LED matrix using shift registers

by barney_1 on June 10, 2008

Table of Contents

Intro: LED matrix using shift registers

This instructable is meant to be a more complete explanation than others available online. Notably, this will provide more hardware explanation than is available in the LED Marquee instructable by led555.

GoalsThis instructable presents the concepts involved with shift registers and high side drivers. By illustrating these concepts with an 8x8 LED matrix I hope to provide you with the tools needed to adapt and expand to the size and layout your project calls for.

Experience and Skills
I would rate this project to be of medium difficulty:

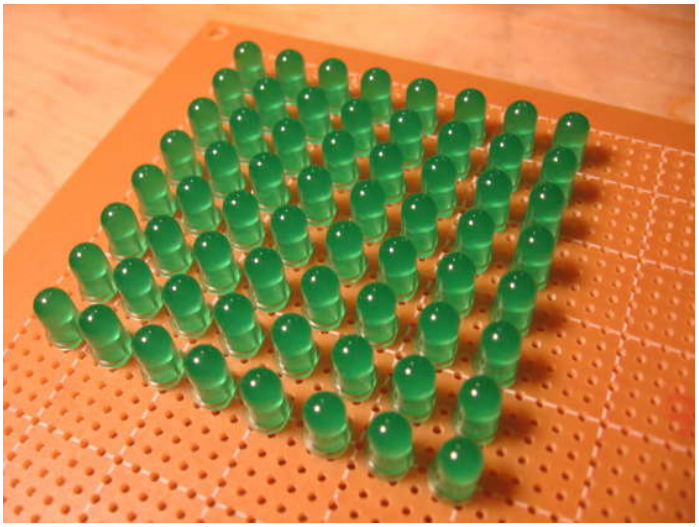
- If you already have experience programming microcontrollers and working with LEDs this project should be fairly easy for you to complete and to scale to larger arrays of lights.
- If you are just starting out with microcontrollers and have flashed an LED or two you should be able to complete this project with some help from our friend google .
- If you have little or no experience with microcontrollers or programming this is probably beyond what you should be getting yourself into. Try out a few other beginner projects and come back when you've got some more experience writing programs for microcontrollers.

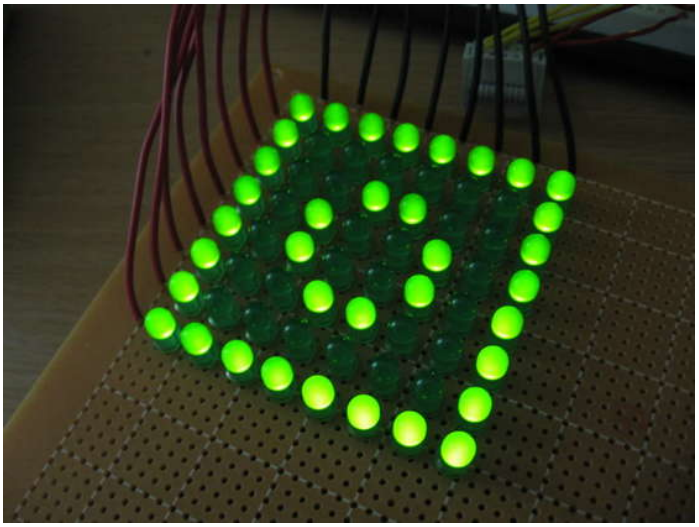
Disclaimer and Credit
First, I am not an electrical engineer. If you see something that is wrong, or not a best practice, please let me know and I'll make the correction.

Do this at your own risk! You should know what you're doing or you can cause damage to your computer, your microcontroller, and even yourself.

I have learned a lot from the internet, particularly from the forums at: http://www.avrfreaks.net

I am using a font set that came with the ks0108 universal C library. Check that out here:
http://en.radzio.dxp.pl/ks0108/





step 1: Parts

Parts List
General PartsTo make an 8x8 grid of LEDs and control them you will need:

- 64 LEDs of your choice
- 8 Resistors for the LEDs
- 1 Shift register for the columns
- 1 Driver array for the rows
- 8 Resistors for switching the driver array
- 1 microcontroller
- 1 clock source for microcontroller
- 1 prototyping board
- 1 power supply
- Hook-up wire

Specific Parts Used HereFor this instructable I used the following:

- 64 green LEDs (Mouser part #604-WP7113GD)
- 8 220ohm 1/4 watt resistors for the LEDs (Mouser part #660-CFS1/4CT52R221J)
- 1 HEF4794 LED driver with shift register (Mouser part #771-HEF4794BPN)
- 1 mic2981 High-Voltage High-Current Source Driver Array (Digikey part #576-1158-ND)
- 8 3.3kohm 1/4 watt resistors for switching the driver array (Radio Shack part #271-1328)
- 1 Atmel ATmega8 microcontroller (Mouser part #556-ATMEGA8-16PU)
- 1 12MHz crystal for the microcontroller clock source (Mouser part #815-AB-12-B2)
- 1 2200-hole prototyping board (Radio Shack part #276-147)
- Converted ATX power supply: See This Instructable
- Solid core 22-awg hook-up wire (Radio Shack part #278-1221)
- Solderless breadboard (Radio Shack part #276-169 (no longer available, try: 276-002)
- AVR Dragon (Mouser part #556-ATAVRDRAGON)
- Dragon Rider 500 by Ecros Technologies: See This Instructable

Notes Regarding Parts
Row and Column Drivers: Probably the most difficult part of this project is picking the row and column drivers. First off, I do not think a standard 74HC595 shift register is a good idea here because they cannot handle the kind of current we want to send through the LEDs. This is why I chose the HEF4794 driver as it can easily sink the current present when all 8 leds are in one row are switched on.

The shift register is present on the low side (the ground pin of the leds). We will need a row driver that can source enough current to string multiple columns together. The mic2981 can supply up to 500mA. The only other part I have found that performs this task is the UDN2981 (digikey part #620-1120-ND) which is the same part by a different manufacturer. Please send me a message if you know of other high-side drivers that would work well in this application.

LED Matrix: This matrix is 8x8 because the row and column drivers each have 8 pins. A larger LED array may be built by stringing multiple matrices together and will be discussed in the "modular concepts" step. If you want a large array, order all of the needed parts at one time.

There are 8x8, 5x7 and 5x8 LED matrices available in one convenient package. These should be easy to substitute for a diy matrix. Ebay is a good source for these. Mouser has some 5x7 units available such as part #604-TA12-11GWA. I used cheap green LEDs because I'm just playing around and have fun. Spending more on high-brightness, high-efficiency LEDs can allow you to produce a much more spectacular looking display... this is good enough for me though!

Control Hardware: The matrix is controlled by an Atmel AVR microcontroller. You will need a programmer for this. Because I am prototyping I am using the Dragon Rider 500 for which I have written both assembly and usage instructables. This is an easy tool for prototyping and I highly recommend it.

step 2: The matrix

I will be building my own LED matrix for this project using 5mm leds and a prototyping board from Radio Shack. It should be noted that you can purchase 8x8 dot matrix led modules from several sources, including ebay. They should work just fine with this instructable.

Construction Considerations**Alignment**
The LEDS need to be aligned so they face the same direction at the same angle. I found the easiest option for me was to put the body of the LED flush to the board and hold it there with a small piece of plexiglass and a clamp. I put a few LEDs in place a couple of inches away from the row I was working on to make sure the plexiglass was parallel with the prototyping board.

Rows and Columns
We need to have a common connection for each row as well as each column. Because of our row and column driver choice we need to have the anode (positive lead of the LED) connected by row and the cathode (negative lead of the LED) connected by column.

Control Wires
For this prototype I am using solid core (single conductor) hook-up wire. This will be very easy to interface with a solderless breadboard. Feel free to use a different connector type to suit your project.

Building the Matrix1. Place the first column of LEDS in the prototyping board.
2. Double check that your polarity for each LED is correct, this will be very difficult to fix if you realize it later.
3. Solder both leads of the LED to the board. Check to make sure they are aligned correctly (not at weird angles) and clip off the cathode leads. **Make sure you do not clip the anode lead, we will need that later so just leave it pointing up.**
4. Remove the insulation from a piece of solid core wire. Solder this piece of wire to each cathode right at board level.

- I tacked this at each end then went back and added a bit of solder at each junction.
- This wire should run past your last LED to make for an easy interface when we add control wires.

5. Repeat parts 1-4 until you have all LEDs in place and all column buses soldered.
6. To create a row bus, bend several of the anode leads at a 90 degree angle so they touch the other anode leads in the same row.

- There are detailed pictures of this below.
- Take care not to let these come in contact with the column buses, creating a short circuit.

7. Solder the leads at each junction and clip off the excess anode leads.

- Leave the last anode sticking past the final LED. This will be used to connect the row driver control wires.

8. Repeat parts 6 & 7 until all rows buses have been soldered.
9. Attach control wires.

- I used red solid core wire for the rows and black for the columns.
- Connect one wire for each column and one for each row. This can easily be done at the end of each bus.

ImportantThis LED matrix does not have any current limiting resistors. If you test this without resistors you will probably burn out your LEDs and all this work will be for nothing.

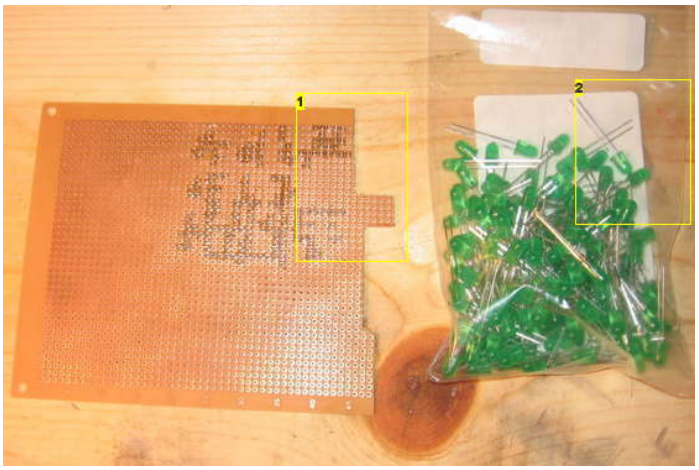


Image Notes
1. I'm reusing this prototyping board. I desoldered the last project and am now ready for this one.
2. Big bag o LEDs.

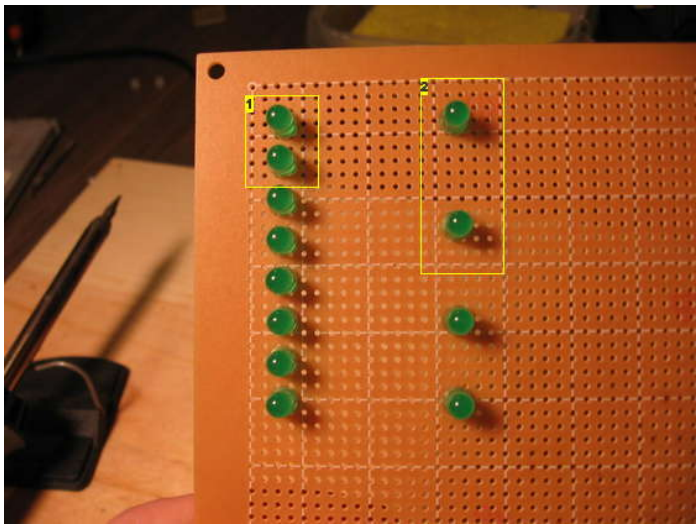


Image Notes
1. LEDs ready for soldering.
2. LEDs used as spacers for plexiglass.

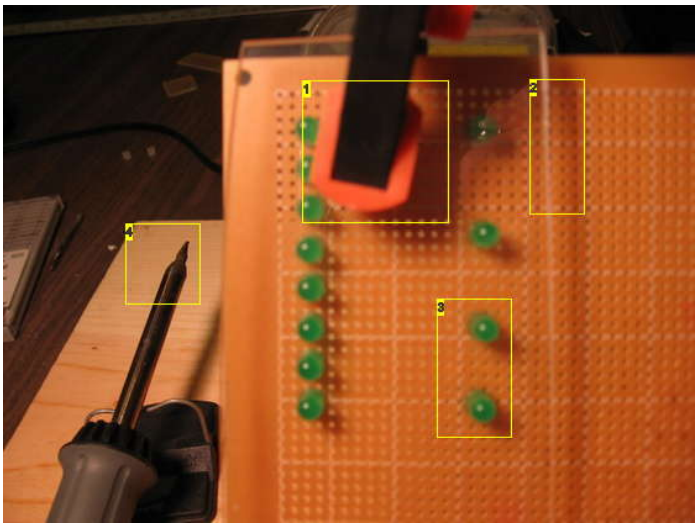


Image Notes
1. Clap
2. Plexiglass holding the LEDs flush to the board.
3. LEDs used for spacing the plexiglass away from board. These will not be soldered.
4. Autofocus on the camera picked the wrong subject. Sorry!

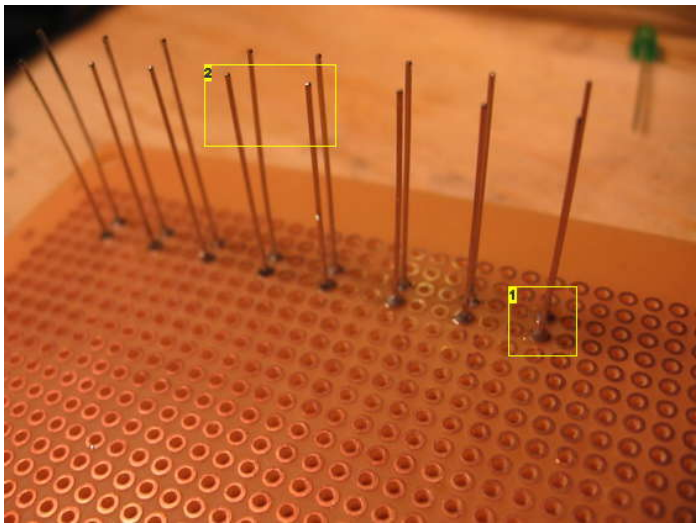
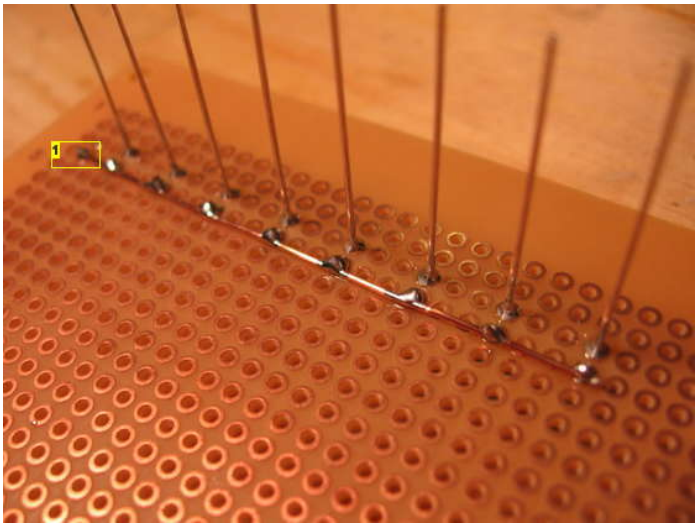


Image Notes
1. All LEDs in this row are now sodered to the board.
2. Double-check to make sure you have the proper polarity. The long legs should be the anode, the short should be the cathode.



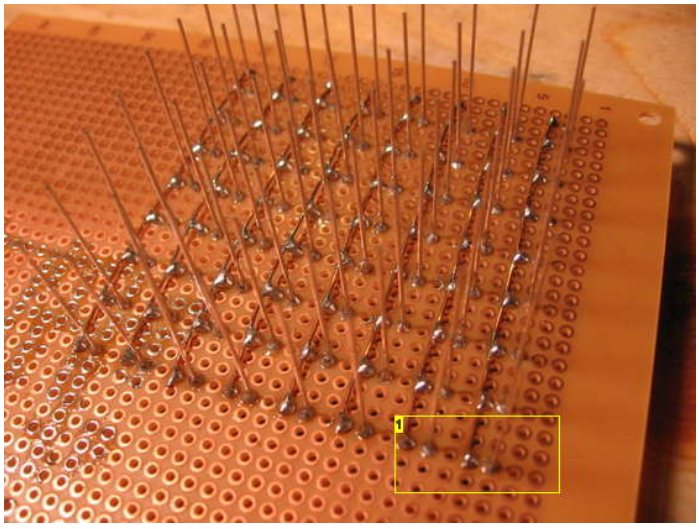


Image Notes
1. Soldered to a pad just past the last LED. This will be a solder point for the control wires.

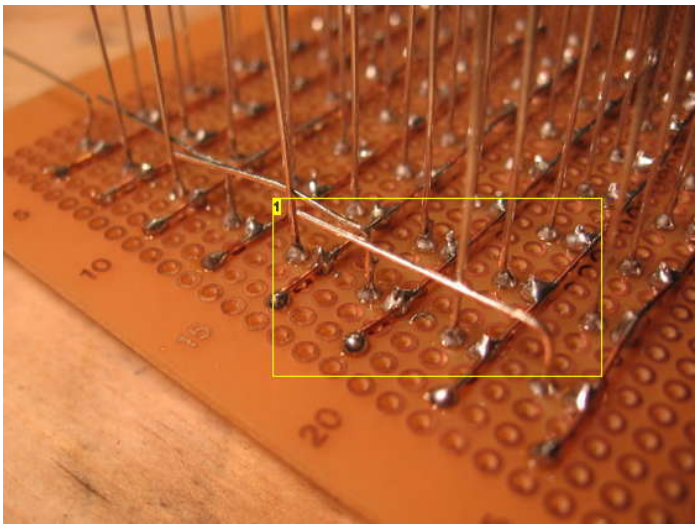


Image Notes
1. Row of anodes have been bent for soldering.

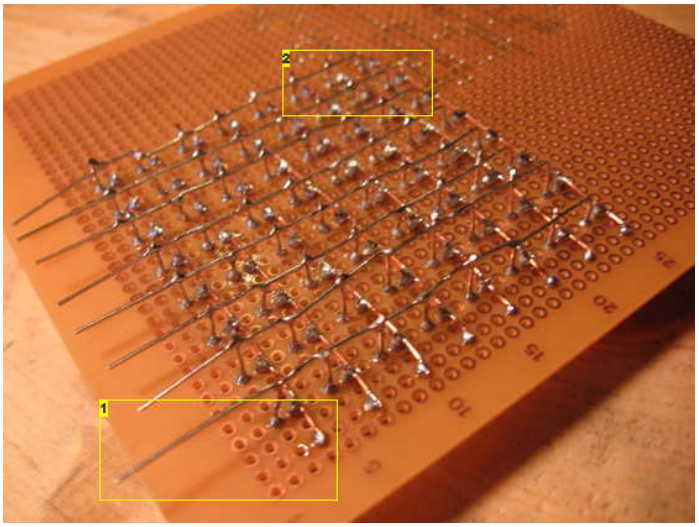


Image Notes
1. Excess left for connection to controll wires.
2. All rows have been bent, soldered, and clipped.

Image Notes
1. Cathode soldering repeated for each column. Anodes are soldered to the board but left sticking up until all of the columns are done.

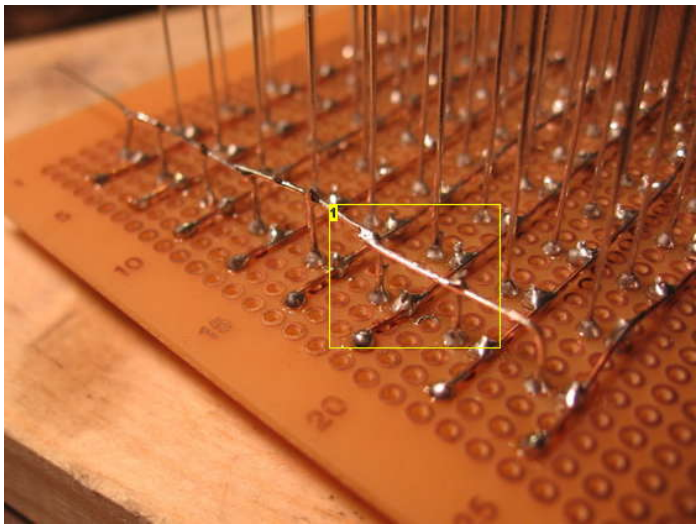
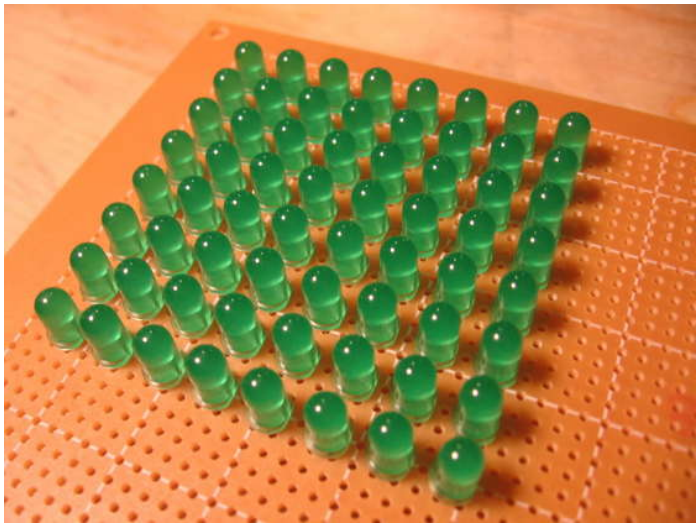


Image Notes
1. Row of anodes has been soldered together creating a row bus. Excess has been clipped off.



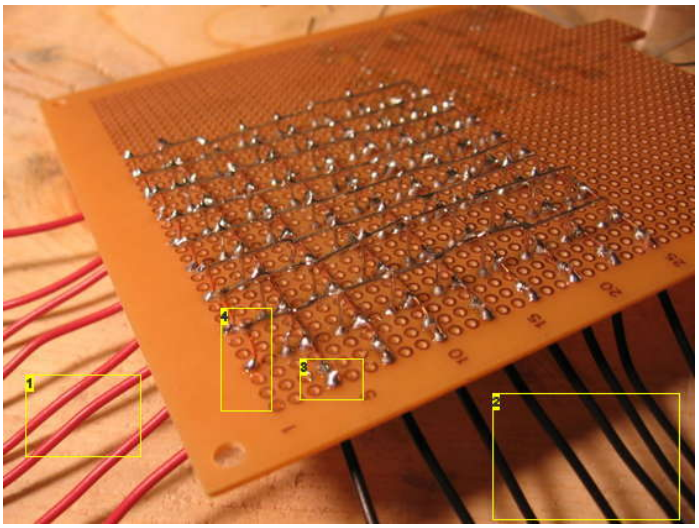
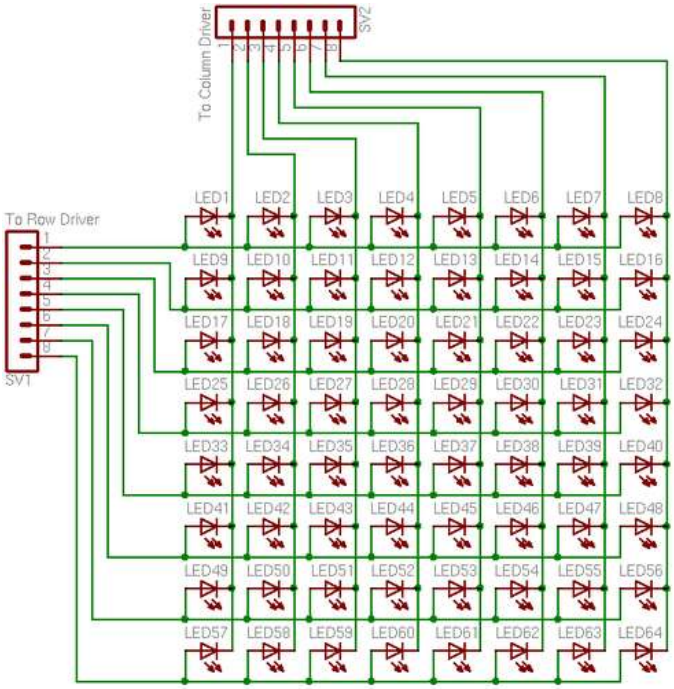


Image Notes
1. Row driver wires.
2. Column wires.
3. soldered to the board and then to the column bus.
4. Row driver wire sticks up past and is soldered to the board as well as to the raised row bus wire.



step 3: The control hardware

We need to control the columns and the rows of our LED matrix. The matrix has been constructed so that the Anodes (voltage side of the LED) constitute the rows, and the Cathodes (ground side of the LED) make up the columns. This means our row driver need to source current and our column driver needs to sink it.

In order to save on pins I am using a shift register to control the columns. This way I can control an almost unlimited number of columns with just four microcontroller pins. It is possible to use only three if the Enable Output pin is tied directly to voltage. I have selected the HEF4794 LED driver with shift register. This is a better option than a standard 74HC595 as it can easily sink the current present when all 8 LEDs are on at one time.

On the high side (current source for the rows) I am using an mic2981. The schematic shows a UDN2981, I believe these two are interchangeable. This driver can source up to 500mA of current. Because we are only driving 1 row at a time this gives a lot of opportunity for expansion, up to 33 columns for this chip (more on that in the "modular concepts" step).

Building the Control Hardware
For this instructable I have just breadboarded this circuit. For a more permanent solution you will want to either etch your own circuit board or use prototyping board.

1. Row Driver

- Place the mic2981 (or UDN2981) in the breadboard
- Connect Pin 9 to Voltage (This is confusing in the schematic)
- Connect Pin 10 to Ground (This is confusing in the schematic)
- insert 3k3 resistors connecting to pins 1-8
- Connect from Port D of the ATmega8 (PD0-PD8) to the 8 resistors
- Connect the 8 row control wires of the LED matrix to pins 11-18 (note that I have connected the lowest row of LEDs to Pin 18 and the Highest row to Pin 11).

2. Column Driver

- Place the hef4794 in the breadboard
- Connect Pin 16 to voltage
- Connect Pin 8 to ground
- Connect 220 ohm resistors to Pins 4-7 and 11-14.
- Connect the 8 column control wires from the LED matrix to the 8 resistors you just connected.
- Connect Pin1 (Latch) to PC0 of the ATmega8
- Connect Pin2 (Data) to PC1 of the ATmega8
- Connect Pin3 (Clock) to PC2 of the ATmega8
- Connect Pin15 (Enable Output) to PC3 of the ATmega8

3. Clock Crystal

- Connect a 12MHz crystal and load capacitors as shown in the schematic

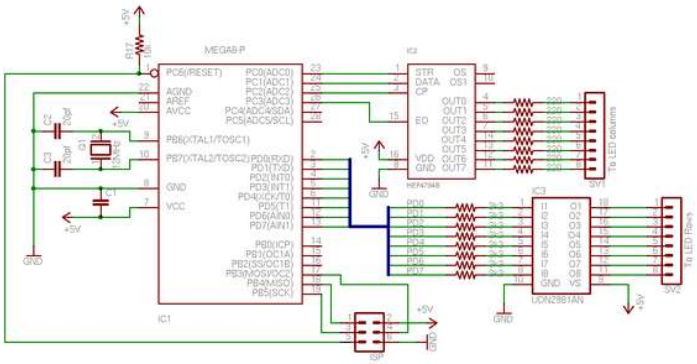
4. ISP

- Connect the programming header as shown in the schematic

5. Filtering Capacitor and Pull-up resistor

- It is best to filter the voltage supplied to the ATmega8. Use a 0.1uf capacitor between Pin 7 & 8 of the ATmega8
- The reset pin should not be left floating as it can cause random resets. Use a resistor to connect it to voltage, anything about 1k should be good. I've used a 10k resistor in the schematic.

6. Make sure you are using +5v regulated power. It's up to you to design the regulator.



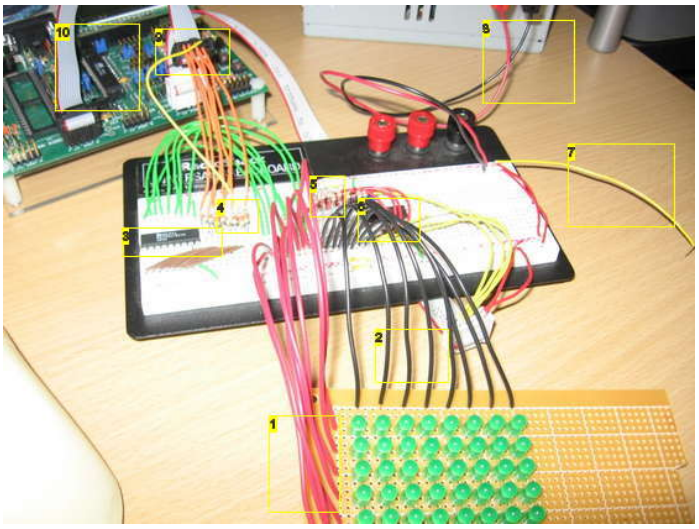


Image Notes
1. Row driver wires.
2. Column Driver wires
3. mic2981 high side driver. Sources current for the rows.
4. 3k3 resistors between the driver input and the microcontroller.
5. 220 ohm current limiting resistors. One for each column.
6. Shift register.
7. oops, left over from measuring current draw. Seems like the average current draw is about 40mA.
8. ATX psu converted to bench-top psu.
9. port D control wires going to the row driver.
10. mega8 sitting happily in its socket aboard the Dragon Rider 500.

step 4: Software

The TrickYes, like everything, there's a trick. The trick is that there are never more than 8 LEDs illuminated at one time.

For this to work well, a bit of crafty programming is needed. The concept I have chosen is to use a timer interrupt. Here's how the display interrupt works in plain english:

- Timer counts up to a certain point, when reached the interrupt service routine is run.
- This routine decides which row is the next one to be displayed.
- The information for the next row is looked up from a buffer and shifted into the column driver (this information is not "latched" so it is not yet displayed).
- The row driver is shut off, no LEDs are currently lit.
- The column driver is "latched" make in the information we shifted in two steps ago the current information to display.
- The row driver then provides current to the new row we are displaying.
- The interrupt service routine ends and program returns to normal flow until the next interrupt.

This happens very very quickly. The interrupt is thrown every 1 mSec. This means that we're refreshing the entire display about once every 8 mSec. This means a display rate of around 125Hz. There is some concern regarding brightness because we are essentially running the LEDs at a 1/8 duty cycle (they are off 7/8 of the time). In my case I get an adequately bright display with no visible flashing.

The full LED display is mapped out in an array. In between interrupts the array can be changed (be mindful of atomicity) and will appear on the display during the next interrupt.

The specifics of writing code for the AVR microcontroller and of how to write code to talk to the shift registers is beyond the scope of this instructable. I have included the source code (written in C and compiled with AVR-GCC) as well as the hex file to program directly. I have commented all of the code so you should be able to use this to clear up any questions about how to get data into the shift register and how the row refresh is working.

Please note that I am using a font file that came with the ks0108 universal C library. That library can be found here: http://en.radzio.dxp.pl/ks0108/

Update:
Shift Registers: How ToI've decided to add a bit about how to program with shift registers. I hope this clears things up for those who haven't worked with them before.

What they do
Shift Registers take a signal from one wire and output that information to many different pins. In this case, there is one data wire that takes in the data and 8 pins that are controlled depending on what data has been received. To make things better, there is an outpin for each shift register that can be connected to the input pin of another shift register. This is called cascading and makes the expansion potential an almost unlimited prospect.

The Control Pins
Shift registers have 4 control pins:

- Latch - This pin tells the shift register when it is time to switch to newly entered data
- Data - The 1's and 0's telling the shift register what pins to activate are received on this pin.
- Clock - This is a pulse sent from the microcontroller that tells the shift register to take a data reading and move to the next step in the communication process
- Enable Output - This is an on/off switch, High=On, Low=Off

Making it do your bidding:
Here's a crash course in the operation of the above control pins:

Step 1: Set Latch, Data, and Clock low

- Setting the Latch low tells the shift register we are about to write to it.

Step 2: Set Data pin to the logic value you want to send to the Shift Register
Step 3: Set Clock pin high, telling the Shift Register to read in the current Data pin value

- All other values currently in the Shift Register will move over by 1 place, making room for the current logic value of the Data pin.

Step 4: Set the Clock pin Low and repeat steps 2 and 3 until all data has been sent to the shift register.

- The clock pin must be set low before changing to the next Data value. Toggling this pin between high and low is what creates the "clock pulse" the shift register needs to know when to move to the next step in the process.

Step 5: Set Latch high

- This tells the shift register to take all of the data that has been shifted in and use it to activate the output pins. This means that you will not see data as it is shifting in; no change in the output pins will occur until the Latch is set high.

Step 6: Set Enable Output high

- There will be no pin output until the Enable Output is set to high, no matter what is happening with the other three control pins.
- This pin can always be left high if you wish

Cascading
There are two pins you can use for cascading, Os and Os1. Os is for fast rising clocks and Os1 is for slow rising clocks. Hook this pin to the data pin of the next shift register and the overflow from this chip will be entered into the next.

End of update

Addressing the displayIn the example program I have created an array of 8 bytes called row_buffer[]. Each byte corresponds to one row of the 8x8 display, row 0 being the bottom and row 7 being the top. The least significant bit of each row is on the right, the most significant bit on the left. Changing the display is as easy as writing a new value to that data array, the interrupt service routine takes care of refreshing the display.

ProgrammingProgramming will not be discussed in detail here. I would warn you not to use a DAPA programming cable as I believe you will be unable to program the chip once it is running at 12MHz. All other standard programmers should work (STK500, MKII, Dragon, Parallel/Serial programmers, etc.).

Fuses:
Make sure to program the fuses to use the 12MHz crystal
hfuse: 0xC9
lfuse: 0xEF

In ActionOnce you program the chip the display should scroll a "Hello World!". Here is a video of the LED matrix in actions. The video quality is pretty low as I made this with my digital camera's video feature and not a proper video or webcam.

File Downloads


8x8_Matrix.zip (9 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to '8x8_Matrix.zip']

step 5: modular concepts

This project is scalable. The only true limiting factor will be how much current your power supply can provide. (The other reality is how many LEDs and register shifters you have available).

MathI am driving the LEDs at about 15mA (5V-1.8vDrop/220ohms=14.5mA). This means I can drive up to 33 columns with the mic2981 driver (500mA/15mA=33.3). Divided by 8 we can see that this allows us to string together 4 shift registers.

Also consider that you do not need to have all 32 columns stretch from left to right. You could instead create a 16x16 array that is wired the same way you would an 8x32 array. This would be addressed by shifting in 4 bytes.... the first two would shift all the way to the leds for the 9th row, the second two bytes would shift into the first row. Both rows would be sourced by one pin on the row driver.

Cascading Shift RegistersThe shift registers used are cascading shift register. This means that when you shift in data, the overflow appears on the Os pin. The becomes very useful as a set of shift registers can be connected to each other, Os pin to Data pin, adding 8 columns with each new chip.

All of the shift registers will connect to the same Latch, Clock, and Enable Output pins on the microcontroller. The "cascading" effect is created when the Os of the first shift register is connected to the Data pin of the second. The programming will need to be altered to reflect the increased number of columns. Both the buffer that stores the information and the function that shifts information in for each column need to be updated to reflect the actual number of columns.

A schematic of this is given below as an example.

Multiple Row DriversThe row driver (mic2981) can source enough current to drive 32 columns. What if you want more than 32 columns? It should be possible to use multiple row drivers without using more microcontroller pins.

We need the row drivers to source enough current to light the LEDs. If you are using more columns than it is possible to light at one time, addition row drivers can supply the needed current. The same input pins from the microcontroller are used so there is no need to alter the scanning of the rows. In other words, each driver controls the rows for an 8x32 block. Even though 64 columns may have the same PHYSICAL row placement, we divide the row buses in two, using one driver for the 8 rows of the first 32 columns, and a second driver for the 8 rows of the second 32 columns and so forth.

A schematic of this is given below as an example.

Potential Missteps:
1. Do not use multiple row drivers with the same number of columns. Doing so would mean that each shift register pin would be driving more than one LED at a time.
2. You must have a set of 8 resistors (3k3) for each row driver, one set for multiple row drivers will not work as it will not provide the necessary current to switch the gates.

For ExampleI decided to expand on the matrix I built earlier. I have added 7 more rows for a total of 15 as that's all I can fit on this protoboard.

I also just found out about a contest that Instructables is doing called "Let it Glow". Here is a video of my take on that. Once again, the digital camera I used to take the video doesn't do it justice. This looks great to the human eye, especially where all the LEDs flash, but doesn't look nearly as good in the video. Enjoy:

Source code for this larger display is included below.

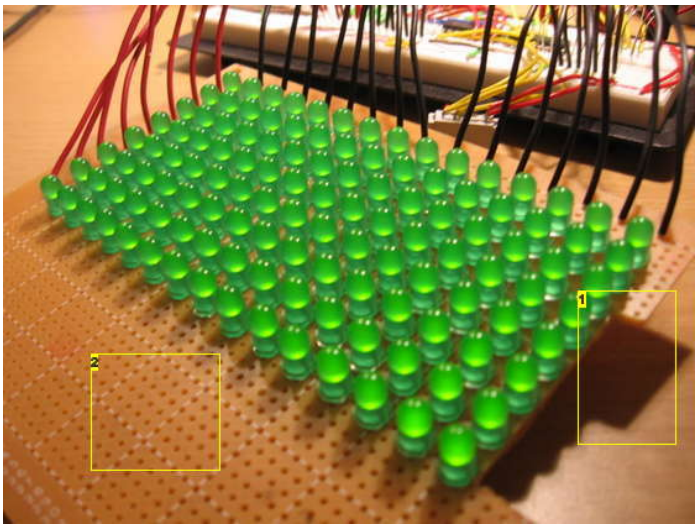


Image Notes
1. I've cut pieces of protoboard off of this end for other projects. I'm limited to 15 columns because of this.
2. I have space for 5 more rows in the future if I wish.... I think I'm just about out of LEDs though. There are 120 on the board right now.

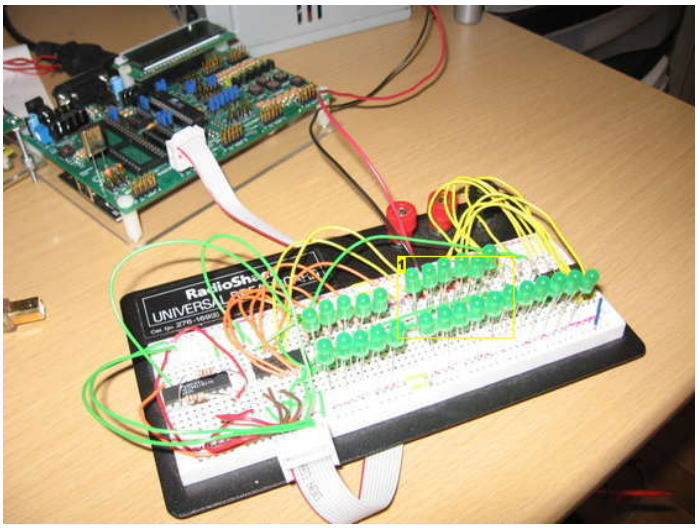
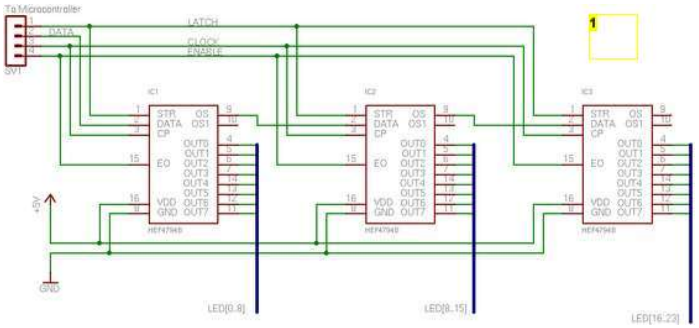


Image Notes
1. Prototype using just two rows but 16 columns. This is enough proof of concept for me to build a full size display.



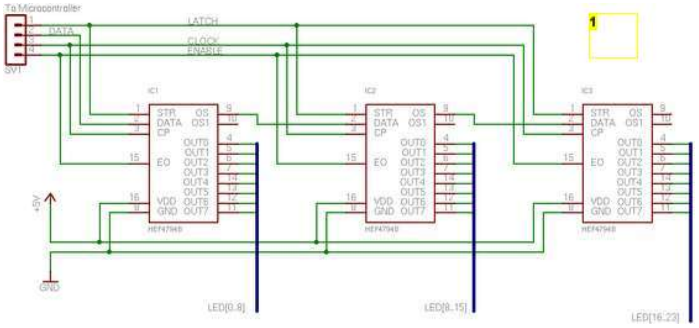


Image Notes
1. This is an example of how to hook up multiple shift registers in a cascading configuration.

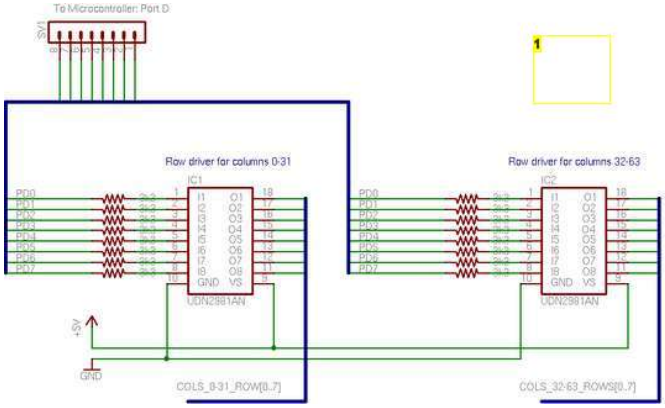


Image Notes
1. An example of hooking up multiple row drivers.

File Downloads

Let_It_Glo.zip (11 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'Let_It_Glo.zip']

step 6: Conclusion

Possible Additions**I2C**
I have left the Two Wire Interface (I2C) pins unused in this design. There are several interesting prospects that can use these two pins. Addition of an I2C EEPROM will allow for storage of much larger messages. There is also the prospect of designing programming to turn the mega8 into an I2C compatible display driver. This would open up the possibility of having a USB enable device to display data on your LED array by passing it over the I2C bus.

Input
There are many pins left over that could be used for buttons or an IR receiver. This would allow for messages to be programmed in via a menu system.

Display
For this instructable I only implemented a couple of display functions. One just writes characters to the display, the other scrolls characters onto the display. The important thing to remember is that what you see in the lights is represented in a data array. If you come up with cleaver ways to change the data array, the lights will change in the same way.

Some tantalizing opportunities include creating a graphing meter out of the columns. This could be used as a signal analyzer with a stereo. Scrolling can be implemented from the top down or bottom up, even left to right. Good luck, have fun!

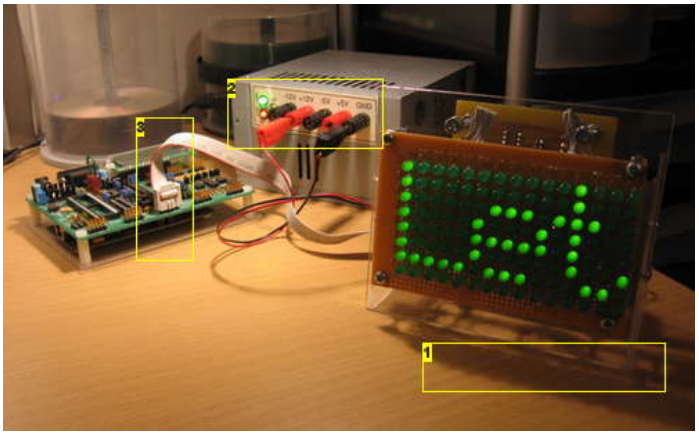
step 7: Follow Up

After letting the controller circuit sit in the breadboard for months I finally designed and etched a few circuit boards to put this prototype together. Everything worked out great, I don't think there's anything I would have done differently.

Circuit Board Features

- Shift registers are on separate boards that can be daisy chained together to increase the size of the display.
- Controller board has it's own power regulator so this can be run by any power source that provides 7v-30v (9v battery or 12v bench supply both work just fine for me).
- 6 pin ISP header included so the microcontroller can be reprogrammed without removing it from the board.
- 4-pin header available for future use of the I2C bus. This could be used for an eeprom to store more messages or even to make this a slave device controlled by another microcontroller (RSS ticker anyone?)
- 3 momentary push buttons are included in the design. I may tweak the firmware in the future to include the use of these buttons.

AssemblyGive me plexiglass, angle brackets, 6x32 machine screws, nuts, and washers, as well as a tap set to thread holes and I can create anything.



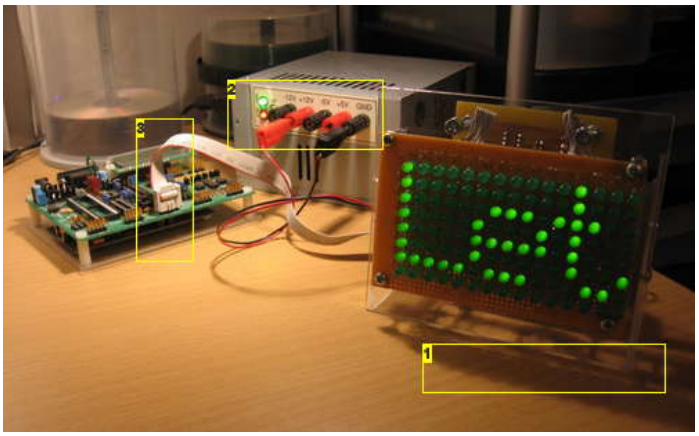


Image Notes
1. Assembled display stand on its own.
2. ATX power supply converted to a bench supply by use of an instructable. I'm putting out +12V and the 5V is being regulated on the controller board.
3. I'm using the ISP adapter I made for the Dragon Rider 500 to program the AVR on the matrix assembly. See my "Using the Dragon Rider 500" instructable for more info on this.

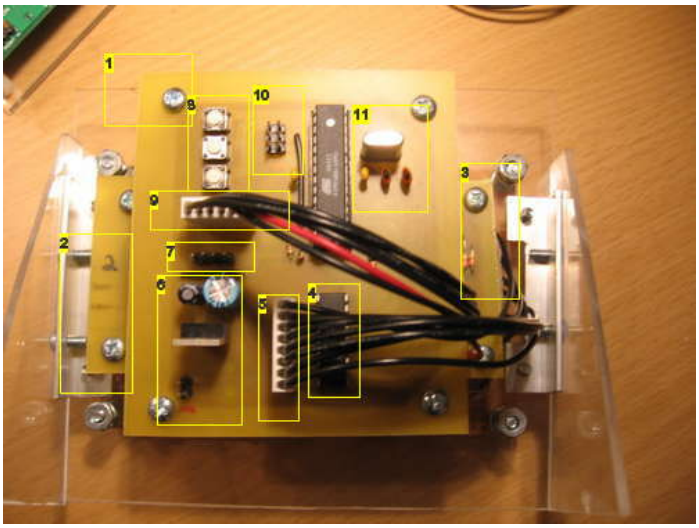


Image Notes
1. Controller Board
2. Shift register board #2
3. Shift Register Board #1
4. Row Driver
5. Connects to the rows of the LED Matrix
6. Power connetion and regulation. I didn't have a power jack on hand and insted used a 2 pin header.
7. I2C connection for future use.
8. Buttons for future use.
9. Connection to the first shift register board.
10. ISP Programming Header
11. Crystal and filtering capacitors.

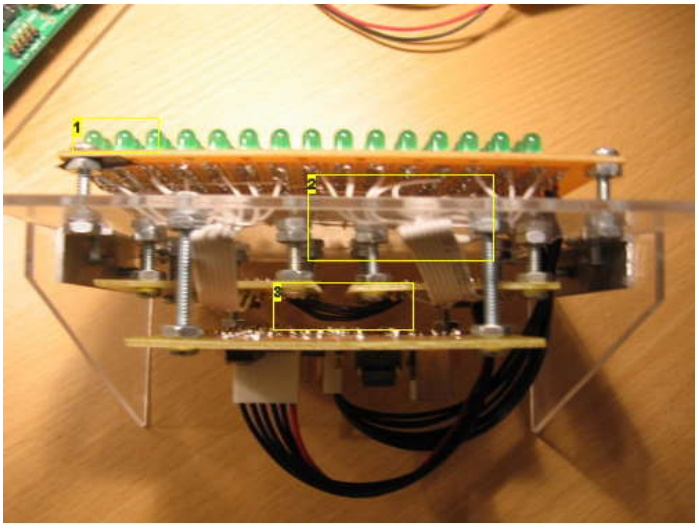
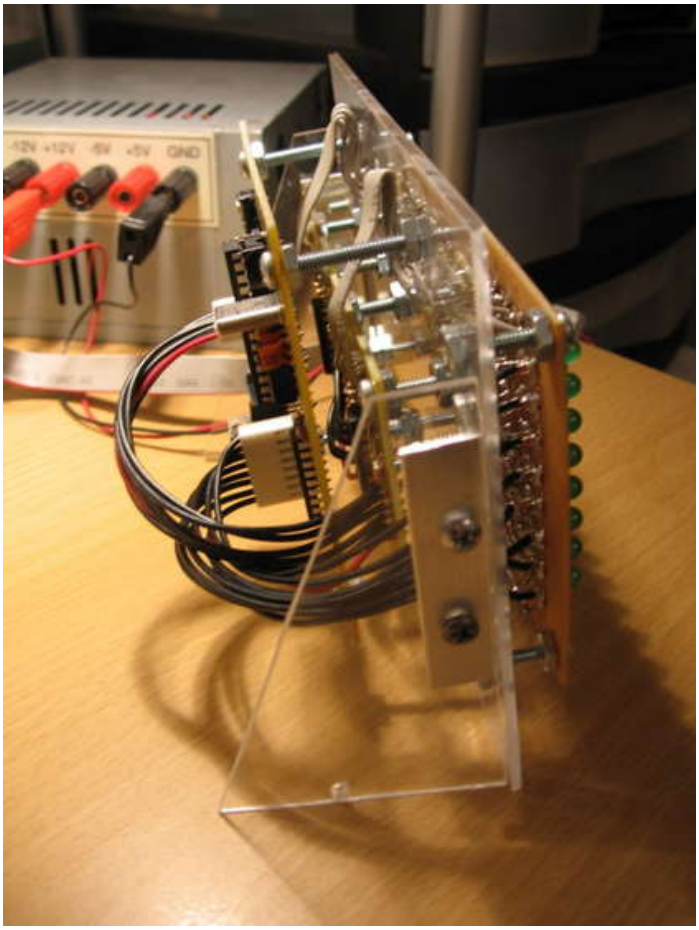


Image Notes
1. LEDs
2. Ribbon cable connecting the columns to the column driver boards.
3. Daisy chain connection (wires) between the two Column Driver boards.



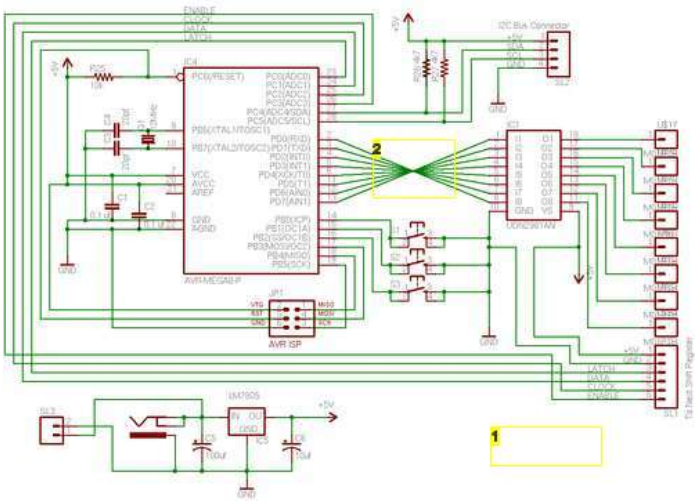


Image Notes
1. Controller Board Schematic
2. Row driver pins have been flipped from the original design to facilitate PCB layout. A simple firmware change flips which row is being scanned to match this change.

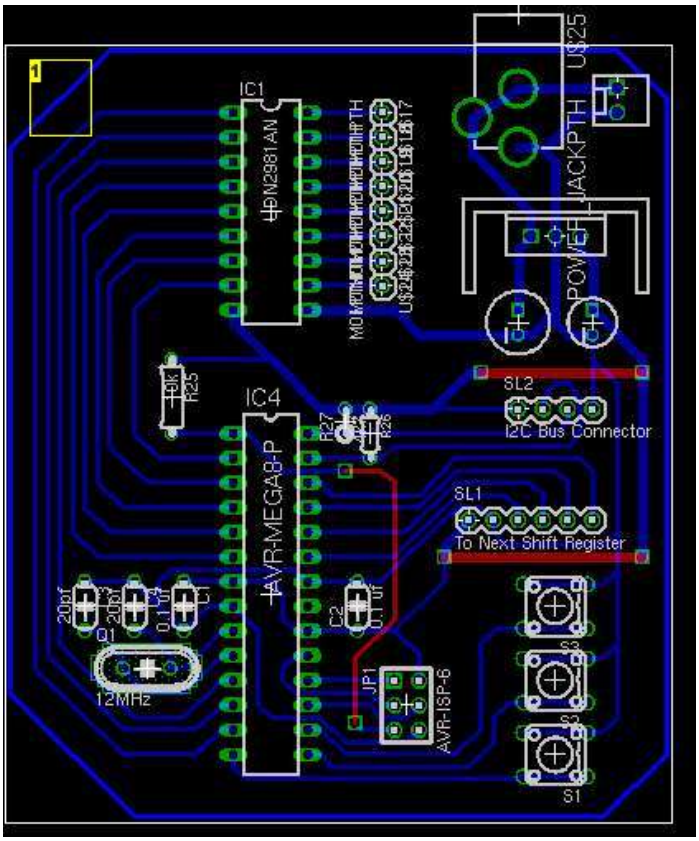


Image Notes
1. Controller board circuit layout using EagleCAD

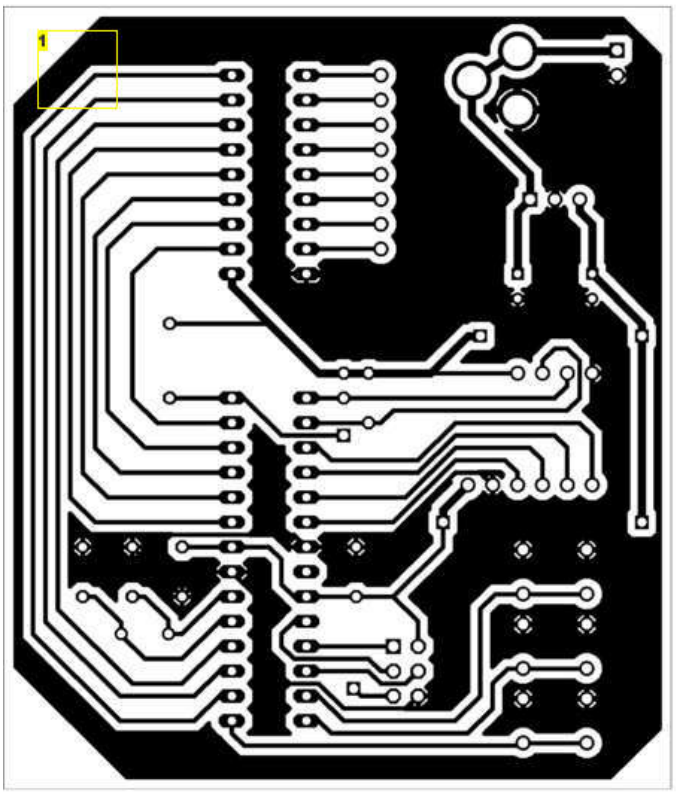


Image Notes
1. The image I used to print the resist onto glossy photo paper.

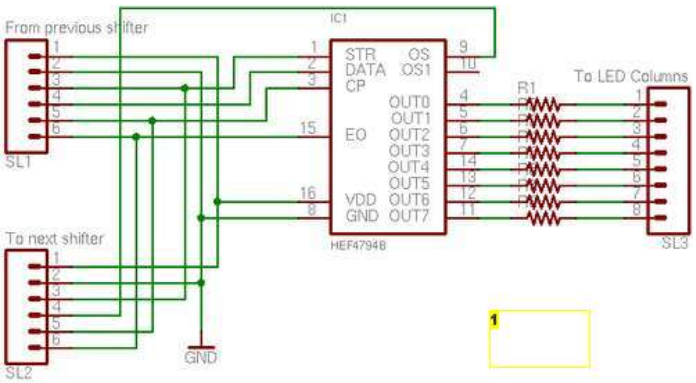


Image Notes
1. Shift Register board schematic. Notice that this has been set up to cascade so multiple boards can be daisy chained. Each board can add up to 8 more columns.

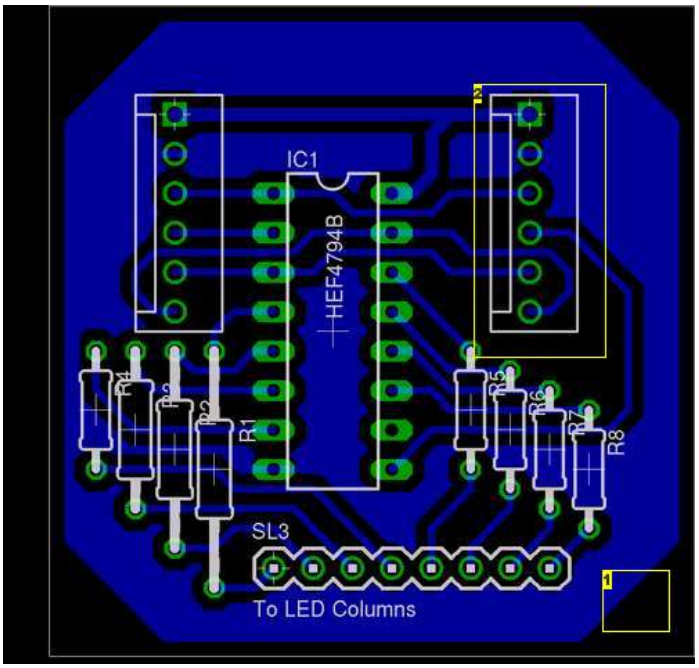


Image Notes
1. Shift Register PCB layout
2. These connections are not labelled but from top to bottom: +5V, GND, Latch, Data, Clock, Enable.

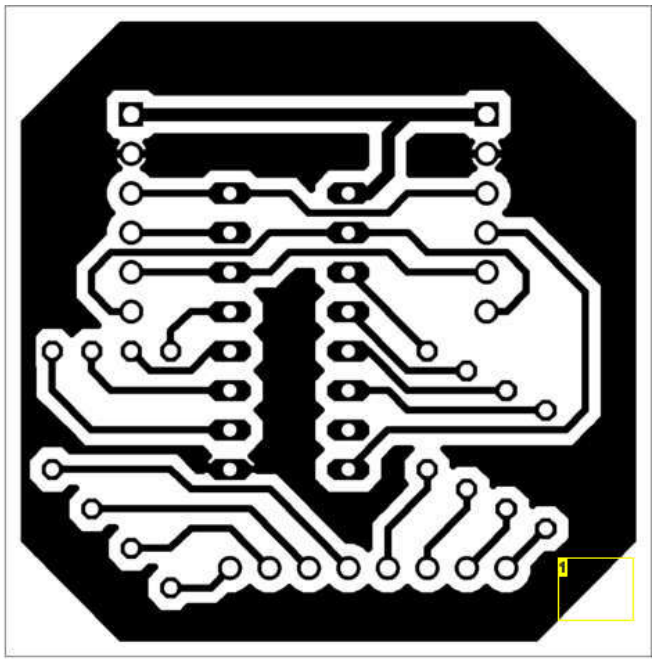


Image Notes
1. Image used to print the resist for PCB making.

File Downloads

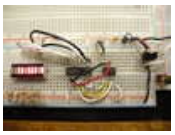
LED_Matrix_PCB_Proto_Firmware.zip (12 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'LED_Matrix_PCB_Proto_Firmware.zip']

Related Instructables

Make a 8x10 L.E.D Matrix by Syst3mX

Using a Dot Matrix LED with an Arduino and Shift Register by nevdull

watch futurama on an 8x8 pixel screen by sethj



How to use a 74HC595 Shift Register with a AVR ATtiny13 by roznerd

The BlokClok Concept - Arduino driven RGB Abstract Clock (video) by earthshine

How to build a 8x8x8 led cube (English version) by agofi

Comments

50 comments **Add Comment** **view all 116 comments**

ji394su3 says: Oct 5, 2010. 5:27 AM **REPLY**
good article!!

pavanbhushan says: Sep 3, 2010. 2:48 AM **REPLY**
what is the additional connection that should be made to make the LEDs glow individual letters???

pavanbhushan says: Sep 3, 2010. 2:47 AM **REPLY**
how can we make the 8*8 led display the letters???? I'm stuck at this point...

Suresh.D.M says: Aug 3, 2010. 12:45 AM **REPLY**
can we add shift registers to the column and shift the bit

pontikakis3 says: Jul 15, 2010. 2:30 AM **REPLY**
Where is the Font Set at http://en.radzio.dxp.pl/ks0108/ ?? Anthony

DanielIt says: Oct 9, 2009. 4:10 PM **REPLY**
Hey there..nice project,easy to acomplish and lot of stuff to learn.Now that i am learning C,i enlarged the project to be 8x32 columns,but i cant get the code to work.Any hints where are the parts that needs to be worked?There are some parts in the code commented that need to be altered,but i cant figure out how.If you can jelp pls send me an PM,or ust reply here.
Thank's and keep up the good work!!

dhimullai says: Jun 1, 2010. 4:57 AM **REPLY**
Hi;
I too tried with expanding 8x32 , it doesn't works as expected., Any one can help me,

knektek says: Apr 24, 2010. 11:54 AM **REPLY**
Could you use a whole load of transistors? For switching something else? Maybe an array of motors or switching relays?

greekwow says: Feb 28, 2010. 6:20 AM **REPLY**
does the leds have good brighness? there is not problem with duty cycle? becouse i have that problem in my project, (i use 4017 decade counter)

ndinitz says: Feb 21, 2010. 11:27 AM **REPLY**
can this be done using parallel load shift registers or does it have to be done with serial load.

if it is to be doable with parallel load registers then the board doesnt have to display only one column at a time, right? that would be so much cooler and you could build modules and connect them with parallel in serial out to serial in parallel out.

leme know what you think...

rammbo69 says: Jan 17, 2010. 5:51 AM **REPLY**
Both are 20 pF.

malth91 says: Sep 29, 2009. 1:25 AM **REPLY**
a very informative and interesting site

roketlights says: Sep 16, 2009. 2:32 AM **REPLY**
what are values for c2 and c3 ????? im relying entirely on this!! please? i got everything but this part and the programing part. please help!!!

jeff-o says: Aug 27, 2009. 8:46 PM **REPLY**
Just to clarify, could I program in a bunch of phrases (like, 30-40) and call up any one of them based on certain inputs? What is the limit on phrase length without any external memory?

roketlights says: Aug 25, 2009. 12:55 AM **REPLY**
i just bought a butt load of registers. im trying to make a 32 x 32 any pointers? on how to connect 8x8 in rows and columns? any help would be appreciated. :)

hype1 says: Jul 23, 2009. 5:06 AM **REPLY**
I am looking into your setup and I noticed this: Why do you use such a driver on the high-side when you are lighting one column at the time during multiplexing? Now you could source rows of 8 LED's at one time, but that isn't the case while multiplexing, right?

monkeydluffy says: May 26, 2009. 9:22 AM **REPLY**
bro barney, can u help me to get HEF4794? i need the HEF4794?? 9 of them to controll my Solar Display. In my country there's no store sell that shift register :(:(Can I get it from u?? I'll pay it later. I'm in Jakarta, Indonesia. Please help me master, this is for my graduation project. thank you very much :)

monkeydluffy says: Apr 13, 2009. 5:34 AM
(removed by author or community request)

merseyless says: Apr 15, 2009. 6:21 PM **REPLY**
u just posted your email on the internet! prepare for spam...

animaster says: May 21, 2009. 10:51 AM **REPLY**
he has misspelled his email ;) i think it has to be GMAIL

ReCreate says: Apr 23, 2009. 7:55 PM **REPLY**
Eh...ahaha!

Sagar Gondaliya says: Apr 23, 2009. 7:15 PM **REPLY**
its probably one he made for just this purpose.

ndegwa says: May 8, 2009. 6:02 AM **REPLY**
I need help programming the micro controller could you please help i would like my display to scroll the following NBI.....VOI.....MBS over and over and can also be made to stop at one of the letters shown.

uberdum05 says: Feb 14, 2009. 1:27 PM **REPLY**
Could you use a MAX232 to connect it to a computer?

EricTheRed16 says: Apr 30, 2009. 5:43 AM **REPLY**
Im trying to get mine working from the COM port, so I am using a MAX232 to convert the data from -3V - 3V to 0V - 5V, then feeding it into a UART (ST16C1450) to get my 8 outputs. I havent had much any luck yet with it yet, as I'm still learning as I go. Right now I'm waiting on new caps for the uart.

ndegwa says: Apr 26, 2009. 6:59 AM **REPLY**
Hi nice idea i was hoping to use this idea but enlarge the font characters where i could use four LEDs in place one LED is that possible

neuromonkey says: Apr 17, 2009. 12:47 PM **REPLY**
I have read and re-read the parts list, but I cannot find the spoon. Where is the spoon?

earthshine says: Apr 19, 2009. 8:47 AM **REPLY**
There is no spoon.

aj218 says: Feb 25, 2009. 10:08 PM **REPLY**
i m making a 16x80 display with 8051 (AT89S52) but as more and more characters scroll, flickering is there why?

agent says: Apr 18, 2009. 2:58 AM **REPLY**
It's not refreshing fast enough. With that size, it would take a lot longer to toggle the LEDs.

format_c says: Apr 3, 2009. 5:08 PM **REPLY**
ok lets chek it out

claudi0 says: Mar 13, 2009. 1:25 AM **REPLY**
Instead of using HEF4794, can I use 74HC595? The outputs of this IC will be connected to the inputs of ULN2003, and the columns of the led matrix to the outputs of ULN2003?

EricTheRed16 says: Mar 28, 2009. 9:07 PM **REPLY**
i use that same setup and it works fine for me.

claudioro says: Mar 29, 2009. 4:15 AM **REPLY**
ok. thanks. i will try.

EricTheRed16 says: Mar 28, 2009. 9:14 PM **REPLY**
I have a question about the resistors. If you put the 1 resistor per 8 leds, wouldnt it get dim when all those 8 leds are lit up at the same time? How do you keep the brightness consistant whether your lighting up just 1 or all 8 leds?

Mario1 says: Feb 26, 2009. 2:35 PM **REPLY**
Can someone make an instructable how to connect this thing to a computer??? Maybe throught the COM port? without those shift registers 'n' stuff.... just row & column wires maybe ??? :D

EricTheRed16 says: Mar 28, 2009. 9:10 PM **REPLY**
I'm working on a LED board to be controlled by the COM port right now, but you need the shift registers to control the LEDs. Otherwise for an 8x8 led board you'd need 64 outputs. In addition to the shift registers you'll also need a UART to talk to the COM port. When i get my first led board version working good i'm going to make a second one and I think it would be cool to do an instructable for it as well.

seamoon7 says: Jan 28, 2009. 8:59 AM **REPLY**
Can I use a AVR-PG2 to interface with my pc

NetReaper says: Jan 10, 2009. 12:33 PM **REPLY**
Simply put, I have no clue how to do this. You lost me after you finished just soldering all the leds together into the rows and columns.

Mario1 says: Jan 5, 2009. 11:29 AM **REPLY**
I like... totaly don't understand this :(If you put a voltage on let's say the 3rd column and 4-th row won't the whole column&row turn on ? not just the leds you need, but the whole 3 column & 4 row

greendude says: Jan 5, 2009. 8:07 PM **REPLY**
No because to light up, electrons need to flow through the LED, meaning they need somewhere to go from and somewhere to go to. If you put a voltage through the 3rd column and 4th row, the LEDs in the 3rd column will all have an electron source and the LEDs in the 4th row will all have a place for the electrons to go to, but only the LED that is in both the row and the column will have a source of electrons and somewhere for them to go (a sink). So only it would light up. I hope this helps, I'm terrible at explaining things :P

greendude says: Jan 5, 2009. 6:36 PM **REPLY**
Is there an easy way to alter the program to run on an arduino, or bootloaded atmega168? I wouldn't think there would be a timing problem since it runs at 16MHz, faster than 12. I'm poking around with the makefile now to see if it's possible and I'll report back. If someone knows how to do this easily I would be grateful if they could do it and post a reply :D Cheers

Tinvarien says: Nov 6, 2008. 10:39 AM **REPLY**
Thanks for the great Instructable! I am stuck here: Fuses: Make sure to program the fuses to use the 12MHz crystal hfuse: 0xC9 lfuse: 0xEF I did this and effectivly "bricked" my atmega8. I have search and scoured AVR Freaks but Dragon RIDER info seems slim and hard to come by. I tried the pp/hvsp mode but I cant seem to set the clock back to the default, I get a long message popup about ensuring the "board" page is within the proper range for high voltage programming... any ideas? I have 2 other atmega8 's that I bought "just in case" but untill I figure this out I dont want to chance "bricking" them as well. I did pop the others in just to check the dragon and the rider and they program and read signatures fine.

barney_1 says: Dec 27, 2008. 12:06 PM **REPLY**
CadaverBSE is correct.

You should also be able to put the chip in the DragonRider with a crystal in the crystal slot and "recover" it without using high voltage programming.

Check out my other instructable for more help:
http://www.instructables.com/id/How-to-use-the-Dragon-Rider-500-with-your-AVR-Drag/

CadaverBSE says: Dec 26, 2008. 9:24 AM **REPLY**
when you say you bricked you Mega8 ... one assumes that you already built the circuit (including at 12MHz crystal) before you tried to program it. It's not bricked, it's expecting a 12MHz input clock. You will need to connect the HV pins accordingly to reprogram it. Have you confirmed that your HV lines are connected per the Dragon (assuming you are using a Dragon) instruction in Studio 4?

greendude says: Dec 9, 2008. 2:01 AM **REPLY**
Hi, this instructable is great and I would love to do it, but I can't find any "nearby" shops that sell the HEF4794 part (it'd be like $30 shipping for $1 chip, stupid mouser), I also don't know enough about LED drivers and shift registers to choose a suitable replacement. Any chance you could suggest a different, more available part?

vandemark_p says: Dec 23, 2008. 3:51 PM **REPLY**

You might try Digikey
Good luck!HEF4794

barney_1 says: Dec 11, 2008. 6:55 AM **REPLY**

You should be able to use any shift register that meets your needs as far as: 1. How many pins you need 2. How much current you need to sink Read the datasheets and choose accordingly.

memuller says: Dec 10, 2008. 10:20 AM **REPLY**

Can your source code be used with an Atmel 89s52 micro? What should i do, analize the code and change the registers used by the interruptions? Thanks a lot!!!

barney_1 says: Dec 11, 2008. 6:54 AM **REPLY**

I believe you should be able to port this over. You are correct that you need to change the registers to match your pinout. You also need to make sure you have the correct vector names for the interrupts, these can be different between different chips. You also may (and I'm not sure here) have a problem with running out of ram as the mega8 has more than the 89s52 does

view all 116 comments