**What do you need?**

* Arduino Uno
* 1 x 8×8 LED matrix
* 2 x 74HC595 shift registers
* 8 x B562 PNP transistors (other PNPs are OK)
* 4 x pushbuttons
* 8 x 330 resistors
* 8 x 1k resistors
* 8 x 10k resistors
* And a lot of wires

**Schematic:**

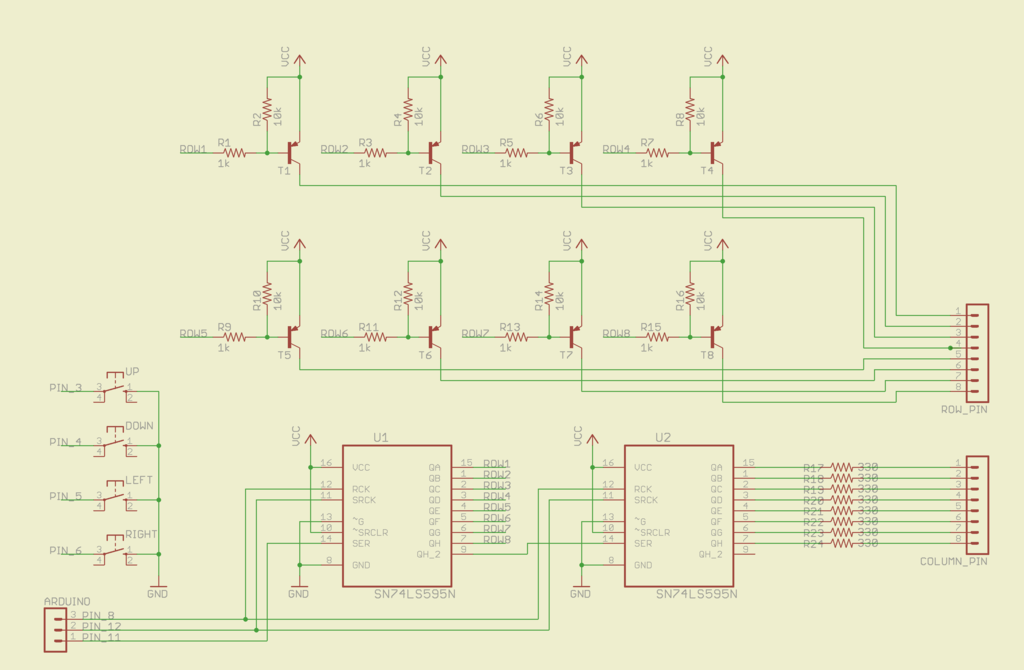
The first thing you should take care of is the pinout of LED matrix. There are various types of LED matrix in the market. The best way is to look at the datasheet of your LED matrix and determine which pins control rows and which pins control collumns.

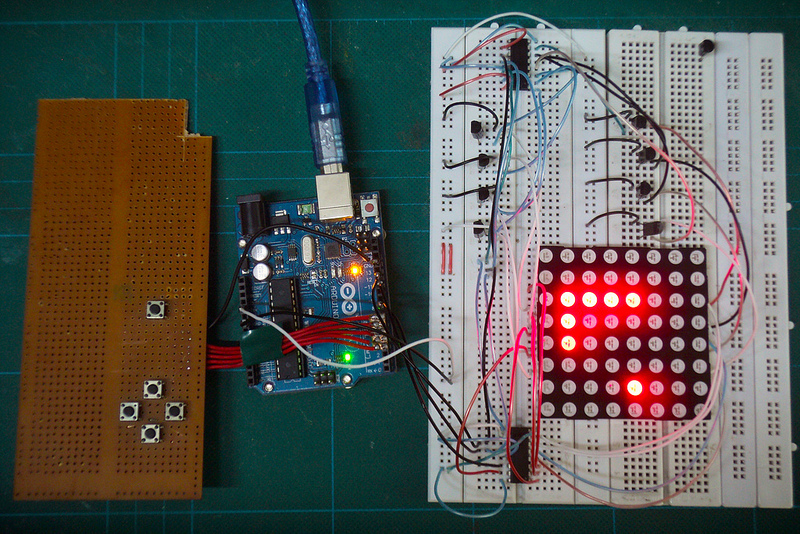
The first 74HC595 controls the rows of LED matrix. We connect its outputs to PNP transitors to ensure the current supply for LED matrix enough. The second 74HC595 controls the column of LED matrix.

**How does it work?**

We use 2 shift registers to control 8 rows and 8 columns of the LED matrix. With 74HC595, we only need 3 Arduino pins to drive 16 output pins, equivalent to 64 LEDs. What a save!

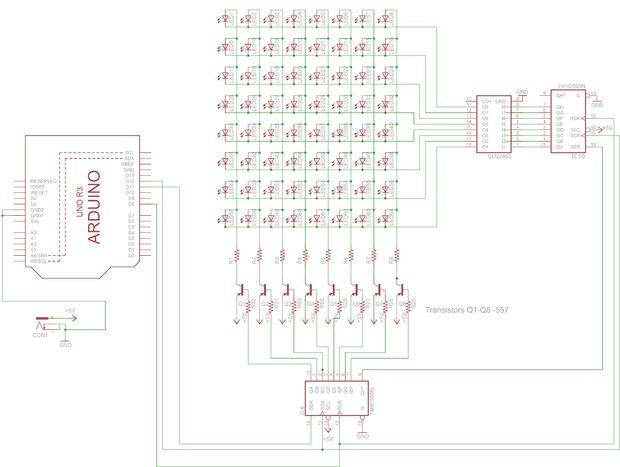
The display data in LED matrix is stores as a bitmap (array) unsigned char led[8] . To send data from Arduino to 74HC595, we use the following functions:





Dot matrix units typically come in either a 5x7 or 8x8 matrix of LEDs. The LEDs are wired in the matrix such that either the anode or cathode of each LED is common in each row. In other words, in a common anode LED dot matrix unit, each row of LEDs would have all of their anodes in that row wired together. The cathodes of the LEDs would all be wired together in each column. The reason for this will become apparent soon. A typical single color 8x8 dot matrix unit will have 16 pins, 8 for each row and 8 for each column. The reason the rows and columns are all wired together is to minimize the number of pins required. If this were not the case, a single color 8x8 dot matrix unit would need 65 pins, one for each LED and a common anode or cathode connector. By wiring the rows and columns together, only 16 pins are required.  
  
However, this now poses a problem if you want a particular LED to light in a certain position. If, for example, you had a common anode unit and wanted to light the LED at X, Y position 5, 3 (5th column, 3rd row), then you would apply a current to the 3rd Row and ground the 5th column pin. The LED in the 5th column and 3rd row would now light. Now let’s imagine that you want to also light the LED at column 3, row 6. So you apply a current to the 6th row and ground the 3rd column pin. The LED at column 3, row 6 now illuminates. But wait…the LEDs at column 3, row 6 and column 5, row 6 have also lit up. This is because you are applying power to row 3 and 6 and grounding columns 3 and 5. You can’t turn off the unwanted LEDs without turning off the ones you want on. It would appear that there is no way you can light just the two required LEDs with the rows and columns wired together as they are. The only way this would work would be to have a separate pinout for each LED, meaning the number of pins would jump from 16 to 65. A 65-pin dot matrix unit would be very hard to wire up and control because you’d need a microcontroller with at least 64 digital outputs. Is there a way to get around this problem? Yes there is, and it is called **multiplexing** (or muxing).   
**Multiplexing** is the technique of switching one row of the display on at a time. By selecting the column that contains the row that contains the LED that you want to be lit, and then turning the power to that row on (or the other way round for common cathode displays), the chosen LEDs in that row will illuminate. That row is then turned off and the next row is turned on, again with the appropriate columns chosen and the LEDs in the second row will now illuminate. Repeat with each row till you get to the bottom and then start again at the top. If this is done fast enough (at more than 100Hz, or 100 times per second) then the phenomenon of  
persistence of vision (where an afterimage remains on the retina for approx 1/25th of a second) will mean that the display will appear to be steady, even though each row is turned on and off in sequence. By using this technique, you get around the problem of displaying individual LEDs without the other LEDs in the same column or row also being lit. By scanning down the rows and illuminating the respective LEDs in each column of that row and doing this very fast (more than 100Hz) the human eye will perceive the image as steady and the image of the heart will be recognizable in the LED pattern. You are using this multiplexing technique in the Project's code. That’s how you’re to display the heart animation without also displaying extraneous LEDs.

## Step 3:

[](http://www.instructables.com/file/FQ9QFFJH54064CC/)

you have to calculate the value of the resistors you can use You should first get some specs on your LEDs, you should know their forward voltage and forward current, you can get this info from the datasheet. The circuit operates on 5V so your Source voltage is 5V which can be obtained from a 5v adapter Download the original file to see the schematics better.(press the "i" icon in the top left corner of the picture)

## Step 4: Making it Work

I have a made the program that displays sentence from the arduino serial monitor on the matrix, my code is very basic. I have made an android application to make the font for the display. Please visit the following page to install the app