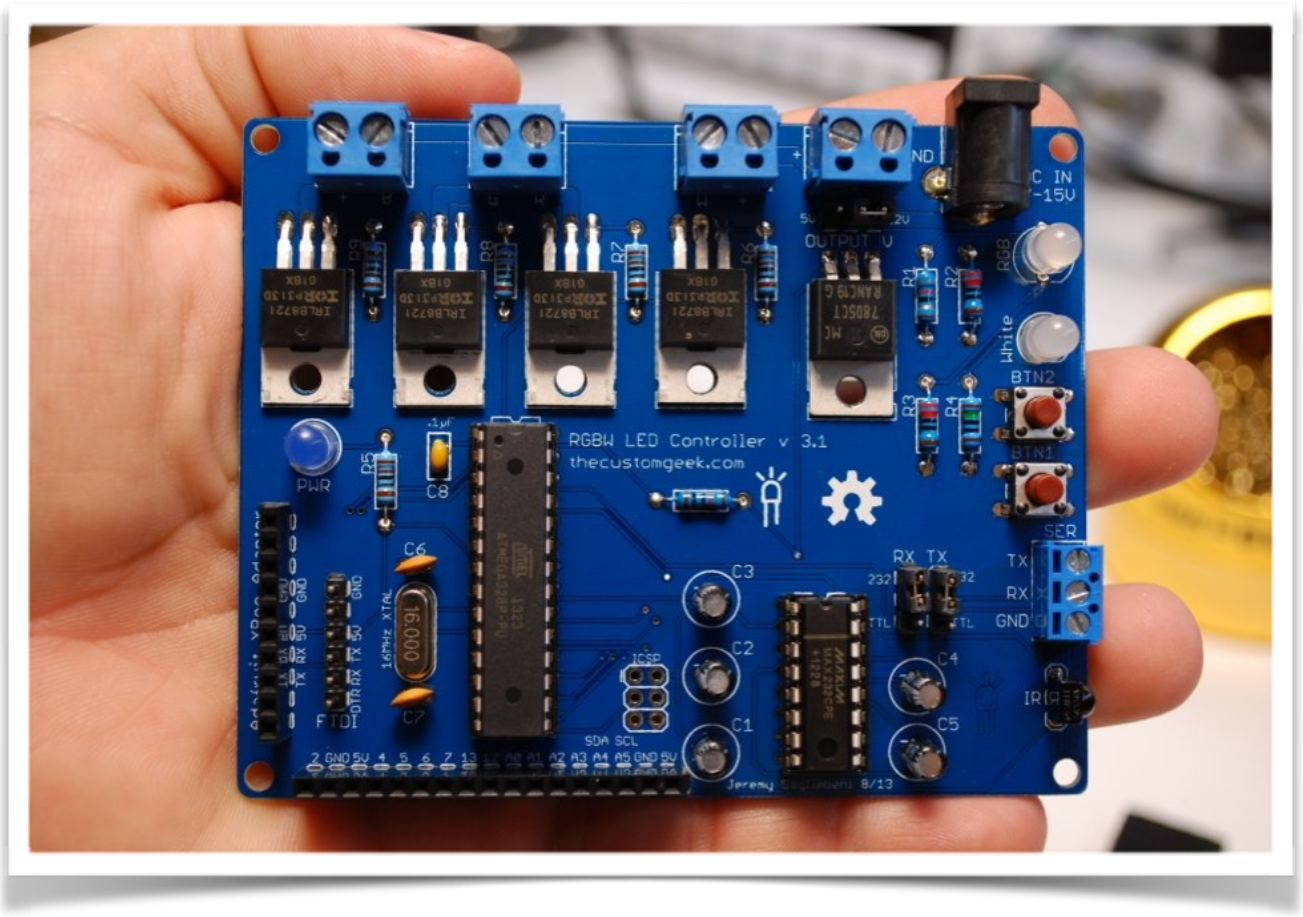
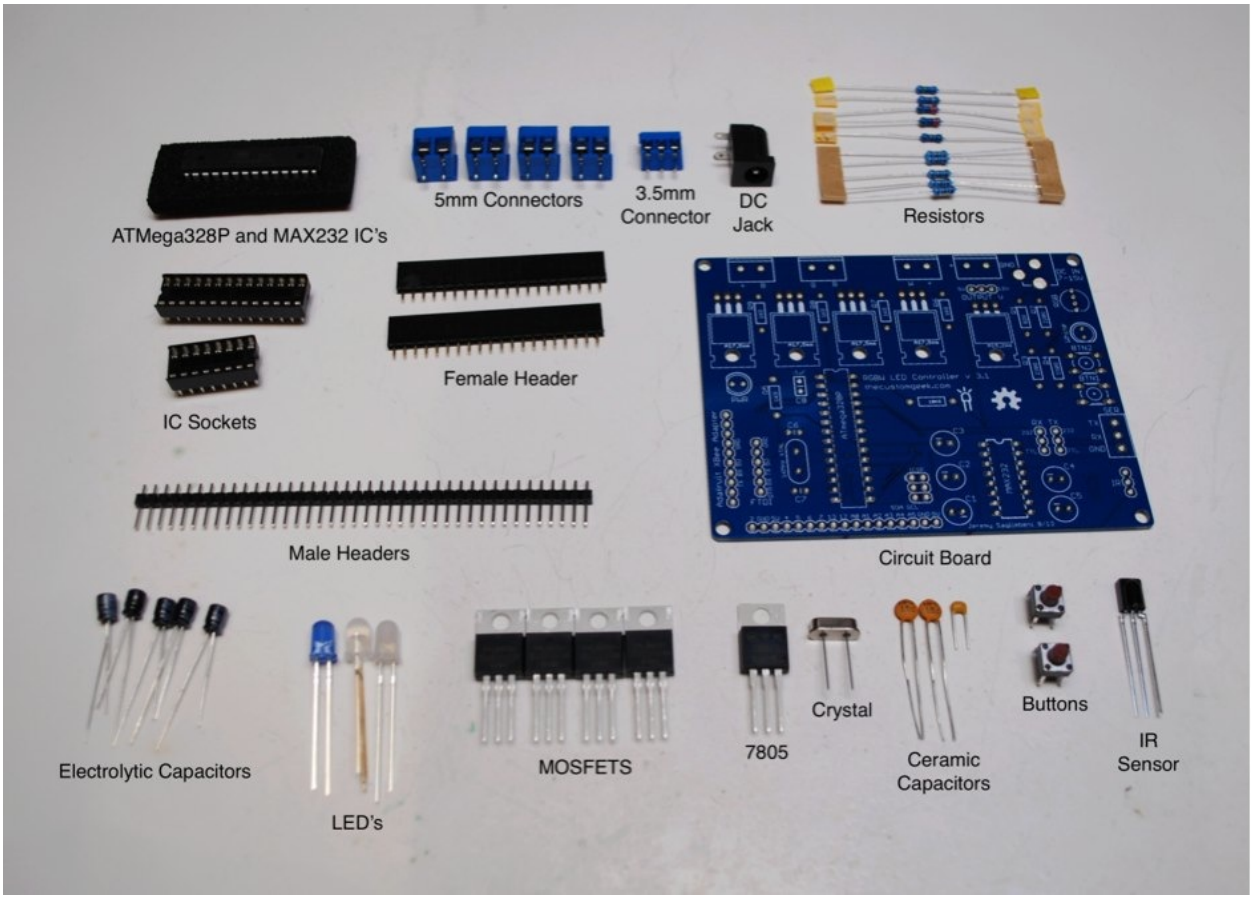
Building the RGBW LED Controller



A guide for the assembly and operation of your RGBW LED Controller.
ver 3.1

Getting Started

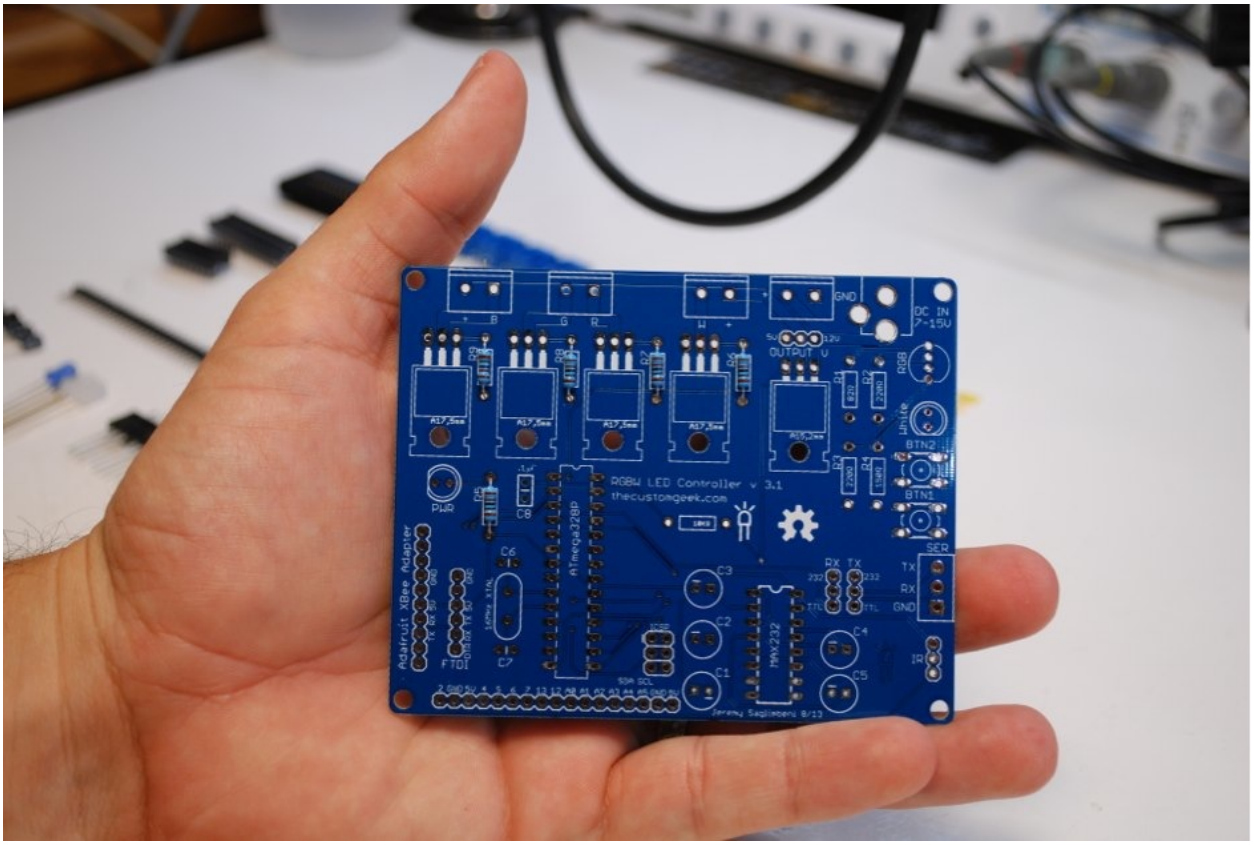
Parts list - You should have received the following parts:
(1) Circuit Board, (1) ATMega328 and (1) MAX232 ICs, (4) 5mm 2 position Connectors, (1) 3.5mm 3 position connector, (1) DC Jack, (10) resistors (5 1KΩ, 1 10KΩ, 2 220Ω, 1 150Ω, and 1 82Ω)(1) 28 Pin IC Socket, (1) 16 Pin IC Socket, (2) strips of Female Header, 1 strip of Male Header, (5) .1µF Electrolytic Capacitors, (3) 5MM LEDs (blue, white, and RGB), (4) MOSFETs, (1) 7805 Voltage Regulator, (1) 16MHz Crystal, (2) 18pF Ceramic Capacitors, (1) .1µF Ceramic Capacitors, (2) buttons, (1) IR Sensor, and 3 jumpers (not pictured).



Resistors - First we will put the resistors in. I personally like to cut them from the 'tape' they are shipped in because it eliminates any adhesive residue that might give us a headache later. If you are not familiar with 5 band resistors, the values are as follows:

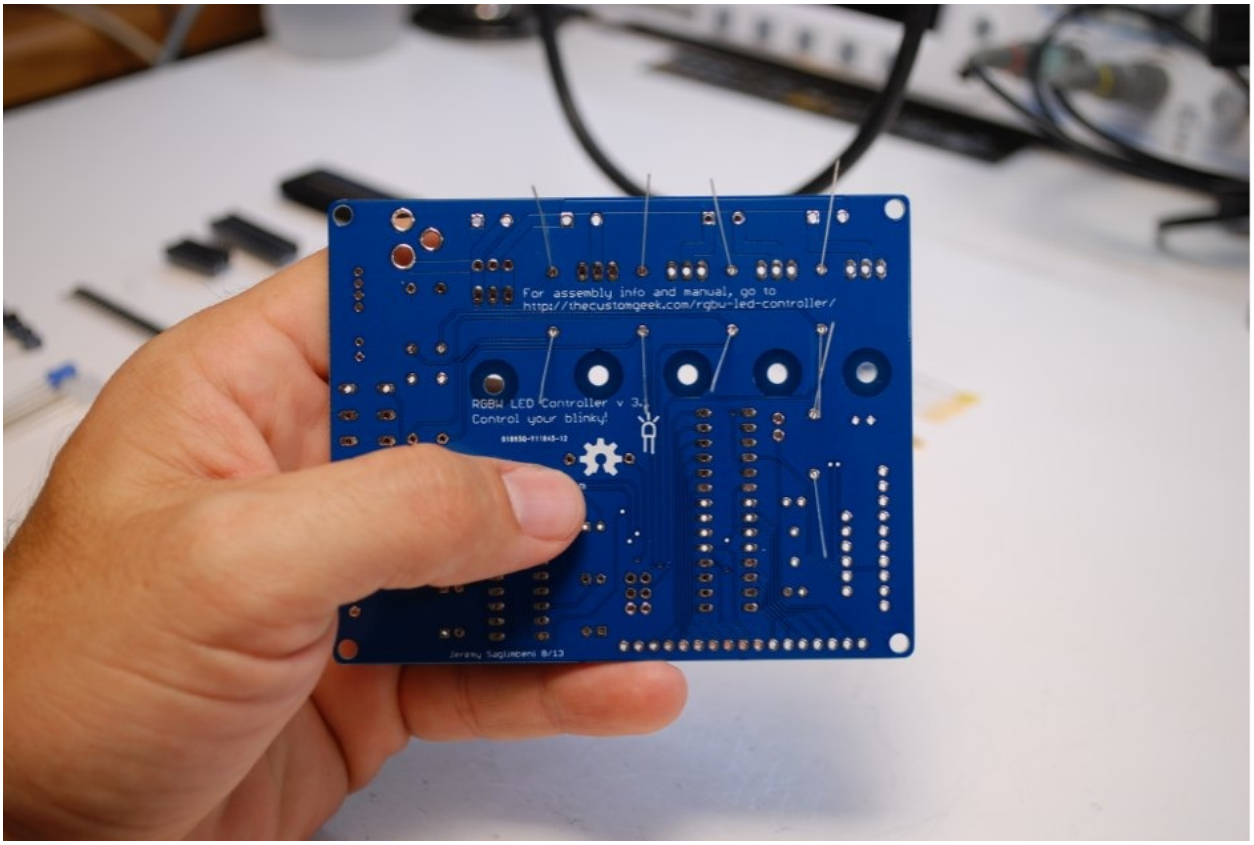
82Ω – Grey, Red, Black, Black, Brown
150Ω – Brown, Green, Black, Black, Brown
220Ω – Red, Red, Black, Black, Brown
1KΩ – Brown, Black, Black, Brown, Brown
10KΩ – Brown, Black, Black, Red, Brown

Lets start with putting the resistors in the board in their spots according to their marked values.

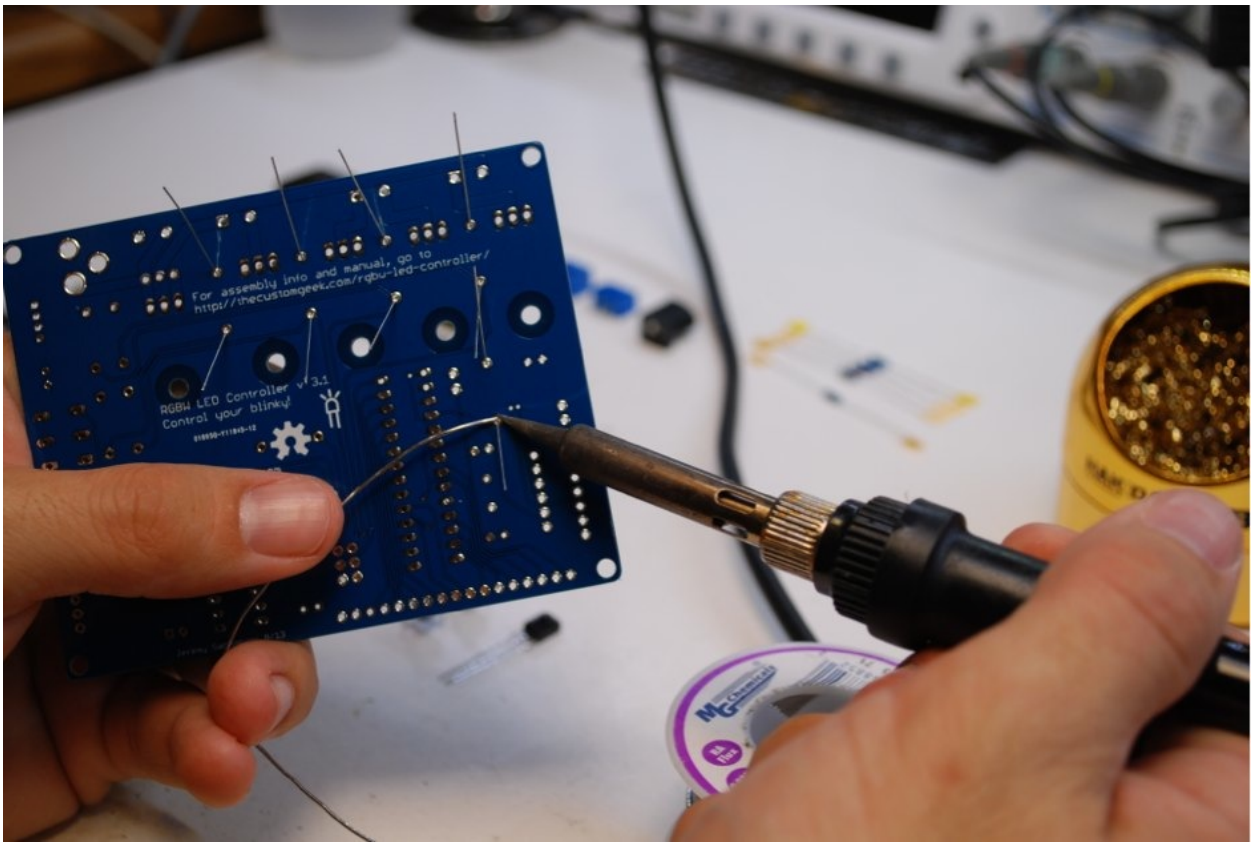


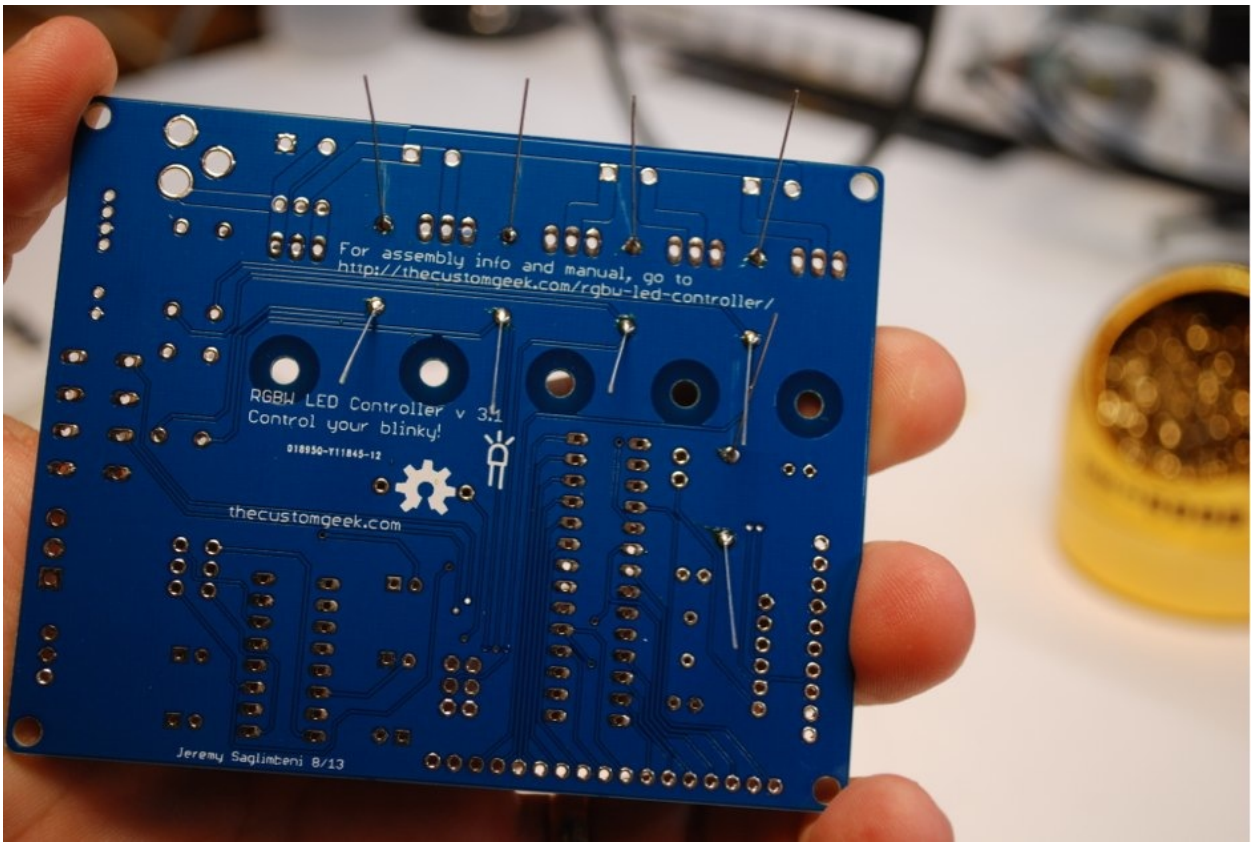
This is what the back of the board should look like. Note that resistors are not polarized, and it does matter witch way they are inserted.

You can slightly bend the resistor leads out, this will help keep the resistor in place when you turn the board over to solder it.You can insert a handful of resistors, then solder them 4 or 5 at a time.

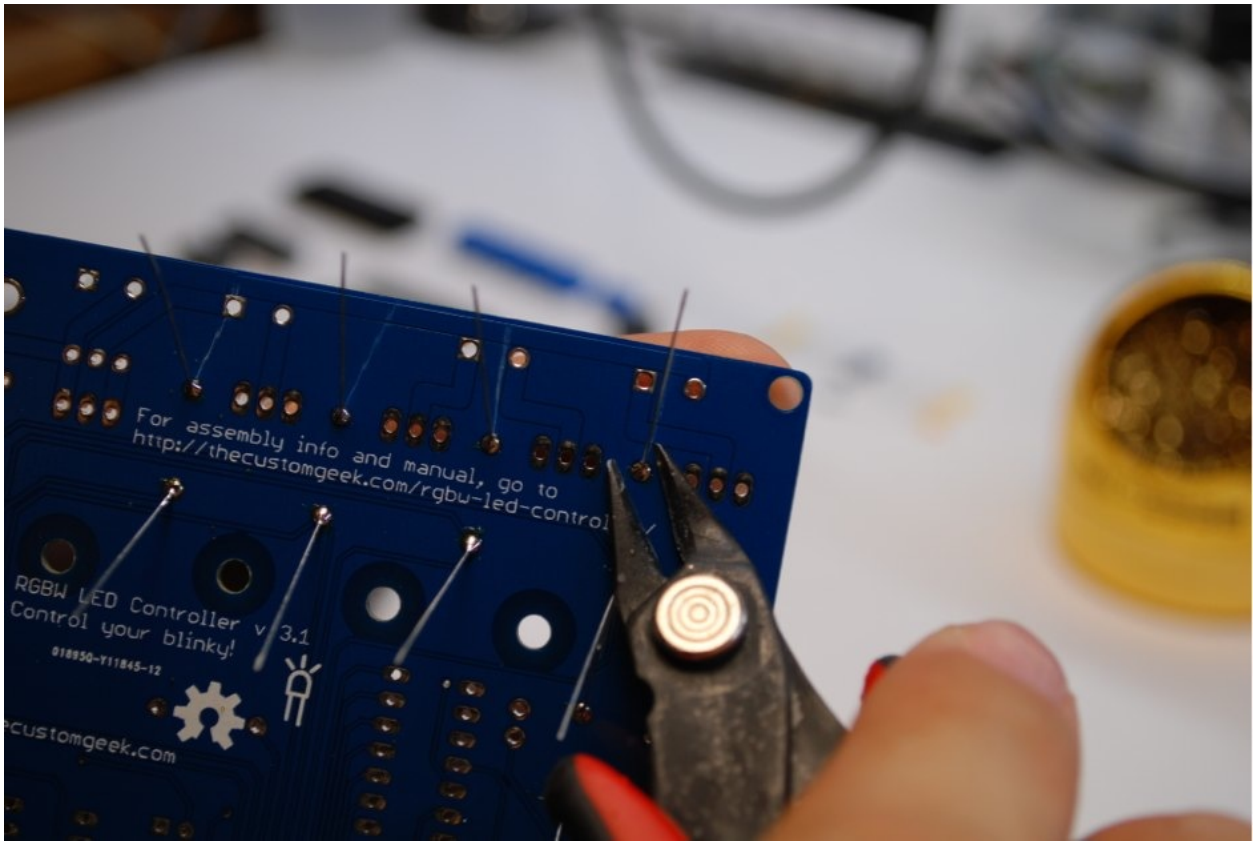


Solder them up good!

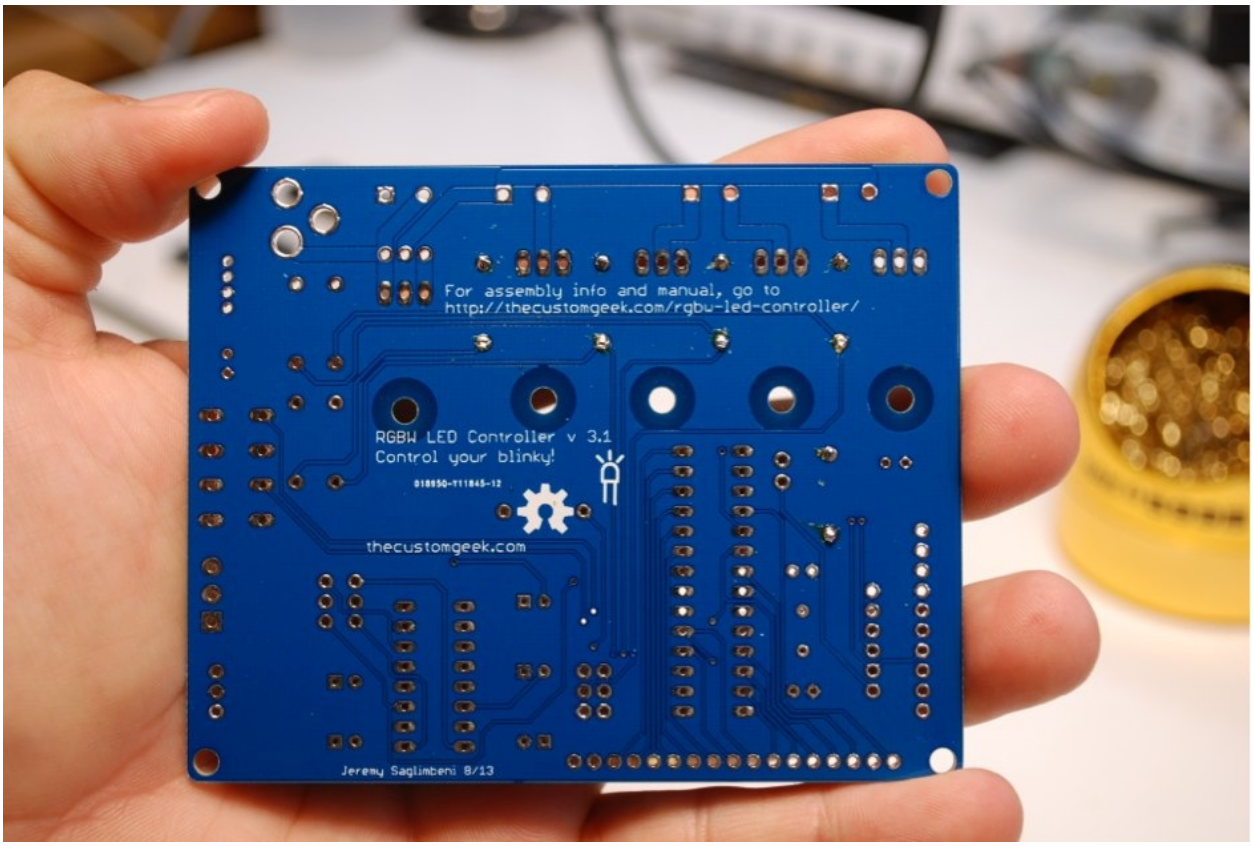


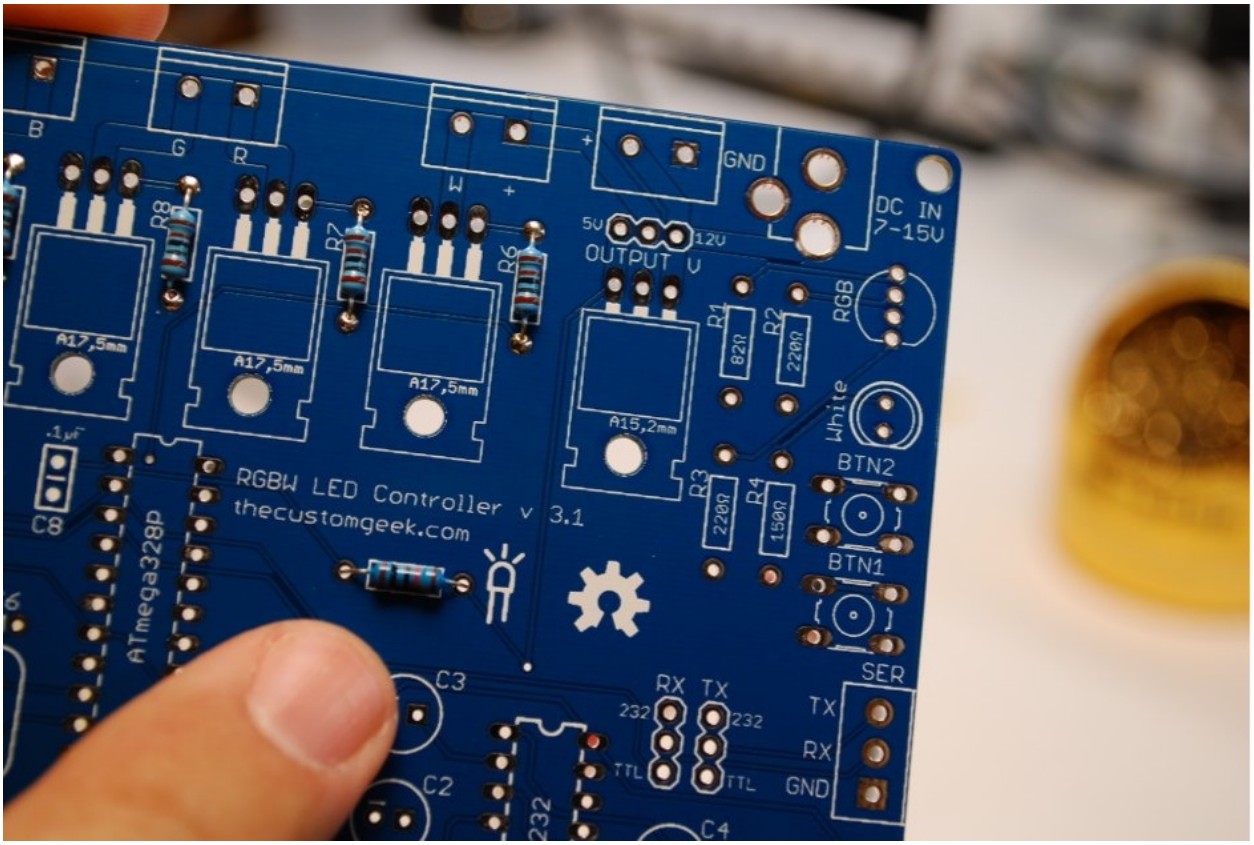


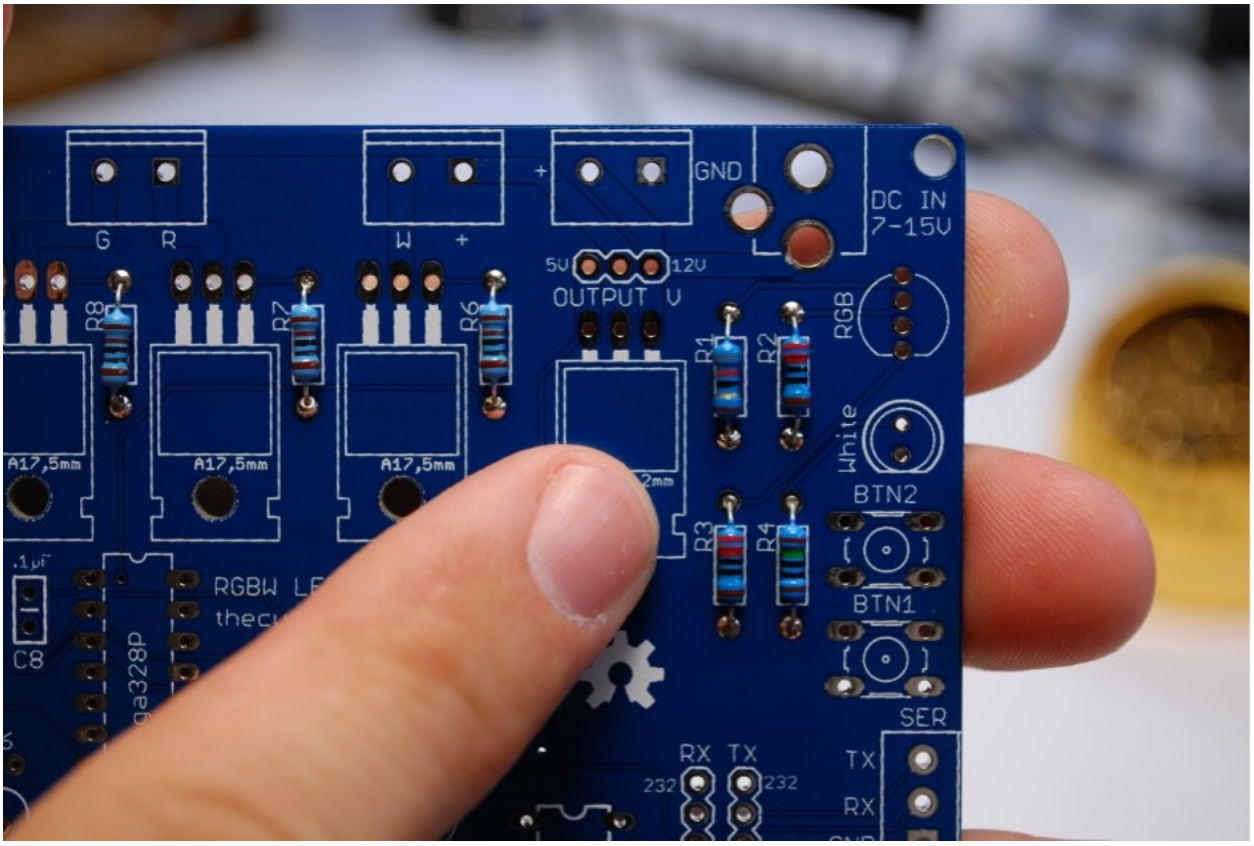
Once soldered, trim the leads with a pair of good quality diagonal cutters.



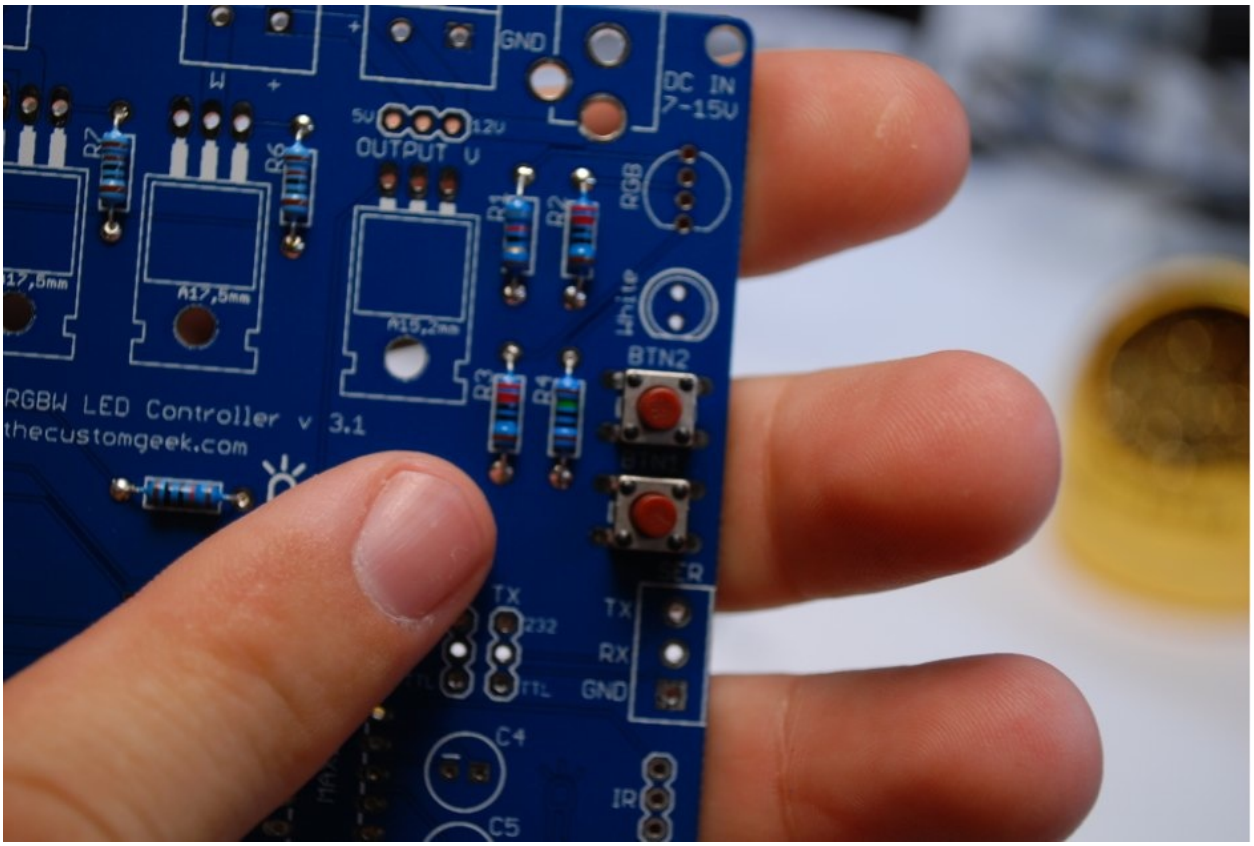
When you are done soldering and trimming, the board should look somewhat like this.



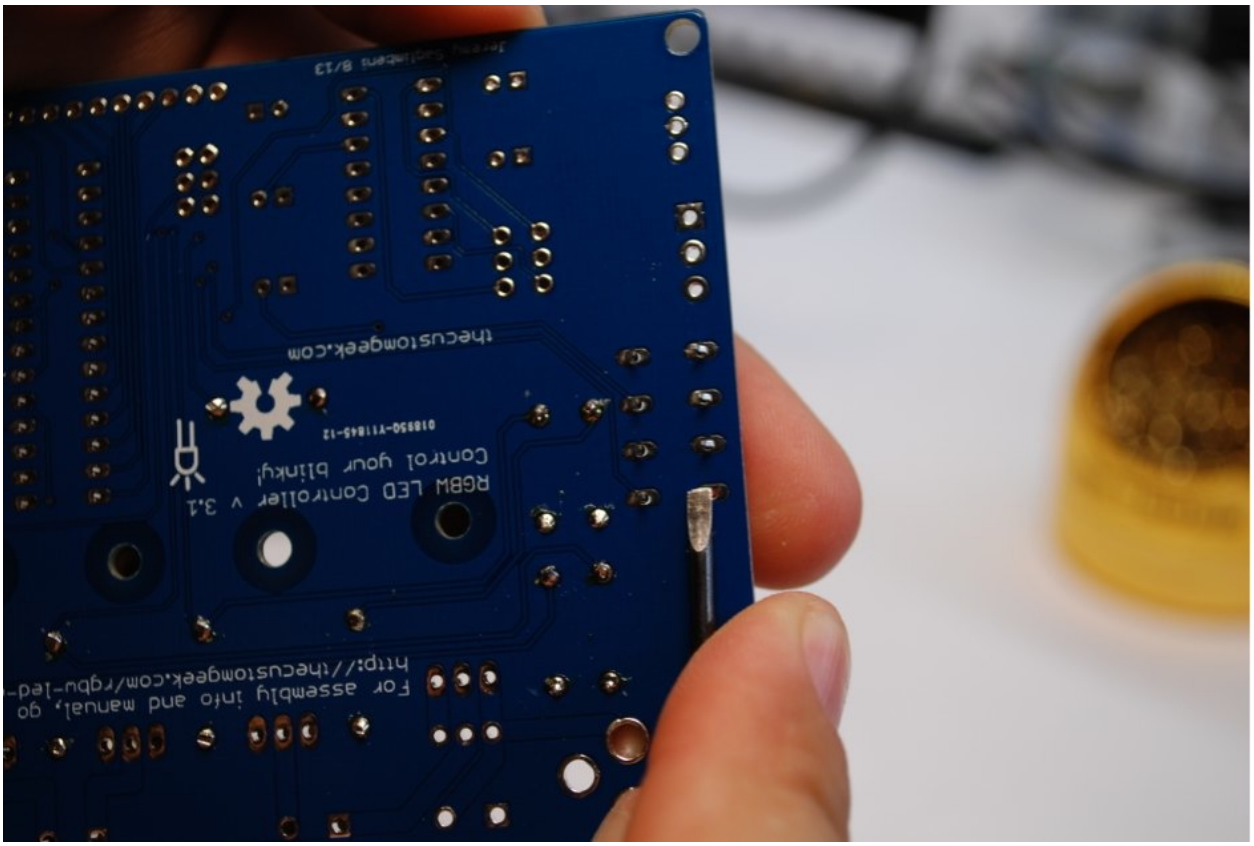




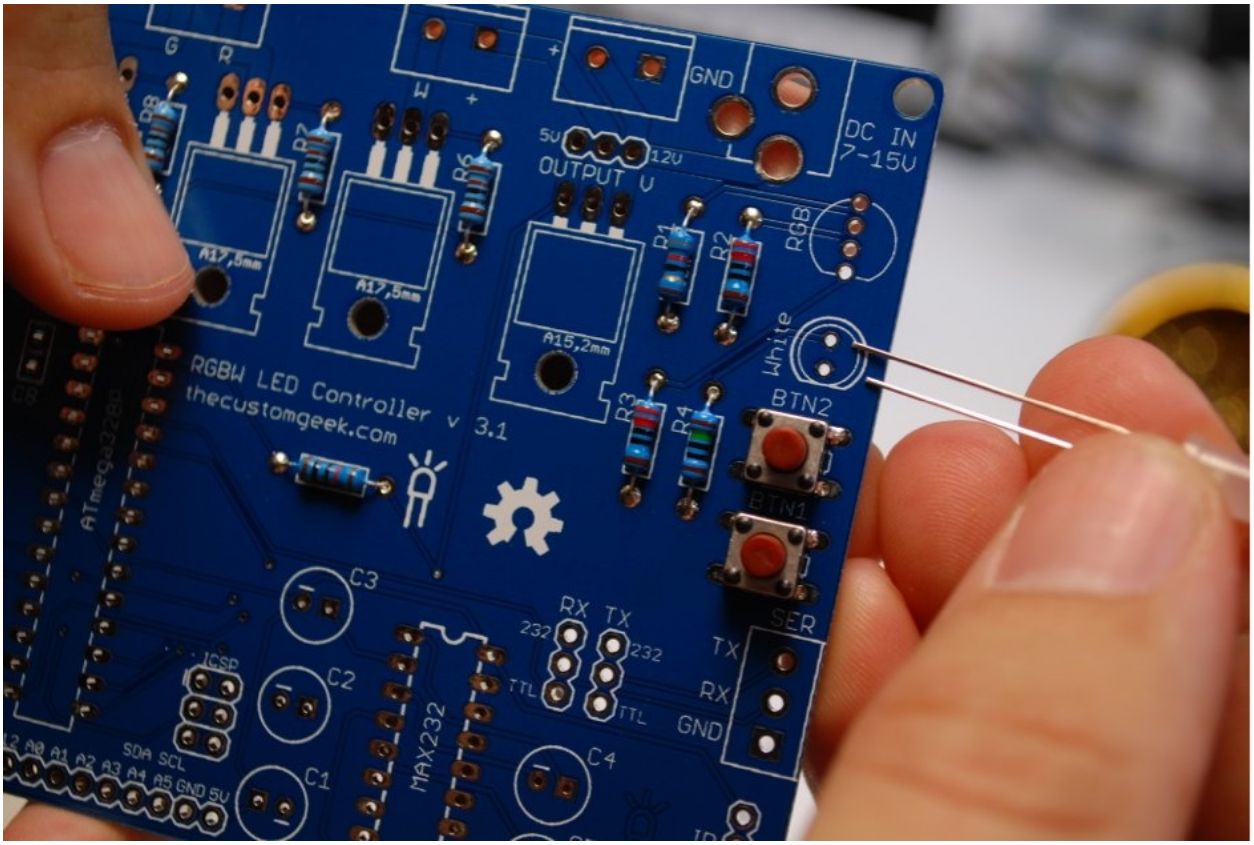
Next insert the 2 buttons into the board. Note that the pin pattern of the buttons are not square, they are rectangular.



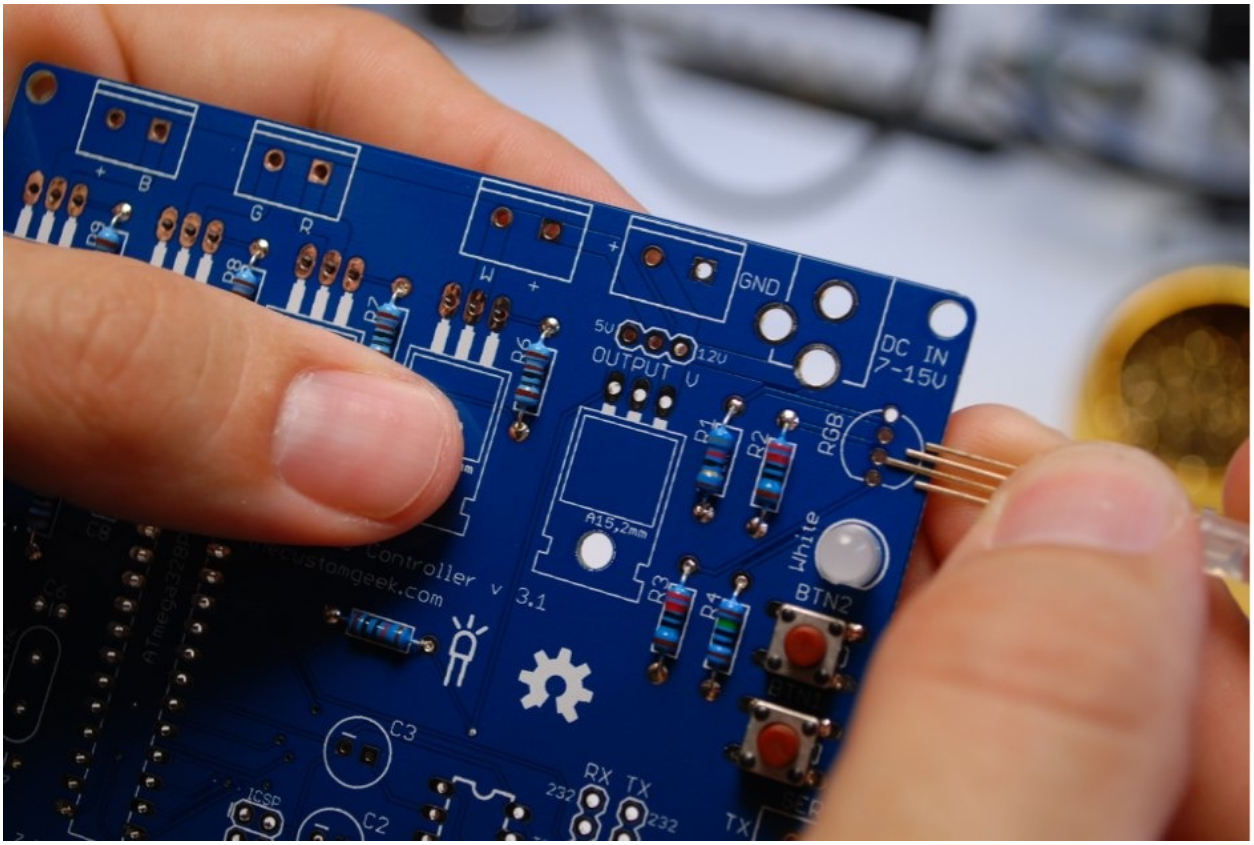
Note that you can bend the pins of the switches after they are snapped in. This is not required, but make for a cleaner profile once everything is soldered. Solder the buttons in.



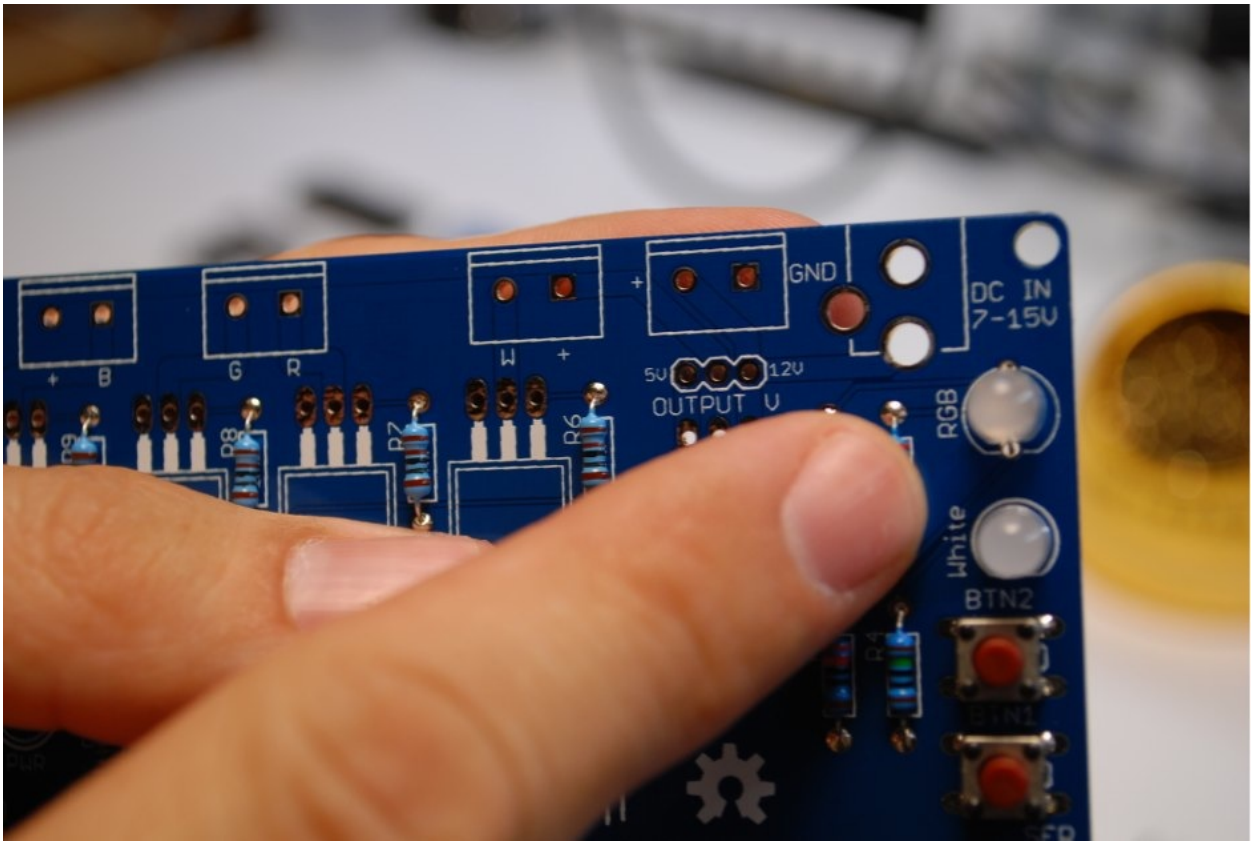
Insert the white LED as shown, the flat side of the LED will be the shorter lead, this is the negative side. It will match the silkscreen on the board.



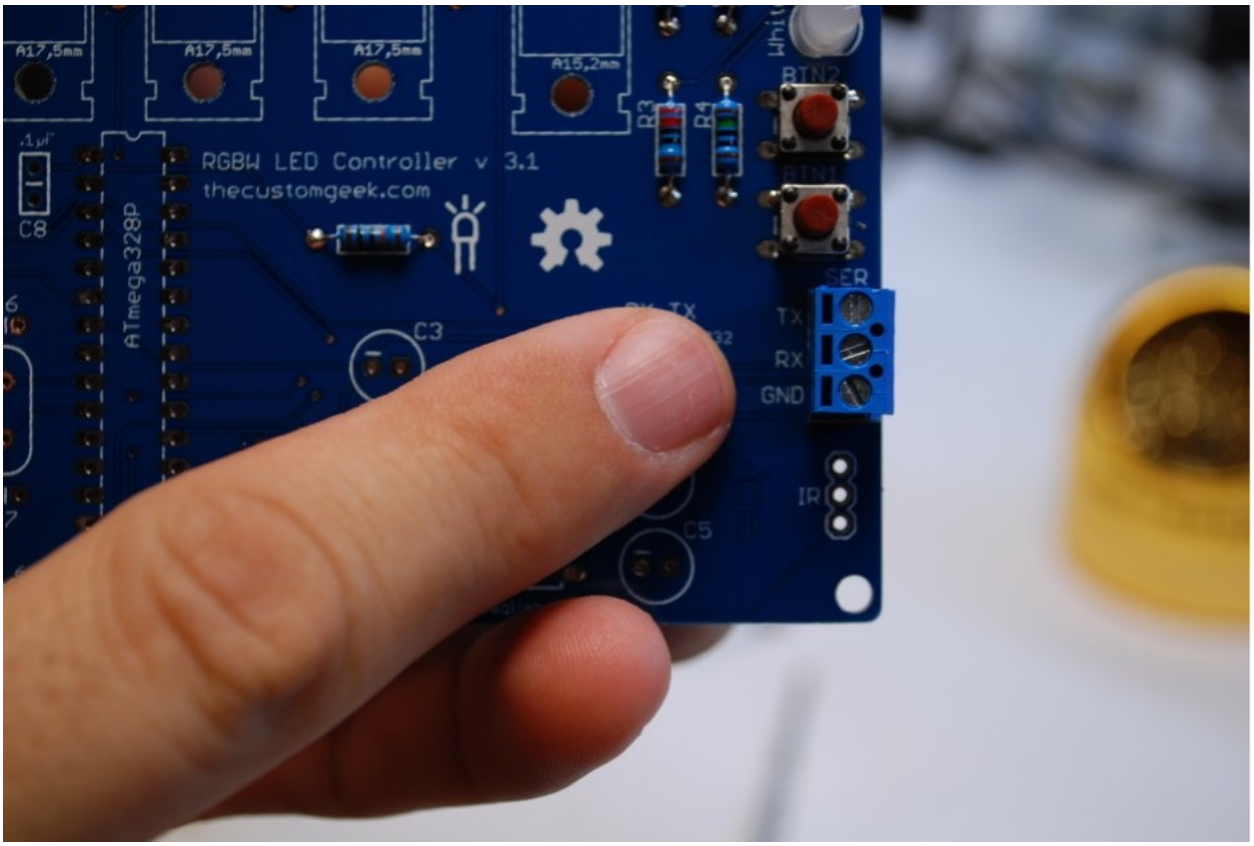
Insert the RGB LED as shown, shortest lead next to to flat side as marked on the silkscreen.



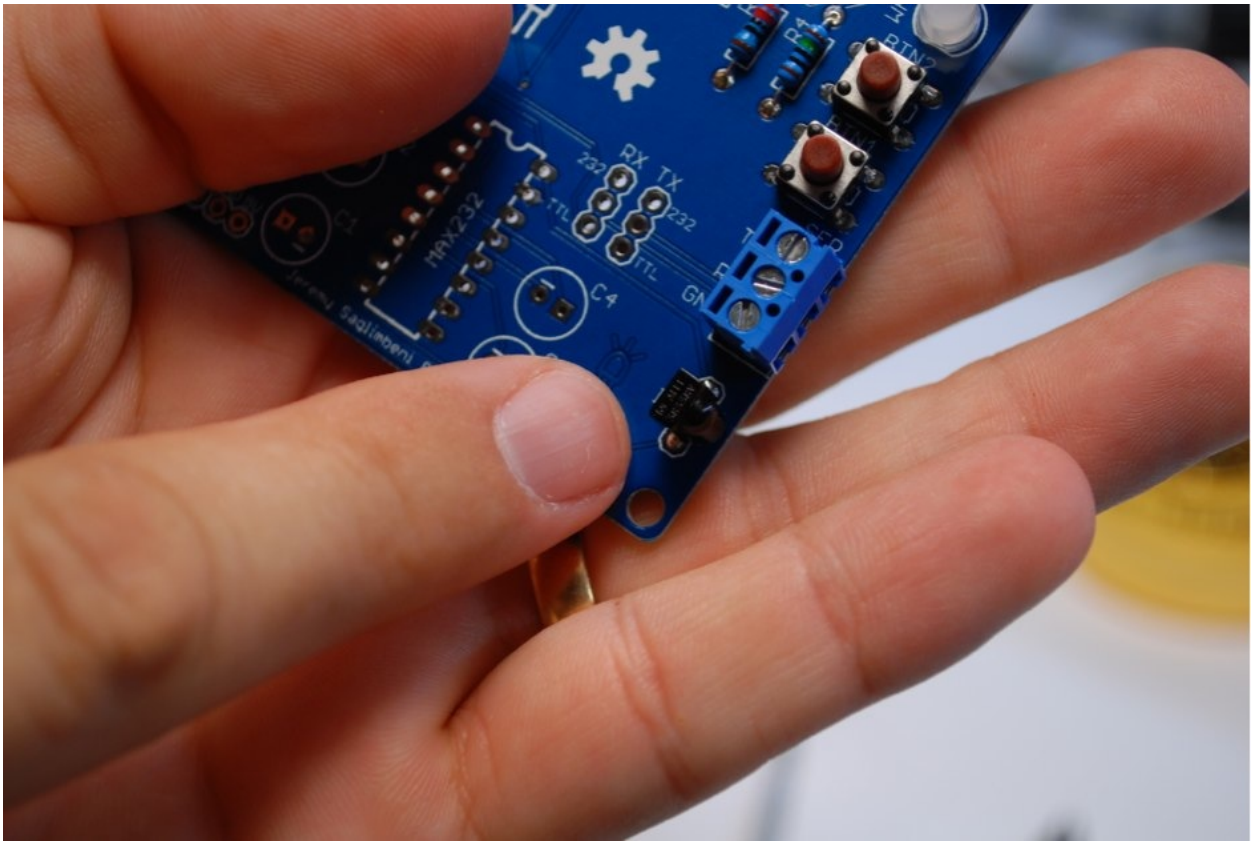
They should look like the picture below when both inserted.



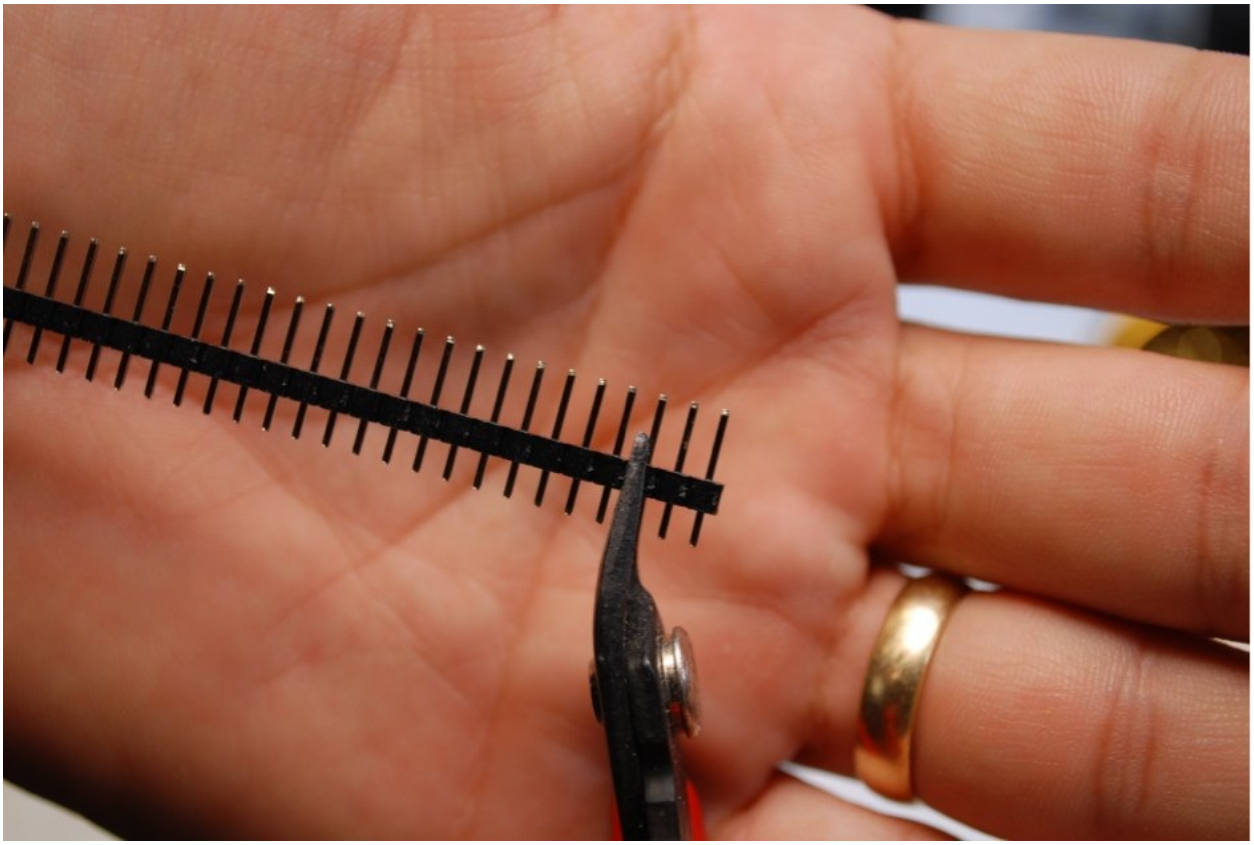
Next solder the 3.5mm 3 position serial connector.

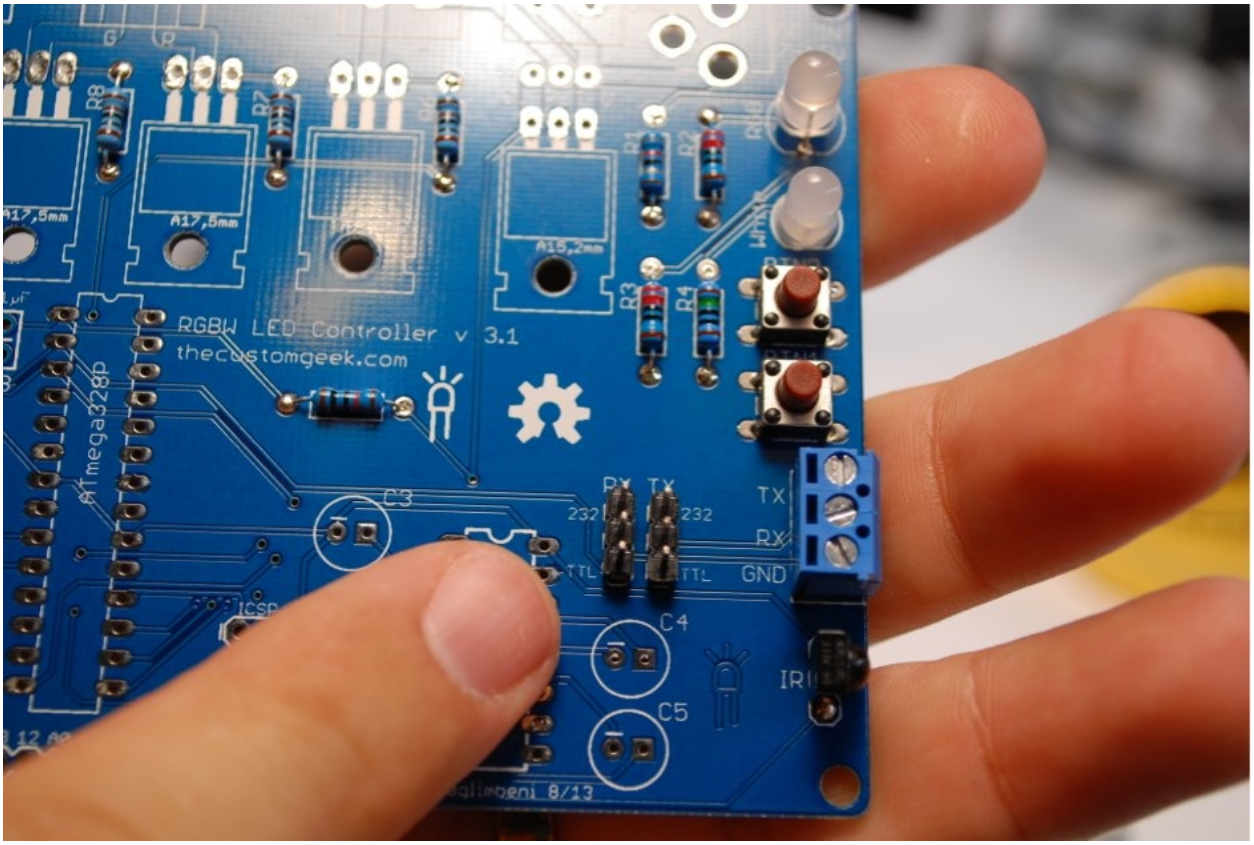


Next, solder the IR sensor as shown, with the rounded side facing out.

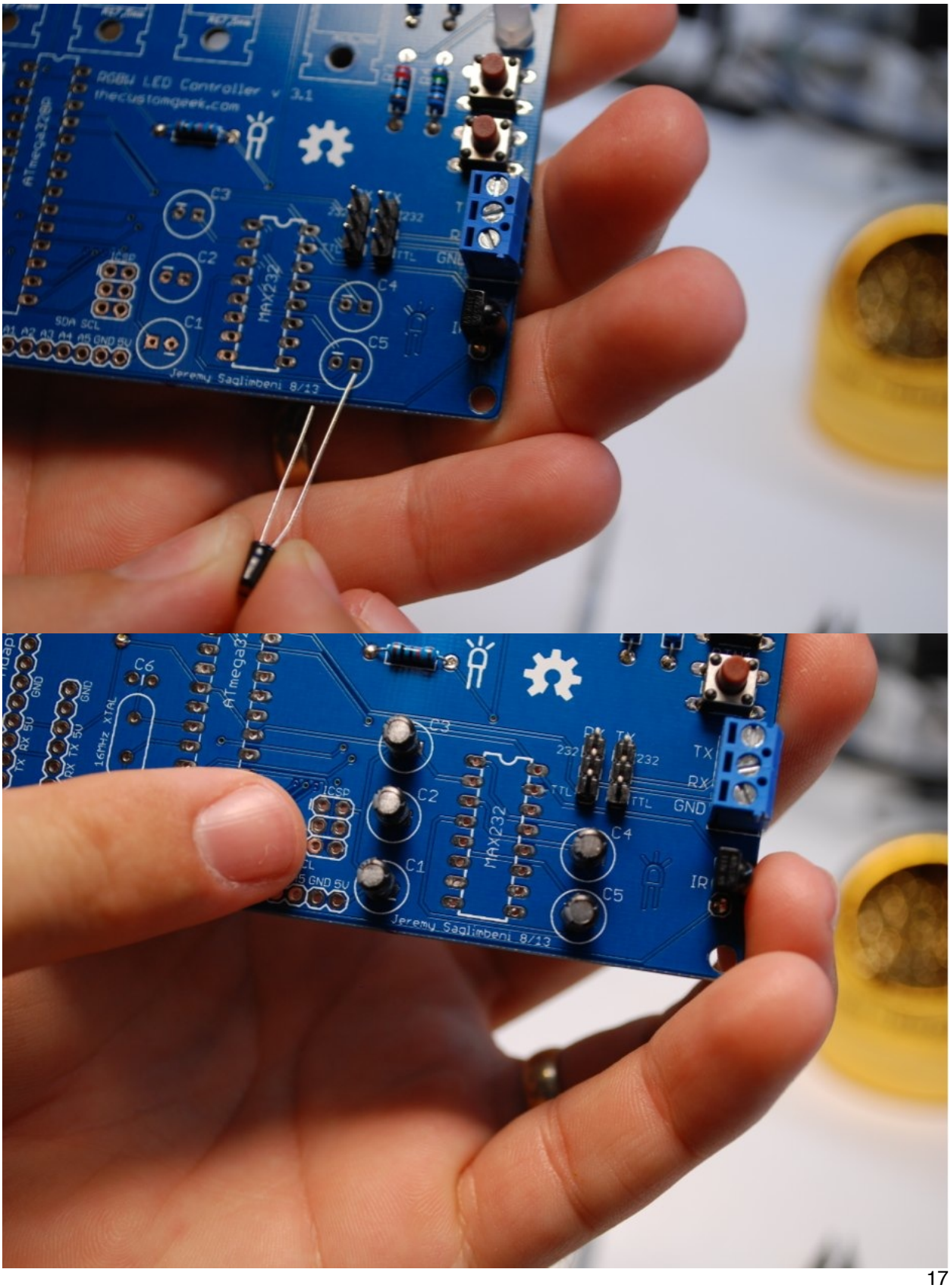


Next, trim the header to two sections of 3 pins each, then solder them to the board.

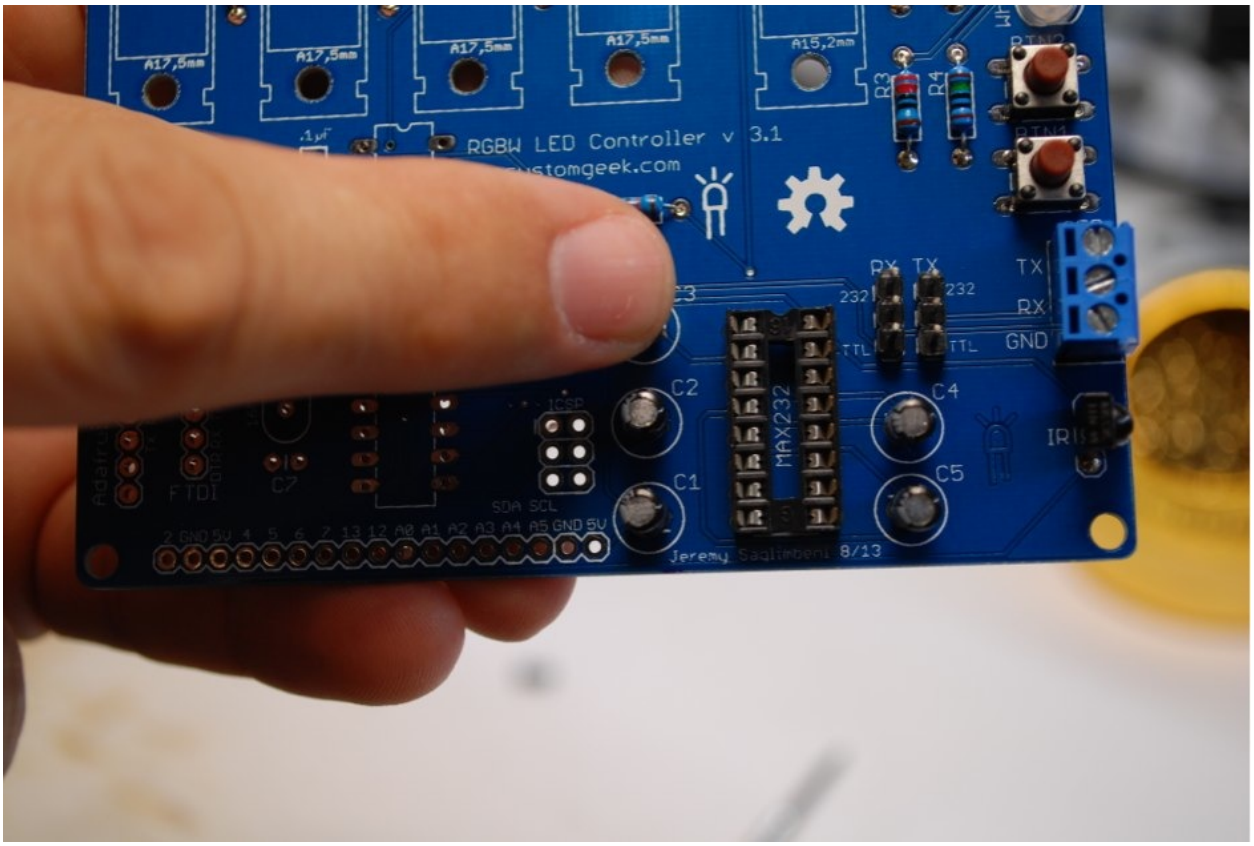




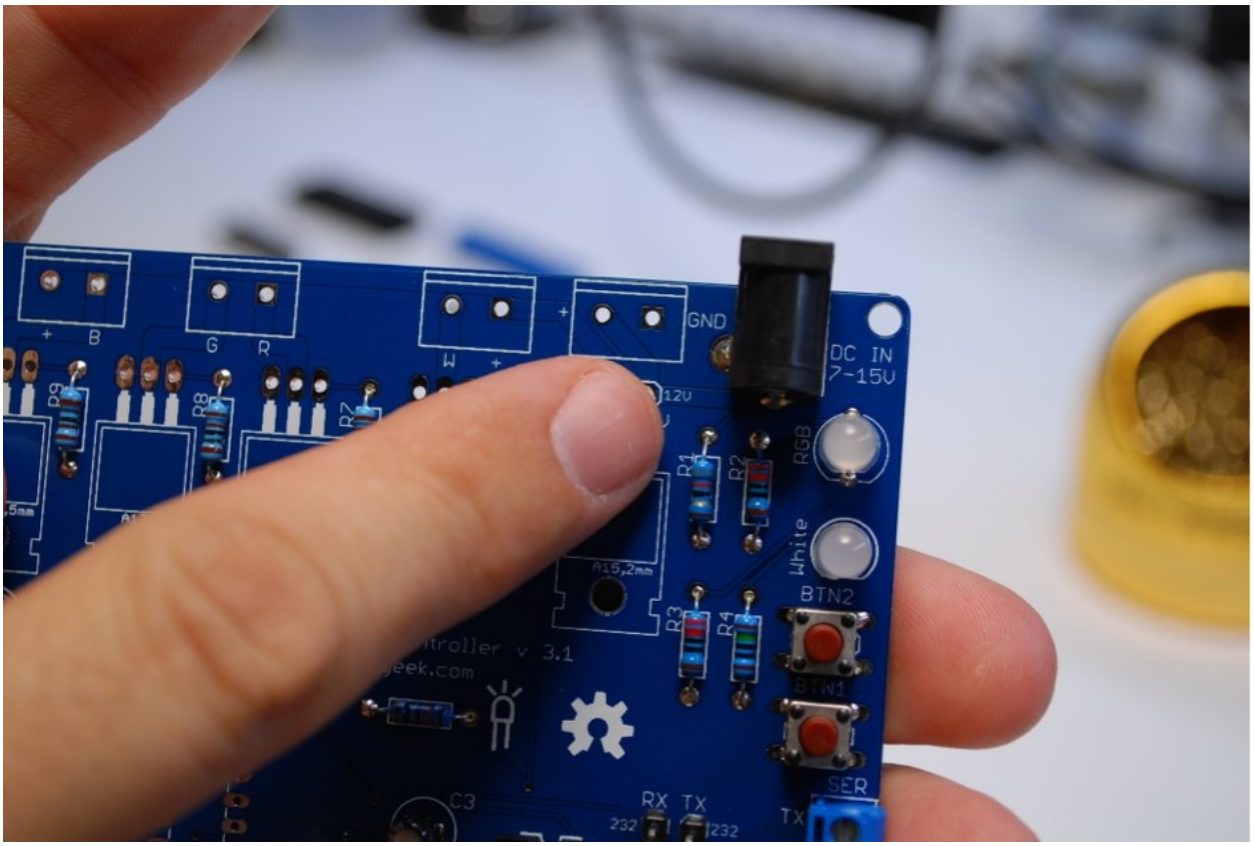
Next, insert and solder the 5 .1µF capacitors to the board as shown. Note the longer lead, the anode, goes in the square hole.



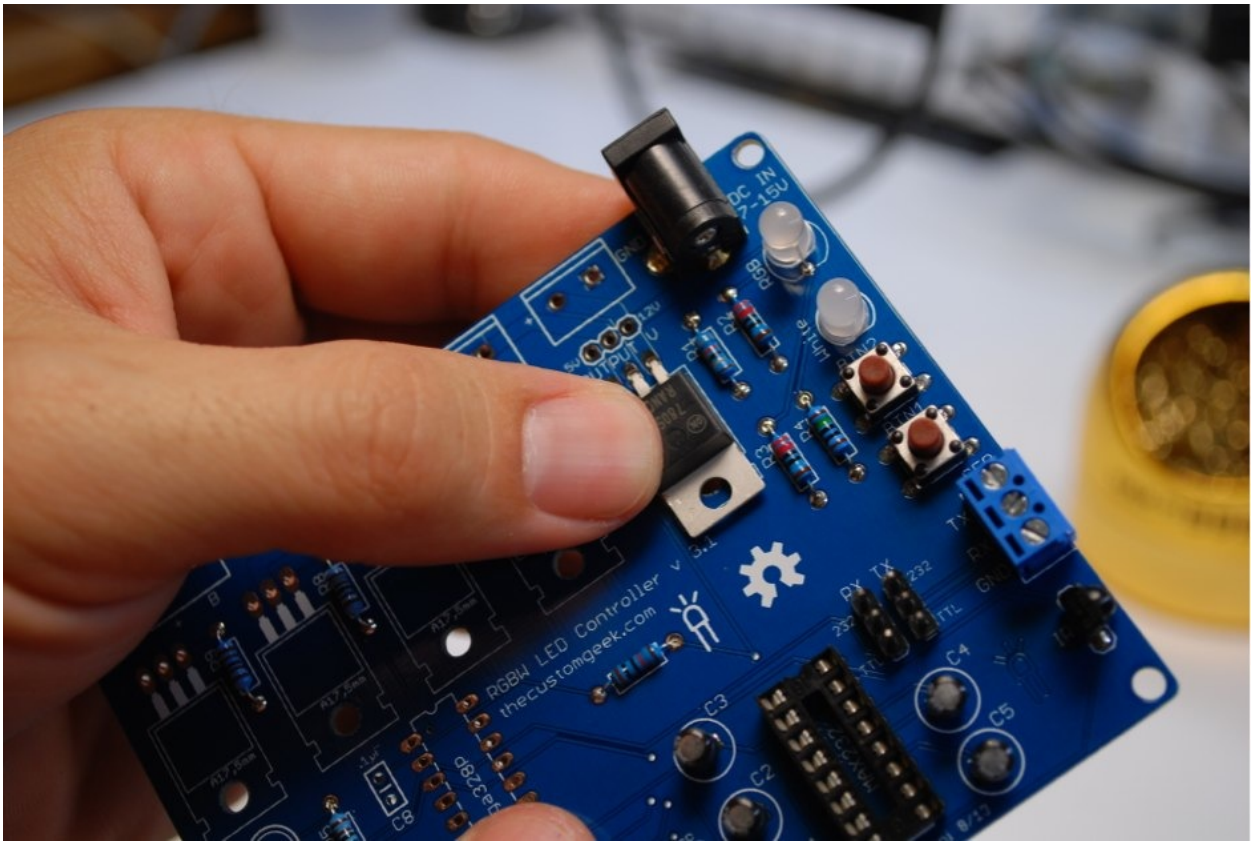
Next, solder the 16 pin DIP socket for the MAX232. Note the notch at the top. We can do the same lead bending trick as we did with the switches. This will help us keep a low profile as well as keep the socket in place while we turn the board over to solder it in place.



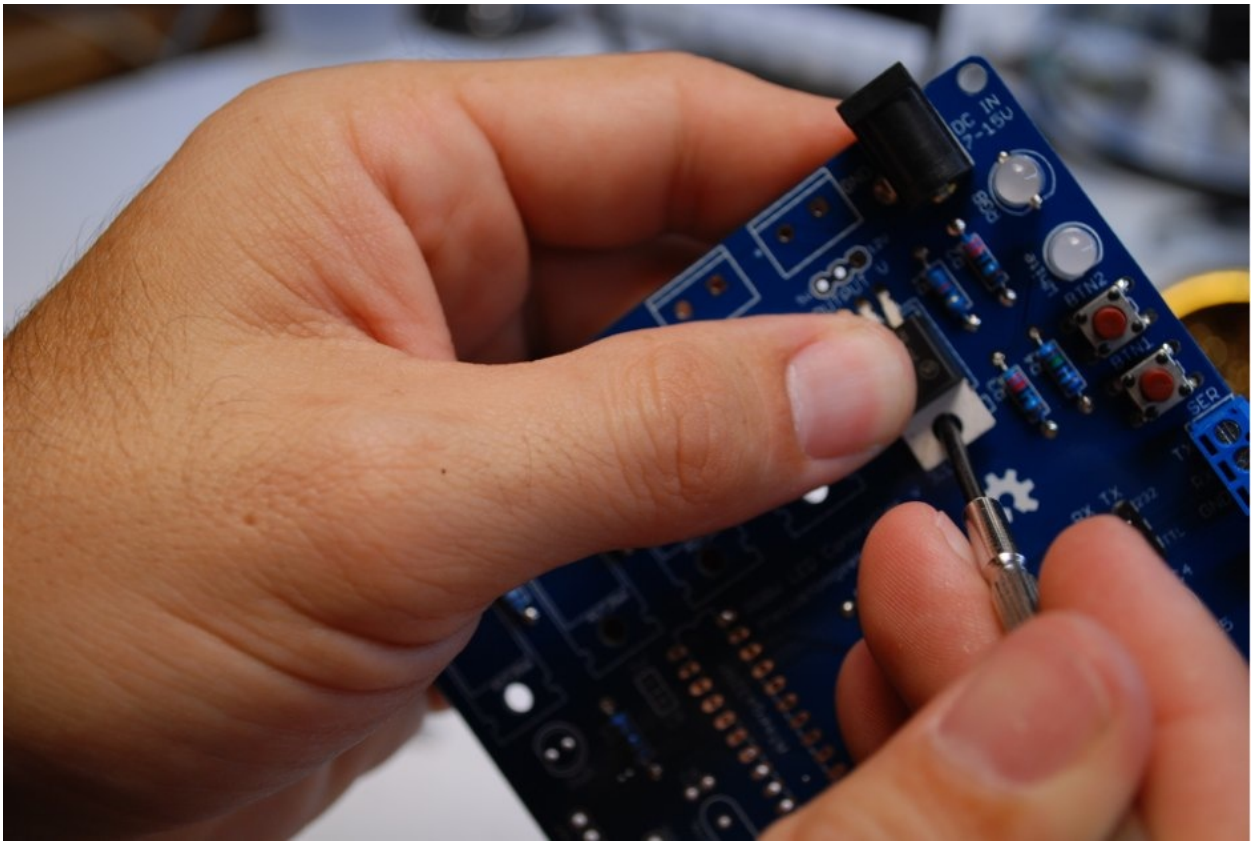
Next, solder the DC power jack into place.

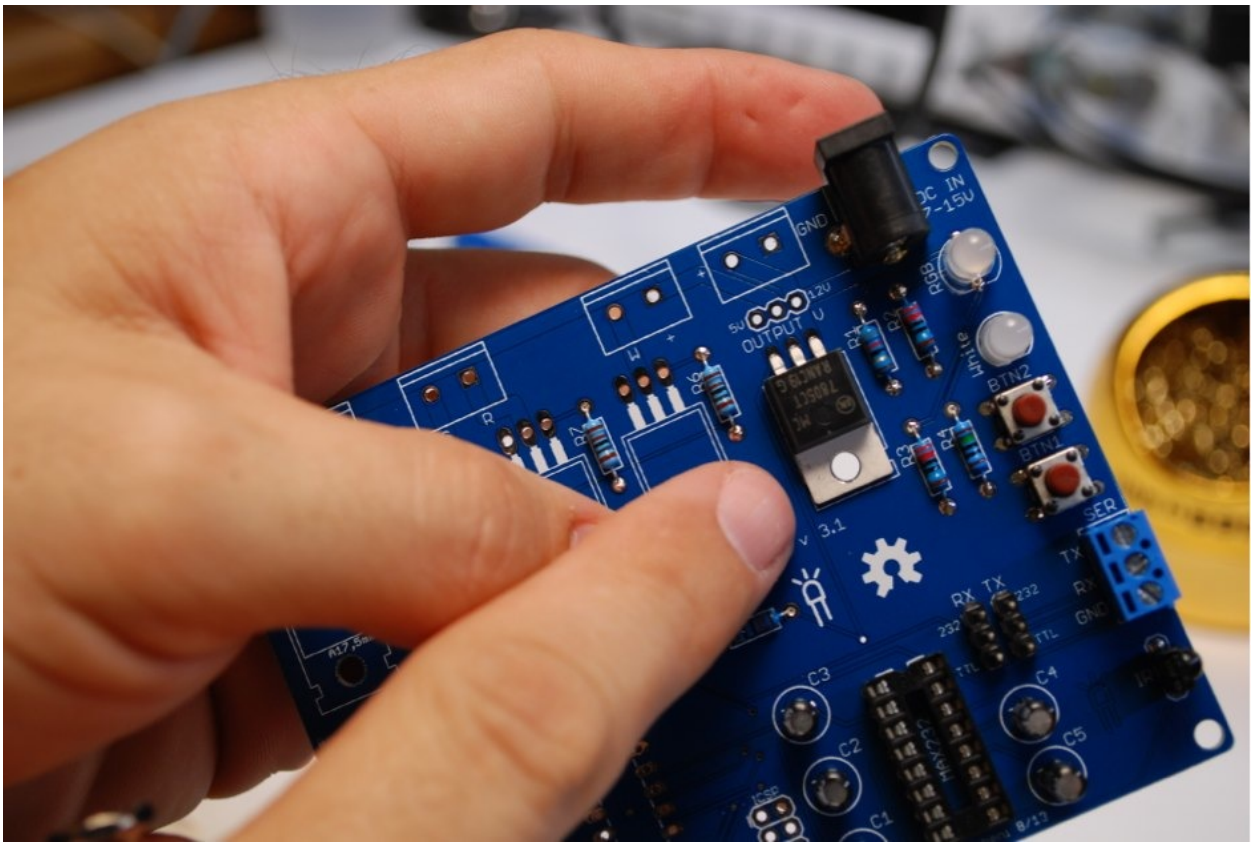


Next, install and solder the 7805 voltage regulator. make sure you use the 7805 and not a IRLB8721PBF MOSFET! They have the same package. ;)

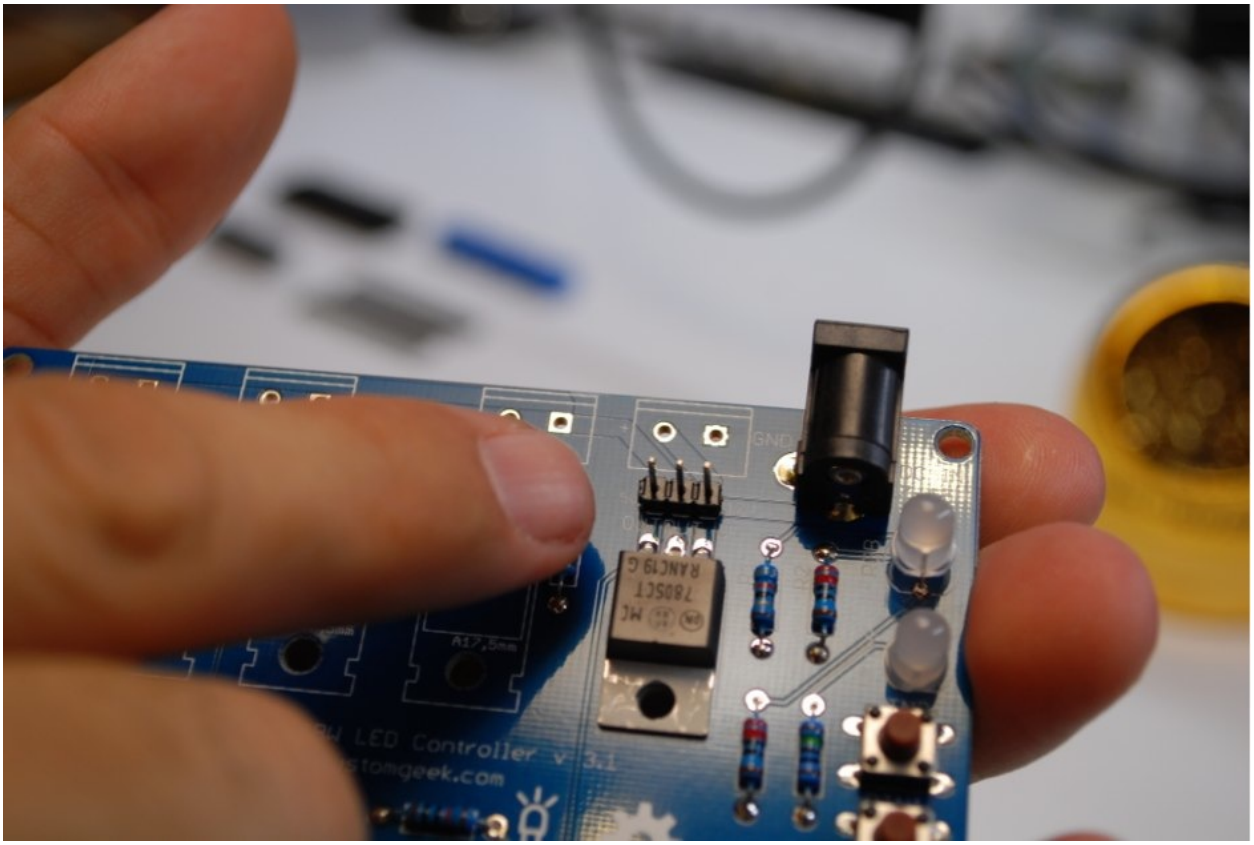


If you insert a screwdriver as shown, you can scoot the 7805 into place and help it sit flat.

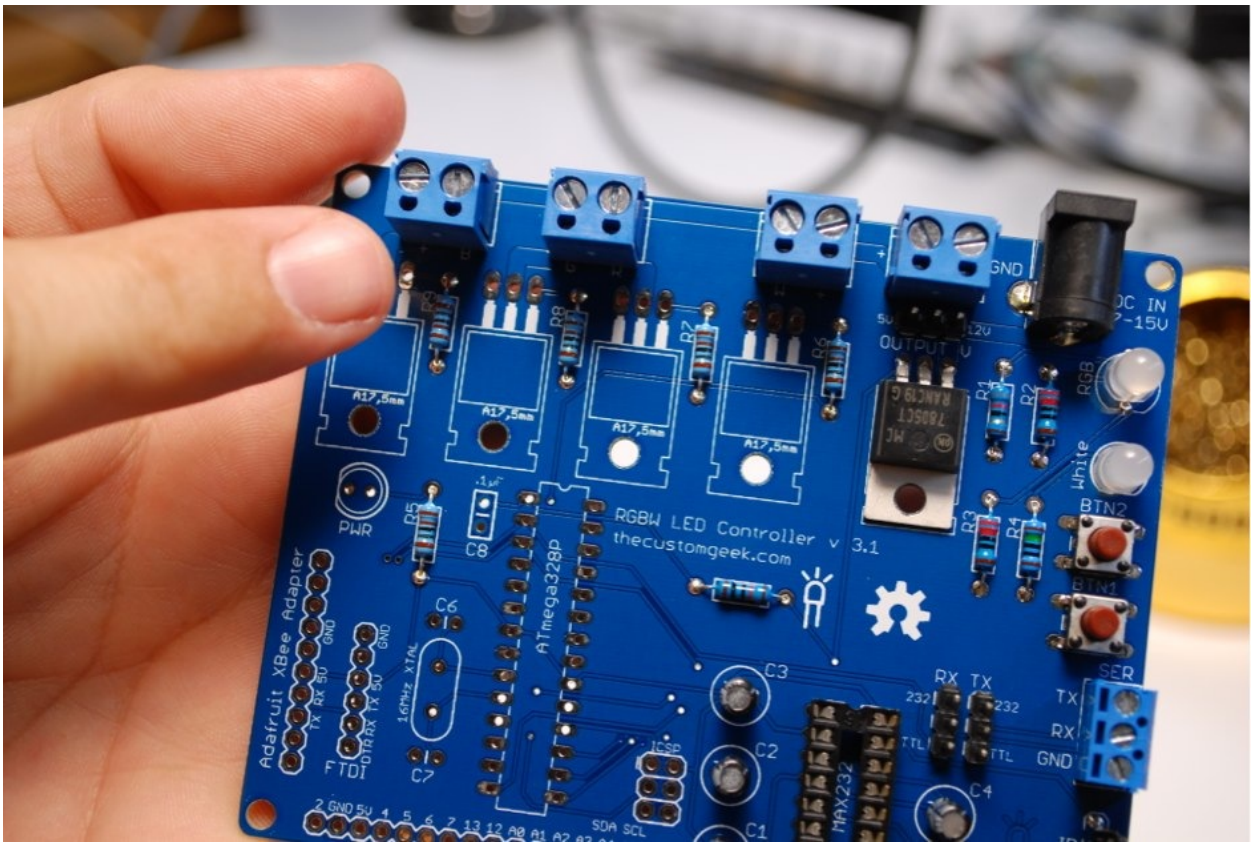




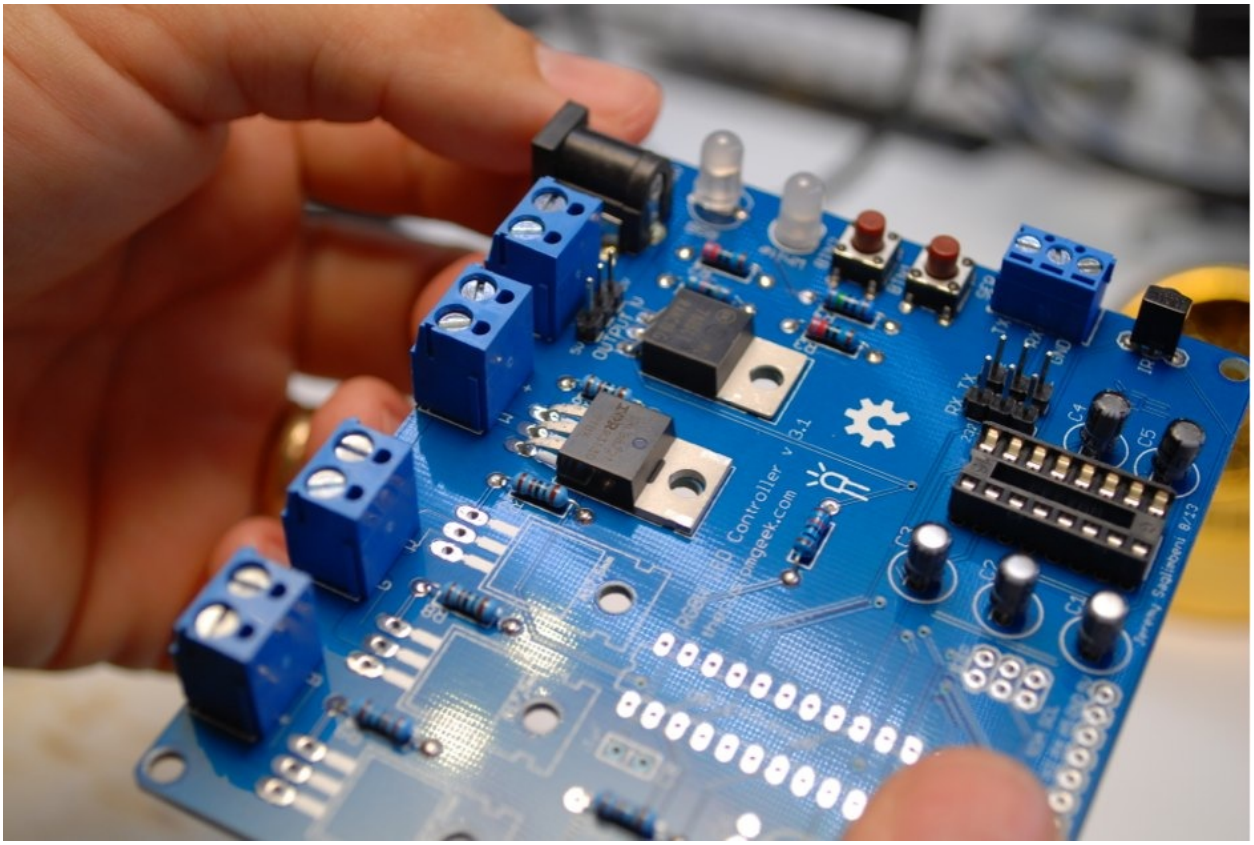
Next, solder a 3 pin male header above the 7805 for the voltage output selection.

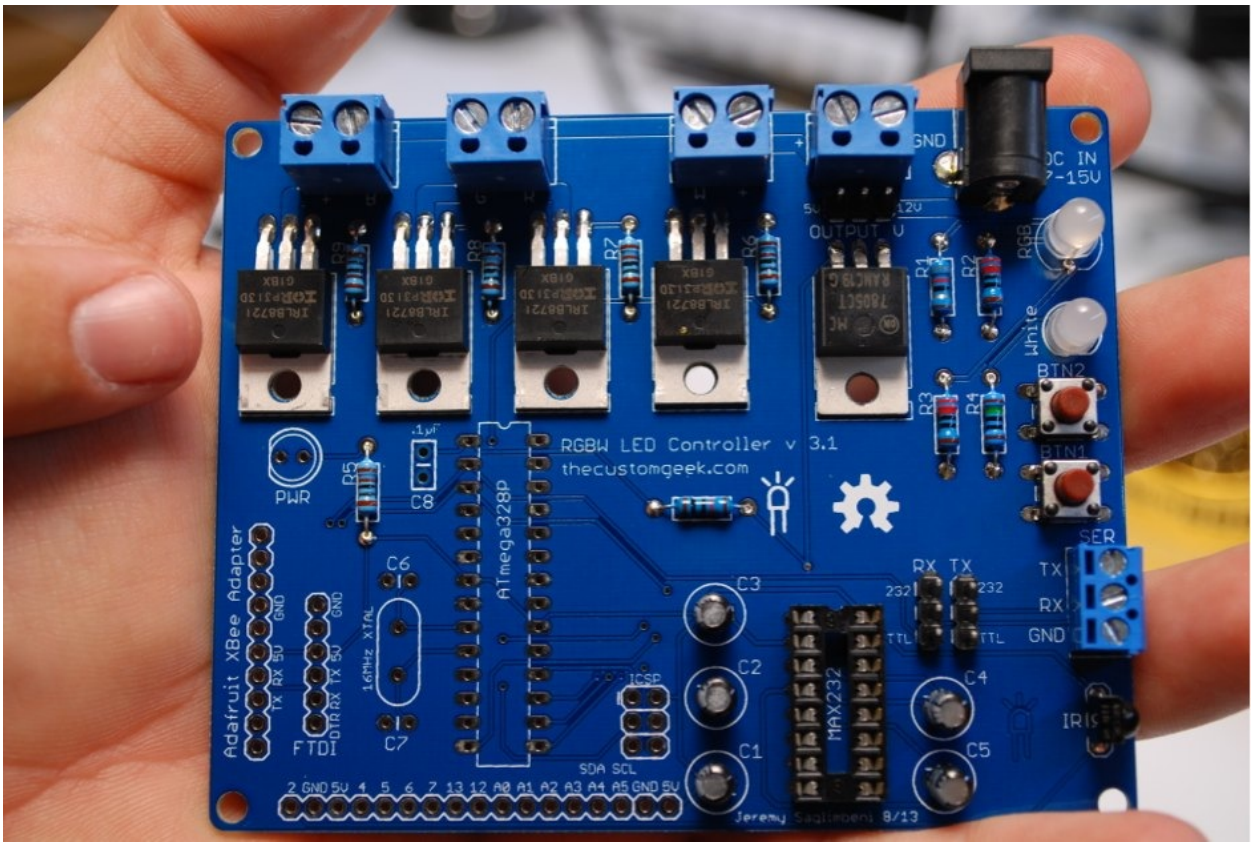


Next, solder the 5mm 2 position connectors as shown.

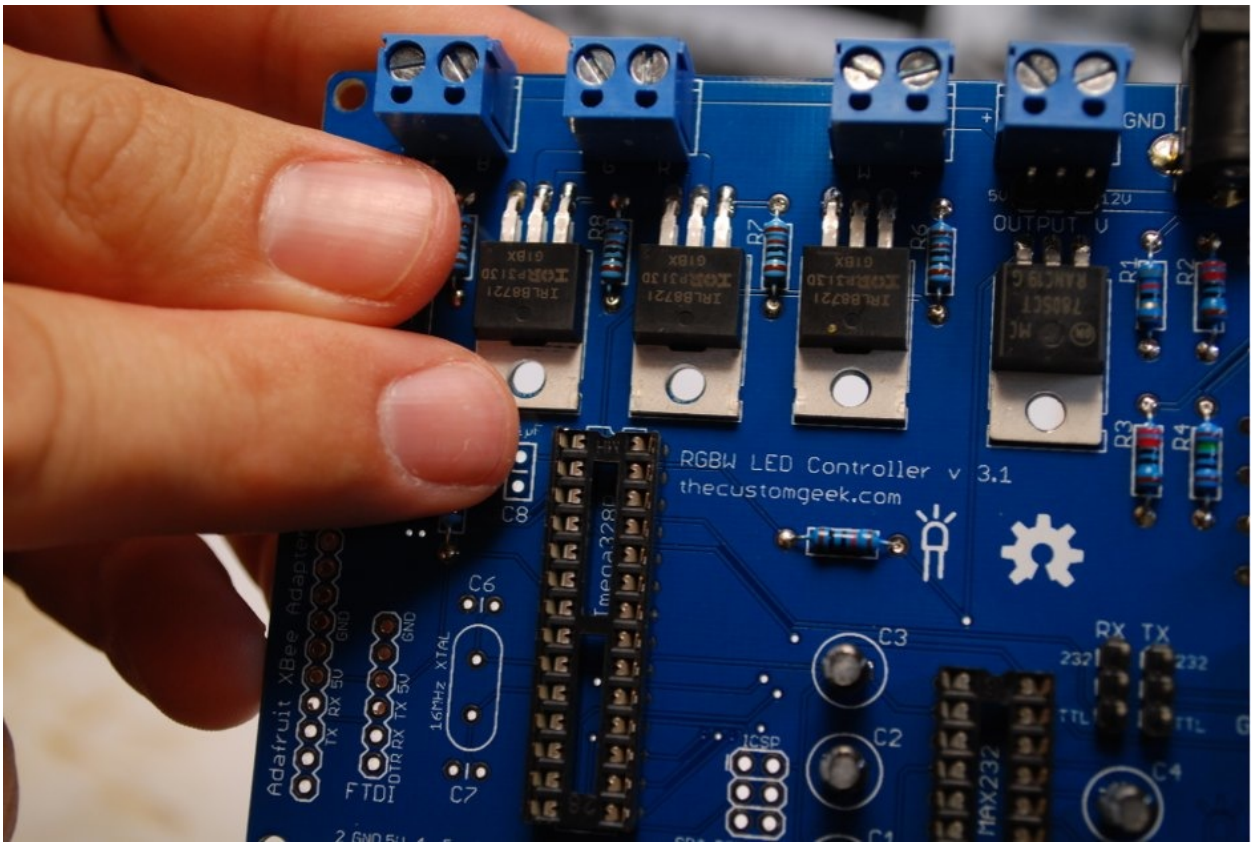


Next, solder the 4 IRLB8721PBF MOSFETs in place.

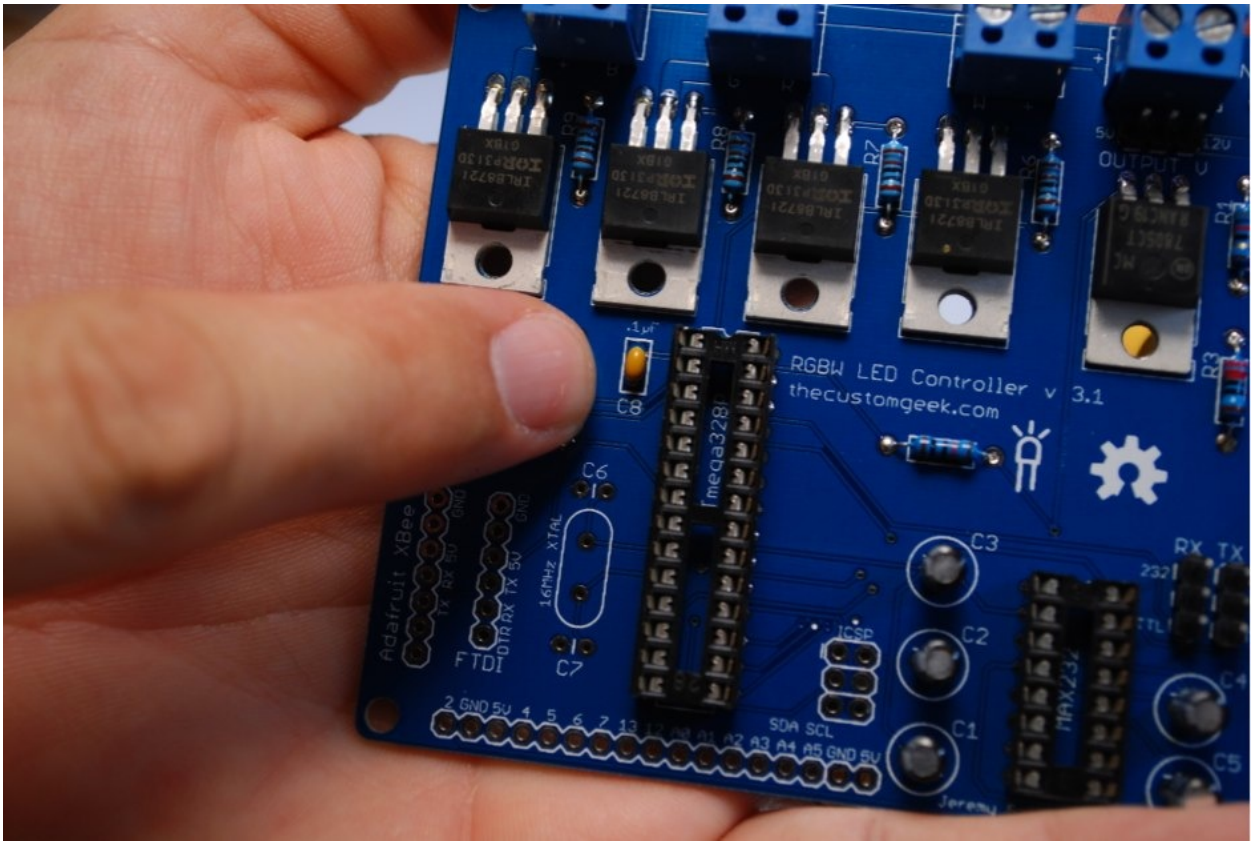




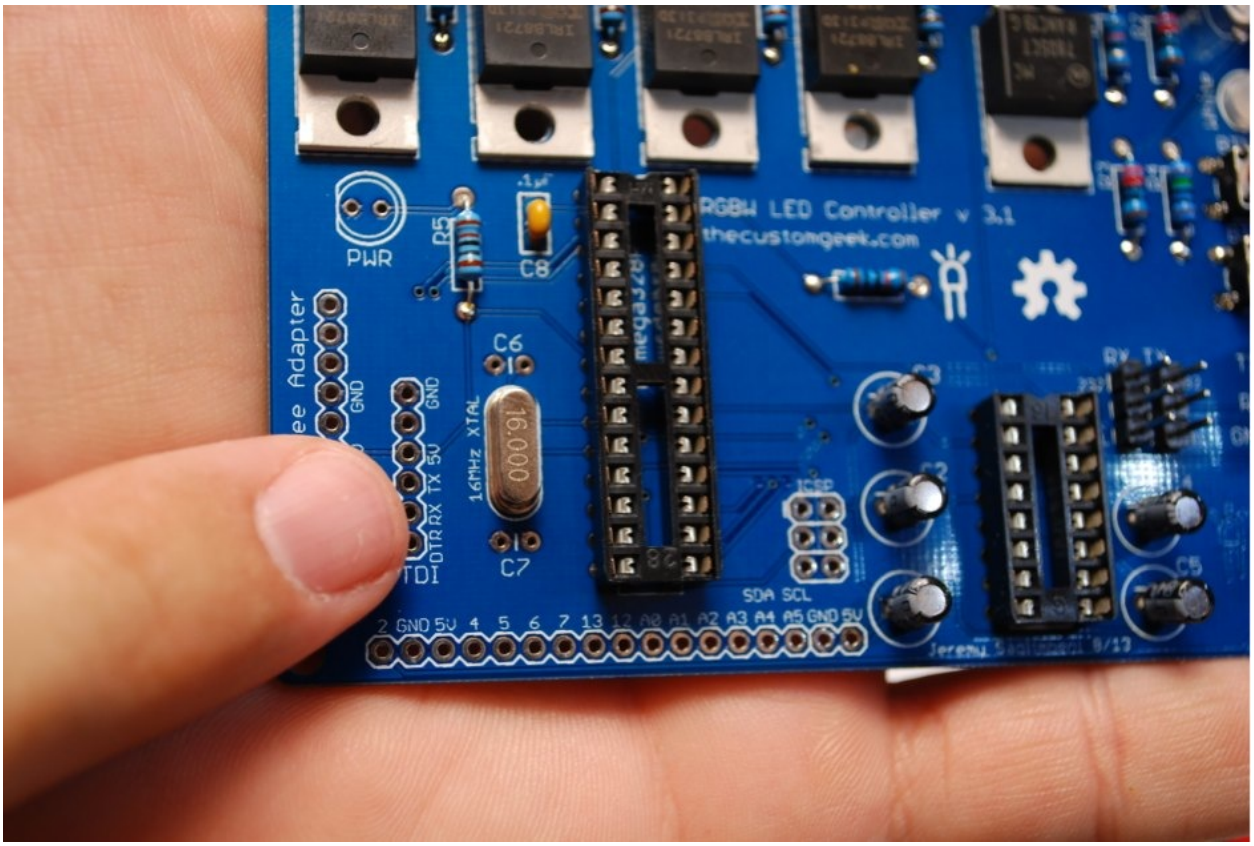
Next, solder the 28 pin DIP socket in for the ATmega328. Note the notch at the top.



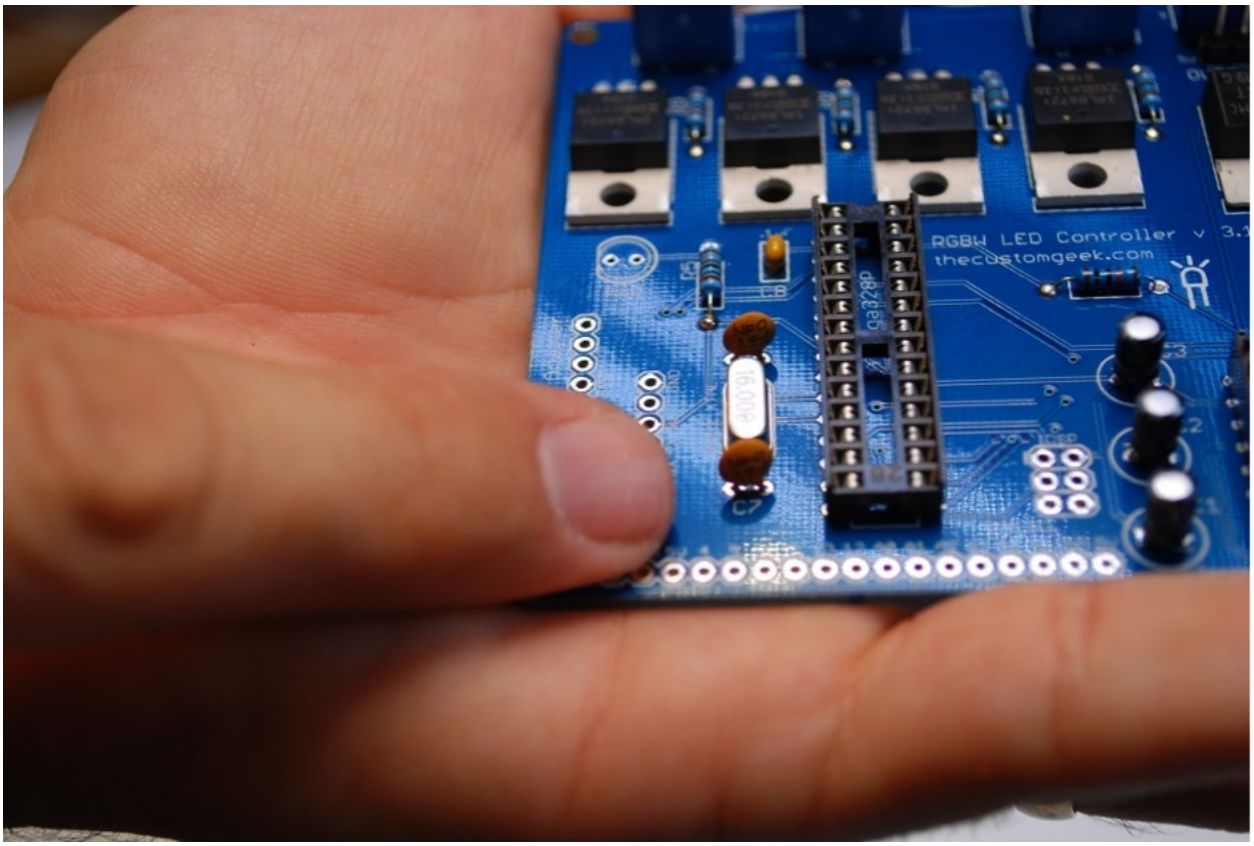
Next, solder the .1µF ceramic capacitor.



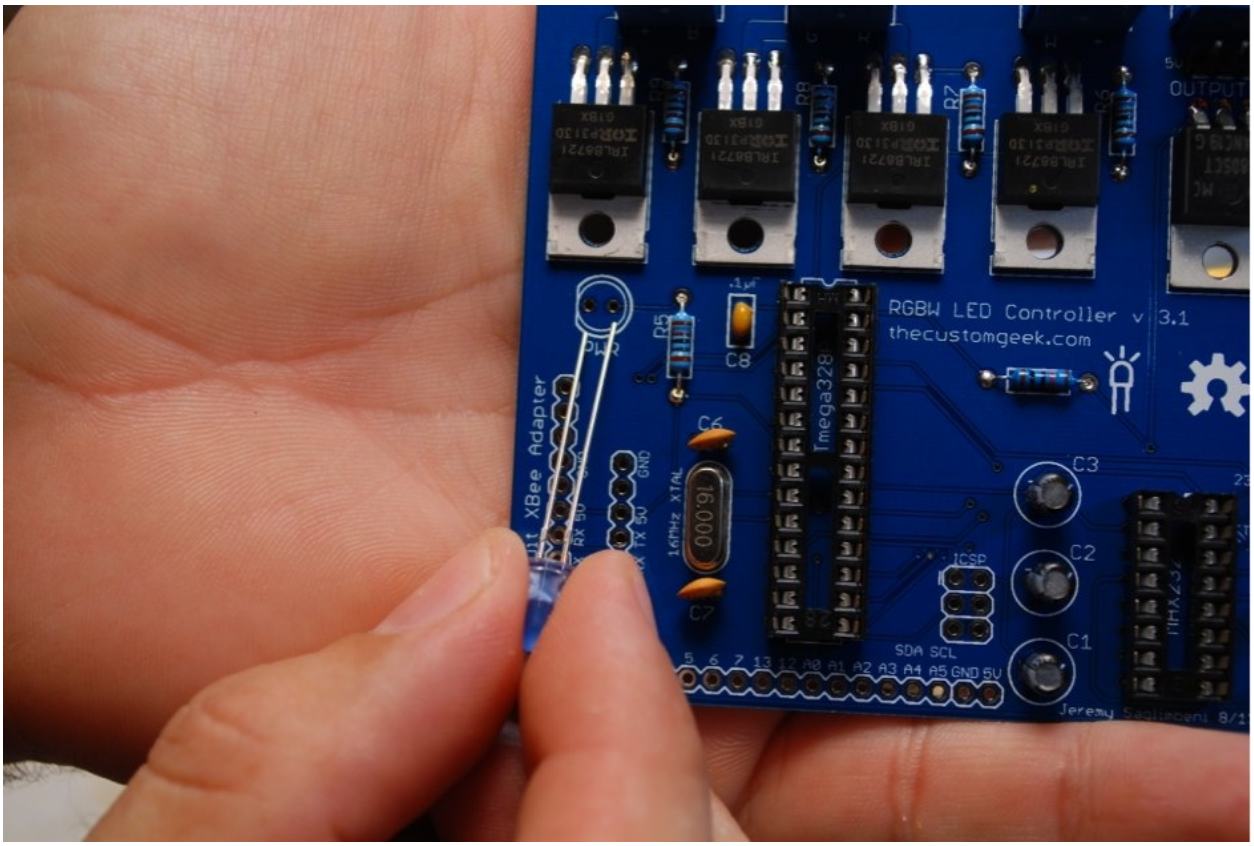
Next, solder the 16MHz crystal. It is not polarized, it can go in either direction.

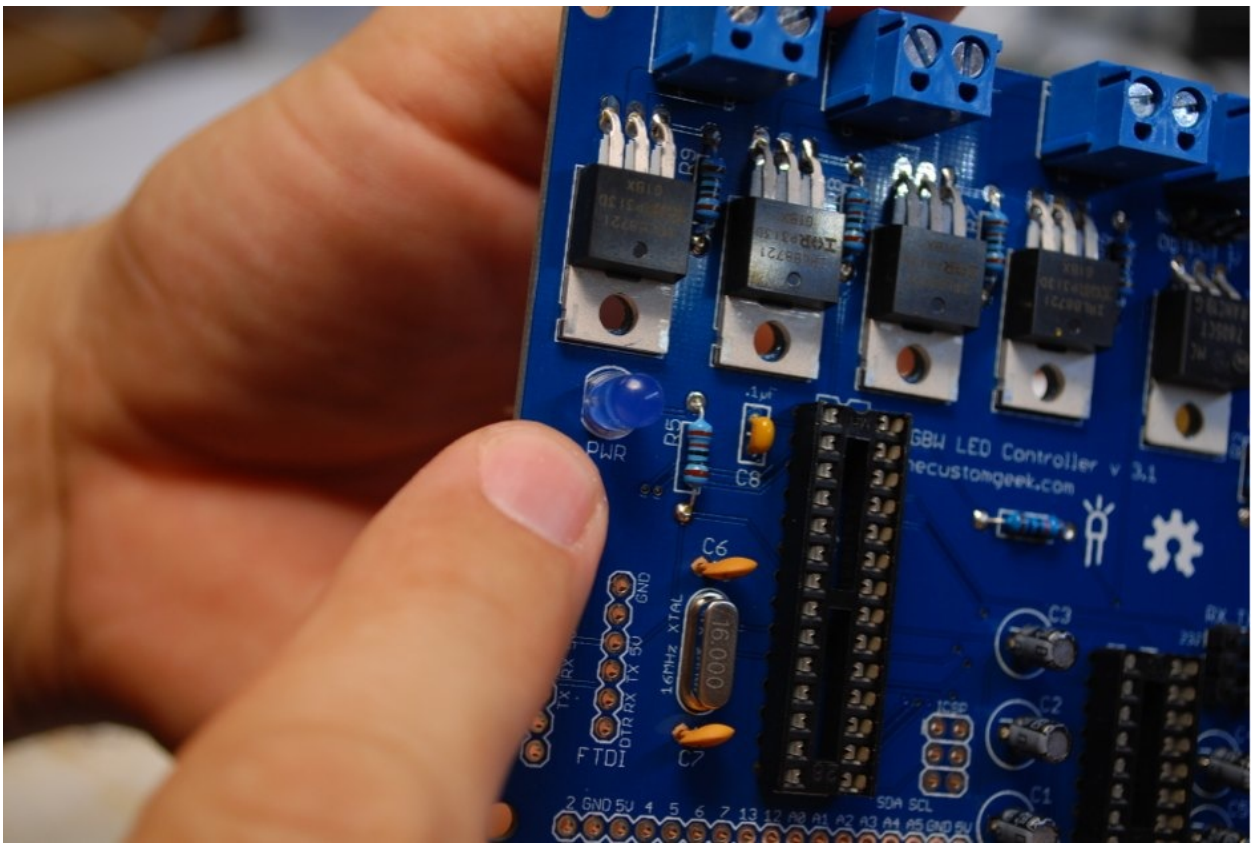


Next, solder the two 22pF ceramic capacitors on both sides of the crystal.

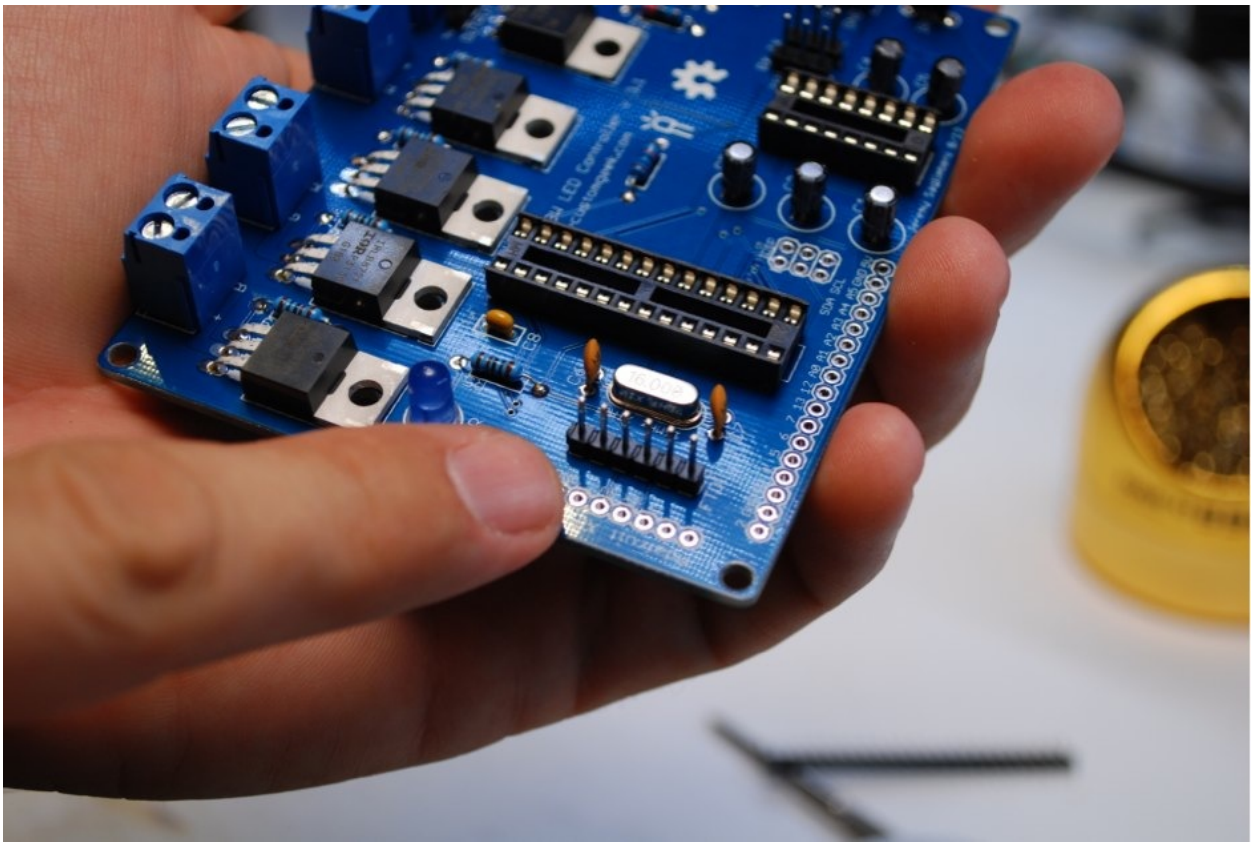


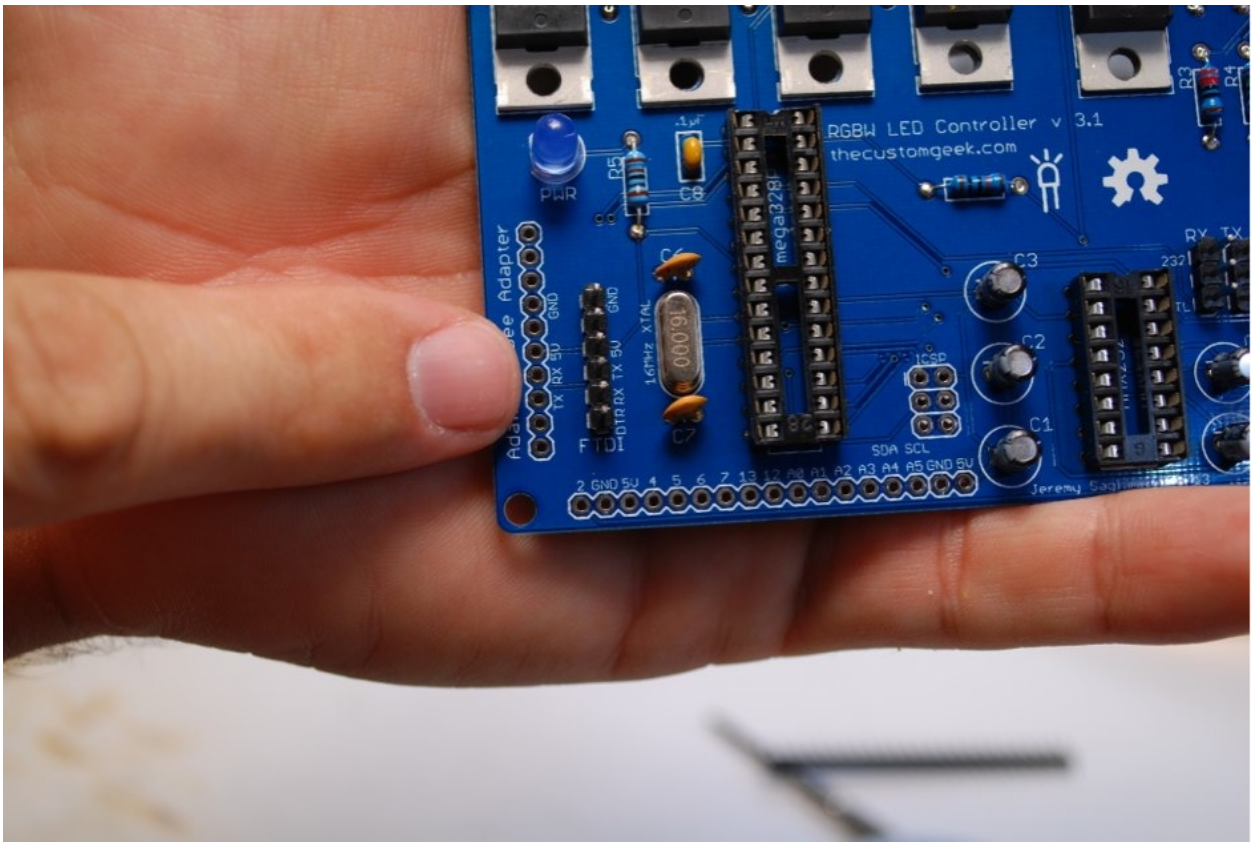
Next, solder the blue power LED as shown below.



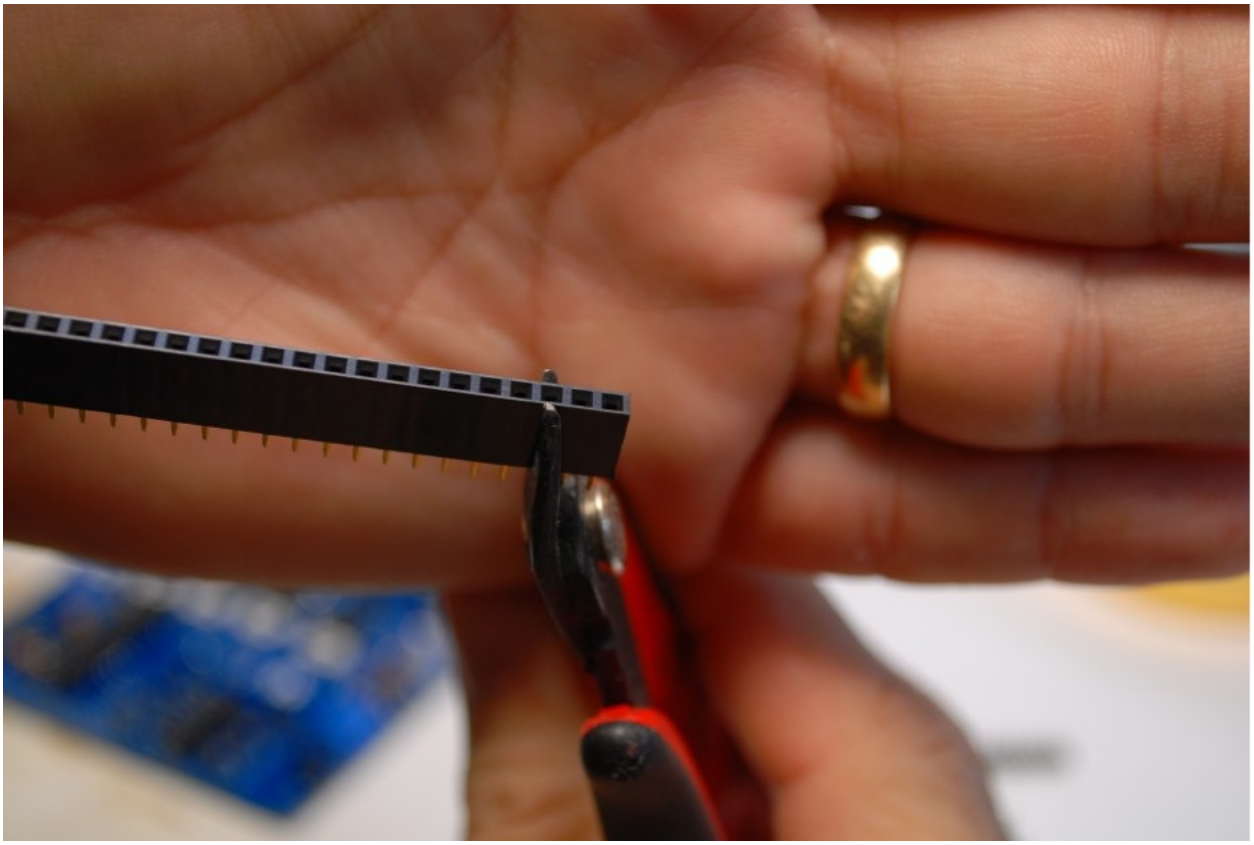


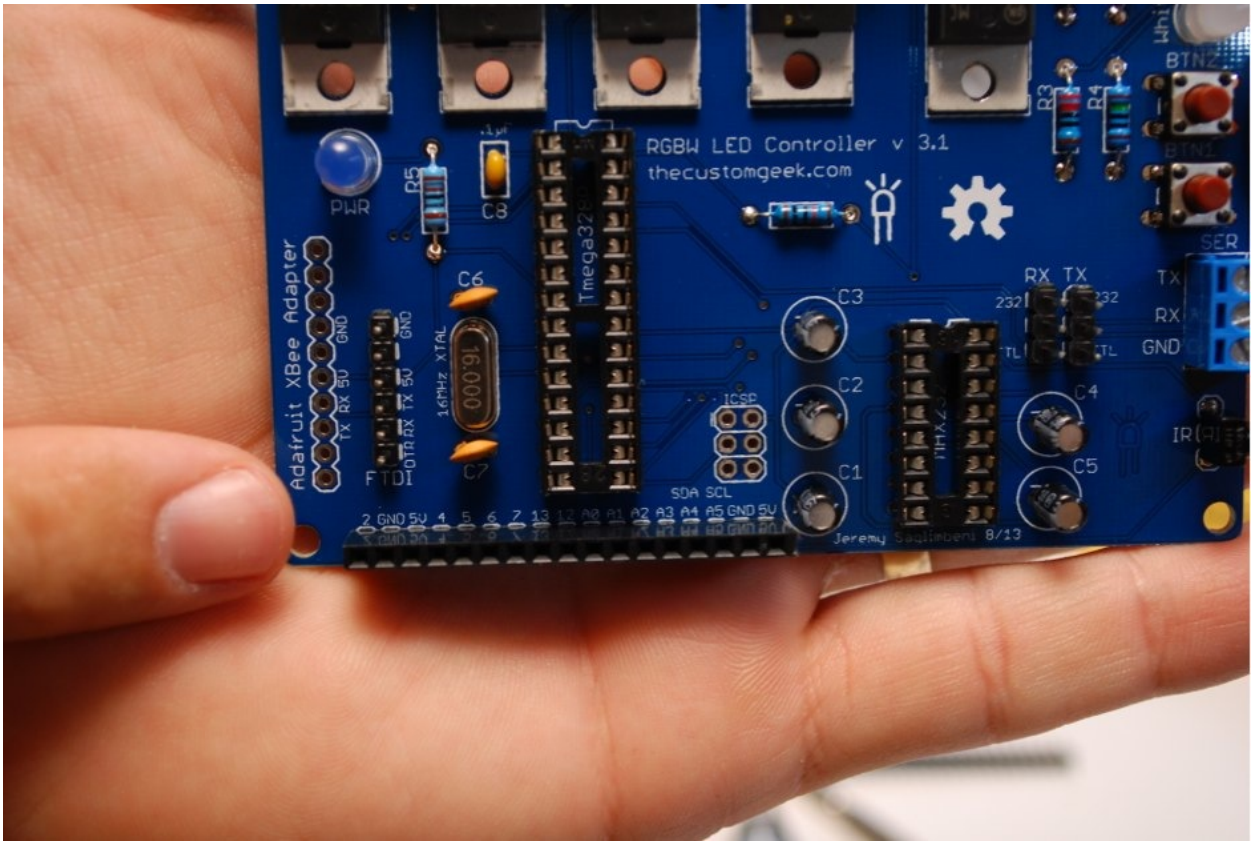
Next, solder a 6 pin male header in the FTDI spot as shown.



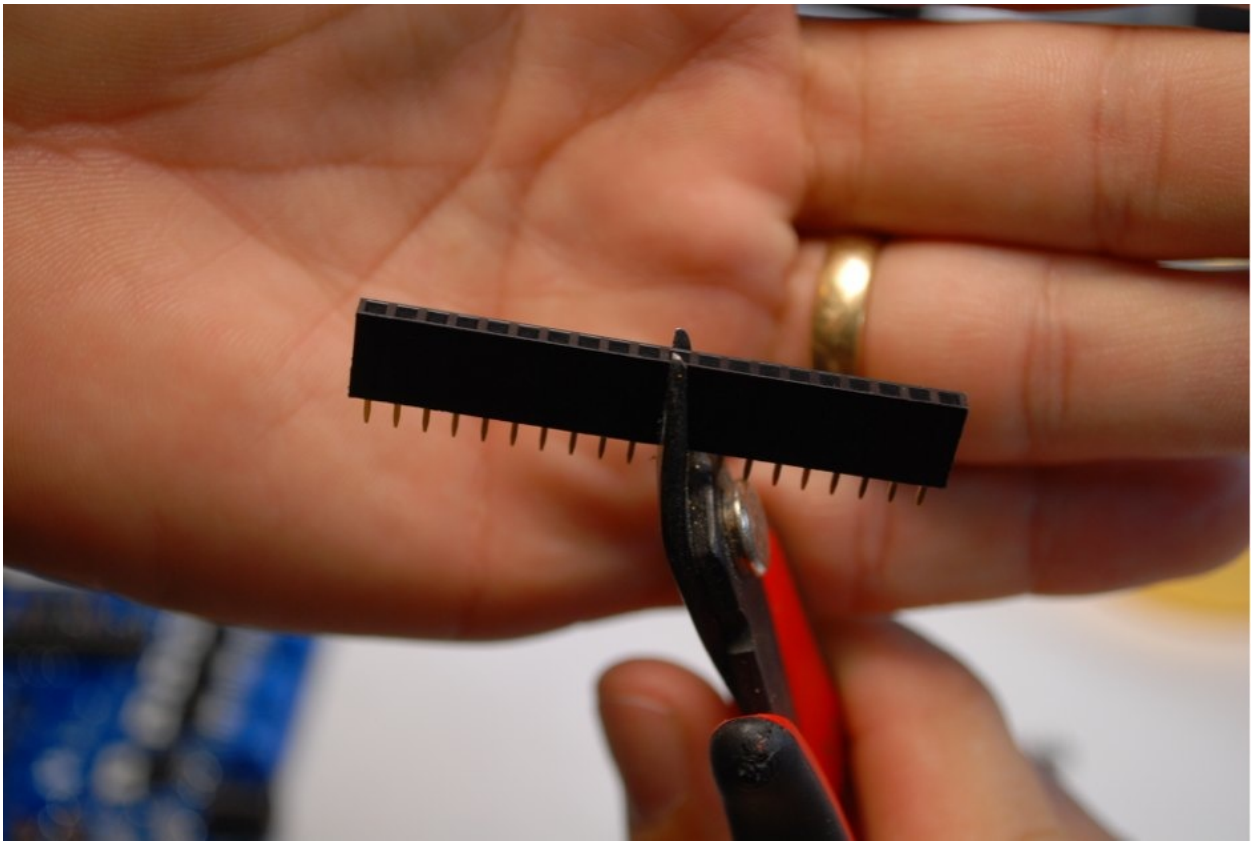


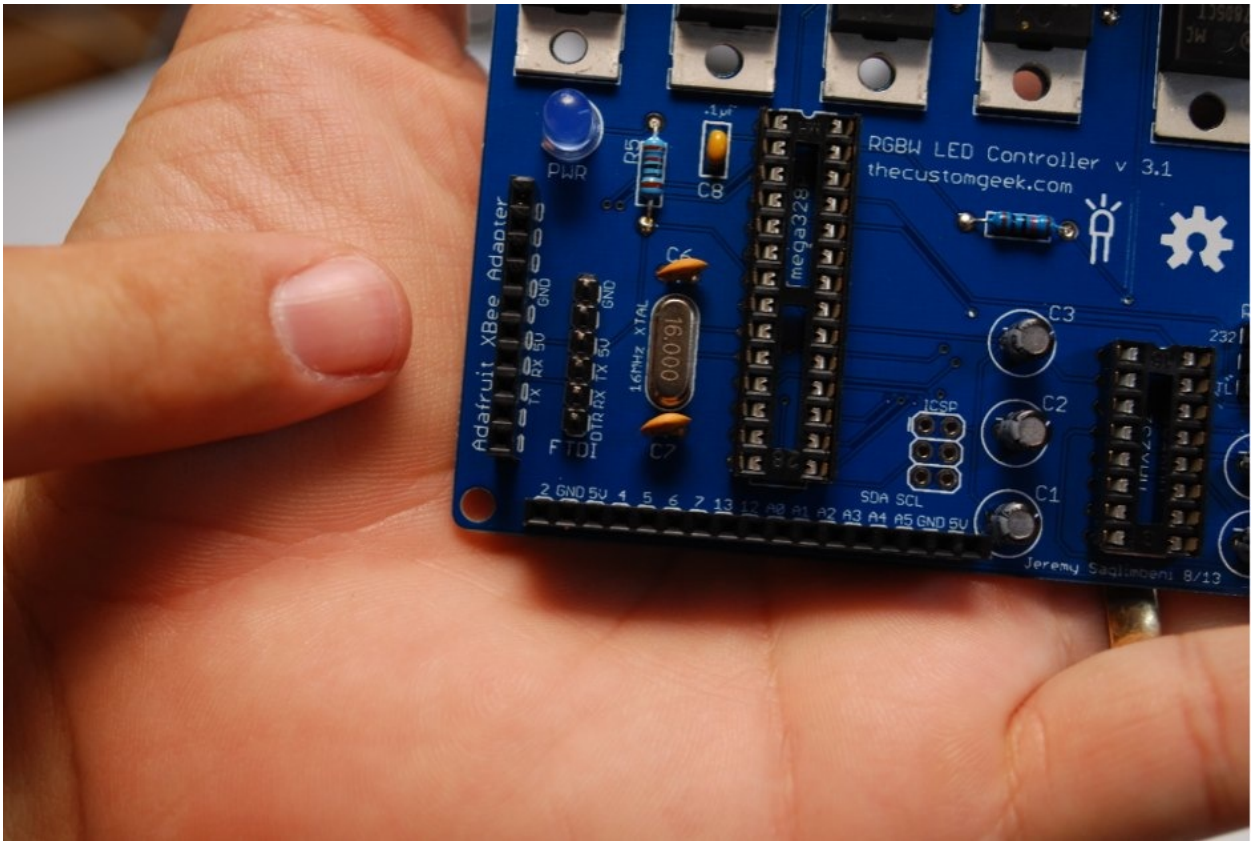
Next, trim one of the female headers as shown, this will be the pin extension header.



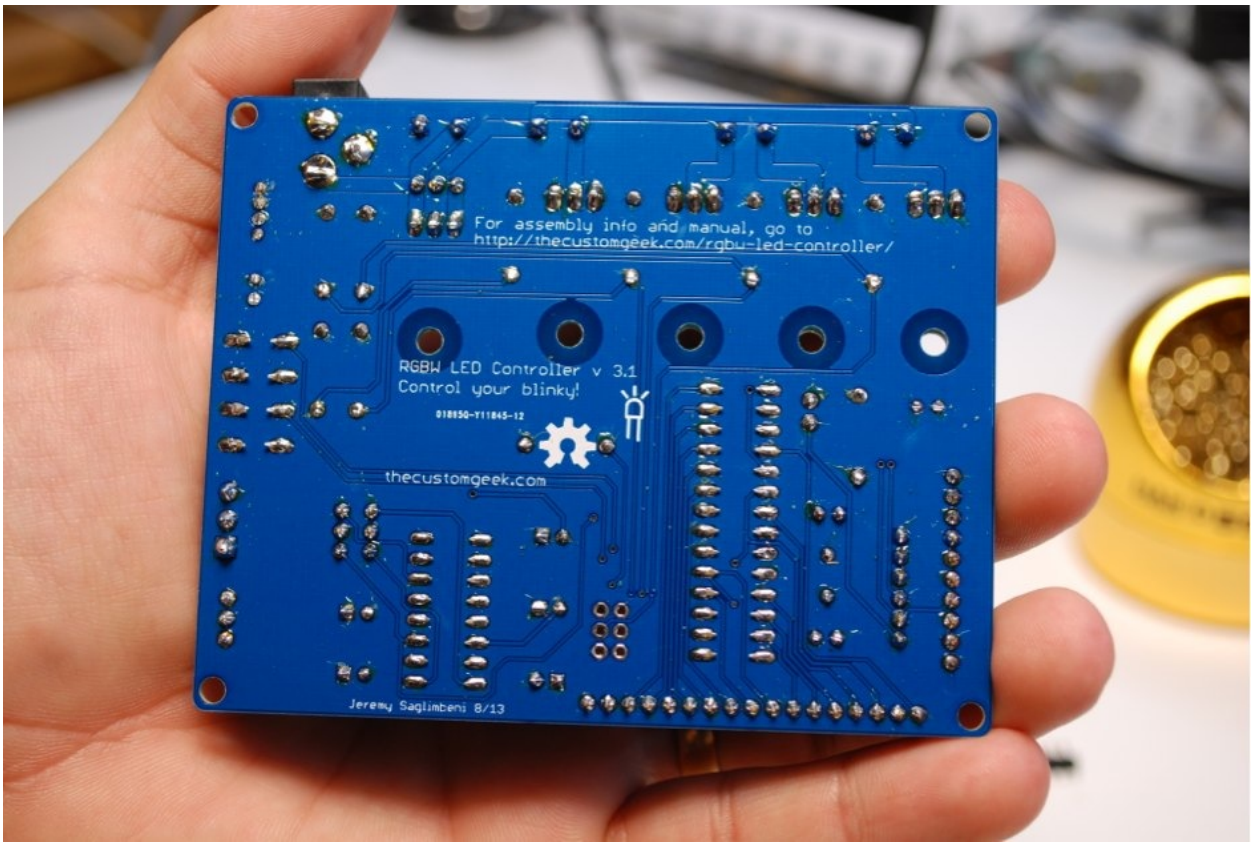


Next, trim and solder the other female header as shown for the Adafruit XBEE adaptor.

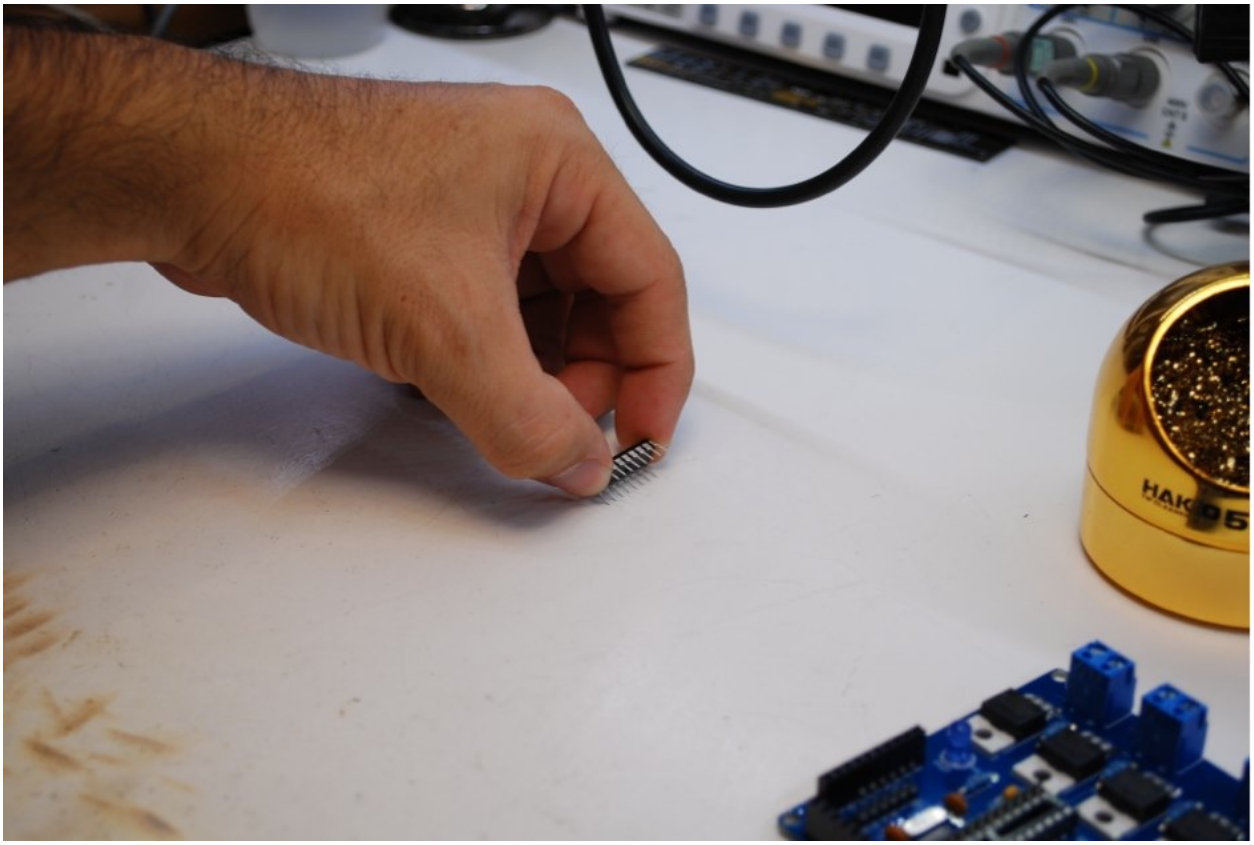




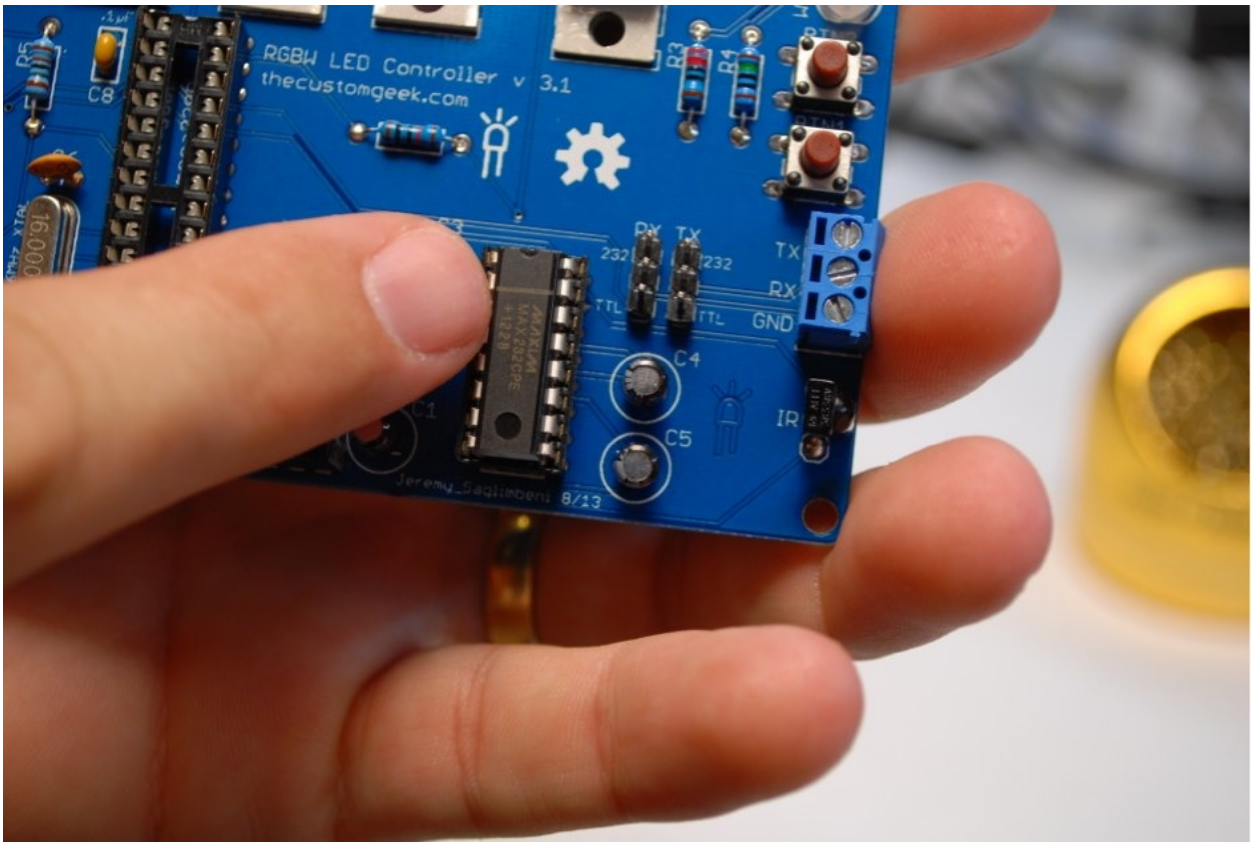
You back of your board should look like this.



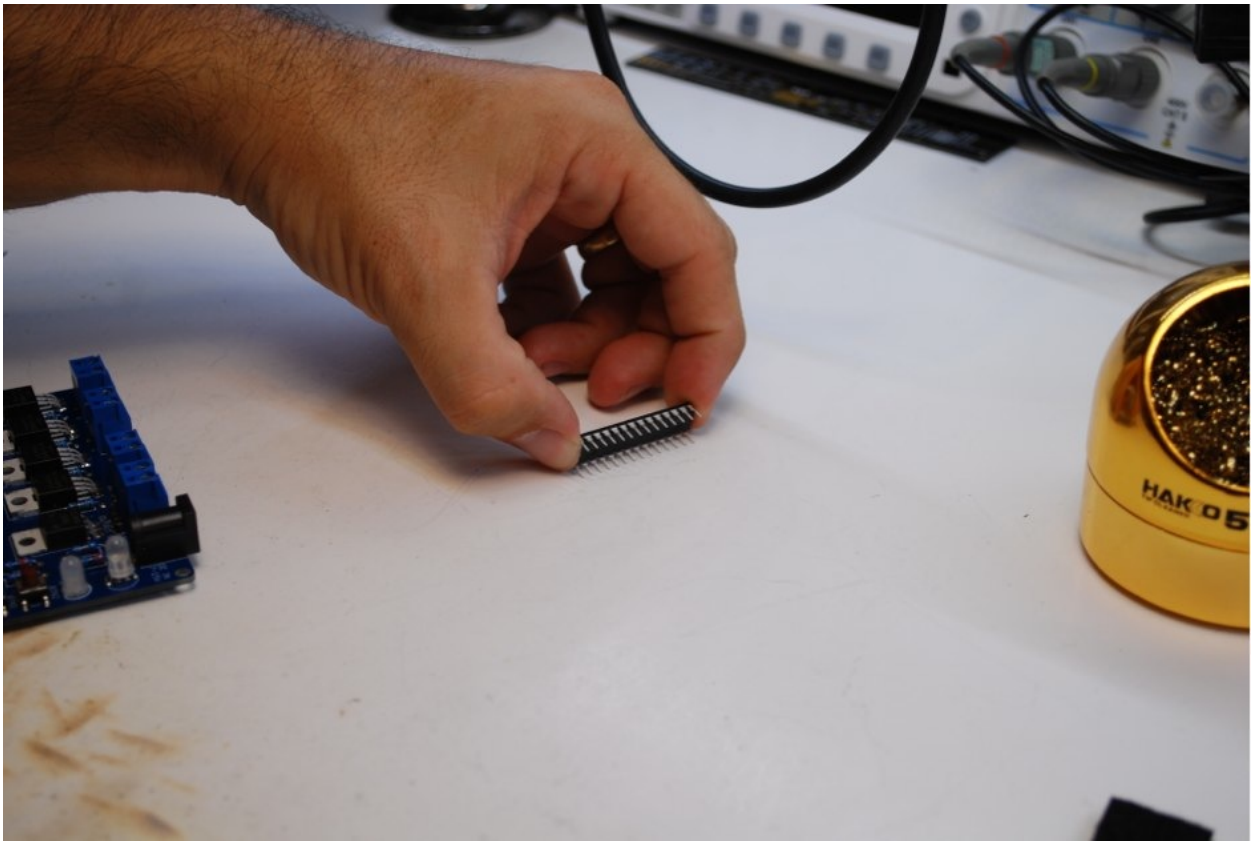
Next, we will insert the ATmega328 and MAX232. You will need to bent the row of pins in slightly for them to fit into the DIP sockets. Make slight adjustments and check the pin spacing often.

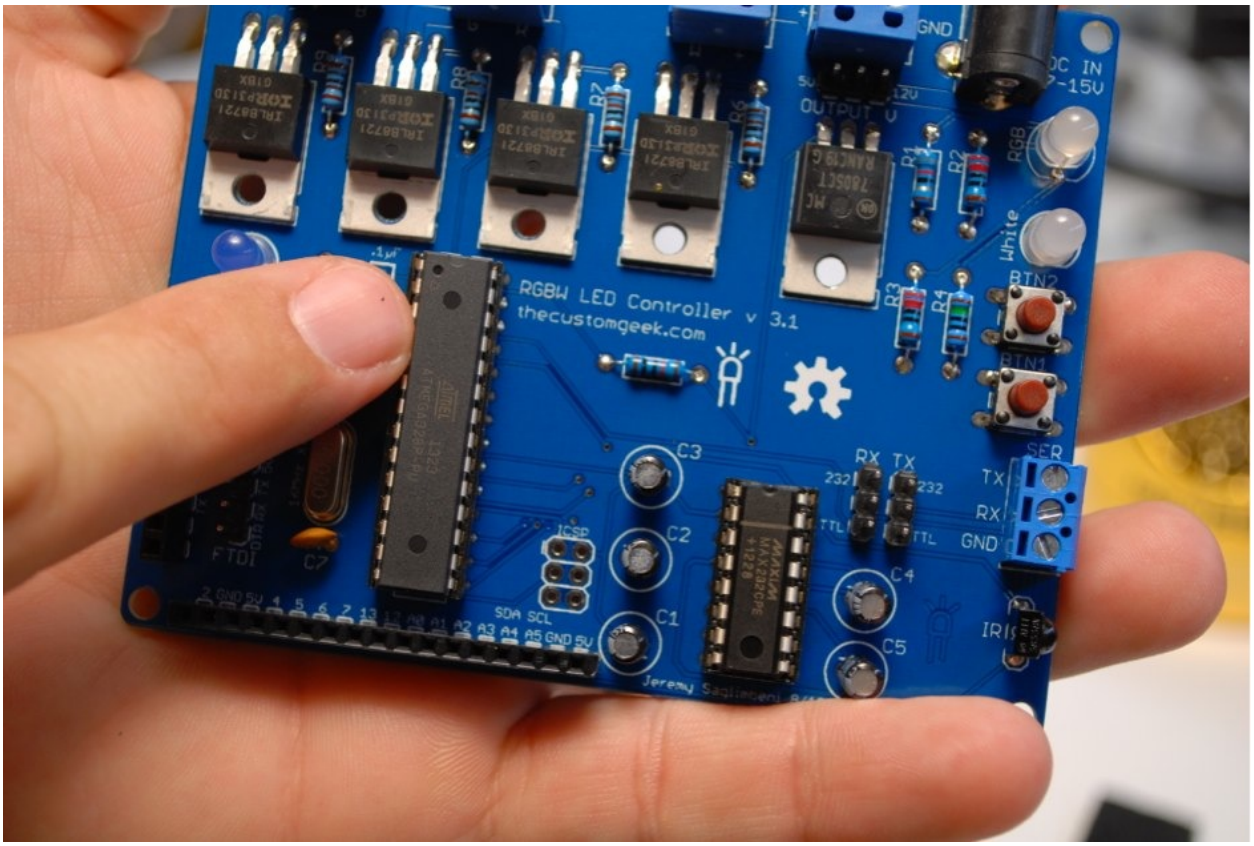


Make sure the notch on the MAX232 IC is toward the top as shown below.

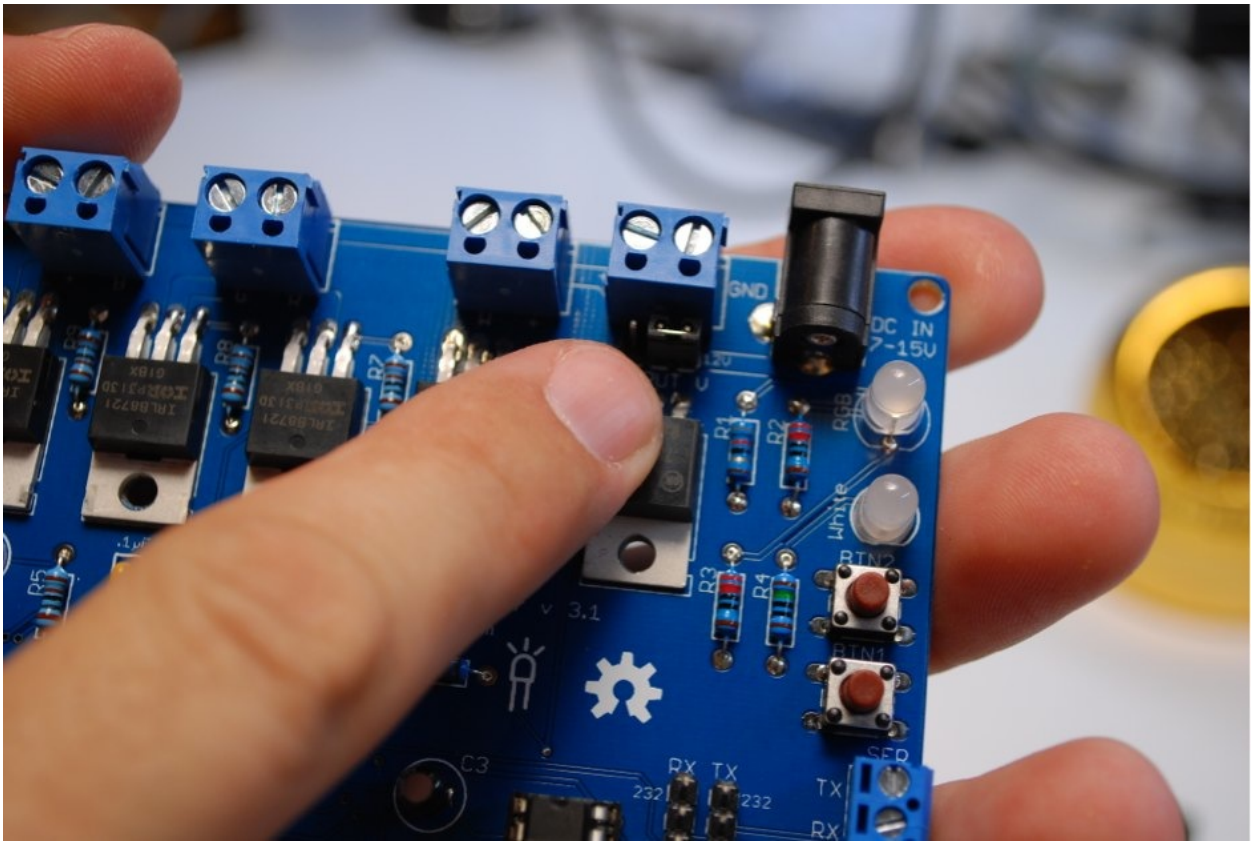


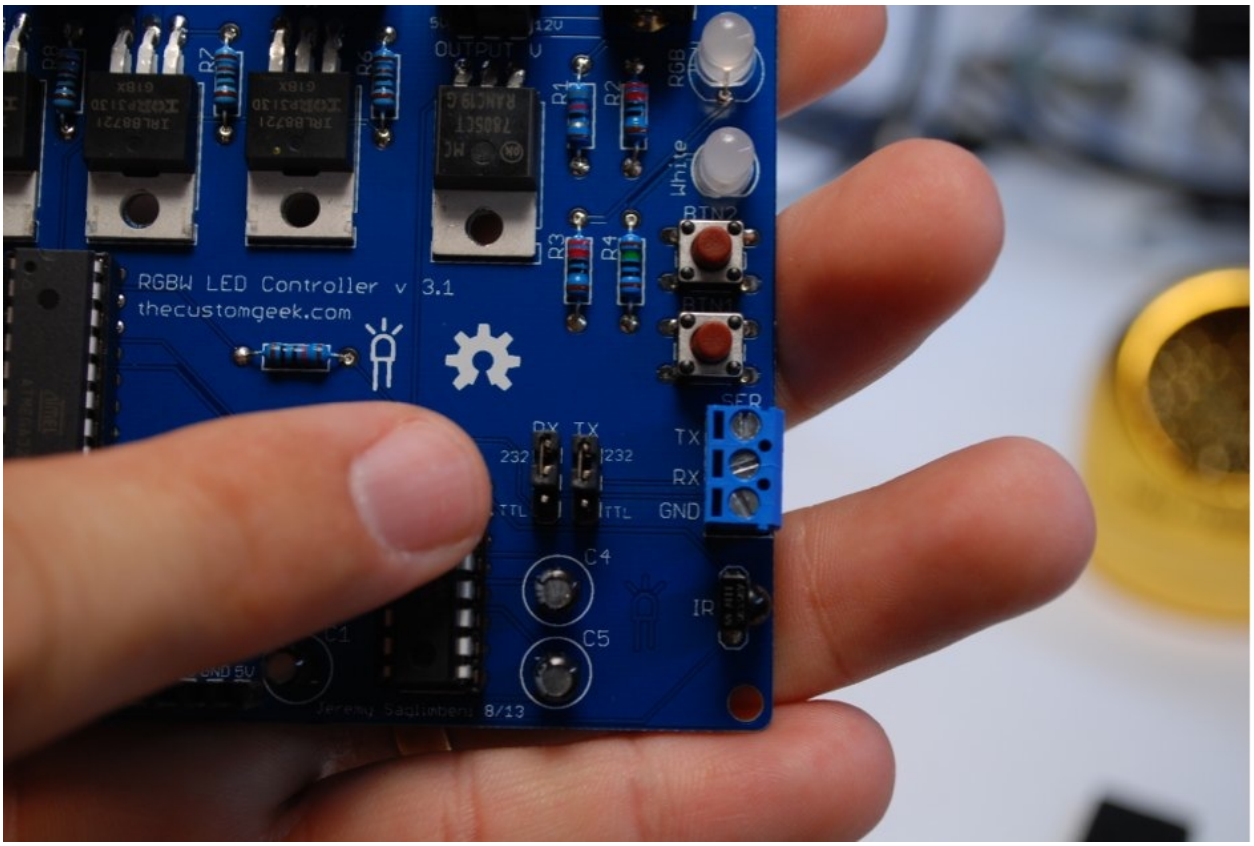
Repeat for the ATmega328, again, notice the notch at the top.



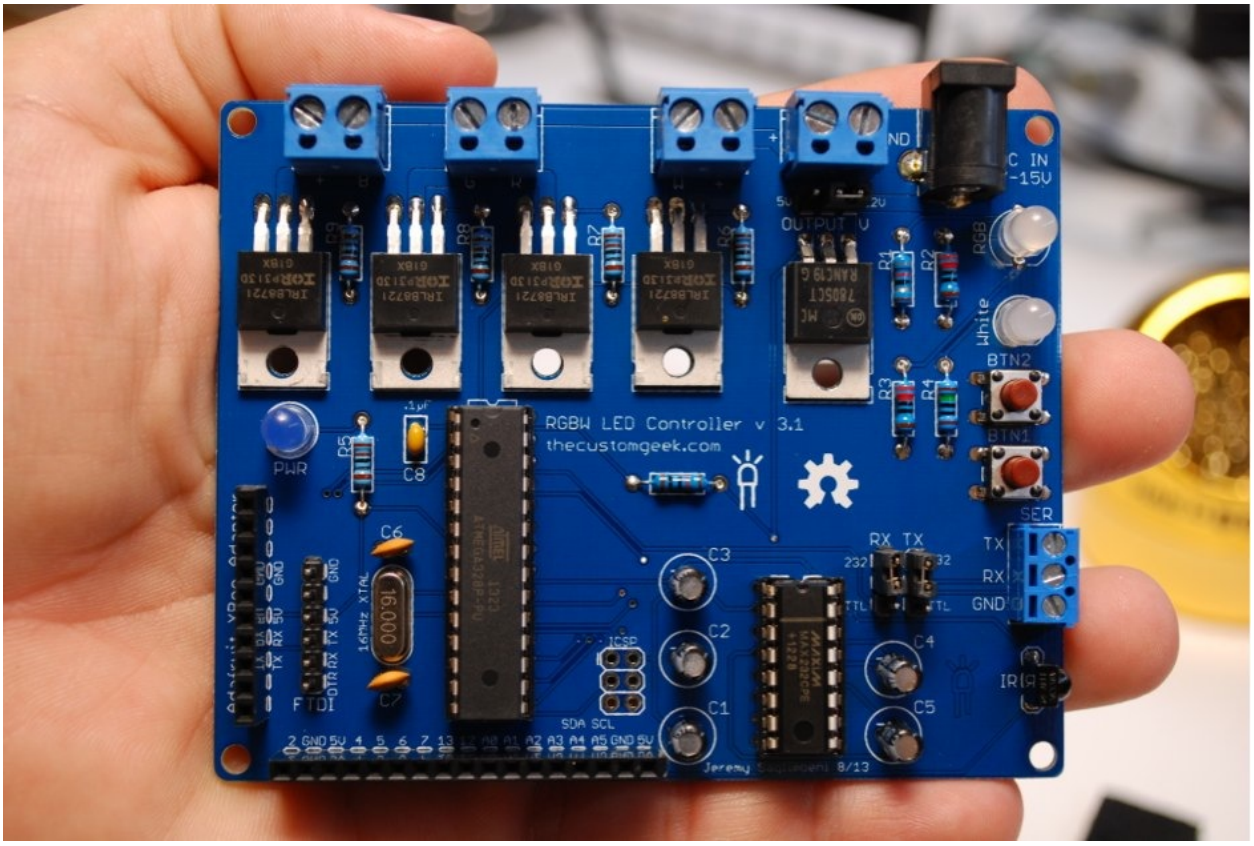


Next, install the jumpers on the voltage and serial select headers.





Finished! - That's it! You're done, you can now take control of your blinky. :)



Function and Operation

Getting Familiar With the Components

Lets learn about this kit! First, lets go through the hardware features and the functions they perform.

First we have the PCB, this is the foundation and has all of the connections so we can avoid using messy wires. Next we have 5 1KΩ resistors, 2 220Ω resistors, 1 150Ω resistor, and 1 82Ω resistor, these will limit current to the LED's and MOSFETSs, and pull the reset line high. Then we have 2 22pF ceramic capacitors, they help the crystal to function properly. Next we have a .1µF ceramic capacitor, it will allow an FTDI interface to auto reset the ATmega328 when uploading new firmware. Next is a 16MHz crystal, this sets the clock for the ATmega328. We then have 2 momentary push button switches, these will allow for manual input. Next we have an ATmega328 with an Arduino bootloader, this behaves like an Arduino, so it is programmed with the Arduino IDE. There is also a MAX232SPE on board. This is a bi-directional RS-232 to TTL converter. The DIP sockets allow you to solder connections for the IC's without them being exposed to a lot of heat. DIP sockets also allow for easy replacement of IC's later if ever need be. Next is 4 IRLB8721PBF MOSFETs, these switch the current the LED output load. The 7805 voltage regulator regulates the voltage to the board to 5VDC. The IR sensor allows for IR control of the board. Female headers allow for easy connection of an optional Adafruit XBee adaptor for easy wireless control and extend the pins of the ATmega328 for further hacking. Male header is used for the voltage select jumper, the serial type select header, and the FTDI header. Three jumper shunts will go on the serial type select and voltage select headers. The white LED will serve as an output indicator for the white channel. The blue LED is a power LED letting you know that there is power on the board. The RGB LED serves an an indicator for the RGB channels. The (qty 5) .1µF electrolytic capacitors act as charge pumps for RS232 output from the MAX232SPE. A two wire terminal connector serves as an alternate power input method. A three wire terminal connector allows for easy connection of serial I/O. Two wire terminal connectors will allow for easily connecting LED's to the outputs. The DC power jack is one option of powering the board.

Once you have assembled your RGBW Controller, you will need to know the basic operations. When you power on the controller, it will return to the last state it was in when powered off. A new controller defaults to all channels at 100%.

Power Requirements - The RGBW LED Controller can be powered 2 different ways. It comes with a power jack that can accept 7-15 volts DC. This also allows the unit to output 12 volts to power 12V LED strips. It also can be adjusted to output 5V to power lower voltage LED setups. Alternatively, the unit can be powered via the FTDI headers. Keep in mind, when providing power this way, the output is limited to 5 volts and the mA rating of the FTDI power source. If that is a typical USB connection, that power is limited to 500mA. This is not much, but it is handy for programming and small loads.

Serial Commands - The firmware, as shipped, provides support for some basic commands for serial control. A list of these commands are as follows:

red(x) – brings level of red to x percent. (x can be 0-100)
green(x) – brings level of green to x percent. (x can be 0-100)
blue(x) – brings level of blue to x percent. (x can be 0-100)
white(x) – brings level of white to x percent. (x can be 0-100)
magenta() – brings the RGB channels to magenta
cyan() – brings the RGB channels to cyan
gold() – brings the RGB channels to gold
rgbwhite() – brings the RGB channels to RGB white (all 100%)
orange() – brings the RGB channels to orange
ltblue() – brings the RGB channels to light blue
ltgreen() – brings the RGB channels to light green
violet() – brings the RGB channels to violet
pink() – brings the RGB channels to pink
rgbww() – brings the RGB channels to RGB warm white
gored() – brings the RGB channels to red, turns off blue and green
gogreen() – brings the RGB channels to green, turns off red and blue
goblue() – brings the RGB channels to blue, turns off red, and green
bright() – brings the RGB channels up 10%
dim() – brings the RGB channels down 10%
stat() – reports levels of the RGB channels
ramp(x) – sets the default rate for LED ramping. The lower the value of x, the faster it will ramp and vice versa. This setting is saved in EEPROM and will remain after being powered off. (x can be 0-999, technically it's the delay in the fade loop) The default value is 4.
cycle() – starts color cycling
pause() – stops color cycling
rate(x) – sets the rate of color fade when color cycling. This setting is saved in EEPROM and will remain after being powered off. (x can be 0 to 999) The default value is 4.
stay(x) – sets the time to stay on a color when color cycling, this value is in SECONDS, not milliseconds. (x can be 0-999) The default value is 0.
alloff() – Ramps LED's to off at the default ramp rate.

Button Commands - The firmware, as shipped, provides support for some basic commands for button control. A 'long press' is a button depressed for more than ½ second, while a 'short press' is a button press that is shorter than a ½ second.

Button 1 – A short press will increase the level of the white LED by 10%. If the white LED is at any level above 0%, a long press will fade the LED to 0. If the white LED is at 0%, a long press will bring the white LED to 100%.

*If the controller is in cycle mode, this button increases the speed of the cycle. After the fastest speed, it will reset to the slowest. A long press of this button while in color cycle mode will reset the speed to the default.

Button 2 – Cycles through red, green, blue, magenta, cyan, gold, white (RGB white), and color cycle. Once in color cycle, a short press will freeze the cycle and retain the current color. At this point, a short press will start the cycle again. If you would like to stop the cycle, do a short press to freeze the cycle, and then a long press to turn the RGB channels off.

Notes

Notes