MINI 7-SEGMENT CLOCK

Assembly Suggestions & Tips

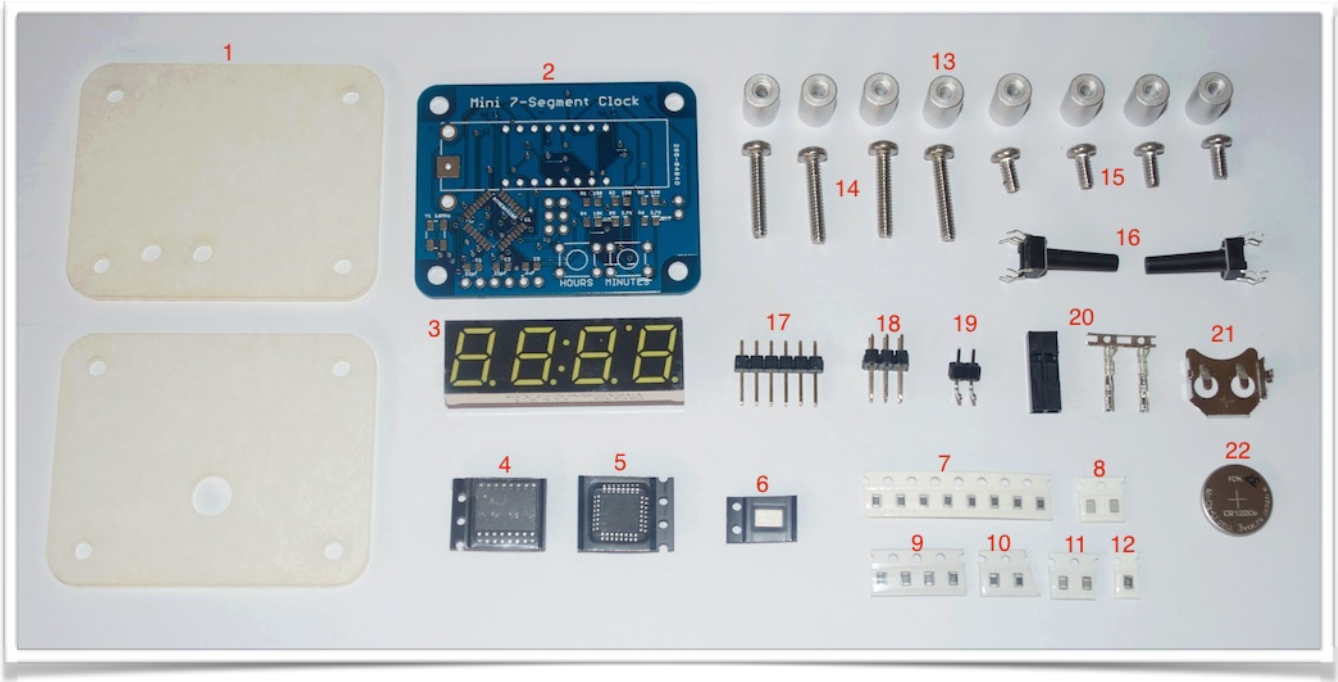
What You'll Need:

1. Soldering Iron
2. Solder
3. Solder Flux (optional)
4. Small Philips head screwdriver
5. ICSP programming cable
6. FTDI Adapter
7. Arduino IDE
8. Power Cord
9. Wire Cutters/Strippers (for your homebrew power cord)

About...

The Mini 7-Segment Clock is an open-source digital clock developed using the Arduino Platform. It's a simple to use clock that utilizes a multiplexed 7-segment display and an extremely accurate DS3231 Real-Time Clock chip that's accurate to +/- 2ppm. By default it runs in 12-hour mode, but with a few code tweaks, it could also display in 24-hour format. Since the DS3231 RTC has an internal temperature-compensated crystal oscillator, the temperature could also be displayed if one chose to implement the feature. The clock has two pushbuttons for setting the hours and minutes. A battery backup keeps the time in the event of a power failure.

ARDUINO
OPEN-SOURCE
COMMUNITY



Kit Contents

1. Acrylic panels (front/back)
2. PCB
3. 7-segment display
4. DS3231 RTC
5. Atmega328p
6. 16 MHz crystal
7. Resistor - 150 ohm - 8 pcs
8. Capacitor - .1uF - 2 pcs
9. Resistor - 10K - 4 pcs
10. Resistor - 2.7K - 2 pcs
11. Capacitor - 22pF - 2 pcs
12. Resistor - 680 ohm
13. 3/8" standoffs - 8 pcs
14. 5/8" 4-40 screw - 4 pcs
15. 1/4" 4-40 screw - 4 pcs
16. 17mm pushbutton switch - 2 pcs
17. 6-pin FTDI header
18. 6-pin ICSP header
19. 2-pin right-angle header
20. 2-pin Dupont connector
21. 12mm battery holder
22. CR1220 3V battery

Data Sheets

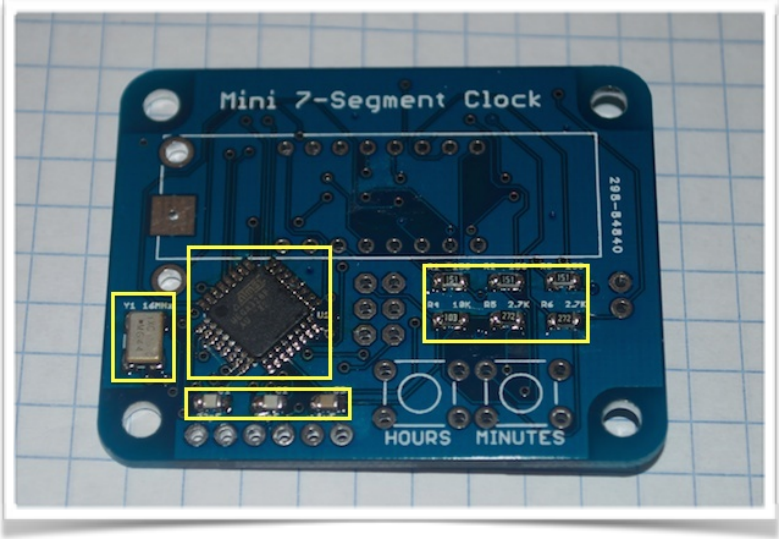
Please consult the datasheets included within the source files for soldering times/temperature.

Components

I've found it's easier to solder all the SMD components first.

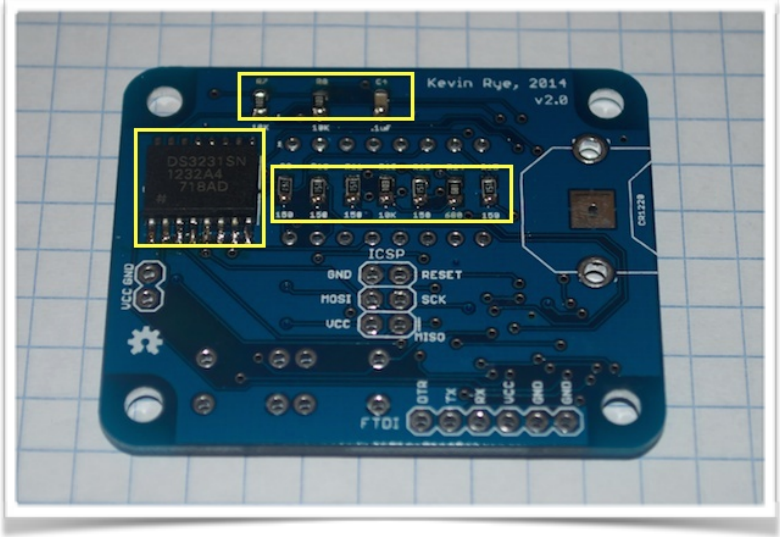
Starting with the front, solder in the:

1. 16 MHz crystal
2. Atmega328p
3. Resistors & capacitors



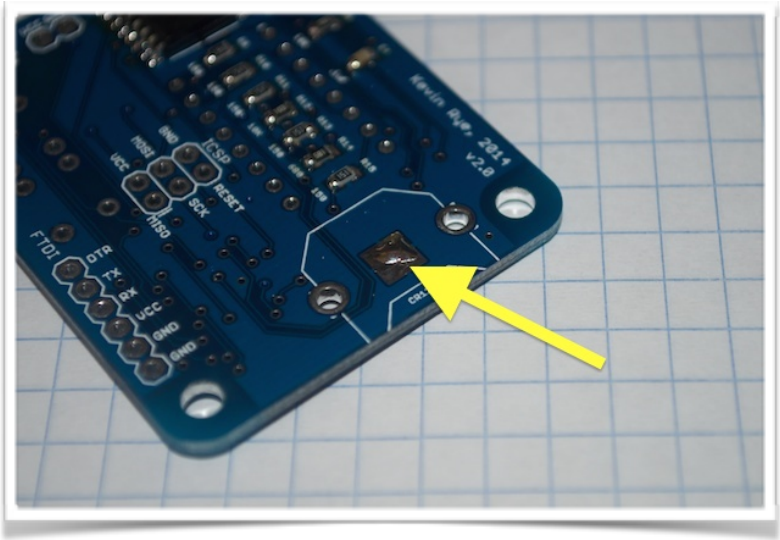
On the back, solder in the:

1. DS3231 RTC
2. Resistors & capacitors



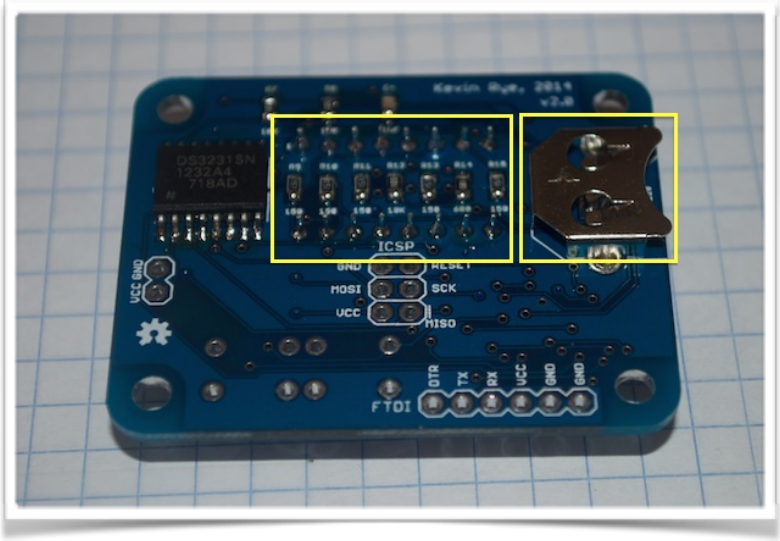
Components (Continued)

I saw a recommendation on an Adafruit tutorial to add a small blob of solder to the ground pad on the battery holder. It gives the battery a tight fit, helps make proper contact, and stops it from sliding out.



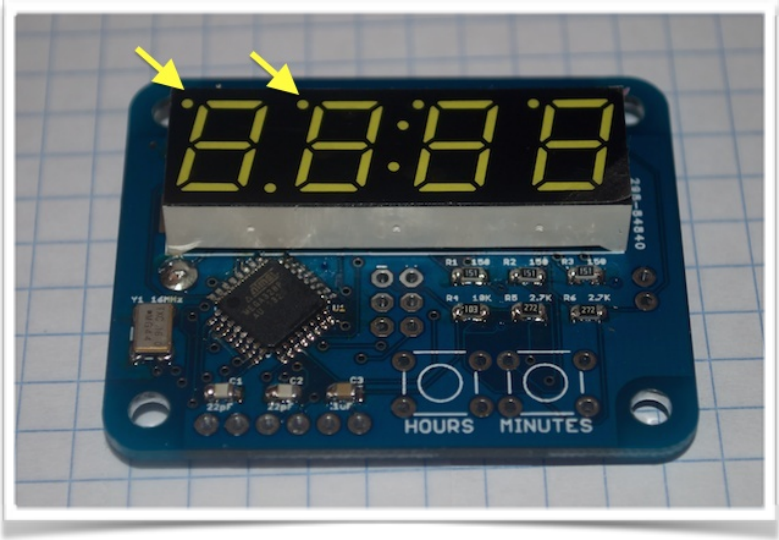
Once you've added a small blob, solder in the battery holder, followed by the 7-segment display. Optionally, you could skip to the next step and solder in all the headers and leave the display until the very end in case you're concerned about scratching it in the process.

Although the display has a protective film on the front, I sometimes like to stick a Post-it to it as an extra precaution.



NOTE: The 7-Segment Display is soldered in UPSIDE DOWN.

Consulting the data sheet would have you insert the display with the decimals on the bottom. However, the decimals in this design are being used for the PM indictor.



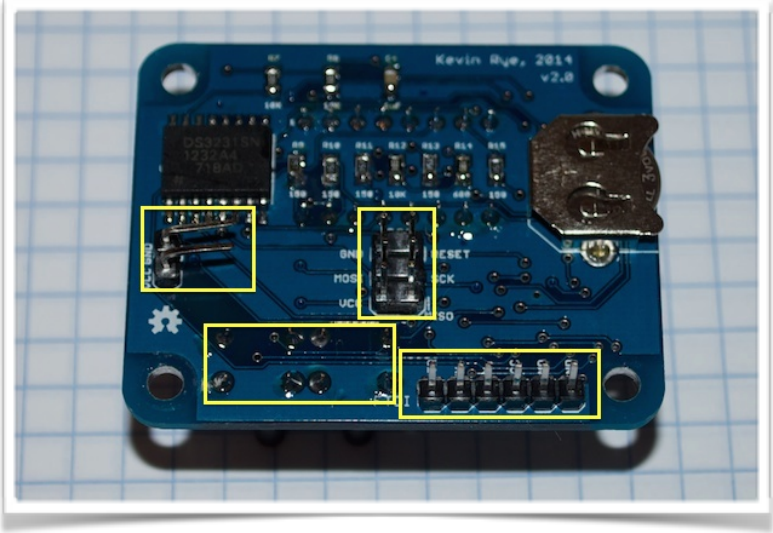
Components (Continued)

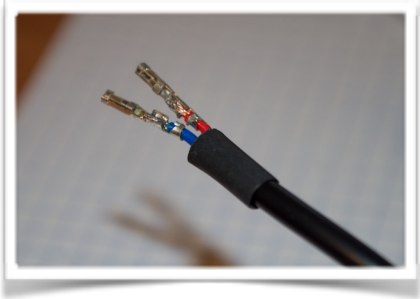
Continuing with the back of the board, solder in the 6-pin headers for ICSP and FTDI.

This of course is optional if you want to use pogo pins for programming.

Additionally, once the bootloader and sketch is uploaded, you might consider unsoldering them or clipping the leads in order to create some additional space for a power cord.

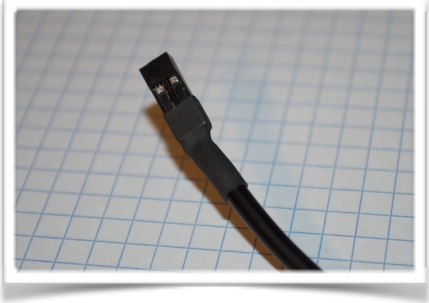
You may feel that the included 2-pin Dupont connector is overkill and choose to simply solder your Vcc and Ground leads directly to the PCB. I'd imagine once the clock is assembled, you won't be taking it apart. So a detachable power cord may be completely unnecessary.





Solder the crimp pins to a cut-off power cord / USB cable and slide on a length of heat shrink tubing.

Attach the Dupont connector.



Shrink the tubing.

There's a little arrow on one of the pins. Take note of which one is which before covering the wires with the tubing. I try to always have the arrow on the Vcc pin, but the choice is yours.

Burning the Bootloader

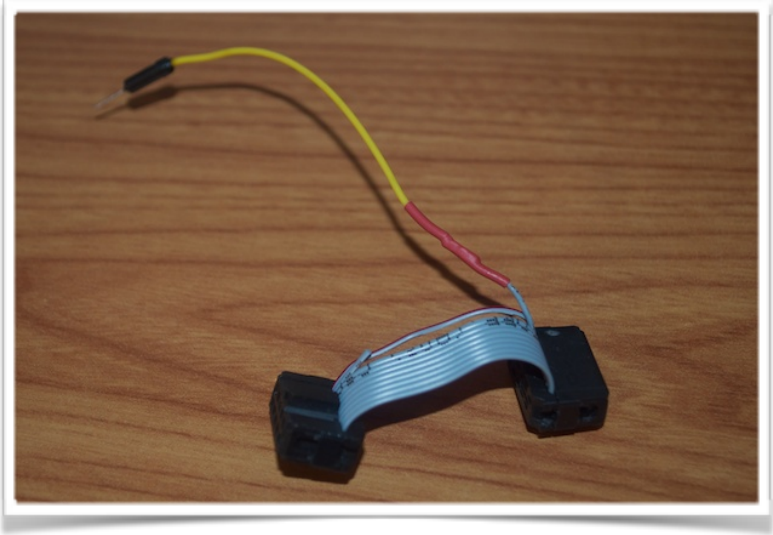
There's many ways to burn the bootloader. Some people choose to breadboard their Atmegas, use an AVR MkII with a command line tool, or use an Arduino as an ISP. I've found using an Arduino as an ISP is the easiest way.

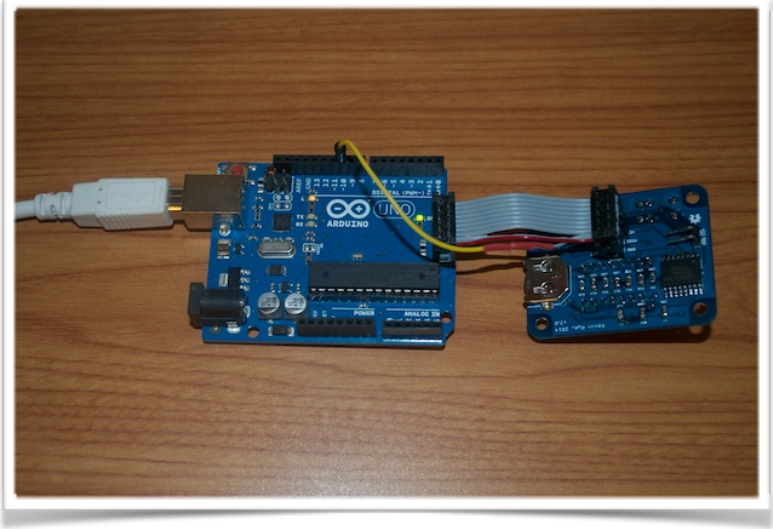
There's no shortage of information on how to set up your Arduino as an ISP and bootload an Atmega, but I've found a pretty straight forward way to do it. I happened across this pretty cool tutorial that walks you through the process.

All you need to do is download the sketch that's linked on the site and connect the ICSP headers together between your Arduino and the Mini 7-Segment Clock. You just have to breakout the Reset pin on the Mini 7-Segment Clock header and connect it to pin 10 on your Arduino.

Run the sketch, answer the prompts in the Serial Monitor window, and you're off to the races.

Here's the homebrew programming cable per the tutorial linked to the left. I didn't have a 6-pin ribbon cable, so I used an 8-pin one. It fits. The extra pins just hang off the header. I cut the wire running to the Reset pin on the target header and soldered a lead to it so that I could plug it into pin 10 on my Arduino.

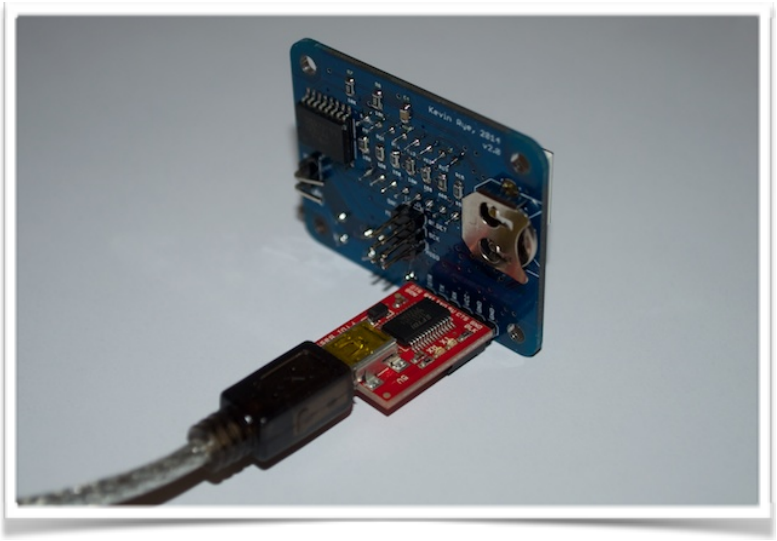




Uploading a Sketch

Again, you could breakout the FTDI header to your breadboard, set up your Arduino as an ISP and upload your sketch, or simply use an FTDI programmer. For $15, it's a huge timesaver.

SparkFun's FTDI Breakout Board works seamlessly with the Arduino IDE.



Button Up

At last! You've soldered it all together, burned your bootloader, and uploaded your sketch. All that remains is to button it up and you can put it on display for all to see!

1. Start with the front panel. Attach 4 standoffs with the 5/8" screws.

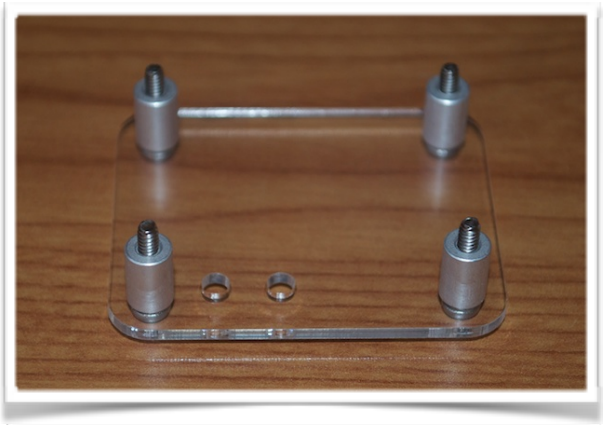
2. Slide the PCB over the 5/8" screws and attach the remaining 4 standoffs. Be careful with those buttons sticking out the front. You probably don't want to mash them on the table.

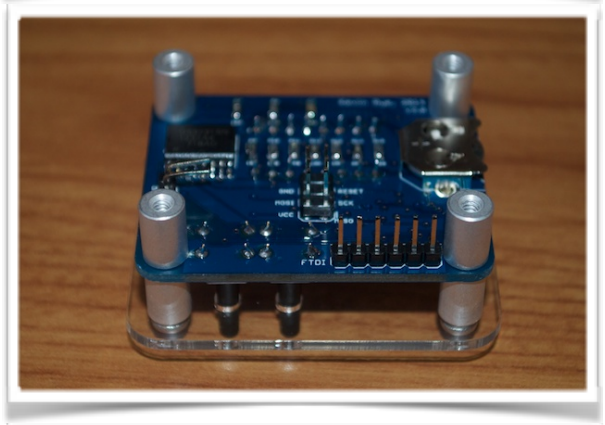
3. With the PCB secured, you can either attach your homebrew power cord using the 2-pin Dupont connector, or solder your leads directly to the PCB. Just remember to slide the cord through the back panel first.

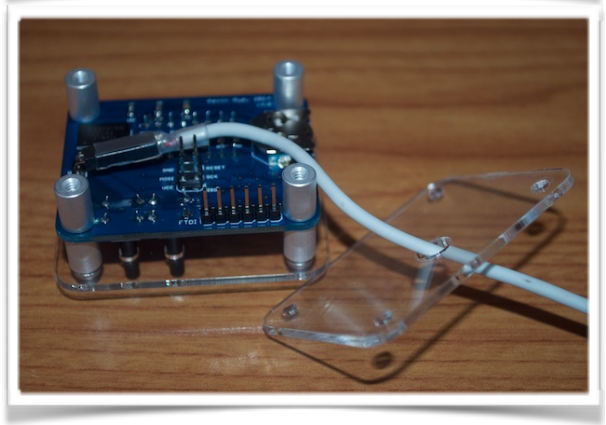
Depending on the thickness of the cord you used, space might be a little tight with the ICSP header in the way. In a future revision, I might move the ICSP header a little more to the right. But for now, it is what it is.

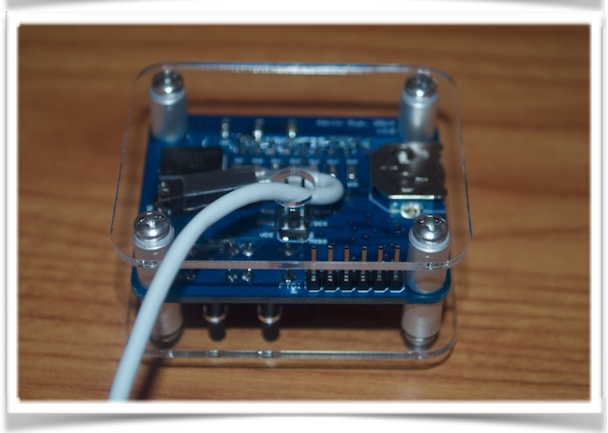
In the images on the right I used a cheap $5 knockoff iPhone cable and just cut the end off. It fit pretty well, and the loop in the cord acts as a pretty effective strain relief.

4. Finally, secure the back panel with the 1/4" screws.









Display

This is a proud moment. You did it!

When you power on the clock, it'll display "12:00". All you have to do is set the time with the HOURS and MINUTES buttons.

That's it.

The time is then stored within the DS3231 RTC and maintained with a battery backup.