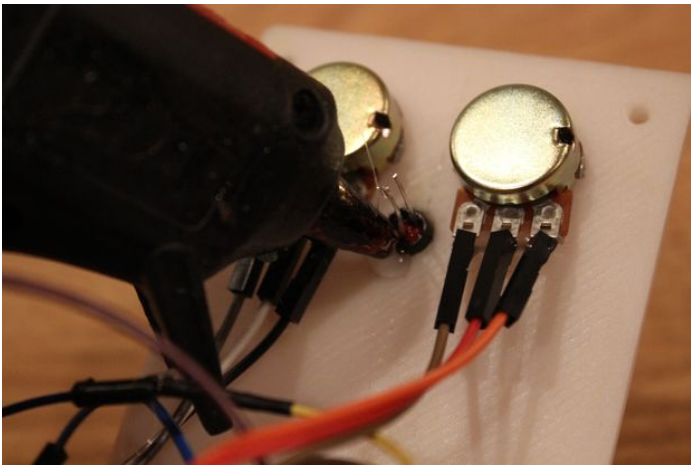
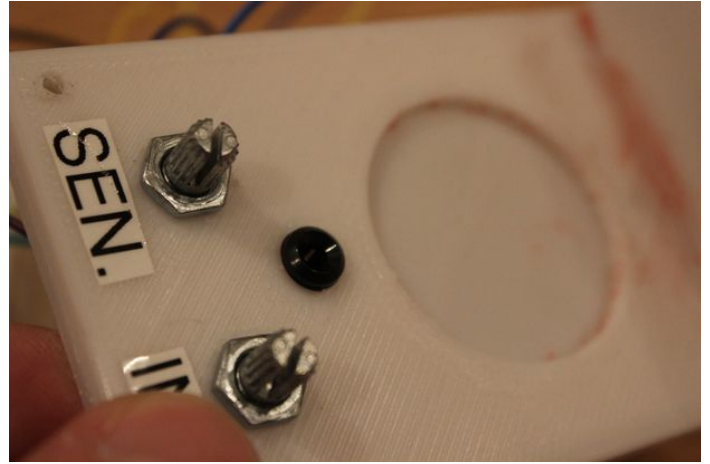
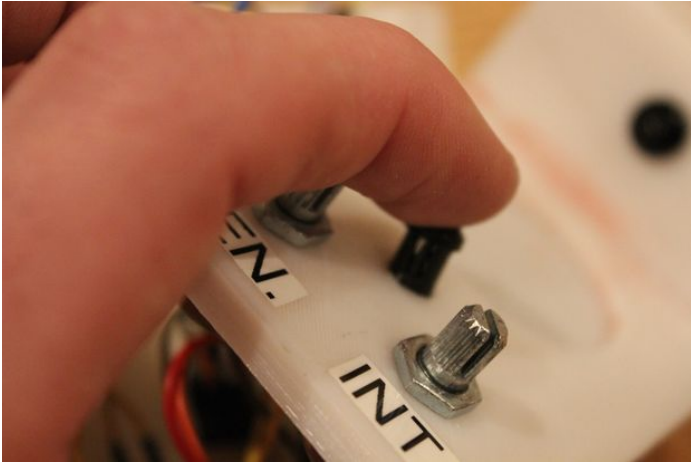


Step 17: Put it All Together

Now we can mount all the knobs, LEDs, and ensure everything is future proof. Using a needle nose pliers, secure the potentiometers, and the reset switch, using nuts. Finally push the indicator LED, and fire sensor's IR LED through their holes and secure them with hot glue.

Make sure everything is nice and tight, as in the next step, we will be closing the case for good!



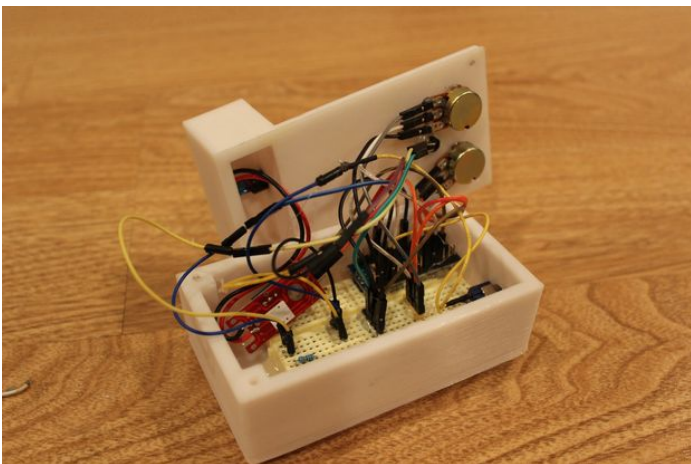


Step 18: Close Up the Case

Awesome! Now is the time to make your circuit working properly so we can close the case. It is a good idea to skip to the next few steps in order to upload some code, and make sure everything is working as expected before we close it. (However, since we have screw holes, it won't be too hard to open it back up for maintenance!)

Begin by orienting the front panel in the correct direction (so the screw holes align). Then make sure any Dupont wires are not sticking too far up or the wrong direct. Using some M3 screws (10mm - 15mm) screw the front panel down to the back.

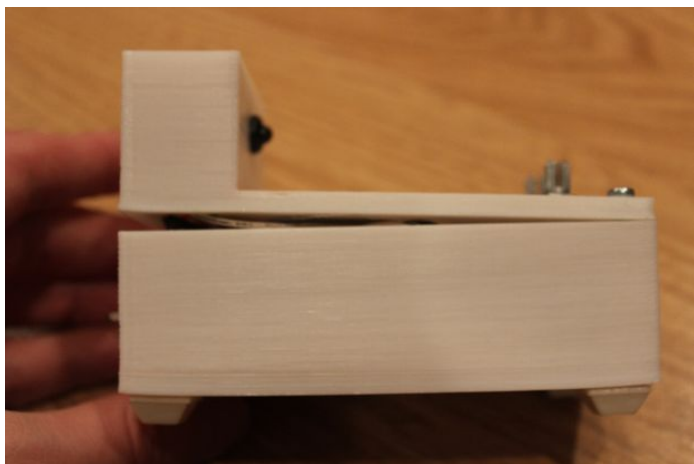
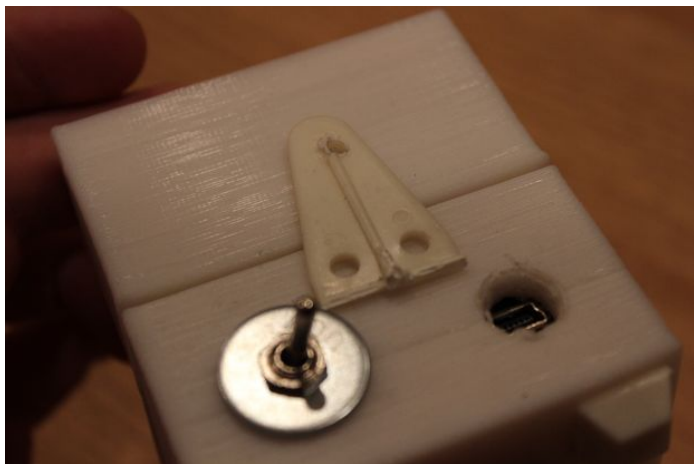
Now the project is looking pretty polished! The next few steps will be working on tidying up the project box to make this controller look even sleeker.



Step 19: Add Bracket (Optional)

A minor flaw in my original design was the lack of a support bracket or screws in the rear of the project box. This causes a slight gap between the front panel and the back. If desired, you can print (or find) a small bracket with screw holes to hold down the front panel. You'll just have to drill two or three 2.5mm holes in the case and secure it with M3 screws.

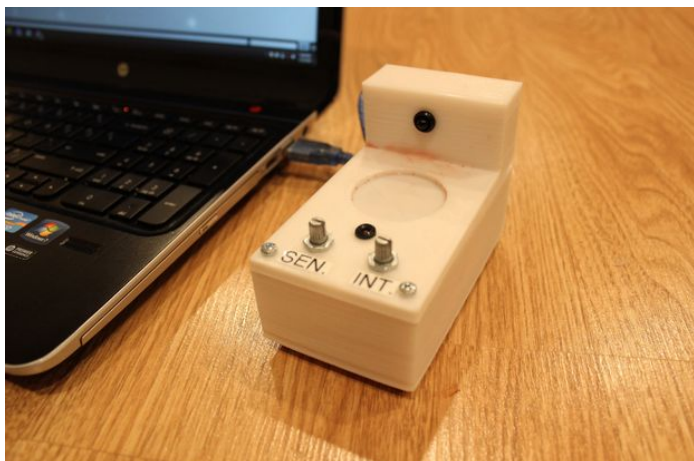
Since while designing the case I added a cavity for the tea candle, adding a bracket is more of a cosmetic touch up than a safety concern. We have one more cosmetic issue to handle before we fire this thing up!



Step 20: Add Potentiometer Caps

To add a little additional usability and style, I added potentiometer caps to my project. It's mainly for looks but also makes it easier to grip the knobs.

Alright. Time to make this controller more than just a pretty paper weight! In the next step we will upload the code!



Step 21: Upload Code

Much of the magic that goes into this MIDI controller is in the code. **For those who don't program, I'll give a general description of how the firmware works:**

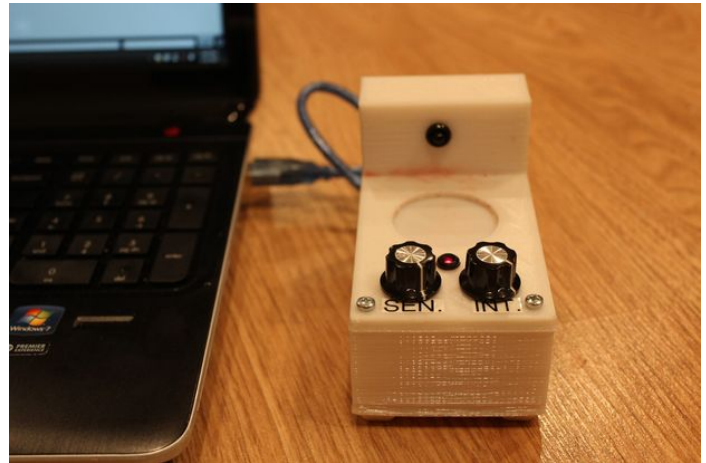
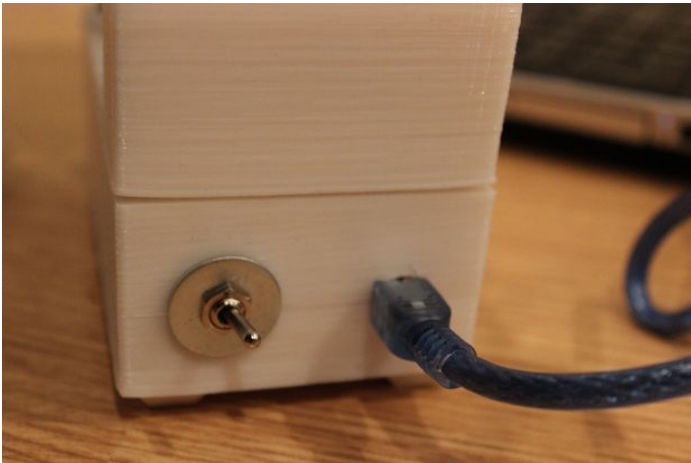
State 1: When first booted, the controller waits for a reading from the flame sensor (asks user to light the candle via the blink pattern on the indicator LED) that is pasted a certain threshold set by the user. Once the threshold is pasted, move onto state 2.

State 2: Take around 5 secs to attempt to take an average of the readings coming from the flame sensor (shows its progress on the indicator LED) for later use. By default, it reads in 50 values, then checks to see which values do not fit the pattern of the readings and discards those values (i.e. if it reads 3, 5, 3, 1, 4, 6, 24. It will throw out the value "24" and continue with the rest). If there are more than 10 values thrown out (since it may alter the accuracy of the average) it restarts state 2. Otherwise, it goes to state 3.

State 3: First read in values from all inputs (flame sensor, potentiometer), based on input values, update what the output is going to be. (Since CC usually works with 127 value resolution, the output is from 0 to 127). Based on the output value, it updates the LED (via PWM) to show the user what value is being sent (it acts to mimic the candle). Finally the output value is sent via serial through the USB and interpreted by the computer for conversion (in Hairless MIDI). State 3 is then repeated unless the flame is removed, or is blown out. If blown out, it goes to state 1 until it is re-lit.

For those of you who do program, **check out the attached code** and comments to learn the details of exactly how the functions and algorithms work.

Once you understand how the firmware functions, boot up Arduino IDE, and upload the code to the Arduino Nano using the appropriate COM port. Next we can set up the computer to convert the serial data to MIDI



File Downloads



fdmc.ino (3 KB)

[NOTE: When saving, if you see .tmp as the file ext, rename it to 'fdmc.ino']

Step 22: Install Hairless MIDI

Hairless MIDI is a program that will convert our Flame-Sense MIDI Controller's serial output into MIDI data. This is needed because the Arduino only sends serial data in the form of strings which does not match the MIDI protocol. If you would like to learn more about MIDI protocol there are plenty of articles about it. My favorite one is [here](#). But anyway, Hairless MIDI is some great free software that we will be using to make the digital audio workstation (DAW) to recognize the data we send to it.

I had no problems installing Hairless MIDI, it works flawlessly and is much easier to install than MIDI Yoke (See next step). To install, [click here](#).

Once installed, you can select an input via the drop down menu and choose your Arduino Nano. The green "light" on the software should blink with each byte string that the software receives. **If you see an error regarding the commands being invalid, make sure you go to options and set the baud rate to 9600.**

Once Hairless blinks green, we need to install one more program, MIDI Yoke which will connect those parsed MIDI commands to actual automation in your DAW!



Step 23: Setup MIDI Yoke

MIDI Yoke software that basically works as a freeway for your MIDI signals. Using Hairless-MIDI we can parse the serial data into MIDI signals, but they have nowhere to go. There is no path connecting them from Hairless to your DAW... well that is unless you install MIDI Yoke.

Visit MIDIYoke.com and download MIDI Yoke. It is freeware, but there are some compatibility issues since the software was originally developed for early versions of windows. You may have to run the installer in compatibility mode for earlier versions in order for it to work. Keep in mind that MIDI Yoke does not have any user interface. It functions more like a driver, where you install it and it will run in the background. To verify that it was installed properly, you should see eight MIDI Yoke inputs and outputs on Hairless-MIDI and your DAW.

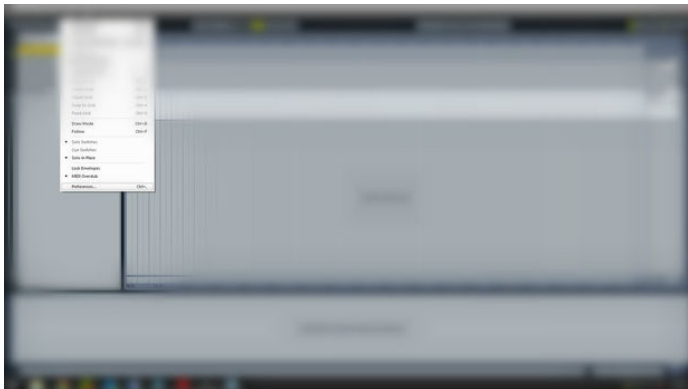
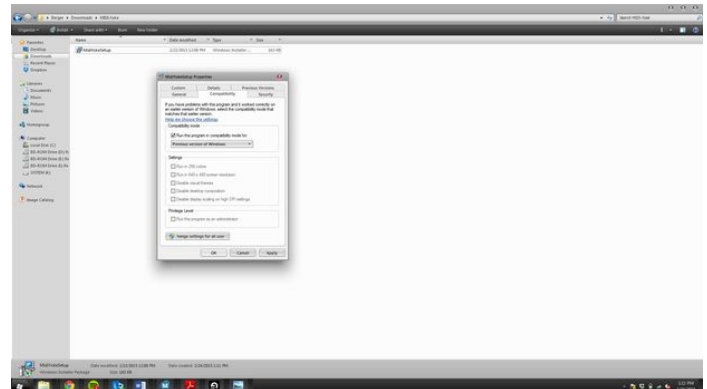
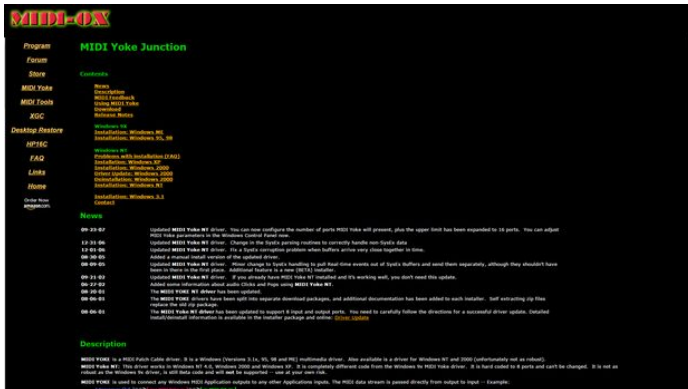
To receive MIDI signals from Hairless-MIDI to your DAW, you'll need to change a few settings in your preferences. I'll show you how to change the settings in Ableton Live, but this should be a similar process for just about any DAW.

First go to Hairless-MIDI and select the output to be MIDI Yoke Output 1
In Ableton, go to Options -> Preferences -> MIDI.

Then set enable MIDI Yoke Output 1 to "Track" and "Sync"

If there is still no input from the device, try restarting Ableton. If that still doesn't work, try to add a control surface with an input from MIDI Yoke 1 and output from MIDI Yoke 1.

Now that we have built this MIDI "freeway" that connects Hairless MIDI to our DAW, lets actually control something!



Step 24: Assign CC in your DAW

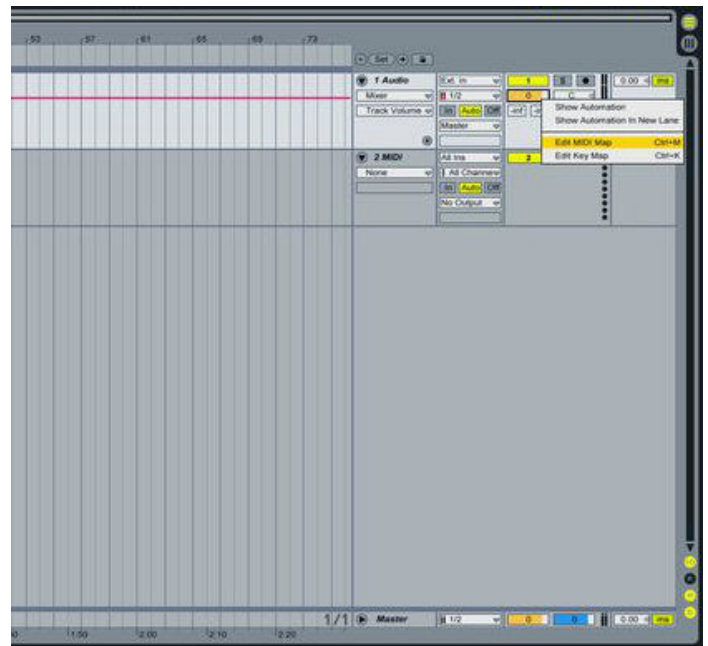
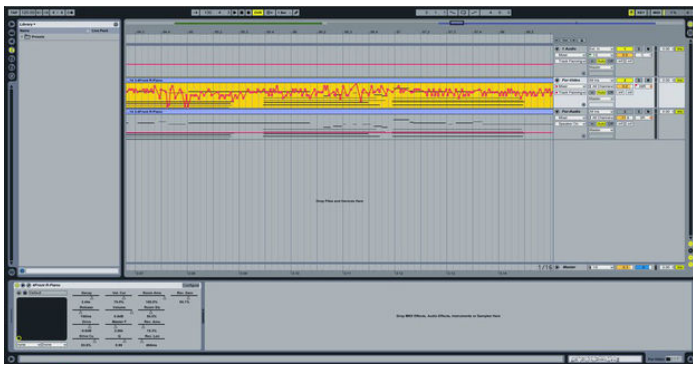
The best part about this Flame-Sense MIDI Controller is that it can control any effect in any DAW. You could use it to control volume, panning, pitch bend, modulation, gain, fuzz, distortion, phaser, etc... The final step to controlling any effect is to assign the midi signal to a virtual input or knob in your DAW. Once again, the instructions will vary depending on your workstation, but I'll demonstrate how to do this in Ableton.

In Ableton, right click on any knob or control on the user interface.

Select "MIDI Map". Light the flame on your controller, wait a few seconds and a CC should populate in the list.

If you still do not see the option for your MIDI controller in the list, double check that Hairless MIDI is running (with the correct input/output) and that the appropriate MIDI Yoke channel is assigned.

Great! Once you see a red line wavering up and down with the candle, you'll know you have completed the project correctly! Not seeing the red line? Double check that you connected everything identically to the schematic. Still stuck? Leave a comment or private message me and I'll gladly help you troubleshoot!



Step 25: Conclusion

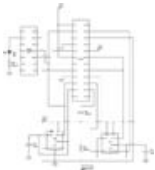
Welcome to the few who can truly say that they have added an authentic analog feel to their digitally produced music. I hope you learned a bit about MIDI commands, using the Arduino, 3D printing, and basic electronics. **If you enjoyed the instructable, and would like to support me, please vote for me in the DIY Audio and Music, Burn it!, and/or Explore Science contests!** Thank you for your interest and support!



Related Instructables



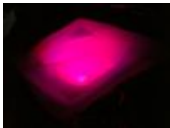
Arcade Button MIDI Controller
by fraganator



MIDI controlled analog FM synthesizer by NYU_MusicTechnc



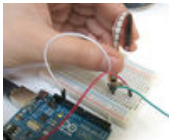
Midi In Me Kit: Midi to Breadboard Access (video) by Tymkrs



Arduino/Ableton Color Organ; MIDI controlled (video) by _Aias



Arduino MIDI Foot Pedal Keyboard by roycetaft



Arduino, Sensors, and MIDI by amandaghassaei

Comments

1 comments Add Comment



tomatoskins says:
This is awesome! I love how the flame gives it such a 'warm' sound. :)

Mar 2, 2015. 8:53 AM [REPLY](#)