

How-To: Make an RGB combination door lock (Part 1)

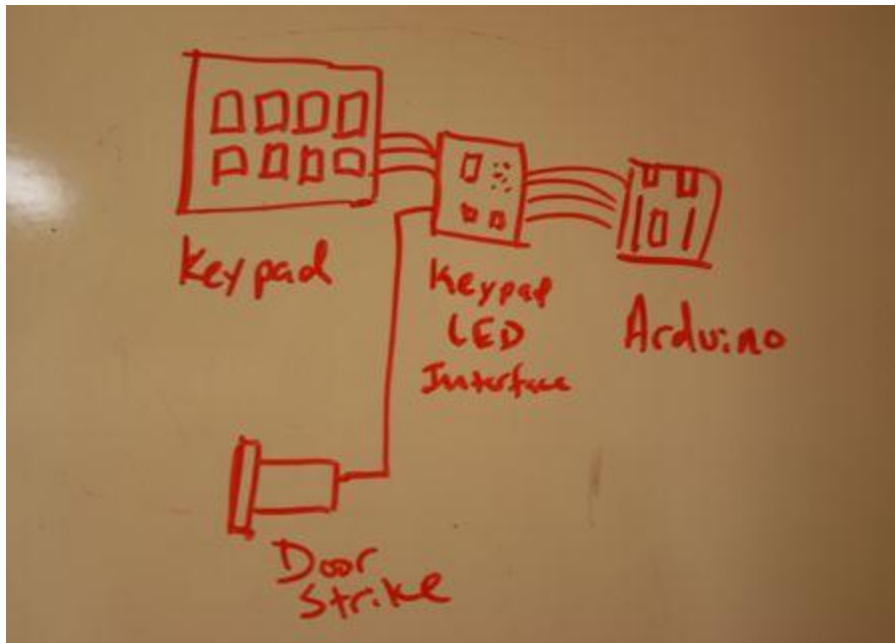


[Part 2 can be found here](#)

Putting a custom designed electronic lock on your space seems like a geek right of passage. For our latest workspace, we decided to skip the boring numbered keypad and build a custom RGB backlit keypad powered by an Arduino. Instead of typing in numbers, your password is a unique set of colors. In today's How-To, we'll show you how to build your own and give you the code to make it all work.

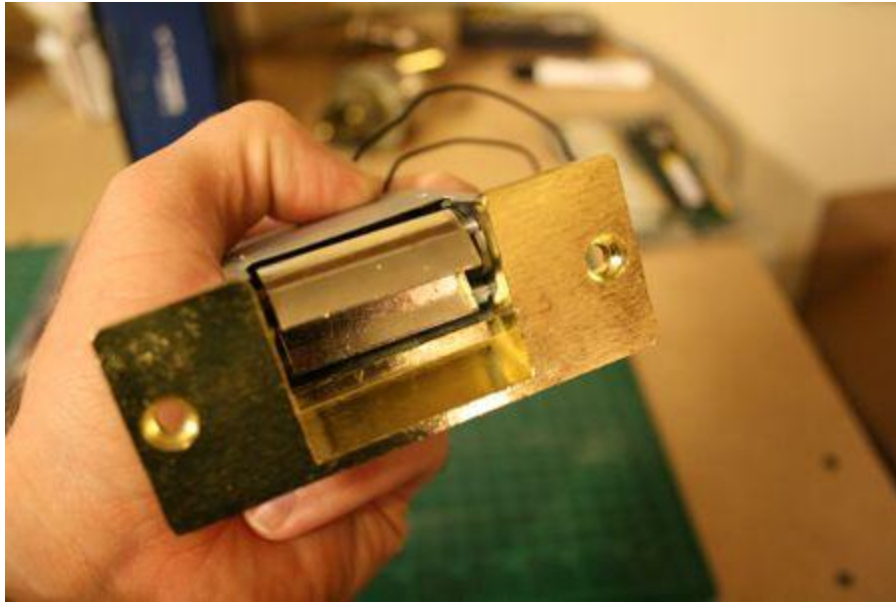


The basic design for the RGB keypad came from [JMG]'s [Arduino based Monome clone](#). He used an Arduino, and multiplexed RGB LEDs with some digital potentiometers to create a color mixing keypad. Since we couldn't fit the complete 4×4 keypad into a standard 2 gang wall box, we chopped the design down to a 2×4 matrix. This cuts down significantly on the cost to build the keypad and makes the code that much easier to digest.

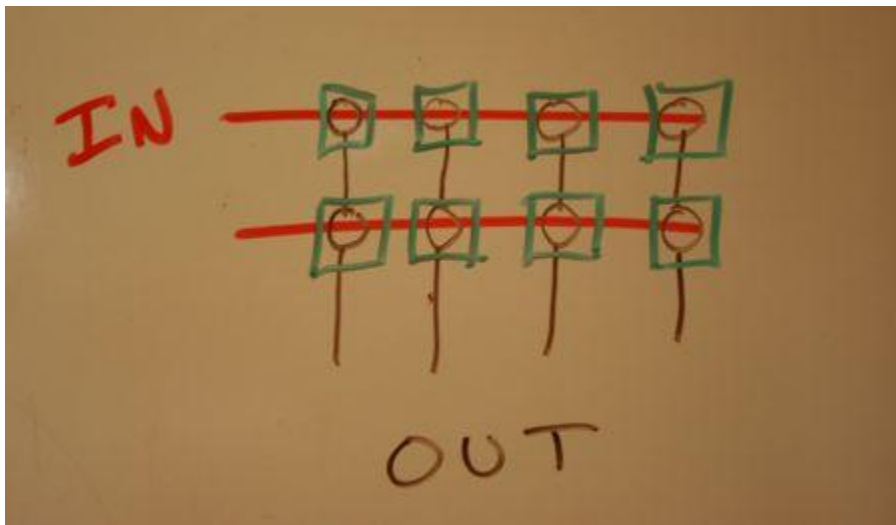


To build your own RGB keypad, you'll need the following:

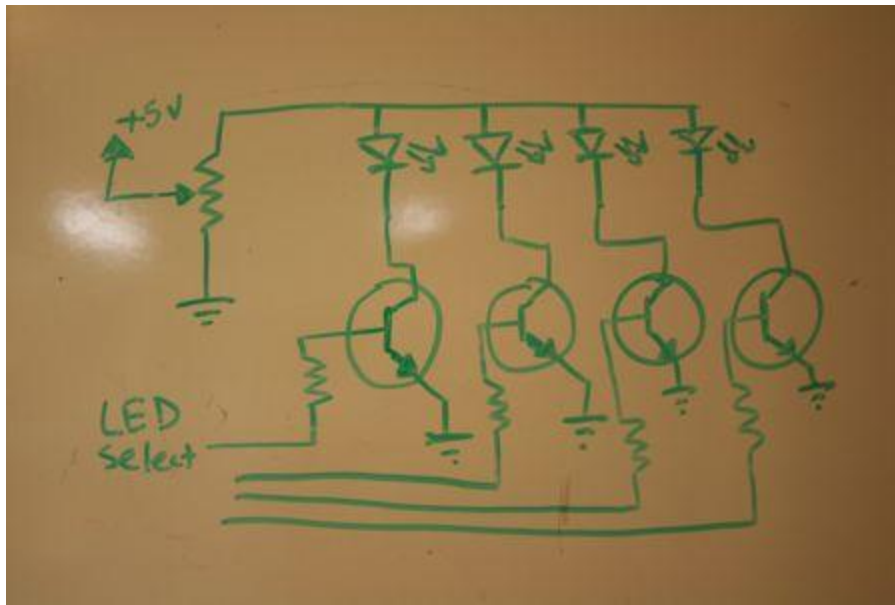
- An electric door strike (Smarthome.com)
- A locking door handle (Any hardware store)
- An Arduino or compatible clone (Sparkfun, adafruit and others)
- 1 TIP120 transistor
- 1 1N4001 diode
- 10 1N4148 diodes
- 4 2n2222 transistors
- 1 Monome style keypad (Sparkfun Electronics)
- 1 Keypad PC board (Sparkfun Electronics)
- 8 RGB LEDs (Sparkfun Electronics)
- 1 7805 voltage regulator
- 4 100 ohm resistors
- 2 150 ohm resistors
- 8 1 kohm resistors



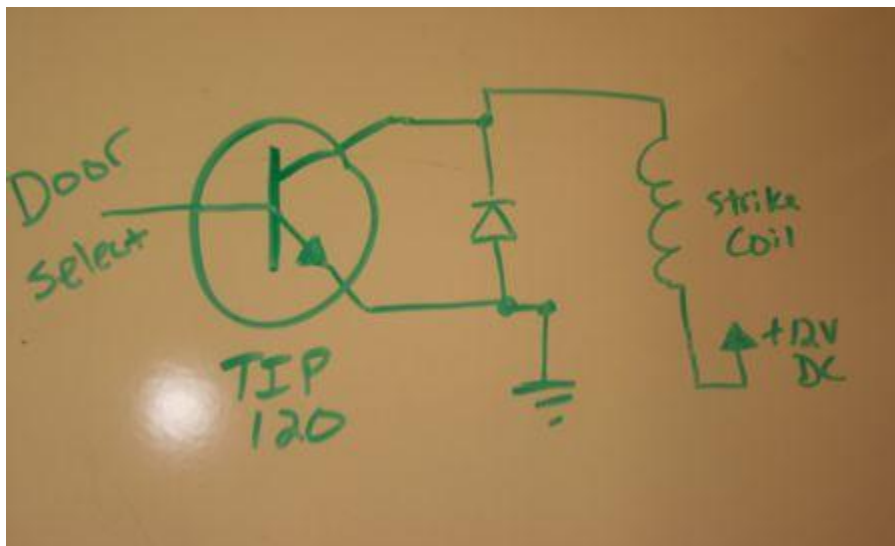
To reliably lock and unlock the door, we ordered an electric door strike. We scored this one as an open box item from [Smarthome.com](https://smarthome.com). It's a 12 Volt DC unit designed just for Schlage commercial door locks. The edge of the strike is slightly recessed from the mounting plate, so it might not work with certain locks. It features a thinner body than the non-recessed version, which will allow us to cut a smaller but deeper hole in the door frame. Without power, the strike stays locked, keeping the locking door shut. When 12 volts is applied to the coil, the strike releases, allowing the door to be pulled open. For the prototype build, you don't have to purchase a strike just yet; you can use a LED and a resistor to indicate the door lock state for testing your code.



The keypad is actually built from two separate circuits that physically overlap. The input circuit is a simple keypad matrix. To read each button push, the Arduino brings one keypad input line high and checks the voltage of the four output lines in order. The diodes on the PC board prevent feedback across the rows and columns.



The RGB LEDs are lit via a completely separate set of circuits. Each row of like colored LEDs is brightness controlled by a digital potentiometer. The digital pot works just like a normal pot, but it's digitally controlled by the Arduino. Meanwhile, each column of LEDs is activated by a separate transistor. By quickly changing the resistance and stepping through the columns, each LED will appear to be individually controlled.



The door strike circuit is pretty simple. Since it contains a coil, we'll treat it like the coil of a stepper motor and use a TIP120 transistor to supply the power. When power is removed from a coil, the collapsing magnetic field creates a current within the coil. To keep the TIP120 from burning out, we'll add a diode to handle the surge created by the field breakdown.

update: [Triffid] pointed out that the diode is better placed in parallel with the coil to handle the transient surge. He's correct, but the circuit here has operated perfectly for several months, so you'll be fine either way.



The traces for the buttons looked a bit challenging to etch at home, so we ordered this PC board that Sparkfun produces for their keypads. Sparkfun helpfully provides the layout for these keys in their eagle library, so you can make your own PCB if you prefer. For reliability, you'll probably want to have it commercially produced. The board wasn't really designed to break apart, but after a review of the traces and vias we decided that we could get away with trimming a couple of rows from the board.



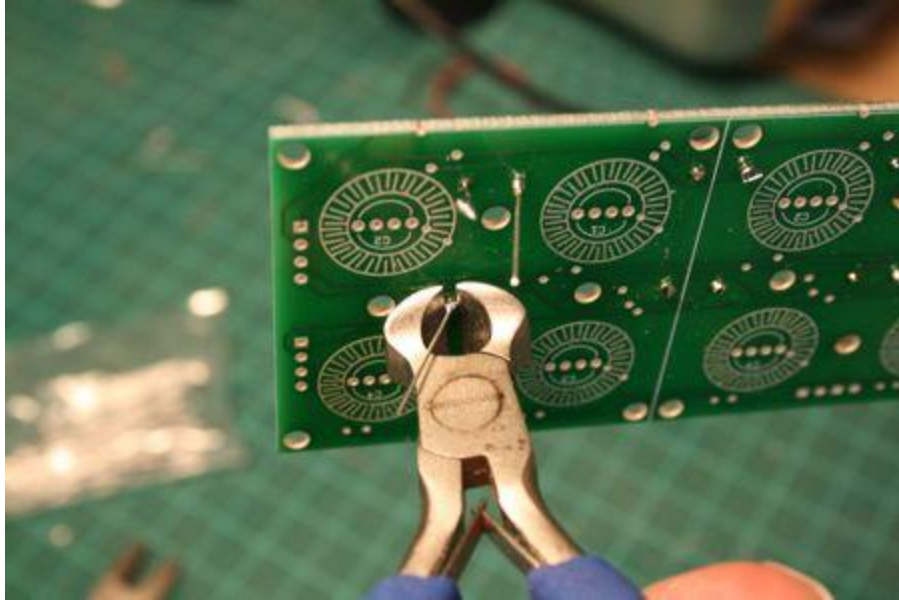
We carefully split the board down the middle with a band saw. If you look closely, you can see where some of the vias were actually cut in half. (A paper cutter might work in a pinch) Don't forget to put on a mask to keep the dust out of your lungs.



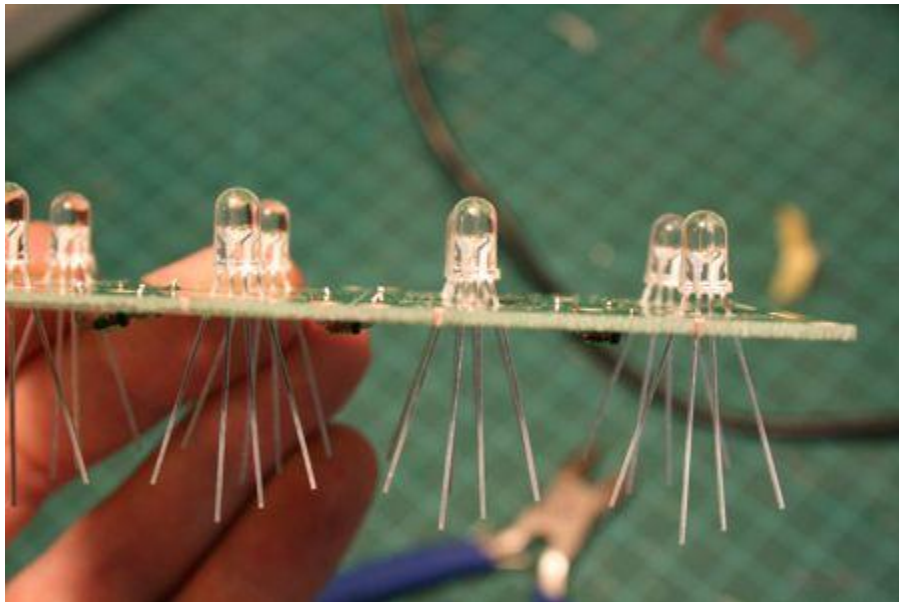
Cutting the button pad is much easier. The pads have pre-scored lines that just need a quick swipe of a sharp knife or scissors to separate them.



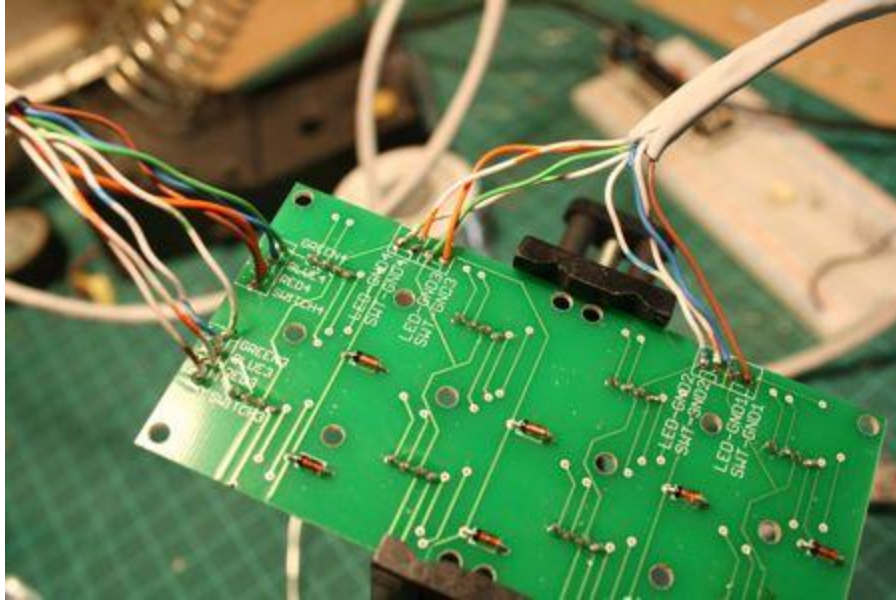
The new shorter PCB only needs a few parts: some 1N4148 diodes and the RGB LEDs. The silkscreen on the board indicates the direction and position of diodes and LEDs.



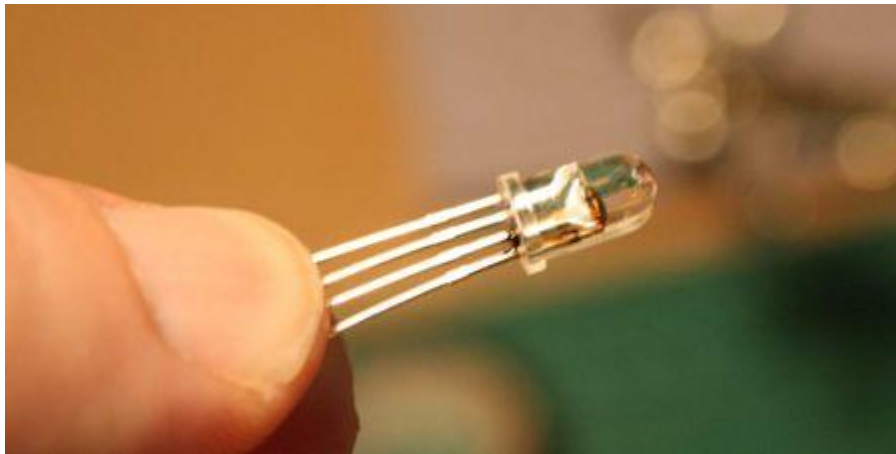
Once you solder on the 1n4148 diodes, cut them as close to the PC board as you can. Flat head cutters like these work extremely well. The keypad will sit on this side of the board and we want to make sure that it can sit as flat as possible.



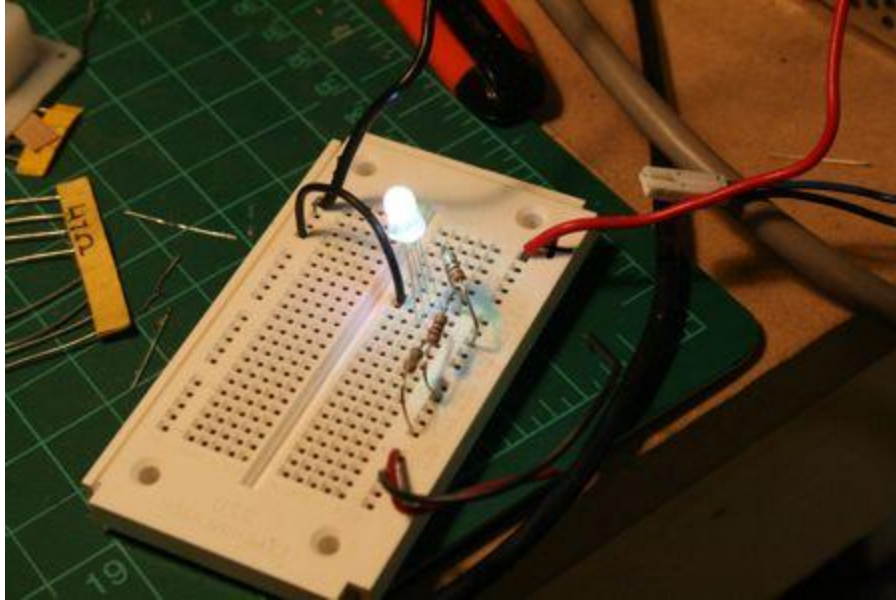
Install the LEDs in the orientation indicated by the silk screen. Carefully push them down into the board until they're inserted just like this. If you let them stick up too high, they'll interfere with the keypad buttons being pushed.



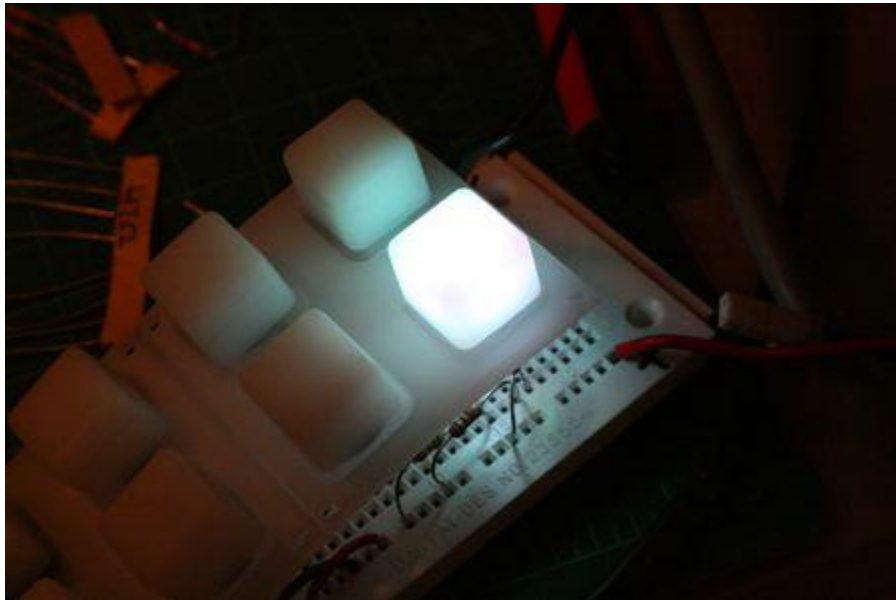
Once you've soldered all the LEDs in place, clip them flush as well. Then you'll need to add some cable to jumper from the keypad to the interface board we'll build. We used some old CAT-5 wiring. Since each axis of the board has eight pins, it's perfect for the application.



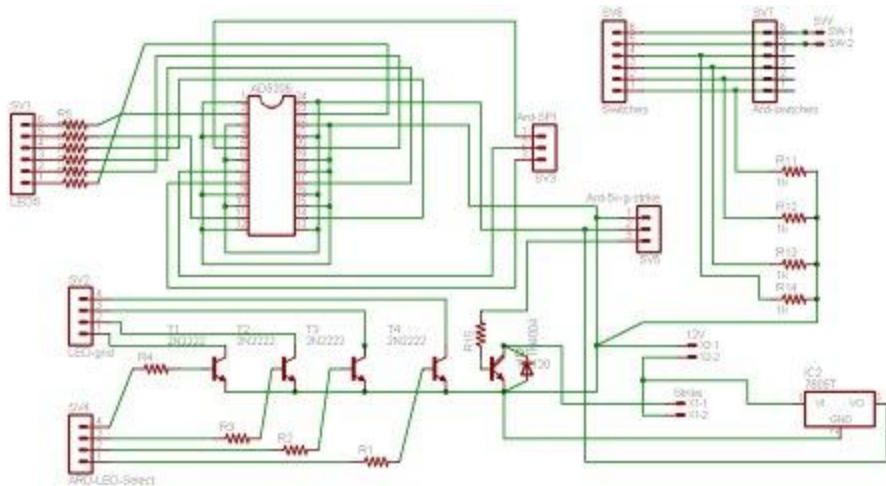
Each RGB LED has three LEDs inside the package. They share a common terminal and have a single separate lead coming out. Because they have different characteristics – that is brightness, current and voltage requirements, we spent some time testing out various combinations. We even murdered a couple of innocent \$2 LEDs just for you. Hey, the other two colors are still usable...



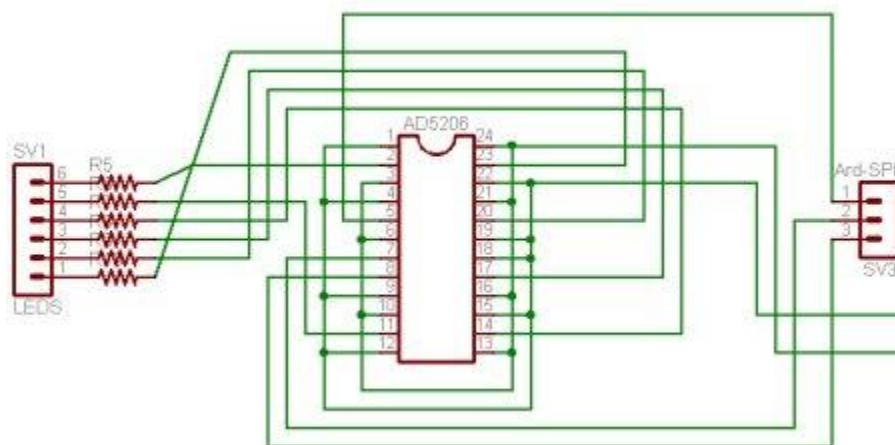
After some experimentation, we managed to find the right combination to create some fairly white light. The requirements will vary between manufacturers, but for the Sparkfun LEDs we found that a pair of 100 ohm resistors and a single 150 ohm resistor blended the red, green and blue fairly well.



The color combination was hard on the eyes until we put the keypad over the LED to double check our findings. In real life, you can see some blending lines from the offset of each LED, but it still looks great.



The circuit has plenty of components, but it's pretty easy to build. We'll break everything up by section to keep things easy. You can download the all of the schematics, Eagle project files, and code for the Arduino [here](#).



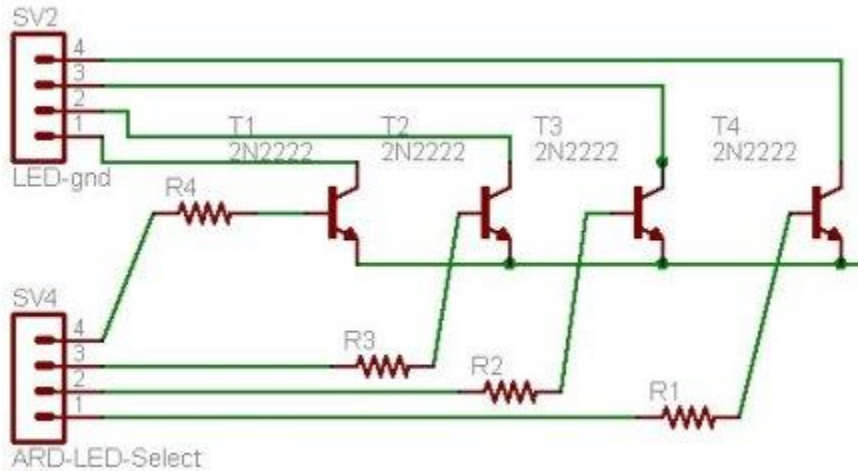
The digital pot has six outputs. Each of these will power a row of red, green or blue LEDs, via a color matching resistor. The digital potentiometer wiring comes directly from [this how-to](#). You can read it if you need more information, or use our quick version:

- Connect AD5206 pins 3, 6, 10, 13, 16, 21 and 24 to 5v.
- Connect pins 1, 4, 9, 12, 15, 18, 19, and 22 to ground.
- Connect pot pin 5 to Arduino pin 10
- Connect pot pin 7 to Arduino pin 11
- Connect pot pin 8 to Arduino pin 13

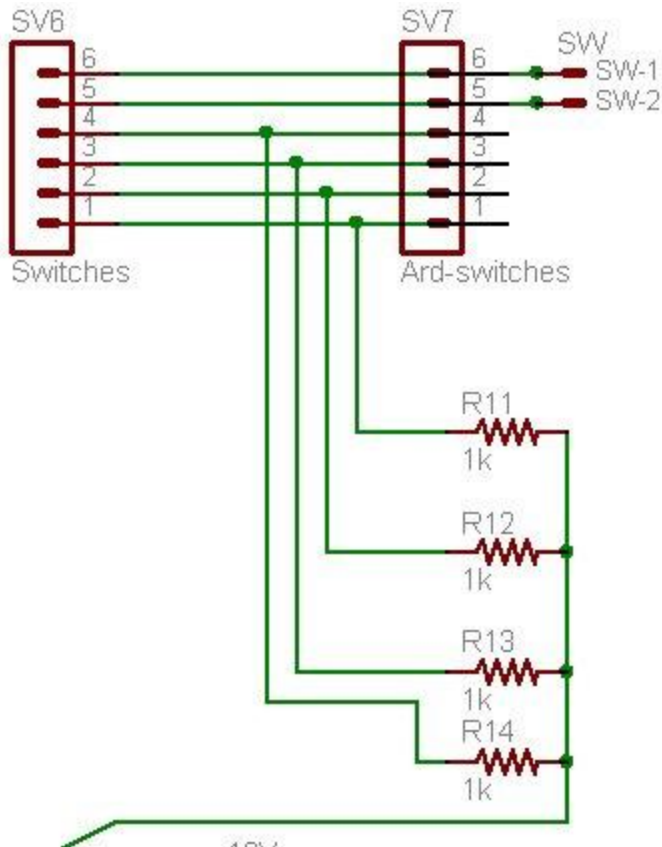
Grab four 100 ohm resistors and two 150 ohm resistors. Place them in the breadboard in a row with each end in a separate bus. (Across the center of the board is easiest) Connect the six LED leads from the keypad to one end of each resistor – reds get the 150's and blue and green into the 100's. Here's the connection order we used.

- RED3 to a 150 ohm resistor to pot pin 14
- GREEN3 to a 100 ohm resistor to pot pin 11

- BLUE3 to a 100 ohm resistor to pot pin 2
- RED4 to a 150 ohm resistor to pot pin 23
- GREEN4 to a 100 ohm resistor to pot pin 20
- BLUE4 to a 100 ohm resistor to pot pin 17



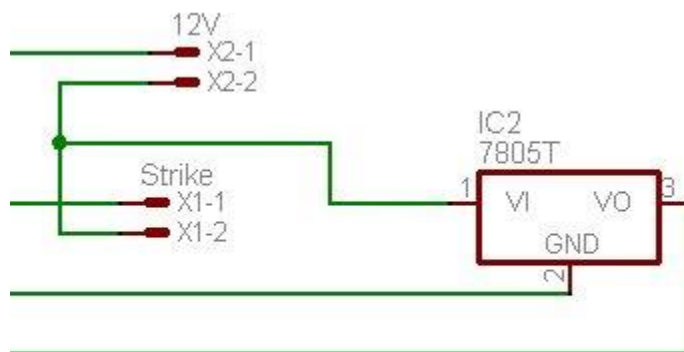
To ground the LED busses, we'll be using four 2N2222 transistors. The Arduino will trigger each transistor individually through a 1Kohm resistor. The collector of each transistor connects to a ground line from the keypad. The emitter of each transistor is connected to the ground. The four transistor select lines connect to Arduino pins 0, 1, 2, and 3. Yes, they're marked Analog in, but it doesn't matter.



The keypad switch matrix is connected in four columns and two rows. Each of the four columns gets a pull-down resistor. We used 1Kohm resistors for R11, R12, R13, and R14; one lead connects to the columns and the other is grounded.

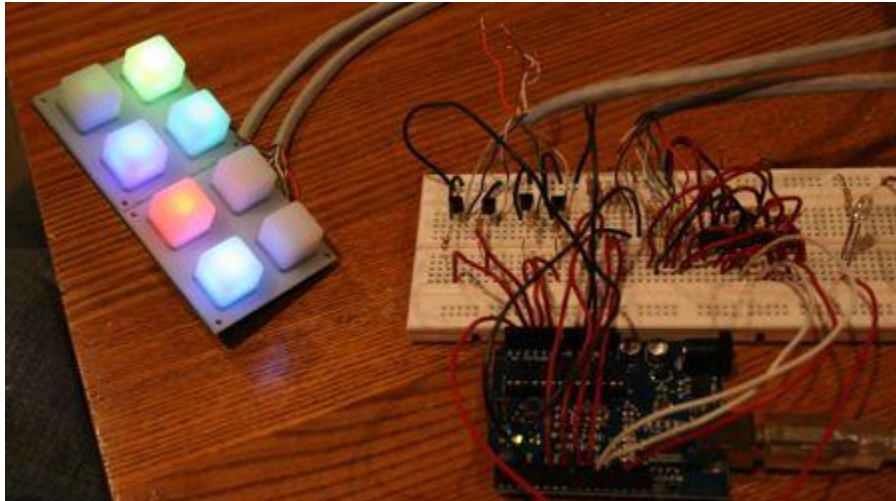
Arduino pins 2 and 3 should connect to the two ungrounded lines, which are marked SWITCH3 and SWITCH4 on the PC board (5 and 6 on the schematic).

Arduino pins 6, 7, 8, and 9 should connect to the four output lines marked SWT-GND1, SWT-GND2, SWT-GND3, and SWT-GND4 (1-4 on the schematic).



The final version of the board takes a 12VDC input to drive the door lock. We added a 7805 to drop the 12V down to 5V for the Arduino. You don't need it for the prototype version unless you

want to test the striker. The Arduino has an on-board regulator, but 7805's are cheap and it helps reduce the load on the Arduino's built in regulator. For code development, we just connected an LED with a resistor to the output line that will control the door lock.



With everything wired in the prototyping board, it's time to test things out. With any luck, you'll soon be rewarded by the pulsing, glowing sight of several RGB LEDs under your tender digits.

```
Arduino  File  Edit  Sketch  Tools  Help
Arduino - 0011
Upload to I/O Board
row_entry_pad_meffect
//based on button_test
//this is a simple color change routine. Everything should be red to start
//As a button is pressed, it will turn green.
//When everything is green, the door is unlocked.

#define DATAOUT 11//MOSI (pin 7 of AD5206)
#define DATAIN 12//MISO - not used, but part of builtin SPI
#define SPICLOCK 13//sck (pin 8 of AD5206)
#define SLAVESELECT 10 //removed the slave switching code entirely.
#define COLS 4 //x axis
#define ROWS 2 //y axis
#define effect_select 1 //choose idle effect

const byte colpin[COLS] = { //pins for led column grounding transistors
  14,15,16,17}; // Using the analog inputs as digital pins (14=A0,15=A1,16
```

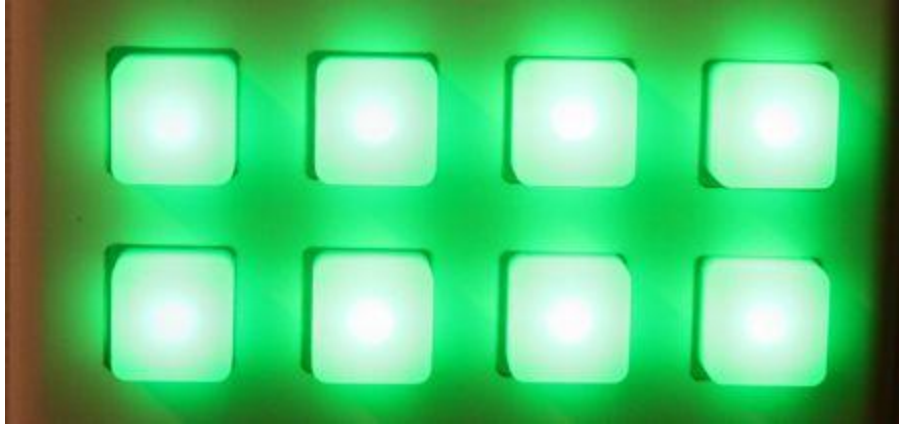
Programming the Arduino is a snap. Just download the software for your OS [here](#). Now follow the [Getting Started guide](#) to get the Arduino software talking to the Arduino board. Once you've enjoyed the blinking LED demo, come back here and get your keypad rolling.

```
++.....  
.....  
.Lock door.....1. 3  
Lock door.....1, 2  
Unlock door.....1, 1  
Lock door.....1, 0  
Unlock  
door.....  
.....  
.....  
.....  
.....
```

Once you've set up and tested your Arduino, it's time to test out your prototype. Download the `button_test` code from [here](#). Paste it into a new sketch and upload it to the Arduino. Click the serial console button and you should start seeing dots accumulating in the window. If you press a button on the pad, the Arduino should print a message to the console and toggle the lock output state.



Once your buttons are tested, you'll probably want to try out your LEDs. Grab the `RGB_light_fade` routine from the [same page](#) and upload it to your Arduino. You should get treated to a nice little light show. This is our favorite demo because it really shows off the color mixing capabilities of the digital potentiometer.



With your LEDs and buttons working, you can grab the `row_entry_pad_meffect` lock code from the same place and upload it. Now the keypad should start flashing blue buttons while it's idle. On key presses, the keys will change colors. By entering the correct color code, the pad will flash green and unlock the door for 10 seconds. If you go over the limit counter, it will flash red for 30 seconds.



Next time we'll show you how to make the permanent version of the keypad, walk through the code for the Arduino, make the PC board, cut a custom wall plate, and install the lock strike.