

GUITAR CHORD CHART

February 19, 2016



GENERAL

You could call this a remix of [Digital Chord Chart](#) from [Instructables](#) although it is basically a complete redesign. I liked the idea of the digital chord chart, but the original Instructable had very limited features, and it didn't include a usable 3D printable model. This document details the construction of my version of the project. This version supports the following:

- All plastic parts are printable on most reasonably sized 3d printers.
- Finding multiple variations of all standard guitar chords.¹

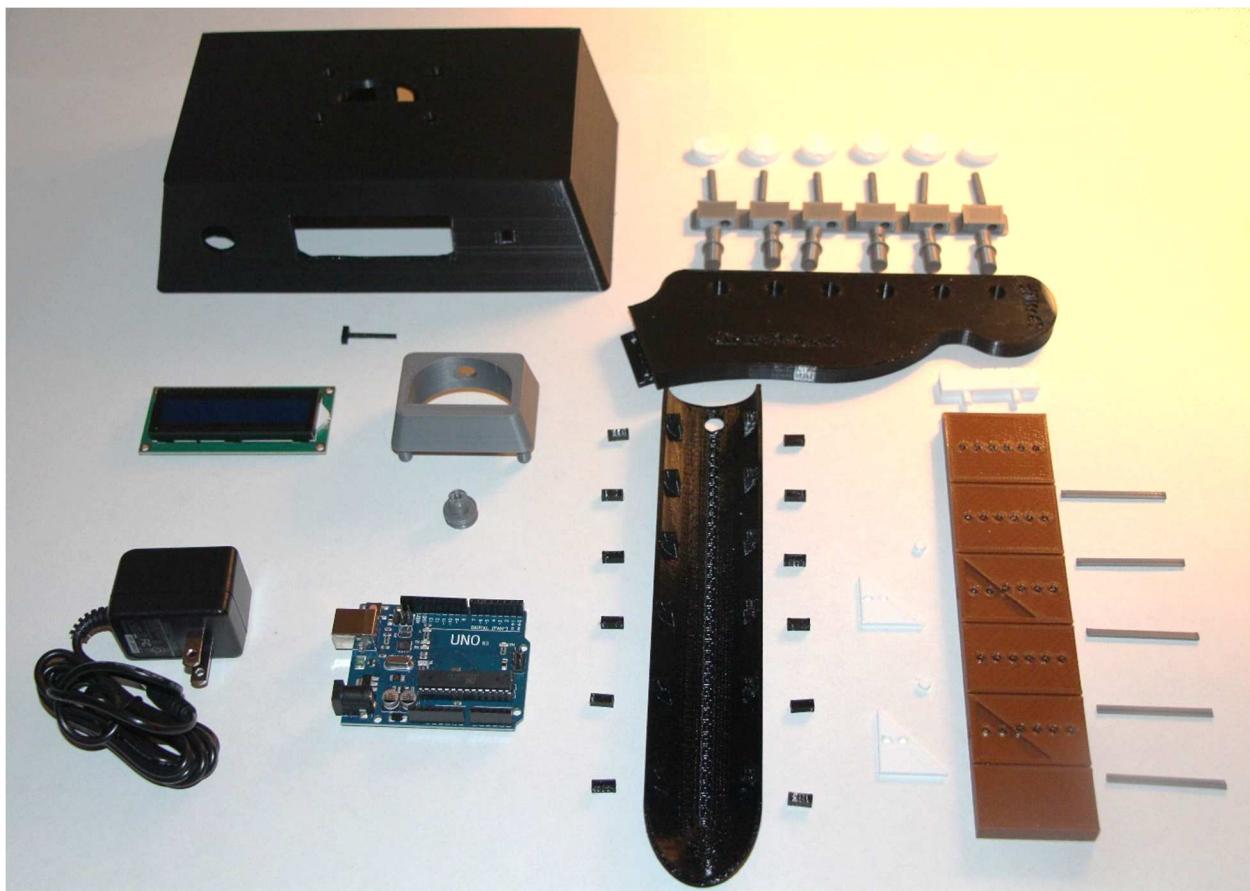
1

¹ [CompleteGuitarChordPoster_WayneChase_FreeEdition](#) was used as a basis for the chord translations. Due to that work being copyrighted, only a small subset of the chord chart is included in the Arduino sketch for this project. Instructions are provided for anyone wanting to extend the set of chords that have been provided.

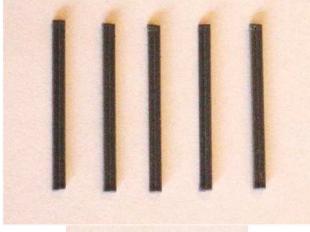
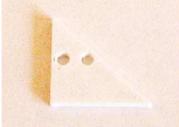
- Reverse chord lookup. The user can specify the string/fret fingering and the device will return the name of the corresponding chord.
- User selectable options for displayed chords.
- A demo mode that can be used to diagnose wiring issues (or just show off).
- A diagnostic mode that can be used to identify IR codes from any IR remote control.
- Adaptable to different IR remote controls.

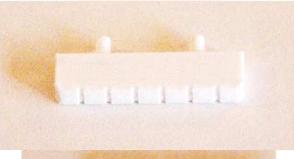
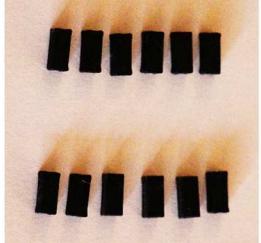
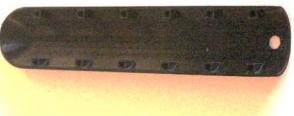
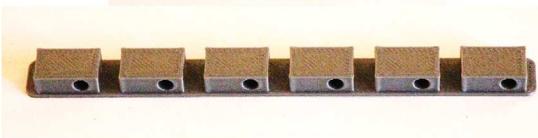
PARTS LIST

All plastic parts can be 3d printed. See Thingiverse???????. Links are provided for all other parts. These are only suggestions. In most cases, other parts can be substituted. The only exceptions are the parts that are exposed on the base (LCD, power jack, power switch, and IR sensor) since the base was designed for these specific parts. If other parts are substituted, the base will probably need to be modified. The LEDs must be 3mm to fit the holes in the finger board, but any 3mm LEDs should work.



Plastic (3d printed) Parts

Part	Quantity	Suggested Color	Need Support ?	Picture
Base	1	Black	Yes	
Input Device Holding Rod	1	Black	No	
Neck Support	1	Silver	No	
Neck Support Pin	1	Silver	No	
Fingerboard	1	Brown	Yes	
Fret	5	Silver	No	
Fret 3 Inlay	1	White	No	
Fret 5 Inlay	1	White	No	

Circular Fret Inlay	2	White	No	
Nut	1	White	No	
Mating Post	12	Black	No	
Neck Back	1	Black	Yes	
Headstock	1	Black	Yes	
Tuner Post	6	Silver	No	
Tuner Assembly	1	Silver	Yes	
Tuner Shaft	6	Silver	No	
Tuning Knob	6	White	No	



Miscellaneous Parts

Part	Quantity	Link	Comment
Arduino Uno	1	Link	Any cheap Arduino will do.
LCD Display (16 columns x 2 rows)	1	Link	The linked display is preferred.
3 mm White LED (50 pack)	1	Link	30 used. Any 3mm LED is OK.
IR Receiver AX-1838HS	1	Link	
Panel Mount 2.1 mm Power Jack	1	Link	Any jack of the same size.
16mm Illuminated Power Switch – Blue	1	Link	Any switch of the same size.
1K Ohm Resistor, 1/8 watt (5 pack)	2	Link	6 1K resistors are needed.
Panel Mount 10K Potentiometer	1	Link	
Stranded Core Wire – 26AWG – Black	1	Link	This wire is great.
Stranded Core Wire – 26AWG – Red	1	Link	
Stranded Core Wire – 26AWG – White	1	Link	
Stranded Core Wire – 26AWG – Blue	1	Link	
Stranded Core Wire – 26AWG – Green	1	Link	
Stranded Core Wire – 26AWG – Yellow	1	Link	
Stranded Core Wire – 26AWG – Grey	1	Link	
Hookup Wire Assortment – 22AWG	1	Link	Only need about 1' of each color.
Female Headers	1	Link	Optional. Could wire to IR.
Breakaway Headers (straight) (10 pack)	1	Link	
Stripboard 50mm x 80mm	1	Link	Optional. Could wire to header.
Power Supply – 9V – 1000ma	1	Link	
4" Cable Ties (100 pack)	1	Link	Only need about 10.
3/32" Heat Shrink Tubing	1	Link	
3/8" Rubber Adhesive Feet (16 pack)	1	Link	Only need 4 feet.
M3-05 x 6mm Nylon Machine Screw (100 pack)	1	Link	Only need 7 screws.

REQUIRED TOOLS

- Soldering Iron
- Solder
- Side Cutters
- Needle Nose Pliers
- Small Wrench or Pliers
- Knife and/or Wire Strippers
- Small Phillips Screw Driver
- Small Files or Sand Paper
- Multimeter
- Ruler
- Helping Hands with Magnifier (optional but recommended)
- Computer with Arduino V1.7.8 or Later Software Installed
- A-B USB Cable (for Arduino programming)
- Zip Tie Tool (optional)
- 3d Printer (optional)

PRELIMINARY WORK

We need to do a few things before work on building the Guitar Chord Chart can begin. This section covers the preliminary work.

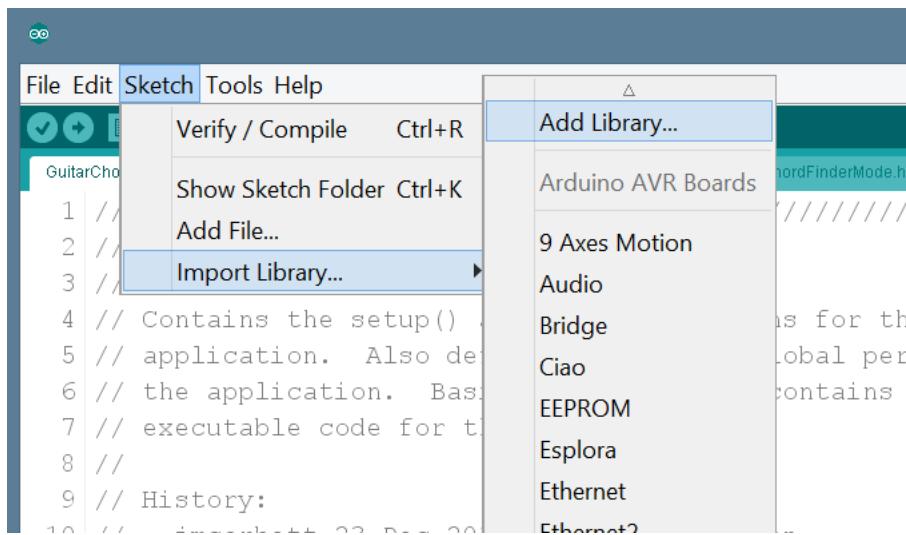
Install the Arduino Libraries

The following libraries are needed in order to run the main Guitar Chord Chart application.

Library	Link
EEPROM	https://github.com/arduino/Arduino/tree/master/hardware/arduino/avr/libraries/EEPROM
IRLib	https://github.com/cyborg5/IRLib
LiquidCrystal	https://github.com/adafruit/LiquidCrystal
TimerOne	https://github.com/PaulStoffregen/TimerOne

In general, to install these libraries, copy the zip file from the provided link. Extract the zipped files to a known location (for example the Desktop) into a folder named the same as the library name in the table above. Copy the new folder to the (hopefully) existing **C:\Program Files (x86)\Arduino\libraries** folder.

If your computer doesn't have this folder, then you'll need to find where you installed the Arduino software and go to the "libraries" folder under it. Then restart the Arduino software and from within the Arduino application, select **Sketch->Import Library...->Add Library...**



Select the folder you just copied to the Arduino libraries folder. That should do it. The library should now be ready for use by the Arduino software.

One exception is the **TimerOne** library. I have not been able to use this library as it is intended. Instead, I've had to copy the **Timer.h** and **TimerOne.cpp** files from the zip archive directly into the **GuitarChordChart** folder. You'll need to do the same.

Preparation – Strip Board (optional)

I had some trouble wiring directly to headers without shorting adjacent pins. As an experiment I tried using strip board, with some success. So the use of stripboard is optional. This section describes how to prepare and use the stripboard option, if you decide to use it.

The stripboard can be used on all connections to the Arduino. I found it better to handle the connections in groups. The groups I used were:

- Power related connections – 4 pins.
- LCD related connections – 6 pins.
- LED fret related connections – 5 pins.
- LED string related connections – 6 pins.
- IR sensor related connections – 1 pin.

The first step is to cut the stripboard to size for each group. I used 3 holes wide for each except the power related group. For the power related group I used 4 holes wide. Thus I cut the stripboard as follows:

Power	4 holes x 4 holes
LCD	6 holes x 3 holes
LED Frets	5 holes x 3 holes

LED Strings	6 holes x 3 holes
IR Sensor	1 hole x 3 holes

To use the cut stripboard, just solder the first row of holes to the male header, then solder connection wires to any of the holes corresponding to the header hole desired.

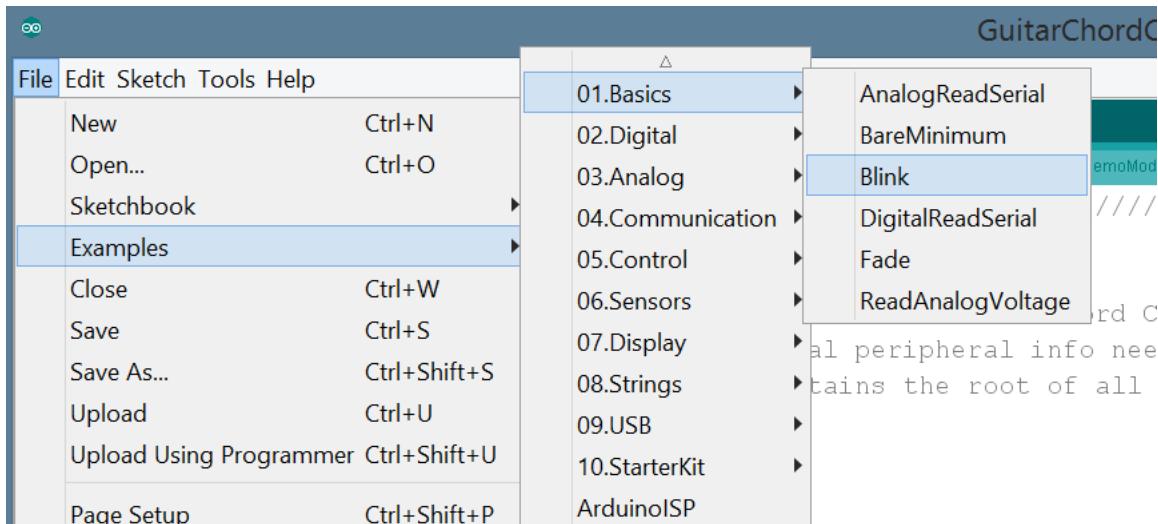
Header wiring

When wiring to a male header, it is easy to melt the plastic that holds the header pins together. For this reason, it is best to insert headers into female header sockets before soldering. This will insure that the plastic on the male header doesn't deform, and will keep all the header pins straight.

Test the Arduino

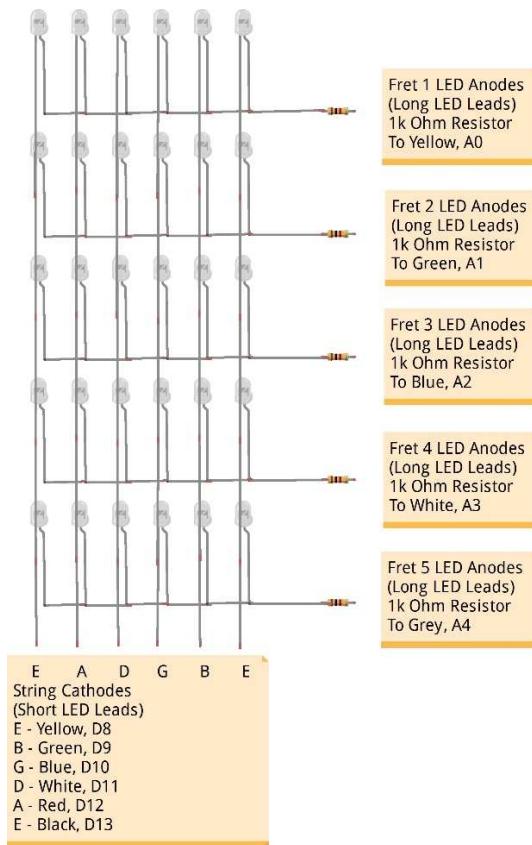
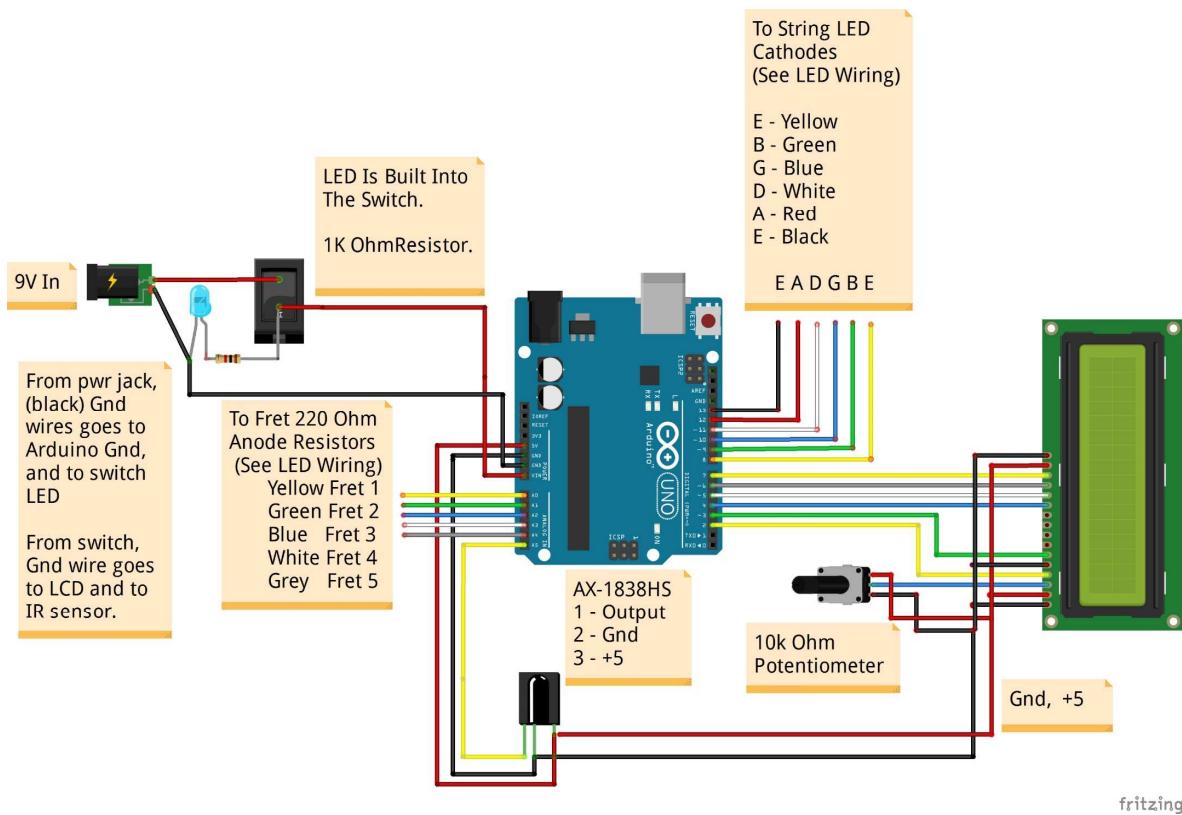
The first step is to test the Arduino to insure that it is functioning correctly. Do the following:

1. Connect the A-B USB cable between your Arduino and computer.
2. Start the Arduino software.
3. Select your Arduino port via the **Tools->Port** selection.
4. Load the Blink example code into your Arduino.
5. Observe that the Arduino LED will be flashing.



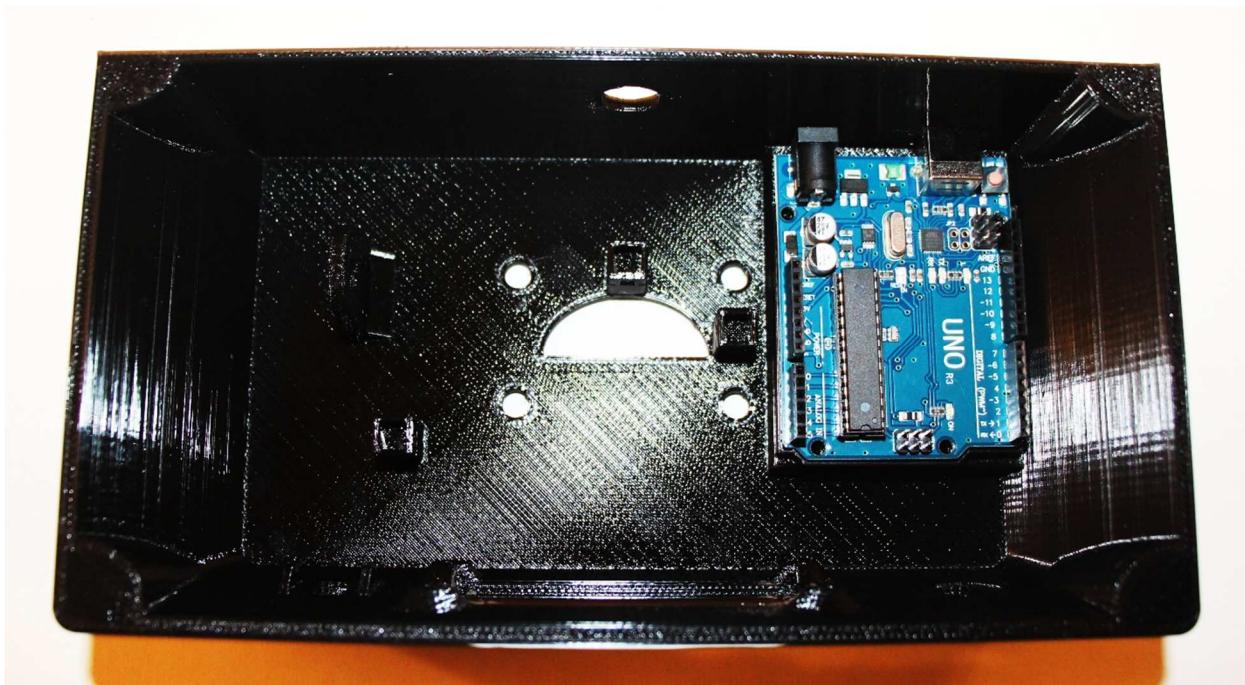
Wiring Diagram

It might be useful to know beforehand what is coming. The following are the wiring diagrams for the device. The first diagram is the wiring for the base, and the second is the wiring for the LEDs in the neck.



MOUNT THE ARDUINO

Now that the Arduino is known to function properly, mount it into the base. Insert the Arduino into the base with the USB connector protruding from the hole in the back of the base. A little force may be needed to get the Arduino to seat properly. Once the Arduino is seated, secure it with 4 nylon M3 x 6mm nylon screws.



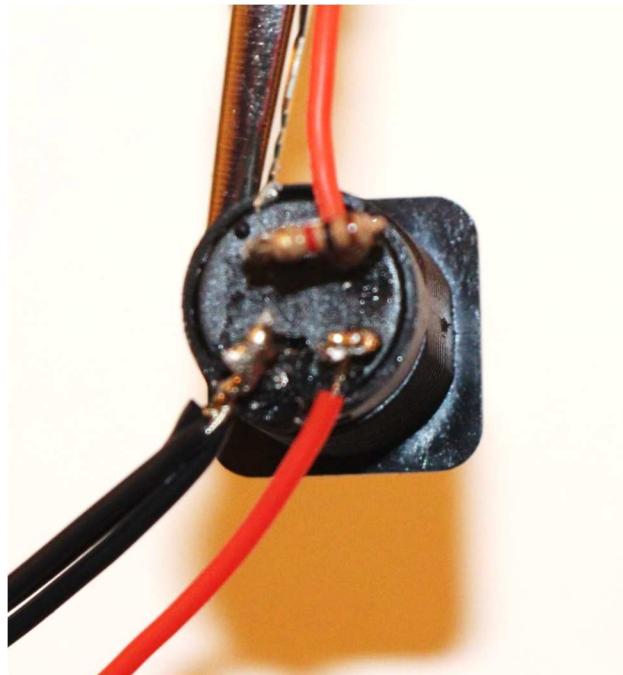
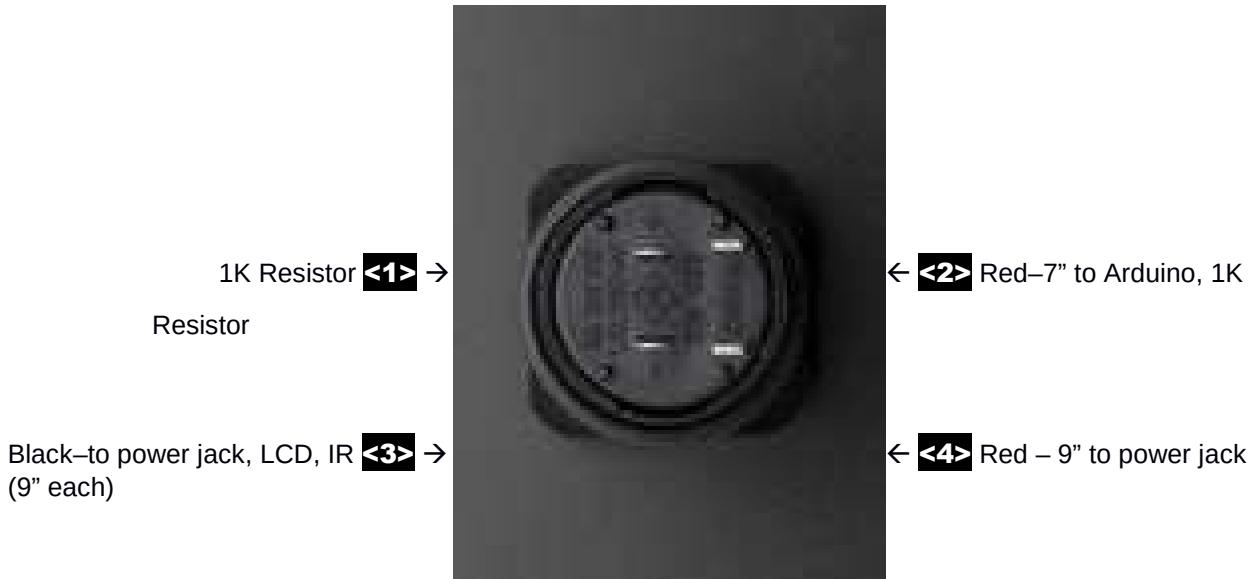
INSTALL THE POWER SWITCH

Pre-wire the Power Switch

See the diagram below. Solder black ground wires, power wires and 1K resistor to the power switch.

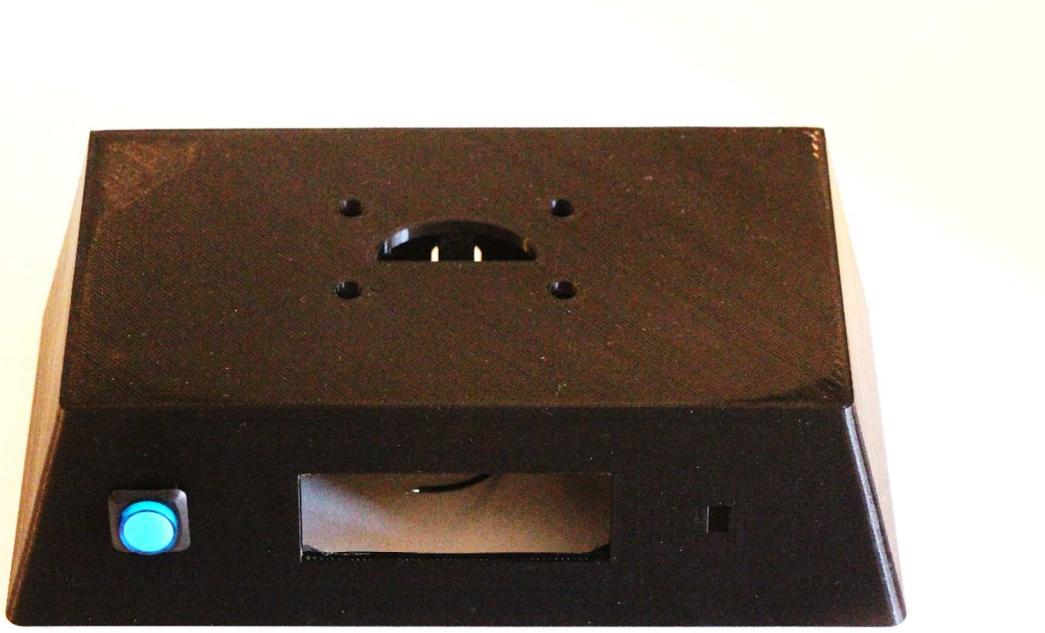
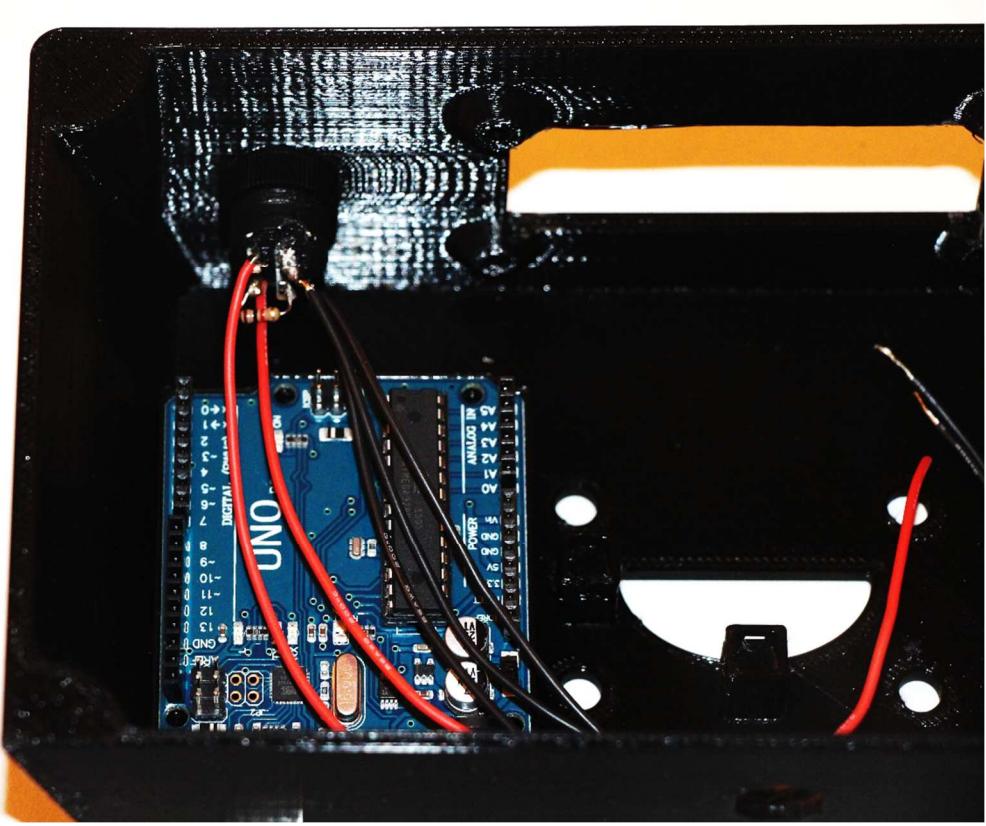
Leave the other end of the wires free for now. Wiring details follow:

- <1>** – Solder one end of a 1K resistor to this pin. Keep the leads as short as possible.
- <2>** – Solder the other end of the 1K resistor to this pin. Also solder a 7" piece of red wire to this pin. This wire will eventually terminate at the Arduino VIN pin, but leave the other end free for now.
- <3>** – Solder 3 9" black (ground) wires to this pin. They will eventually terminate at the power jack, the LCD, and the IR sensor. For now, leave the other ends free.
- <4>** – Solder a 9" red (+9V) wire to this pin. This wire will eventually terminate at the power jack, but leave the other end free for now.



Mount the Power Switch

Remove the mounting nut from the power switch if it is not already removed. Fish the power switch wires through the power switch hole in the base. Insert the switch into the power switch hole being careful to observe the keying. The power switch is keyed via 2 flat sides on the hole and on the power switch. Make sure that the switch is oriented so that the flat sides of the power switch match the flat sides of the power switch hole. Once the power switch is seated in the base, fish the wires through the power switch mounting nut, and screw the mounting nut onto the switch. Note that the mounting nut has one side that has bumps. The mounting nut should be oriented so that it the bumps are against the base.



INSTALL THE POWER JACK

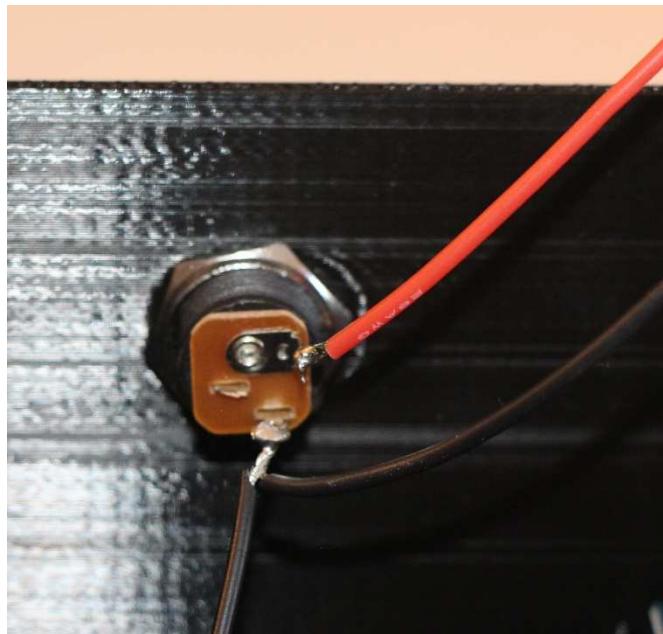
Mount the Power Jack

Remove the mounting nut from the power jack and insert it into the base being careful to observe the keying (the power jack and its base mounting hole are flat on 2 sides). Secure the power jack with the mounting nut.



Wire the Power Jack

- Solder the red +9V wire from <4> of the power switch pre-wiring section to the large post on the back of the power jack.
- Cut a 5" length of black wire and solder it and one of the black ground wires from <3> of the power switch pre-wiring section to the lower post on the back of the power jack.
- Note that the middle post on the power jack is unused. It has been cut off in the picture below.



Check Connections

Using the multimeter, check that there are no shorts in any of the connections that have been wired so far. Also check that all black wires are connected by verifying that there is continuity between the ends of each of the black wires. Likewise, check that all red wires are connected when the switch is closed by verifying that there is continuity between the ends of each of the red wires. With the switch closed, verify that there is no continuity between any of the red wires and any of the black wires.

Apply Power and Test

Now that the wiring has been verified, it is safe to apply power. Connect the power adapter to the power jack and verify that the switch lights up when it is on, and that the light goes out when the switch is off. In addition, it is a good idea to check that the multimeter shows +9 volts between any red and black wires when power is attached and the switch is on.

Run Power to the Arduino

Now that the incoming 9V power has been verified, it is safe to apply the 9V power to the Arduino's VIN input. Be sure to disconnect the power supply before performing this step.

Solder the remaining free red wire from the power switch (+9 V) to one end of a 4 pin male header (or use the stripboard method). Solder the remaining free black wire from the power jack (Ground) to the third pin of the 4 pin male header. Now insert the 4-pin header into the Arduino such that the red wire aligns with the Arduino VIN pin, and the black wire aligns with one of the Arduino GND pins.

Assuming that the Blink program is still installed on the Arduino, when power is applied, and the switch turned on, the Arduino's LED should be seen to blink.

INSTALL THE LCD

Pre-fit the LCD and Potentiometer in the Base

The base was designed to fit the LCD snugly. Due to variances in 3d printing, the openings for the LCD and potentiometer may need some slight adjustments. Holding the base firmly, attempt to insert the LCD into its opening in the base (connections towards the top of the base). If this is too difficult, inspect the opening to determine where the LCD is binding. Use a file or sandpaper to remove any offending plastic till the LCD fits snugly into its opening.

Similarly, the hole for mounting the potentiometer may be too small to accept it. If this is the case, use a file or sandpaper to enlarge the hole until the potentiometer easily slides into it.

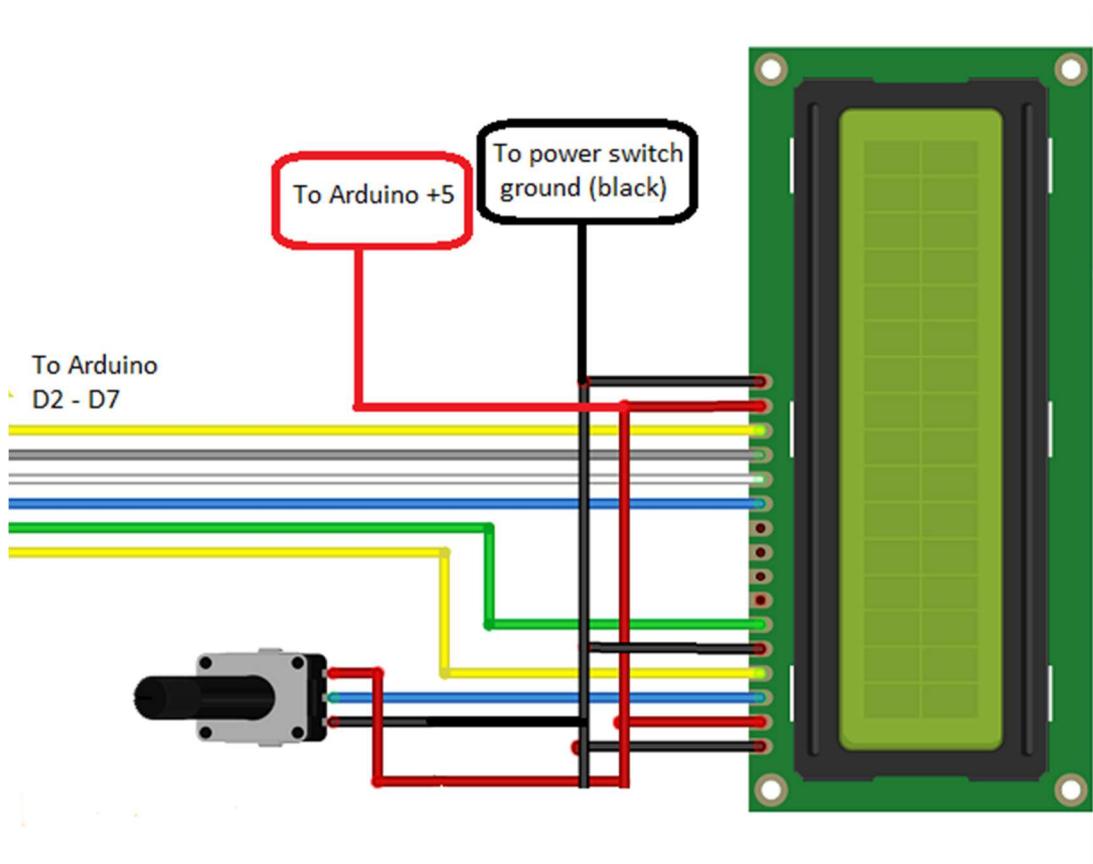
Pre-wire the LCD

All of the pre-wiring of the LCD can be done before the LCD is mounted to the base. See the diagram below for details. Note the following:

- Pin 1 of the LCD module is at the bottom of the diagram, pin 16 is towards the top.
- All wires should be soldered to the back of the board in order to leave room on the front of the board for mounting the LCD module to the base.
- Each signal wire (the wires that will end up at Arduino pins 2 through 7) should be 7" long.
- The black ground jumper wires that connect between pins 1 and 5, and between pins 5 and 16 of the LCD module should be about 2" each.
- The red +5 V jumper wire that connects between pins 2 and 15 of the LCD module should be about 3".

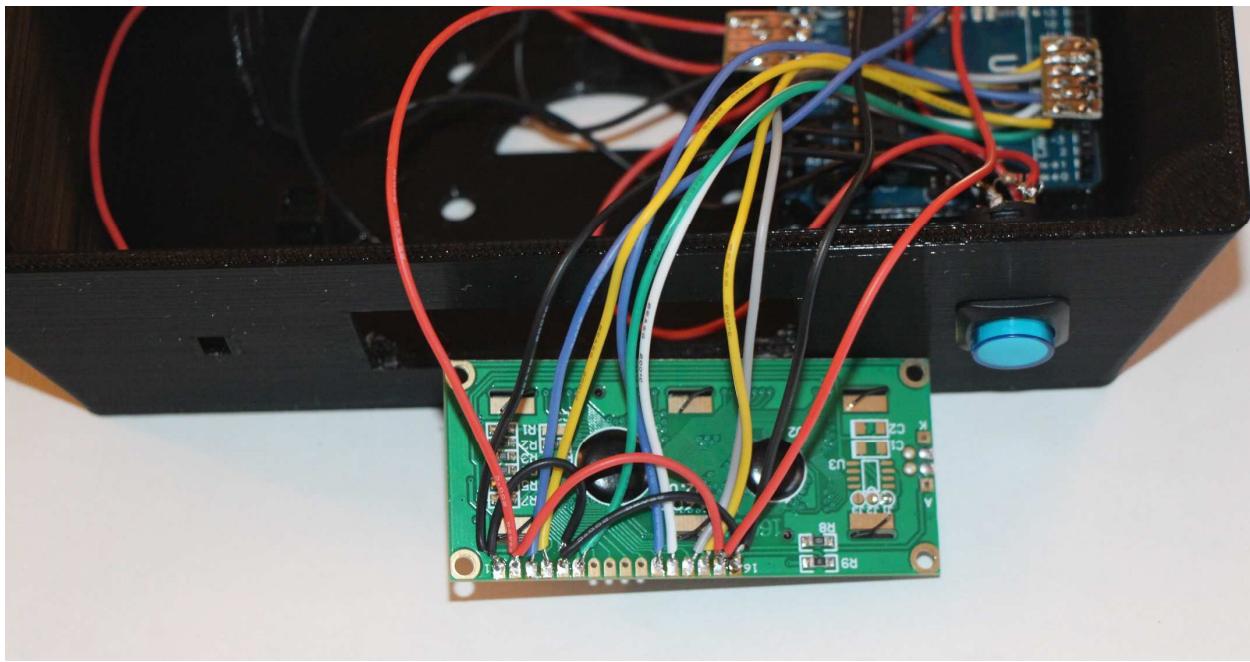
Follow these steps to pre-wire the LCD module:

1. Solder the signal wires to the LCD module. Use the colors shown on the diagram. Leave the other end of the signal wires free for now.
2. Solder a 7" length of blue wire to pin 3 of the LCD module. Solder the other end of the blue wire to the center post of the 10K potentiometer.



3. Twist the end of one of the ground wires from the power switch (see **<3>** of the power switch pre-wiring section above) to one of the 2" black jumper wires, and solder them to pin1 of the LCD module.
4. Twist the second 2" black jumper to the free end of the other 2" black jumper, and solder them to pin 5 of the LCD module.
5. Cut a piece of black wire to a length of 6". Twist the end of this wire to the free end of the black 2" jumper from the previous step, and solder them to pin 16 of the LCD module.
6. Solder the other end of the 6" black wire to either end post of the potentiometer.
7. Cut two 7" lengths of red wire. Twist one end of one of these wires to one end of the 3" red jumper wire, and solder them to pin 2 of the LCD module. Twist one end of the other 7" red wire to the other end of the 3" red jumper, and solder it to pin 16 of the LCD module.
8. Solder the other end of the 7" red wire from pin 16 of the LCD module (see the previous step) to the empty post of the 10K potentiometer.
9. Solder the other end of the 7" red wire from pin 2 of the LCD module to the fourth pin of the Arduino power group header (or stripboard). This red wire should connect to the end of the header that currently has nothing else attached to it.
10. Solder the other end of the signal wires (see step 1 above) to a 6 pin header (or stripboard) in the order shown in the diagram (see the table below).

LCD Pin	Arduino Pin	Header Pin	Color
4	2	1	Yellow
6	3	2	Green
11	4	3	Blue
12	5	4	White
13	6	5	Grey
14	7	6	Yellow

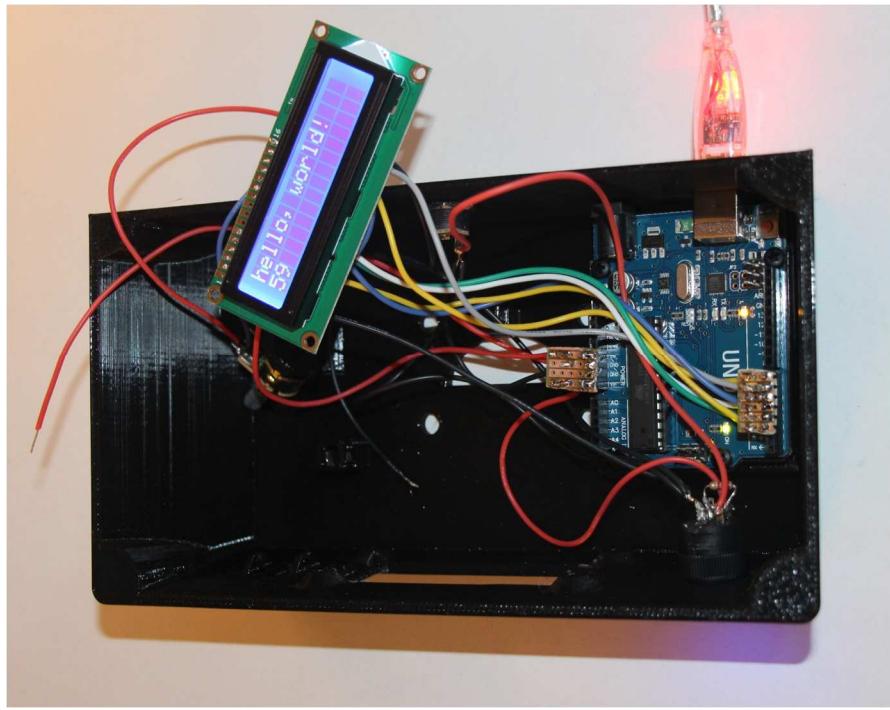


Check the LCD Wiring

Use a multimeter to check for shorts and continuity on all connections. If all connections look OK, then insert the header into the Arduino such that pin 1 of the header aligns with D2 on the Arduino, and pin 6 of the header aligns with D7 of the Arduino. Also insert the power header into the Arduino if it is not already there, making sure that the +5V pin is toward the back of the base.

Test the LCD wiring

Attach the A-B USB cable between the computer and the Arduino. Load the IrTest program into the Arduino. Turn the potentiometer until characters can be seen on the LCD display. The LCD should now display "hello world" on first line, and the number of seconds since power-up counting up on the second line.



Mount the LCD

Remove power from the base before continuing.

Now that the LCD is working it is time to mount it into the base. Carefully insert the LCD into the base.

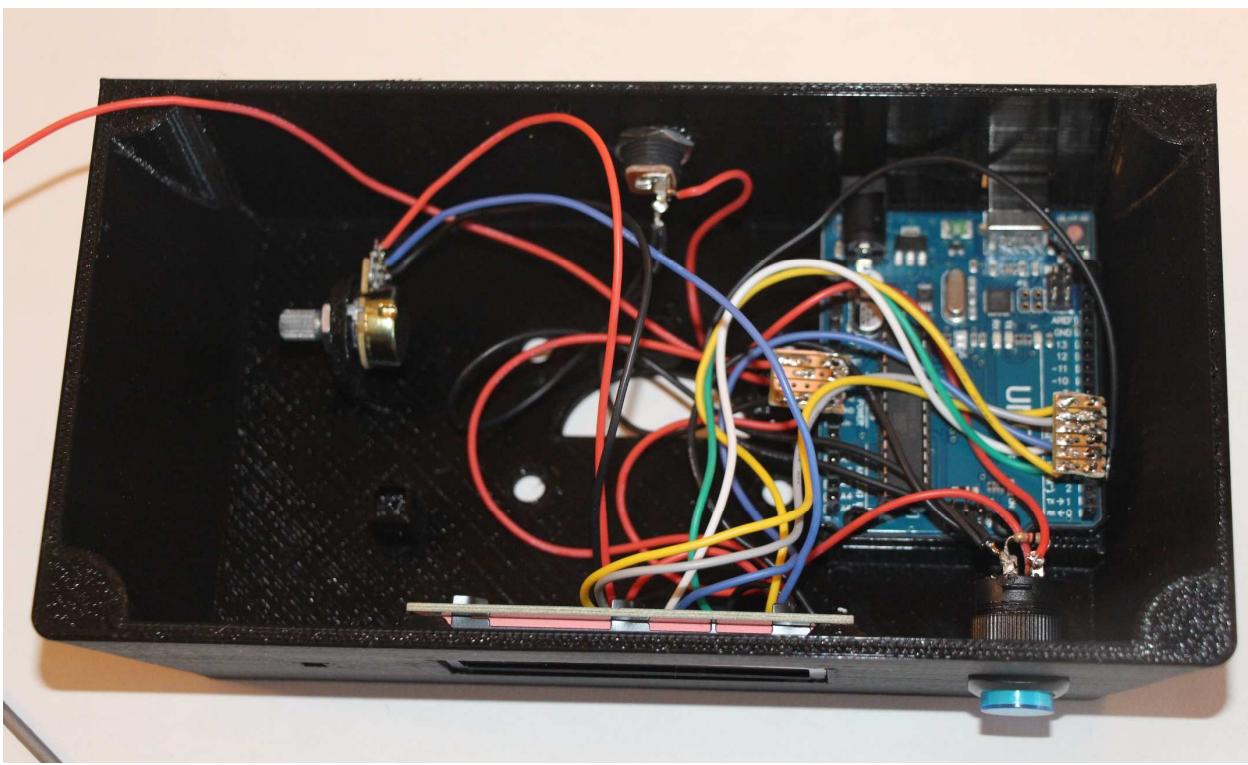
Mount the potentiometer in its mounting bracket. Note that the potentiometer mounting bracket contains a small keying hole on top which can accept a tab that the potentiometer contains. Do not yet secure the potentiometer with its mounting nut, just in case a problem is introduced while mounting the components.

Re-apply power and verify that the LCD is still working correctly. If not, remove the LCD and verify all connections.

Secure the LCD with 3 M3 x 6mm nylon screws. Note that it will not be easy inserting the top left screw, and it will be impossible to insert the top right screw. Due to the snug fit, using 3 screws to secure the LCD should be sufficient.

Secure the potentiometer via tightening its mounting nut.

Re-apply power and verify the LCD is still working.



INSTALL THE IR SENSOR

Mount the IR Sensor

Insert the IR sensor into the base with its leads facing the bottom of the base. It is possible that the IR sensor hole in the base could need some filing in order to make the IR sensor fit. Once the IR sensor is in place, secure it by inserting the Input Device Holding Rod into the through the holes in the two holding rod brackets that are on either side of the IR sensor. This may require the application of some force on the back of the IR sensor to seat it properly into the base.

Once the IR sensor is mounted, use the needle nose pliers to bend the three 3 IR sensor leads toward the back of the base. The bend should be about $\frac{1}{2}$ " below the sensor. Using the side cutters, trim the 3 IR sensor leads to about $\frac{1}{2}$ " after the bend.



Wire the IR Sensor

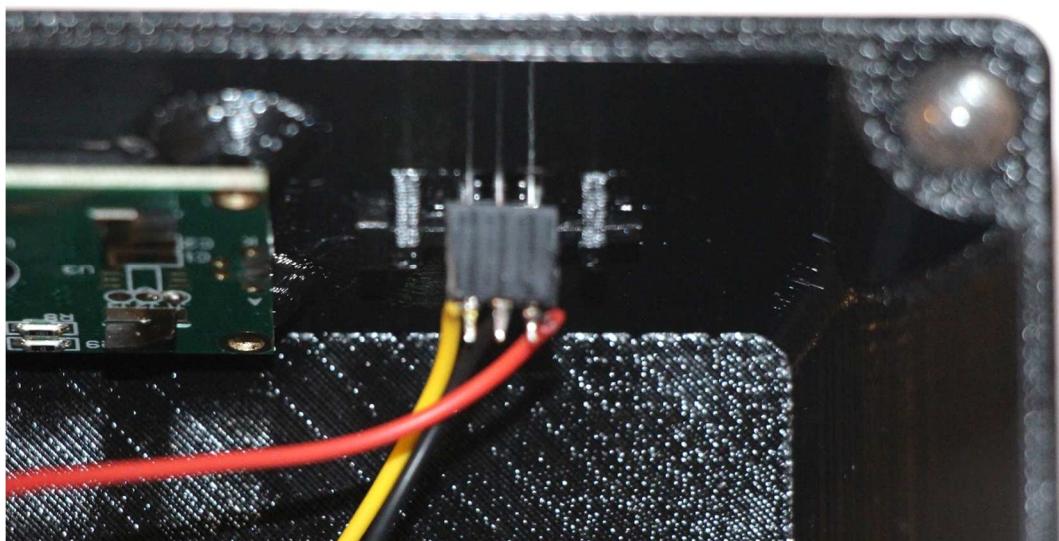
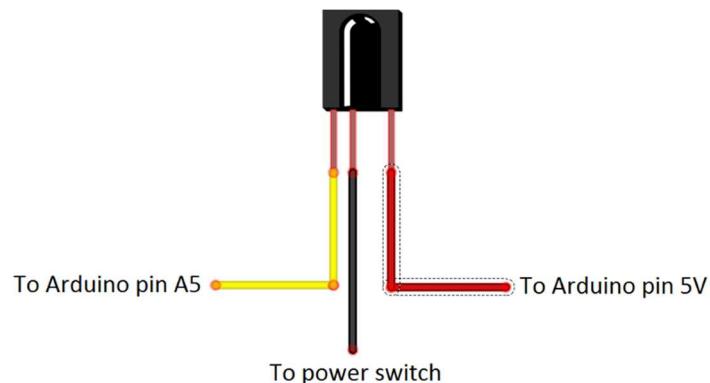
Two options exist for wiring the IR sensor.

- Solder directly to the IR sensor's leads.
- Solder to a 3 pin female header which will mate with the 3 IR sensor leads. I prefer this method.

Regardless of the wiring method used, the following must be done:

1. Cut a 9" yellow wire and solder one end to the leftmost pin of the IR sensor (or female header). Solder the other end to the single pin male header. It will be attached to Arduino pin A5.
2. Solder the remaining black wire from the power switch to the middle pin of the IR sensor (or female header).

3. Cut a 9" red wire and solder one end to the rightmost pin of the IR sensor (or female header). Solder the other end to the Arduino power group 5V pin.



Check the IR Sensor Wiring

Using the multimeter, verify that there are no shorts between the pins of the IR sensor (or female header). Also verify that there is continuity between the IR sensor pins and Arduino pin A5, GND, and Arduino pin 5V respectively. Once verified, connect the female header (if used) to the IR sensor, observing correct orientation. Also connect the 1 pin male header to Arduino pin A5.

Test the IR Sensor

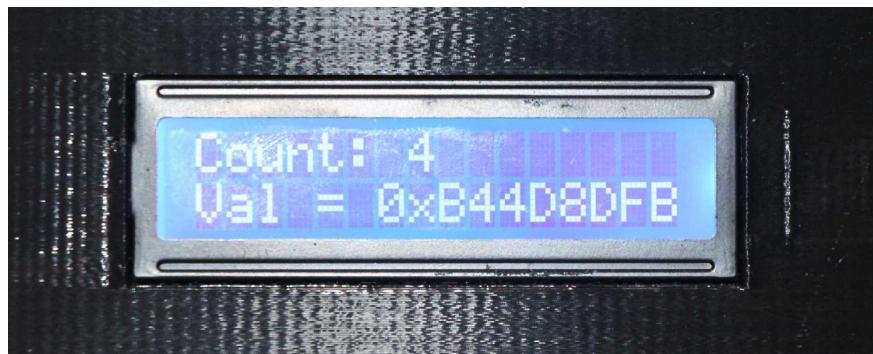
Be sure the Arduino IRLib library has been installed as described earlier.

Load the IrTest sketch into the chord chart's Arduino and apply power.

When the IrTest sketch starts, the LCD display should show the following:



At this point it is waiting to see a key press from the IR remote control. Press any key on the IR remote. The LCD display will show the count of the number of IR key presses along with the hexadecimal code of the last pressed key:



Note that the 8-digit hex code should be unique for each key that is pressed. At this point, the value of the IR code is not important. The important thing is that the IR sensor recognized the receipt of an IR code, and that it is repeatable and unique for each unique IR remote key.

The sketch will continue to accept key presses and return their associated value forever.

Note that in my experience, the only reliable IR codes are Sony codes. And even these don't always work correctly all the time. The important thing is that IR codes are seen by the IR sensor.

BUILD THE NECK ASSEMBLY

Pre-test the LEDs

It is difficult to replace an LED once the fingerboard wiring is completed. For this reason, it is advisable that all LEDs be tested before installing them into the fingerboard. I have found that some of the cheap LEDs from the web have a non-zero chance of being dead on arrival.

An easy way to test an LED is to set the multimeter to continuity testing mode and connect the leads to the LED. Some voltmeters supply enough current in continuity mode to light an LED. If yours doesn't do this, then a battery could be used in its place. If the LED doesn't light, then reverse the leads and try again. If the LED doesn't light in either case, then it is probably bad. Discard it.

You'll need 30 good LEDs for the next step.

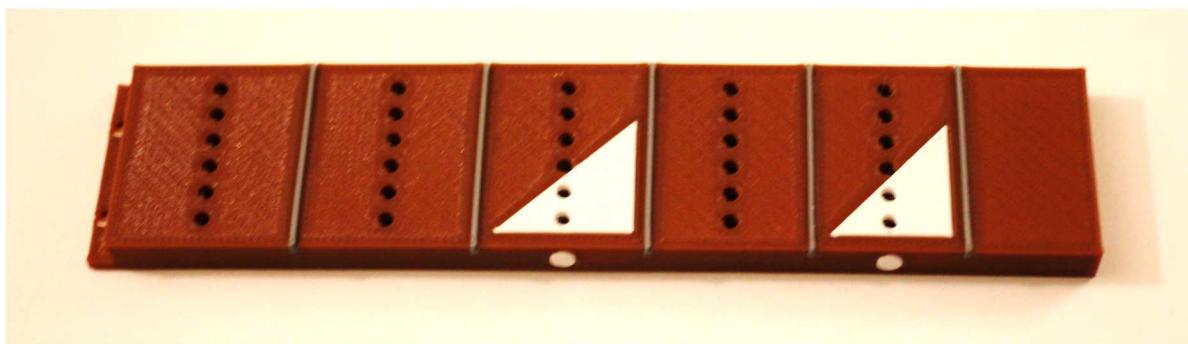
Prepare the Fingerboard

Before installing the LEDs, it is best to install the frets and the inlays into the fingerboard. The frets will keep the fingerboard from lying flat on the working surface, and give the LEDs a bit of room to protrude from the fingerboard when inserted. The inlays should be installed so that the LEDs that are associated with them will penetrate the inlays when inserted. This will eliminate the possibility of having a pre-installed LED get pushed out if the inlay is later added.

The frets fit snugly into the fingerboard fret slots. Some 3d printers may leave a small lip on either side of the fingerboard that can interfere with installing the frets. In this case, it has been found that using a sharp knife or X-acto knife to remove such lips is helpful. Once the fret channel is cleared, simply slide a fret into each of the fret slots as far as possible. Once they are all installed, push them all in flush with the fingerboard by holding the fingerboard on its side against a desk or other flat surface, and apply enough force to move all frets into the correct position.

The inlays should be installed next. Note that the fret 3 triangular inlay is slightly larger than the fret 5 inlay. The triangular inlays should be installed with the shiny side up. Install the triangular inlays by seating one edge of the triangle, then forcing the opposite corner into position. A wooden handle may be used to "bump" the inlay into place by tapping it with sufficient force. In the worst case, it may be useful to use a file or some sandpaper to remove a little plastic from one edge of the inlay. If this is done, be careful to not remove too much material.

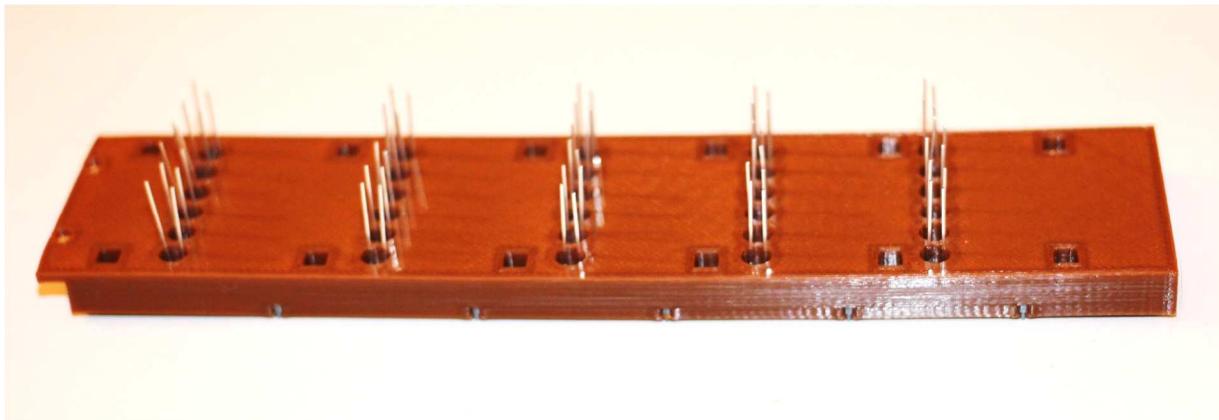
It is also a good time to insert the circular side inlays. Both of these inlays are the same size, so either one can be installed into either fingerboard side inlay hole. As with the triangular inlays, the circular inlays will be very snug. They may need to be tapped in to get them to seat flush with the fingerboard.



Install the LEDs Into the Fingerboard

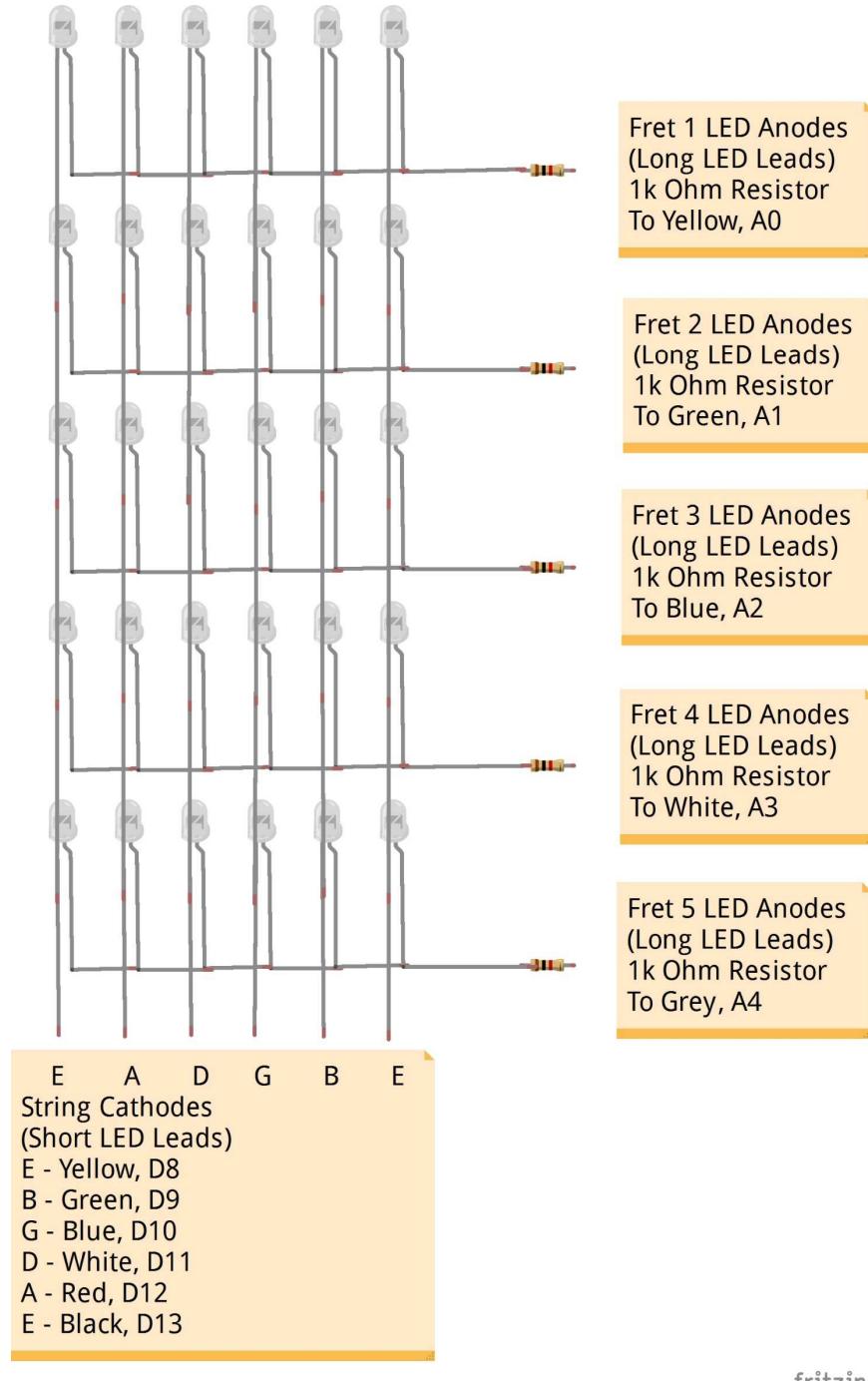
It is time to install the LEDs into the fingerboard. The 30 holes in the fingerboard are designed to snugly fit the 30 3mm LEDs. Each LED must be mounted with its longer lead toward the top of the fingerboard, and its shorter lead toward the bottom. It takes a moderate amount of force to install each LED. The best way to insert them is to hold the LED with needle nose pliers by one of leads. Now, with the LED oriented

with the longer lead up, press the LED down into its corresponding fingerboard socket with enough force to get it to “snap” or “pop” into place. Be sure that all LEDs are seated to the same depth. This is best done by viewing the front of the fingerboard and verifying that all LEDs protrude the same distance.



Wire the LEDs

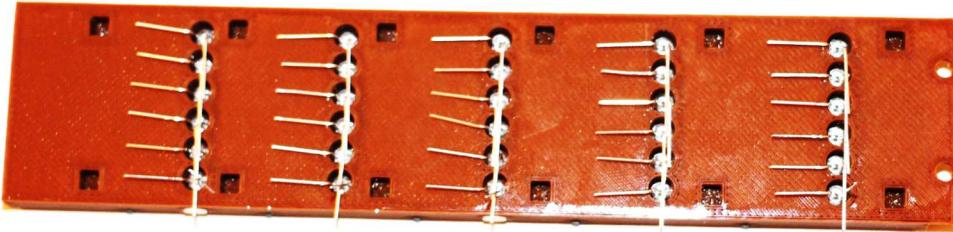
Note: The following steps require the LED leads to be bent. Use care when bending the LED leads. Bending them more than once or twice in the same spot may cause them to break off. Also, while bending the LED leads, be careful not to pull the LED out of its mounting hole in the fingerboard.



fritzing

Bend the LED Leads

Proceed to bend the LED leads as follows:



1. Starting with the leftmost LED from the top fret, bend the top, longer, LED lead to the right so that it is parallel with the back of the fingerboard and touches the next LED to the right's longer lead. The bend should be made close to the fingerboard, but not touching it (about 1/8" away). This will allow sufficient space so that soldering the leads, in a future step, will not melt the fingerboard, but will still provide enough room to mount the neck back.
2. Repeat step 1 with the rest of the LEDs of the first fret, except the rightmost LED. Do not bend the rightmost LED's longer lead. It will be handled in a later step.
3. Repeat steps 1 and 2 on the rest of the fret LEDs, being sure to leave the rightmost LED lead unbent.
4. Starting with the leftmost LED from the top fret, bend the bottom, shorter, LED lead down toward the corresponding LED of fret 2. Again, the bend should be made about 1/8" above the fingerboard.
5. Repeat step 7 with the rest of the LEDs. The LED of the fifth fret should be bent down similar to all the other LEDs.

Solder the Fret (Horizontal) LEDs

1. Starting with the leftmost LED of the top fret, solder the bent upper lead to the adjacent LED's bent lead.
2. Repeat step 1 for all LEDs in the top row except the rightmost. The LED to the left of the rightmost LED should be soldered to the unbent longer lead of the rightmost LED.
3. Repeat steps 1 and 2 with the LED of all the remaining frets.
4. Trim the LED lead from the next to last LED so that it does not protrude beyond the rightmost LED.

Wire the String (Vertical) LEDs

Note that it is recommended to use the solid wires for the string (vertical) LEDs. This should make it easier to perform the following steps. However, any kind of wire will do in a pinch.



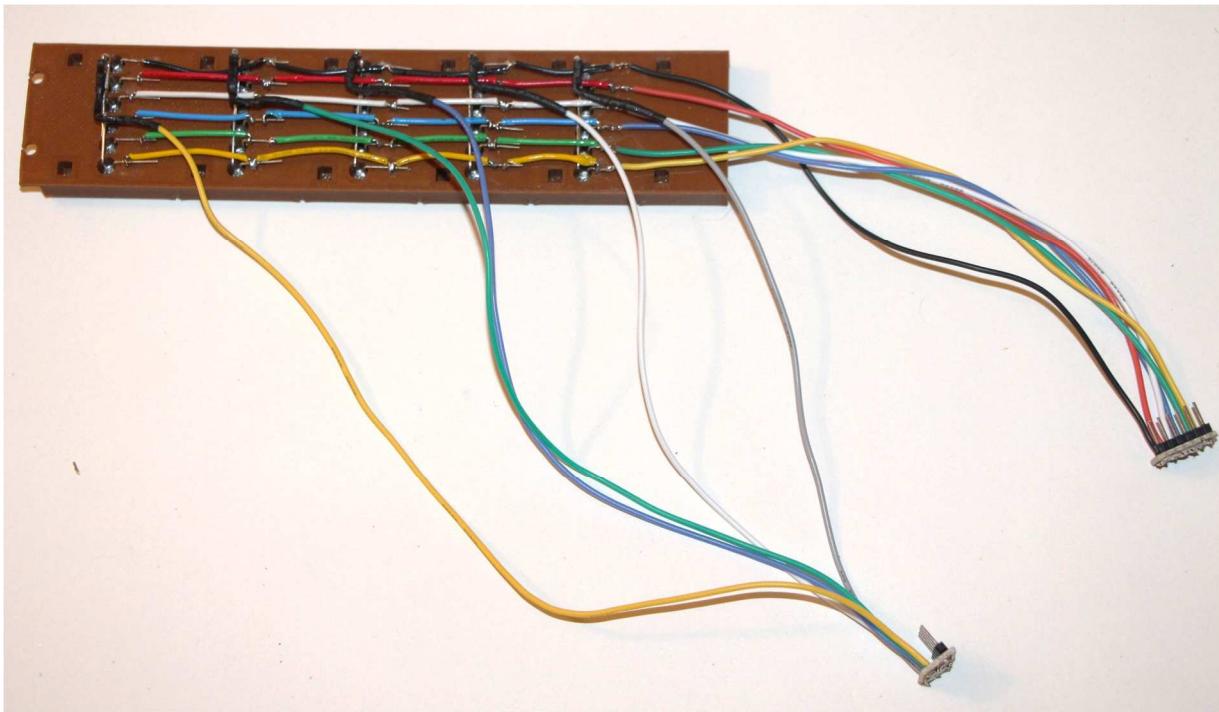
1. Cut a piece of yellow wire about $1 \frac{1}{2}$ " long, and make a small loop on each end. The following picture shows an example in which black wire was used.



2. Carefully slide the jumper wire from step 1 over the first LED leads of fret 1 and fret 2 (see the top right jumper of the picture above).
3. Insure that the jumper is about half way over the bent leads of the two LEDs and solder both ends.
4. Repeat steps 1 through 3 for all other connections as shown in the picture above. Note how the black and yellow wires in the picture are slightly bent away from the edges of the fingerboard. This is to allow room for the connection supports on the neck back, which will be added later.

Install the String Connections

This section will install the connection wires and header that will connect the string (vertical) LED leads to the Arduino.



1. Cut an 8" length of each color of stranded wire.
2. Solder one end of the wires from step 1 to each of the corresponding colored fret 5 LEDs as shown in the picture. For example, the lower row of previously installed solid wires in the picture is yellow. Solder one end of the 8" wire from step 1 to the corresponding fret 5 LED lead.
3. Repeat step 2 for each fret 5 LED lead (green, blue, white, red, black).
4. Solder the other end of each of the wires from the preceding steps to a male 6-pin header, in the same order as the string LED wires (yellow, green, blue, white, red, black).

String	Color	Arduino Pin
1	Yellow	D8
2	Green	D9
3	Blue	D10
4	White	D11
5	Red	D12
6	Black	D13

Install the Fret Connections

Use the picture from the preceding section as reference.

1. If the 1K ohm resistors are taped together, cut the ends of the resistors to insure that the sticky stuff from the tape is removed. This sticky stuff will interfere with soldering the parts.
2. Cut stranded wires as specified in the following table:

Fret	Color	Length	Arduino Pin
1	Yellow	12"	A0
2	Green	11"	A1
3	Blue	10"	A2
4	White	9"	A3
5	Grey	8"	A4

3. Solder one end of each 1K ohm resistor to the LED lead that is unbent on each fret.
4. Solder the wires that were cut in step 2 above to the other end of the resistors as specified in the table above.
5. Solder the other end of each of the wires from the preceding step to a male 5-pin header, in the same order as the fret LED wires (yellow, green, blue, white, grey).

Check the LED Wiring

Visually inspect all connections for shorts and opens. Using the multimeter, check for shorts and opens. Shorts are most likely to occur on the header connections. Some voltmeters supply enough current in continuity mode to light an LED. If yours doesn't do this, then a battery could be used in its place. A good test is to use the multimeter in continuity mode (or battery) to test the LED connections. While holding the black (common) probe on a string header pin, run the red (positive) probe over the fret header pins. You should see LEDs associated with the selected string light one-by-one down the frets. If any LED doesn't light, then there is an open connection to that LED. If more than one LED lights up at a time, then there is a short. These conditions must be fixed before proceeding.

Test the LEDs

Plug the 5-pin fret header into pins A0-A4 of the Arduino, with the yellow wire associated with Arduino pin A0. Note that it might be necessary to file/sandpaper the 5-pin header and/or the 1 pin IR connector from Arduino pin A5. They might not mechanically fit in to the Arduino header socket without a little filing/sanding.

Plug the 6-pin string header into pins A8-A13 of the Arduino, with the yellow wire associated with Arduino pin A8.

Be sure that all the needed Arduino libraries have been installed as described earlier.

Load the GuitarChordChart sketch into the chord chart's Arduino and apply power.

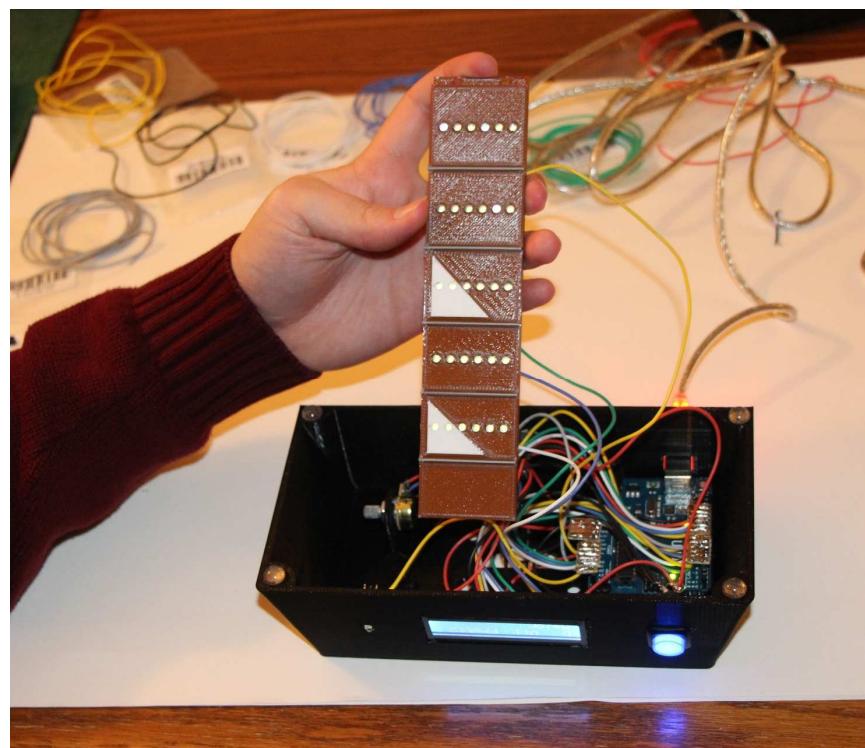
When the GuitarChordChart sketch starts, the LEDs should run a test cycle in which each LED successively turns on until all LEDs are lit, then the LEDs will successively turn off until all LEDs are unlit. Finally, the LEDs should light up in the pattern of a C major chord.

The LCD display should show the following until the LED test above completes:



Then the LCD should display the following:





Construct the Neck and Headstock



1. Unplug the 6-pin string header and the 5-pin fret header from the Arduino.
2. Insert the 6 Tuner Posts into the Headstock. They must be inserted from the top of the Headstock (the side with the lettering). It may take a moderate amount of force to get the Tuner Posts seated properly into the Headstock. When properly seated, the flange on the Tuner Post should be flush with the top surface of the Headstock, and the Tuner Post should protrude from the bottom of the Headstock.
3. Insert the 6 Tuner Shafts into the corresponding holes in the Tuner Assembly. A significant amount of force may be required to get the Tuner Shafts to insert into the Tuner Assembly.

4. Attach the 6 Tuning Knobs to the ends of the 6 Tuner Shafts. A significant amount of force may be required to get the Tuning Knobs to seat correctly onto the Tuner Shafts.
5. Attach the Tuner Assembly onto the end of the protruding Tuner Posts making sure that the holes for the Tuner Shafts face toward the near edge of the Headstock.
6. Insert the 6 Tuner Shafts into the corresponding holes in the Tuner Assembly. A significant amount of force may be required to get the Tuner Shafts to insert into the Tuner Assembly. The Tuner Knobs should all end up at the same distance from the Tuner Assembly.



7. Insert the Nut into the Fingerboard.
8. Attach the Headstock to the Fingerboard.
9. Insert the Mating Posts into the Fingerboard. Not all of the Mating Posts are necessary. I recommend inserting one at each of the corners, and at least one in the middle of each side (a total of at least 6). Note that a good deal of force may be needed to get the Mating Posts to seat properly into the Fingerboard.

10. Align the Mating Posts with the Mating Post Holes in the Neck Back, and force the Neck Back onto the Mating Posts. The middle of the Neck Back will not support a lot of force, so it is best to apply pressure to the edges when getting the Neck Back and the Fingerboard to mate properly.

Mount the Neck

1. Insert the Neck Support into the Base. A significant amount of force is required to achieve this. Once the Neck Support is started, turn the Base over, and push the Base onto the Neck Support. This will keep from putting too much pressure on the top of the Base which may cause damage.
2. Fish the string and fret headers and wires through the neck hole in the Base, and seat the Neck assembly into the Neck Support.
3. Secure the Neck assembly to the Neck Support by inserting the Neck Support Pin into the Neck Support. The Neck Support Pin should be inserted all the way into the Neck Support so that it engages the Neck, and keeps it from sliding.

Wire the LEDs to the Arduino

1. Plug the 5-pin fret header into pins A0-A4 of the Arduino, with the yellow wire associated with Arduino pin A0.
2. Plug the 6-pin string header into pins A8-A13 of the Arduino, with the yellow wire associated with Arduino pin A8.

Test the LEDs

Load the GuitarChordChart Arduino sketch into the Guitar Chord Chart and apply power. The LEDs should run through their power-up sequence, and the LCD should display the messages shown in the previous “Test the LEDs” section.

INSTALL THE ARDUINO FIRMWARE

Program the Remote Control

All IR remote controls use different codes and different methods of programming. This section describes the code and programming method for the Vivitar 8-in-1 Remote Control that I got from Microcenter (Mfg. Part # VIV-URC-713).

After much testing, I settled on a Sony TV code 5321. To program the device, do the following:

1. Press the button corresponding to the device you want to use for controlling your Guitar Chord Chart (TV, DVD, DVR, SAT, CBL, or AUX).
2. Press and hold the SET button until the remote control's red LED lights.
3. Enter the 4-digit program code (5-3-2-1).

See the next section if you are using a different remote control. It describes how to determine the key values that are sent by your remote control, and how to modify the Arduino sketch to accommodate them.

Modify the Arduino Sketch as Needed

There are 3 main reasons that you may wish to modify the Arduino sketch:

- You are using a different remote control than the one that has been specified.
- You are adding or modifying chord patterns.
- You want to add or modify the behaviour of the sketch.

Each of these is discussed in detail below.

Modify IR Codes

There are 2 steps required to change the IR codes that the Guitar Chord Chart responds to.

1. Determine the key code values of that are sent by the remote control that you are using. The Guitar Chord Chart uses the keys in the following table. If your remote control does not contain any of the specified keys, then you'll need to find another key to substitute for it. The following table can be used to help gather the IR key codes for your remote control. Use the IrTest Arduino sketch to gather the equivalent key codes for your device, and enter them into the table below. Only keys that are used in the sketch are required. The others are displayed to show which keys may be used if you want to modify the existing functionality.

IR Key	Sketch Name	Used?	Current Code	Your Code
POWER	POWER	NO	0xf124b0b6	
MUTE	MUTE	NO	0x78f479a7	
REWIND	REWIND	NO	0x2bdc941f	
PLAY	PLAY	YES	0x633dd8c4	
FAST FORWARD	FFWD	YES	0xbd293f06	
STOP	STOP	NO	0xc502105e	
PAUSE	PAUSE	NO	0x36d87f5b	
MENU	MENU	NO	0x3ac82a07	
VOLUME UP	VOL_UP	YES	0xb44d8dfb	
VOLUME DOWN	VOL_DOWN	YES	0x43898cc0	
CHANNEL UP	CH_UP	YES	0xbc392255	
CHANNEL DOWN	CH_DOWN	YES	0xfffafb8dc	
UP ARROW	UP	YES	0xec27d43d	
DOWN ARROW	DOWN	YES	0x086bd99c	
LEFT ARROW	LEFT	YES	0x1a422e43	
RIGHT ARROW	RIGHT	YES	0xa23bd824	
OK (SELECT)	OK	YES	0x7295a904	
INFO	INFO	YES	0xdc68e43f	
INPUT	INPUT_CH	NO	0xba00cb82	
BACK	BACK	YES	0x99c7ec8f	
GUIDE	GUIDE	YES	0xd1c67b61	
1	NUM_1	YES	0x207bef0f	
2	NUM_2	YES	0xe8455d8e	
3	NUM_3	YES	0xcbb7e949	
4	NUM_4	YES	0x2c1f3172	
5	NUM_5	YES	0x905ed4f5	
6	NUM_6	NO	0x8ce1e3fc	
7	NUM_7	NO	0x315b1905	
8	NUM_8	NO	0xc67c01b6	
9	NUM_9	NO	0xb418d969	
0	NUM_0	NO	0x3b0b8330	
-	DASH	NO	0xbe980a4b	

- Now that you have collected the key codes for your remote control, you must enter them into the IrCodes.h file that can be found in the GuitarChordChart directory. The relevant portion of the existing code follows:

```

static const uint32_t POWER      = 0xf124b0b6;      //
static const uint32_t MUTE       = 0x78f479a7;      //
static const uint32_t REWIND     = 0x2bdc941f;      //
static const uint32_t PLAY       = 0x633dd8c4;      //
static const uint32_t FFWD       = 0xbd293f06;      //
static const uint32_t STOP       = 0xc502105e;      //
static const uint32_t PAUSE      = 0x36d87f5b;      //
static const uint32_t MENU       = 0x3ac82a07;      //
static const uint32_t VOL_UP    = 0xb44d8dfb;      //
static const uint32_t VOL_DOWN   = 0x43898cc0;      //
static const uint32_t CH_UP     = 0xbc392255;      //
static const uint32_t CH_DOWN   = 0xfffafb8dc;      //
static const uint32_t UP         = 0xec27d43d;      //
static const uint32_t DOWN       = 0x086bd99c;      //
static const uint32_t LEFT       = 0x1a422e43;      //
static const uint32_t RIGHT      = 0xa23bd824;      //
static const uint32_t OK         = 0x7295a904;      //
static const uint32_t INFO       = 0xdc68e43f;      //
static const uint32_t INPUT_CH  = 0xba00cb82;      //
static const uint32_t BACK       = 0x99c7ec8f;      //
static const uint32_t GUIDE      = 0xd1c67b61;      //
static const uint32_t NUM_1      = 0x207bef0f;      //
static const uint32_t NUM_2      = 0xe8455d8e;      //
static const uint32_t NUM_3      = 0xcbb7e949;      //
static const uint32_t NUM_4      = 0x2c1f3172;      //
static const uint32_t NUM_5      = 0x905ed4f5;      //
static const uint32_t NUM_6      = 0x8ce1e3fc;      //
static const uint32_t NUM_7      = 0x315b1905;      //
static const uint32_t NUM_8      = 0xc67c01b6;      //
static const uint32_t NUM_9      = 0xb418d969;      //
static const uint32_t NUM_0      = 0x3b0b8330;      //
static const uint32_t DASH       = 0xbe980a4b;      //

```

Simply replace the existing hex code with the corresponding code for your remote control, and re-download the GuitarChordChart sketch into the device. Your remote controller should now be understood by the application.

Add or Modify Chords

The supplied source code supports only 2 variations of the M, m, 6, m6, 7, M7, m7, and 9 chord types. However, the code is written to be able to accommodate any number of variations of the following chord types:

M, m, 6, m6, 7, M7, m7, 9, O, +, sus2, 7sus2, sus4, 7sus4, 5, -5, 7b5, m7b5, 7#5, 7b9, 7#9, 7b9#5, 7/6, 9b5, 9#5, M9, m9, 9/6, m9/6, add9, 11, m11, 11+, 13, 13b9, and 13b9b5.

In order to extend the supported chords, two files will need to be modified. These files are ChordChartData.h and ChordChartData.cpp.

ChordChartData.h contains the following lines of code:

```
const unsigned NUM_COMMON_CHORDS      = 8;
const unsigned NUM_COMMON_CHORD_VARIATIONS = 2;
const unsigned NUM_SUPP_CHORDS        = 0;
const unsigned NUM_SUPP_CHORD_VARIATIONS = 0;
```

These lines will need to be modified to indicate the number of chords and variations that will be supported. For example, if the O, +, and sus2 types are added and each of them uses 2 variants, then the code would change to:

```
const unsigned NUM_COMMON_CHORDS      = 8;
const unsigned NUM_COMMON_CHORD_VARIATIONS = 2;
const unsigned NUM_SUPP_CHORDS        = 3;
const unsigned NUM_SUPP_CHORD_VARIATIONS = 2;
```

If, in addition, the number of common chord variations is changed to 4, then the code would change to:

```
const unsigned NUM_COMMON_CHORDS      = 8;
const unsigned NUM_COMMON_CHORD_VARIATIONS = 4;
const unsigned NUM_SUPP_CHORDS        = 3;
const unsigned NUM_SUPP_CHORD_VARIATIONS = 2;
```

The chords have been split into COMMON and SUPPLEMENTAL groups since in general it may be desired to have more variations of the COMMON types, and less variations of the SUPPLEMENTAL types.

Note that due to the architecture of the code, if chord types are added, they must be added in the following order:

O, +, sus2, 7sus2, sus4, 7sus4, 5, -5, 7b5, m7b5, 7#5, 7b9, 7#9, 7b9#5, 7/6, 9b5, 9#5, M9, m9, 9/6, m9/6, add9, 11, m11, 11+, 13, 13b9, and 13b9b5. In addition, all chords in the COMMON group must contain the same number of variations, and all chords in the SUPPLEMENTAL group must contain the same number of variations.

ChordChartData.cpp contains the array of Chord structures for each chord. The following is an excerpt of the data from the array:

```
static const uint8_t Keys[sizeof(Key) * NUM_KEYS] PROGMEM =
{
    // C Major
    1, 0x00, 0x02, 0x08, 0x30, 0x00, 0x00,      3, 0x00, 0x31, 0x00, 0xe0, 0x00, 0x00,
    // C Minor
```

```

1, 0x30, 0x0a, 0x00, 0x01, 0x00, 0x00,      3, 0x00, 0x31, 0x02, 0x0c, 0x00, 0x00,
// C Sixth
1, 0x00, 0x02, 0x0c, 0x01, 0x00, 0x00,      5, 0x00, 0x0f, 0x00, 0x10, 0x20, 0x00,
// C Minor Sixth
1, 0x20, 0x0a, 0x00, 0x01, 0x00, 0x00,      4, 0x30, 0x02, 0x0d, 0x00, 0x00, 0x00,
// C Seventh
1, 0x00, 0x02, 0x08, 0x34, 0x00, 0x00,      3, 0x00, 0x35, 0x00, 0xa, 0x00, 0x00,
// C Major Seventh
2, 0x00, 0x08, 0x30, 0x00, 0x00, 0x00,      3, 0x00, 0x31, 0x04, 0xa, 0x00, 0x00,
// C Minor Seventh
1, 0x20, 0x0a, 0x00, 0x15, 0x00, 0x00,      3, 0x00, 0x35, 0x02, 0x08, 0x00, 0x00,
// C Ninth
2, 0x00, 0x08, 0x37, 0x00, 0x00, 0x00,      5, 0x20, 0xe, 0x01, 0x00, 0x00, 0x00,
// C Sharp Major
1, 0x00, 0x25, 0x02, 0x08, 0x10, 0x00,      4, 0x00, 0x31, 0x00, 0xe, 0x00, 0x00,
// C Sharp Minor
1, 0x30, 0x04, 0xa, 0x00, 0x00, 0x00,      4, 0x00, 0x31, 0x02, 0xc, 0x00, 0x00,

```

Each entry is composed of the 7 bytes of the Chord class:

```

uint8_t m_Fret;                      // Starting fret of the chord.
uint8_t m_Unplayed;                  // Bit pattern of unplayed strings.
uint8_t m_Chord[NUM_CHORD_FRETS];    // Each element represents a fret.

```

So, for example, looking at the first C Sharp Minor entry above, the following is relevant:

- `m_Fret = 1` – this is the base fret for the displayed chord.
- `m_Unplayed = 0x30` – this is the bit pattern of strings that will not be played (the upper 2 strings).
- `m_Chord[0] = 0x04` – this specifies the strings of the first fret. In this case, the G string (0x04) is specified.
- `m_Chord[1 – 4] = 0x0a, 0x00, 0x00, 0x00` – these specify frets 1 through 5.

For `m_Unplayed` and `m_Chord`, the set bits indicate:

- E = 0x01
- B = 0x02
- G = 0x04
- D = 0x08

- A = 0x10
- E = 0x20
- The remaining bits (0x40 and 0x80) are unused.

Modify Sketch Behavior

This section provides only high level information regarding the design of the GuitarChordChart sketch. A competent software person should be able to figure out how to modify it as desired.

The following table contains a list of files that comprise the GuitarChordChart application, and a brief description of each:

File	Description
ChordChartData.h	<p>Defines several classes and constants used to describe and save the chord pattern information. The main class that will interest most is the “Chord” class. It defines the chord pattern that is used in ChordChartData.cpp. Its data members are:</p> <ul style="list-style-type: none"> • m_Fret – the starting fret of the chord. • m_Unplayed; - a bit pattern of unplayed strings. • m_Chord[NUM_CHORD_FRETS] – the chord layout. Each element represents a fret starting at m_Fret. <p>The values used by m_Unplayed and m_Chord are bit patterns, with each bit representing a string:</p> <ul style="list-style-type: none"> • E => 0x01 • B => 0x02 • G => 0x04 • D => 0x08 • A => 0x10 • E => 0x20 <p>The remainder of the bits are unused.</p>
ChordChartData.cpp	<p>This file contains the chord patterns for all the chords known by the chord chart. When adding chords, or modifying chords, this is the file that will need to be changed. The “Keys” array contains all of the known chords. It is implemented as an array of uint8_t instead of an array of Chord due to limitations in the Arduino tool set that don’t allow storing structures in non .ino files into PROGMEM (at least I couldn’t figure out how to make it work).</p>
ChordFinderMode.cpp, ChordFinderMode.h	<p>These files implement the Chord Finder Mode. This is the main mode of operation in which the user can traverse the Keys array and display the desired Chord information.</p>

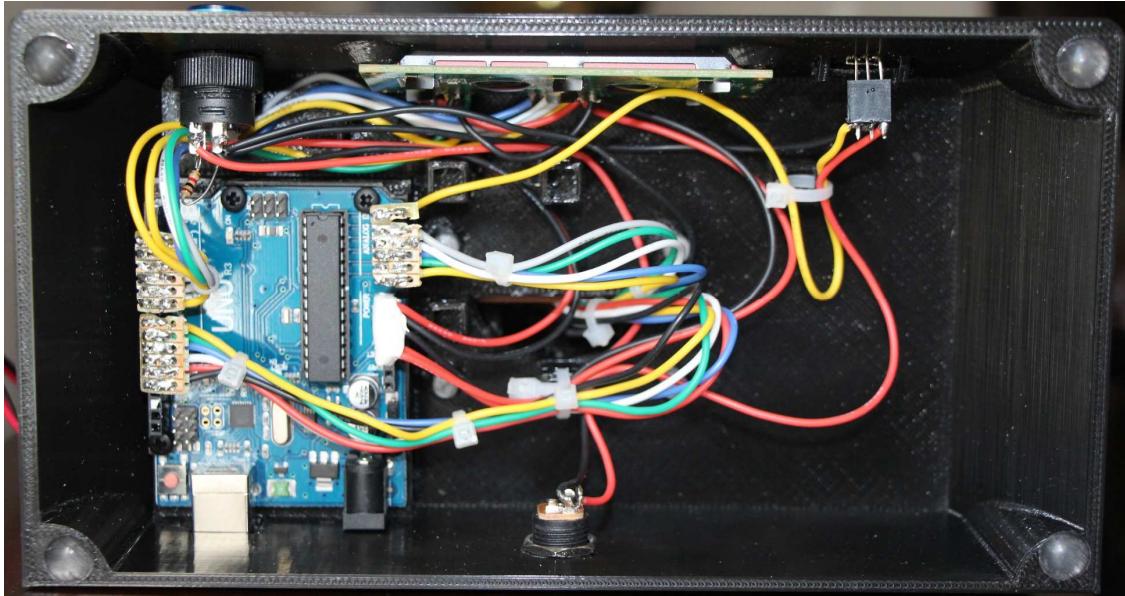
DemoMode.cpp, DemoMode.h	These files implement the Demo Mode. This mode displays various patterns on the fingerboard LEDs. It is mainly used to show off, or to verify that the LEDs are functioning correctly.
Display.cpp, Display.h	These files implement the Display class. It mainly contains methods to display data and strings on the LCD.
EepromConfig.cpp, EepromConfig.h	These files implement the EepromConfig class. This class is used to save and restore configuration information to/from the Arduino EEPROM memory.
GuitarChordChart.ino	This is equivalent to the “main” function of a C program. It contains the startup and loop code that implements the entire GuitarChordChart application.
IrCodeDisplayMode.cpp, IrCodeDisplayMode.h	These files implement the IR Code Display Mode. This mode reports the raw key codes that are received. This mode cannot be exited because it disables the decoding of IR keys. To exit this mode, power must be cycled.
IrCodes.cpp, IrCodes.h	These files are used to specify all the known IR codes. The .h file can be modified to adapt the application to other IR remote controls.
IrDriver.cpp, IrDriver.h	These files implement the IrDriver class. This class manages the decoding of IR input. It derives from the IRdecodeHash class that is part of the Arduino IRLib library.
LedDriver.cpp, LedDriver.h	These files implement the LedDriver class. This class manages the interface to the LEDs.
Mode.h	This file defines the abstract Mode class, from which all modes derive.
ModeManager.cpp, ModeManager.h	These files implement the ModeManager class. This class manages the selection and execution of the various modes supported by the application.
PolledDelay.h	This file implements the PolledDelay class. Rather than waiting while a timer times out, this class polls to determine whether or not a specified amount of time has expired.
ReverseChordFinderMode.cpp, ReverseChordFinderMode.h	These files implement the ReverseChordFinderMode class. This mode allows the user to specify the chord pattern of interest. It then looks it up the pattern in the Key table to report which chord, if any, it pertains to.
SettingsMode.cpp, SettingsMode.h	These files implement the SettingsMode class. This mode handles the setting of configurable values that are remembered across power cycles.
SpecialChars.cpp, SpecialChars.h	These files implement the SpecialChars class. This class defines and installs the special characters that get displayed on the LCD.

	These characters include the flat, augment, diminish, and sharp symbols.
Strings.cpp, Strings.h	These files implement the Strings class. This class is used to fetch strings from FLASH memory. Strings are stored in FLASH in order to save limited RAM space.
TimerOne.cpp, TimerOne.h	These files are from the Arduino TimerOne library. I was not able to ever get them to work as a library. I needed to include them in the application's source directory instead. These are the only files listed that were not written by me.

Run the Final Test

Download the (possibly modified) GuitarChordChart sketch and verify that it powers up correctly and that your remote control works correctly with it. See the Appendix for information on using the application.

ADD THE FINAL TOUCHES



1. Attach the 4 rubber feet to the base.
2. Clean up the wiring by using zip ties. Secure the wires to the base by using the zip tie support posts inside the base.

POSSIBLE ENHANCEMENTS

The Arduino Uno is maxed out with this application. All the I/O pins are used, and nearly all the memory is used. With another computer, say an Arduino Mega, or a Raspberry Pi Zero, it wouldn't take a lot of effort to add some really cool features.

Add “Unplayed String” LEDs

The options for selecting how unplayed strings will be displayed are a kludge at best. It would be much more useful and aesthetically pleasing to add another set of LEDs, possibly red and placed in the nut, to denote strings that are not to be played. This would not take much effort, but would really add to the usefulness of the device.

Add Bluetooth

The IR remote is functional, but clunky. If Bluetooth were added, it would be a relatively simple matter to write a phone app that could control the chord chart. The interface could be made much more usable, and the user would not need to maintain a separate remote control.

Play a Selected Chord

If an amplifier and speaker were added, it wouldn't take much effort to make the chord chart actually play the specified chord. In this way, the user could hear what the chord should sound like as well as see the finger positions.

Display a Chord Based on Audio Input

This would require a computer with a bit more horsepower, but would be realizable with a Raspberry Pi or a Beagle Bone. It should be possible to have the computer listen to a chord, and determine which chord it is, and what the associated finger positions were.

Help Tune the Guitar

If a microphone and preamp were added, it would be a relatively simple matter to add a tuning mode. In this mode, the computer would listen to each string, and notify the user when the string was tuned properly.

APPENDIX

User Manual

The Guitar Chord Chart main application supports 5 modes of operation:

1. Chord Finder Mode – This is the normal mode of operation. In this mode, the user can traverse the chord tables and find the desired chord finger pattern by specifying the key, type, and variation of the chord. This mode is entered after powering up.
2. Reverse Chord Finder Mode – In this mode, the user can specify a finger pattern and base fret, and the application will look up and report the name, type, and variation of the specified chord if found.
3. Settings Mode – This mode allows the user to specify the values of several persistent parameters that will be remembered across power cycles.
4. Demo Mode – This mode displays various patterns on the fingerboard LEDs. It is mainly used to show off, or to verify that the LEDs are functioning correctly.
5. IR Key Display Mode – While in this mode, the LCD reports any raw key codes that are received. This mode cannot be exited because it disables the decoding of IR keys. To exit this mode, power must be cycled. Due to the inability to exit this mode, there is a user configurable setting that disables this mode.

Each of these modes is described fully in the sections below.

Chord Finder Mode

This mode can be entered from any other mode, except IR Key Display Mode, by pressing '1' on the IR remote control.

The following table describes the IR keys that are active while in Chord Finder Mode.

IR Key	Behavior
Up Arrow	Next key. Selects the next chord key. Key sequence is: C -> C# -> D -> Eb -> E -> F -> F# -> G -> Ab -> A -> Bb -> B -> C ...
Down Arrow	Previous key. Selects the previous chord key. Key sequence is: C -> B -> Bb -> A -> Ab -> G -> F# -> F -> E -> Eb -> D -> C# -> C ...
Left Arrow	Previous type. Selects the previous type of the current key. Type sequence is: M -> 13b9b5 -> 13b9 -> 13 -> 11+ -> m11 -> 11 -> add9 -> m9/6 -> 9/6 -> m9 -> M9 -> 9#5 -> 9b5 -> 7/6 -> 7b9#5 -> 7#9 -> 7b9 -> 7#5 -> m7b5 -> 7b5 -> -5 -> 5 -> 7sus4 -> sus4 -> 7sus2 -> sus2 -> + -> O -> 9 -> m7 -> M7 -> 7 -> m6 -> 6 -> m -> M -> ... Note that the code that accompanies the Guitar Chord Chart only supports 2 variations of M, m, 6, m6, 7, M7, m7, and 9. The code will need to be enhanced to support the remainder of the known chord types.
Right Arrow	Next type. Selects the next type of the current key. Type sequence is: M -> m -> 6 -> m6 -> 7 -> M7 -> m7 -> 9 -> O -> + -> sus2 -> 7sus2 -> sus4 -> 7sus4 -> 5 -> -5 -> 7b5 -> m7b5 -> 7#5 -> 7b9 -> 7#9 -> 7b9#5 -> 7/6 ->

IR Key	Behavior
	<p>9b5 -> 9#5 -> M9 -> m9 -> 9/6 -> m9/6 -> add9 -> 11 -> m11 -> 11+ -> 13 -> 13b9 -> 13b9b5 -> M -> ...</p> <p>Note that the code that accompanies the Guitar Chord Chart only supports 2 variations of M, m, 6, m6, 7, M7, m7, and 9. The code will need to be enhanced to support the remainder of the known chord types.</p>
OK (Select)	First type and variation. Selects the first type (Major) and the first variation of the current key. For example, if C 7/6 variation 2 is currently selected, and the OK key is pressed, the information for C Major variation 1 will be displayed.
Channel Up	Next variation. Selects the next variation of the current key and type. When the last variation is reached, the display wraps back to the first variation of the same key/type.
Channel Down	Previous variation. Selects the previous variation of the current key and type. When the first variation is reached, the display wraps back to the last variation of the same key/type.
Info	Toggle type. Switches between verbose and brief display of the selected chord's display. For example, brief display might be "13b9", while the corresponding verbose display would be "13 th Minor 9 th "
Guide	Next unplayed string option. Cycles through the unplayed string display options. The sequence is: Light All Frets -> Light Last Fret Unused -> Light Last Fret Used -> Ignore ...
Volume Up	Brighten LEDs. Increases the brightness of the LEDs.
Volume Down	Dim LEDs. Decreases the brightness of the LEDs.
2	Select Reverse Chord Finder Mode.
3	Select Settings Mode.
4	Select Demo Mode.
5	Select IR Key Display mode if enabled. See Settings Mode.

Reverse Chord Finder Mode

This mode can be entered from any other mode, except IR Key Display Mode, by pressing '2' on the IR remote control.

In this mode, the user needs to select the strings to be included in the match, the selected base fret, and the fingering pattern to search for. Once these have been entered, the application will attempt to find a chord that matches the description. If found, the information pertaining to the specified chord will be displayed on the LCD.

The following table describes the IR keys that are active while in Reverse Chord Finder Mode.

IR Key	Behavior
Up Arrow	<p>Next base fret/Previous fret. While entering the base fret specification, this key will increment the fret as follows:</p> <p>ANY -> 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> 11 -> ANY -> ...</p> <p>While entering the chord finger pattern, this key will cycle to the previous fret as follows:</p> <p>1 -> 5 -> 4 -> 3 -> 2 -> 1 ...</p> <p>Note that setting any of the LEDs in the fifth fret indicates that that string is unplayed.</p>
Down Arrow	<p>Previous base fret/Next fret. While entering the base fret specification, this key will decrement the fret as follows:</p> <p>ANY -> 11 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1 -> ANY -> ...</p> <p>While entering the chord finger pattern, this key will cycle to the next fret as follows:</p> <p>1 -> 2 -> 3 -> 4 -> 5 -> 1 ...</p> <p>Note that setting any of the LEDs in the fifth fret indicates that that string is unplayed.</p>
Left Arrow	<p>Next string. While entering the match pattern, the left arrow will increase the strings to match. The pattern is:</p> <p>E -> EB -> EBG -> EBGD -> EBGDA -> EBGDAE</p> <p>Note that this pattern does not wrap after EBGDAE is reached.</p> <p>While entering the chord finger pattern, this key will cycle to the next string as follows:</p> <p>E -> B -> G -> D -> A -> E -> E ...</p>
Right Arrow	<p>Previous string. While entering the match pattern, the right arrow will decrease the strings to match. The pattern is:</p> <p>EBGDAE -> EBGDA -> EBGD -> EBG -> EB -> E</p> <p>Note that this pattern does not wrap after E is reached.</p> <p>While entering the chord finger pattern, this key will cycle to the previous string as follows:</p> <p>E -> E -> A -> D -> G -> B -> E ...</p>

IR Key	Behavior
OK (Select)	Select/toggle. While entering the base fret, this key finalizes the selection. While entering the fingering pattern, this key toggles the on/off state of the currently selected fret/string.
Play	Begin/Continue search. Once the base fret and fingerboard pattern are set up, pressing this key will cause the lookup to begin. After a match is found, pressing this key will search for another match.
Info	Toggle type. Switches between verbose and brief display of the selected chord's display. For example, brief display might be "13b9", while the corresponding verbose display would be "13 th Minor 9 th "
Guide	Next unplayed option. Cycles through the unplayed string display options. The sequence is: Light All Frets -> Light Last Fret Unused -> Light Last Fret Used -> Ignore ...
Volume Up	Brighten LEDs. Increases the brightness of the LEDs.
Volume Down	Dim LEDs. Decreases the brightness of the LEDs.
1	Select Chord Finder Mode.
2	Re-start Reverse Chord Finder Mode.
3	Select Settings Mode.
4	Select Demo Mode.
5	Select IR Key Display mode if enabled. See Settings Mode.

Settings Mode

This mode can be entered from any other mode, except IR Key Display Mode, by pressing '3' on the IR remote control.

There are 5 options that may be set in this mode. Once the options are saved, they will take effect the next time the device powers up. The options that may be set are:

1. Chord Display Option – This option specifies how chord types are displayed. The possibilities are VERBOSE and ABBREVIATED. If VERBOSE is selected, then for example, the display of a chord might be "13th Minor 9th", as opposed to "13b9" if ABBREVIATED is selected.
2. Unplayed String Option – With some chords, not all 6 strings should be played. This option specifies how unplayed strings will be displayed. The possibilities are IGNORE, LIGHT UNPLAYED, LIGHT PLAYED, LIGHT ALL FRETS. The IGNORE selection will do nothing to mark unplayed strings. The LIGHT UNPLAYED selection will cause LEDs in the fifth fret to be lit for any strings that should not be played. The LIGHT PLAYED selection is the opposite of the LIGHT

UNPLAYED selection. That is, the LIGHT PLAYED option will cause LEDs in the fifth fret to be lit if the string is played. The LIGHT ALL FRETS selection will cause all LEDs of all frets of strings that are not to be played to be lit.

3. Demo Option – This option specifies how the Demo Mode will behave. The possibilities are CONTINUOUS, RANDOM, and REPEAT CURRENT. The CONTINUOUS selection will cause the demos to run in order continuously. The sequence is:
 - ALL ON – Lights all LEDs.
 - ALL OFF – Turns off all LEDs.
 - WALK LED ON – Starts with all LEDs off, and sequentially turns on each LED briefly before turning it back off. There is always exactly one LED lit, and it walks across strings and frets till it hits the last fret/string.
 - SWEEP LED ON – Starts with all LEDs off, and sequentially turns on each LED and leaves them on. This continues until all LEDs are lit.
 - WALK LED OFF – Starts with all LEDs on, and sequentially turns each LED off before turning it back on. There is always exactly one LED unlit, and it walks across strings and frets till it hits the last fret/string.
 - SWEEP LED OFF – Starts with all LEDs on, and sequentially turns off each LED and leaves them off. This continues until all LED are unlit.
 - WALK FRET ON – Similar to WALK LED ON, but using frets instead of single LEDs.
 - SWEEP FRET ON – Similar to SWEEP LED ON, but using frets instead of single LEDs.
 - WALK FRET OFF – Similar to WALK LED OFF, but using frets instead of single LEDs.
 - SWEEP FRET OFF – Similar to SWEEP LED OFF, but using frets instead of single LEDs.
 - WALK STRING ON - Similar to WALK LED ON, but using strings instead of single LEDs.
 - SWEEP STRING ON– Similar to SWEEP LED ON, but using strings instead of single LEDs.
 - WALK STRING OFF – Similar to WALK LED OFF, but using strings instead of single LEDs.
 - SWEEP STRING OFF – Similar to SWEEP LED OFF, but using strings instead of single LEDs.
 - RANDOM CHORD – Briefly shows random chords.
4. LED Brightness Option – This option selects the default LED brightness (10% through 100% in steps of 10%).
5. IR Mode Permission Option – This option selects whether or not the '5' key will cause the IR Key Display Mode to be entered. Possibilities are FORBID and PERMIT. If FORBID is selected, then IR Key Display Mode cannot be entered. The PERMIT selection allows the mode to be entered. The reason that one might choose to FORBID entry to the IR Key Display Mode is that once that mode is entered, it can never exit. A power cycle is required to exit the mode.

Once all options have been selected, the user is given a chance to SAVE or CANCEL the selections. If SAVE is selected, the selections that were made will be saved to EEPROM so that they will be active on the next power-up.

The following table describes the IR keys that are active while in Settings Mode.

IR Key	Behavior
Up Arrow	<p>Cycle to the previous value for the currently active configuration option.</p> <p>While entering the Chord Display option, the sequence is: VERBOSE -> ABBREVIATED -> VERBOSE ...</p> <p>While entering the Unplayed String option, the sequence is: LIGHT ALL FRETS -> IGNORE -> LIGHT PLAYED -> LIGHT UNPLAYED -> LIGHT ALL FRETS ...</p> <p>While entering the Demo option, the sequence is: CONTINUOUS -> RANDOM -> REPEAT CURRENT -> CONTINUOUS ...</p> <p>While entering the LED Brightness option, the sequence is: 10% -> 20% -> 30% -> 40% -> 50% -> 60% -> 70% -> 80% -> 90% -> 100% Note that the LED brightness option does not cycle back to 10% after 100% is reached. It stays at the 100% value.</p> <p>While entering the IR Key Permission option, the sequence is: FORBID -> PERMIT -> FORBID ...</p> <p>While entering the Save selection, the sequence is: CANCEL -> SAVE -> CANCEL ...</p>
Down Arrow	<p>Cycle to the next value for the currently active configuration option.</p> <p>While entering the Chord Display option, the sequence is: VERBOSE -> ABBREVIATED -> VERBOSE ...</p> <p>While entering the Unplayed String option, the sequence is: LIGHT ALL FRETS -> LIGHT UNPLAYED -> LIGHT PLAYED -> IGNORE -> LIGHT ALL FRETS ...</p>

IR Key	Behavior
	<p>While entering the Demo option, the sequence is: CONTINUOUS -> REPEAT CURRENT -> RANDOM -> CONTINUOUS ...</p> <p>While entering the LED Brightness option, the sequence is: 100% -> 90% -> 80% -> 70% -> 60% -> 50% -> 40% -> 30% -> 20% -> 10% Note that the LED brightness option does not cycle back to 100% after 10% is reached. It stays at the 10% value.</p> <p>While entering the IR Key Permission option, the sequence is: FORBID -> PERMIT -> FORBID ...</p> <p>While entering the Save selection, the sequence is: CANCEL -> SAVE -> CANCEL ...</p>
OK (Select)	<p>Select. When this key is pressed, the currently displayed value for the active configuration option is posted, and the next configuration option is selected. The sequence is:</p> <p>Chord Display Option -> Unplayed String Option -> Demo Option -> Brightness Option -> IR Key Display Permission Option -> Save Selection -> Chord Display Option ...</p>
Volume Up	Brighten LEDs. Increases the brightness of the LEDs.
Volume Down	Dim LEDs. Decreases the brightness of the LEDs.
1	Select Chord Finder Mode.
2	Select Reverse Chord Finder Mode.
3	Go back to the start of Settings Mode.
4	Select Demo Mode.
5	Select IR Key Display mode if enabled. See Settings Mode.

Demo Mode

This mode can be entered from any other mode, except IR Key Display Mode, by pressing '4' on the IR remote control.

The behavior of this mode is selected via the Demo option setting. See the previous section for details.

The following table describes the IR keys that are active while in Demo Mode.

IR Key	Behavior
Up Arrow	Next Demo. Cycle to the next demo. The sequence is: ALL ON -> ALL OFF -> WALK LED ON -> SWEEP LED ON -> WALK LED OFF -> SWEEP LED OFF -> WALK FRET ON -> SWEEP FRET ON -> WALK FRET OFF -> SWEEP FRET OFF -> WALK STRING ON -> SWEEP STRING ON -> WALK STRING OFF -> SWEEP STRING OFF -> RANDOM CHORD -> ALL ON ...
Down Arrow	Previous Demo. Cycle to the previous demo. The sequence is: ALL ON -> RANDOM CHORD -> SWEEP STRING OFF -> WALK STRING OFF -> SWEEP STRING ON -> WALK STRING ON -> SWEEP FRET OFF -> WALK FRET OFF -> SWEEP FRET ON -> WALK FRET ON -> SWEEP LED OFF -> WALK LED OFF -> SWEEP LED ON -> WALK LED ON -> ALL OFF -> ALL ON ...
Volume Up	Brighten LEDs. Increases the brightness of the LEDs.
Volume Down	Dim LEDs. Decreases the brightness of the LEDs.
1	Select Chord Finder Mode.
2	Select Reverse Chord Finder Mode.
3	Select Settings Mode.
5	Select IR Key Display mode if enabled. See Settings Mode.

IR Key Display Mode

This mode can be entered from any other mode, when enabled, by pressing '5' on the IR remote control.

This mode operates exactly like the IrTest sketch that was used earlier to test the IR functionality.

When the IR Key Display Mode starts, it is waiting to see a key press from the IR remote control. Press any key on the IR remote. The LCD display will show the count of the number of IR key presses along with the hexadecimal code of the last pressed IR key:



Note that the 8-digit hex code should be unique for each key that is pressed. This mode will continue to accept key presses and return their associated value forever. The only way to exit from this mode is to cycle power.