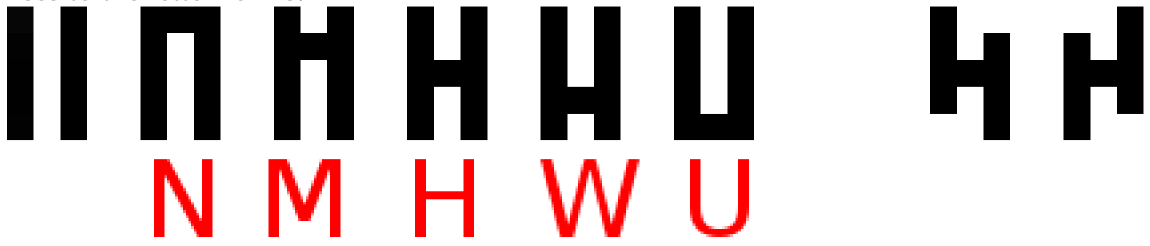
Escript
Electronic Script

Page 1-3 : Introduction
Skip to page 4 to get straight into the script

The “problem”

Escript is a constructed script aimed at electronic display, specifically optimized for LED pixel display.

Displaying text on an led matrix requires a relatively large resolution. For truly accurate characters the best is 5x5. It is possible to make smaller resolutions work, but it requires significant sacrifices to the letter forms.



In the image above you can see the 3x5 matrix alphabet just barely functions within reasonable limits. (the most overlapping letter form cluster NMHWU is shown)

It generally requires at least 5x5 to create reasonable letters.

Even with 3x5 spaces between characters are of course needed, so a word requires ((N*4)-1) x 5, where N is the number of letters in the word, and at bare minimum 2x5 for spaces between words.

Some letters won’t require the full 3x5. The letter “I” can be done using 1x5 for example.

It should also be noted that while a few decades ago this was a more significant issue, nowadays we have easy access to high resolution screens so the matter has become less important. Screen resolution has become so high that our text density limit is entirely determined by out vision, not the pixel count.

The availability of high resolution screens does not however make it a useless endeavor. Not only are larger LEDs, low resolution LED matrices and LED strips cheaper, they also require cheaper and less complex hardware. Driving a screen requires an complex system of chips and connections, the higher the resolution the harder it is to drive(Drive is the verb used to describe the electronic process of controlling screens and other hardware) .

Anyone who has experience in electronics, or even just playing with hobby dev boards like Arduinos, will quickly realize just how complex it is to control a large quantity of lights. In most cases a specialized controller module is used, it will take a simple serial data input and do all the complex signaling and control. Moreover the screens and matrices becomes quite complex, usually a method known as multiplexing is used to reduce the number of controlled output signals is used.

In short there are still advantages and savings to be had by reducing pixel count, particularly for hobbyists or when resources like processing power, size, or cost are determining factors.

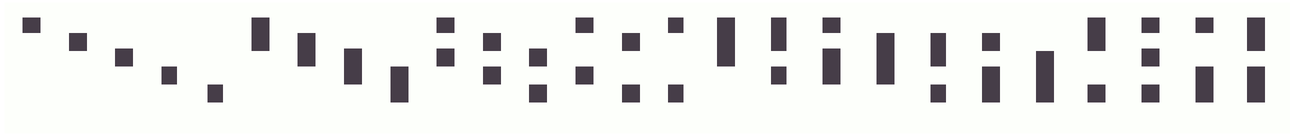
In the past, and still today at times, the solution is use single LEDs and print labels to identify the meaning of that LED, but it would greatly advantageous to able to display dynamic letters or text with a minimal amount of lights.

Solutions

Creating an entirely new script to represent the alphabet is far from a new idea. It is, however, definitely not an easily accepted solution for the simple and obvious reason… learning a new script is a rather large investment and not likely to be widely accepted.

The most well know recent attempt is perhaps Dotsies. http://dotsies.org/

Dotsies stacks 5 pixels vertically, each pixel represents a binary digit, allowing for a total of 32 characters (2^5=32).



In the image above are the letters A-Z used in Dotsies. As you can see they are not indexed in a binary counting method but still in a systematic way,

Dotsies is an interesting approach, using 5 bits is actually a common data compression method used to compressed text data. Using only 5 bits restricts you to 32 characters but it allows for 1.6 letters/byte compared to standard ascii at 1 letter/byte.

The downsides to Doties, in my opinion, are:

- **Very difficult to write by hand** - Writing Dotsies by hand is difficult and impractical.
- **High precision required** - because there are 5 vertical layers high precision is required for reading and writing. It would be very hard, for example, to distinguish a single A from a single B because you would need to count exactly how many blank pixels are in the vertical gap, this is a little easier when there are many letters together which can provide a spacial context, but this is still a task not well suited for human vision.
- **Character similarity –** Relying on precise pixel count and gap sizes is hard because human reading traditionally relies on shapes and relative position, not absolute position and size.

Dotsies is difficult to read, and even harder to write. It would make more sense as a script for computer vision and digital display, not humans. But lets be honest, if a human is not involved then why not just transmit raw digital data over a wire or wireless protocol?

If the end goal does not include humans there are QR codes, which include multiple encoding formats and redundancy (so even if the image is partially obstructed or corrupt the data can still be read in its entirety without error or loss). So anything not well suited for human use will be put aside for now. Computer to computer communication is a well developed field and a completely separate topic.

The Dotsies goal (as stated on their site) was to compress information on the screen. I feel it falls short on proving a practical solution because it is hard to read and learn. I say hard to learn because being poorly suited to hand writing makes it difficult to practice the script, I believe engaging with motor skills and having motions that can be used in associations dramatically helps, the more associations the better, and proactive writing is better than passive reading alone.

I have previously designed a script aimed at being easy for “both humans and computers”, Cscript. Cscript is designed to be both easy to read and write by hand, but also be easy to OCR and compress spatially.

Cscript PDF : http://dscript.org/cscript.pdf

Cscript could be used for pixel matrix display, but it was not optimized for that. Many Cscript letters would require 3x3 pixels each. Cscript could be optimized using colors, greyscale, or blinking, but Cscript was not optimized for this application so to achieve the best results I feel its best to design a new script optimized for this from the start.

There are many more similar attempts at optimized scripts, a large portion use the triple-trit like Cscript. A trit is like a bit but it has 3 values instead of 2, so triple trit(three trits) provides 3x3x3=27 characters, ideal for the English alphabet.

Here is an example triple-trit system suggested on reddit
https://www.reddit.com/r/neography/comments/4uhjp3/less_terrible_minimal_dot_script/

The goal was to improve upon the weaknesses of Dotsies. I think some significant improvements were made. Dropping the vertical stack from 5 to 3 helps greatly in my opinion because I have found it is reasonably easy to distinguish thirds (top middle bottom) compared to fifths, five is just too many levels to be practical for human vision. This system however does not make an attempt to make it human writable.

There are many other scripts that might be suitable for low resolution. Like the centuries old Pigeon cipher (https://en.wikipedia.org/wiki/Pigpen_cipher) or a recent derivative of pigeon cipher Elianscript (http://www.omniglot.com/conscripts/elianscript.htm), or one of the many constructed scripts out there, many archived at Omniglot (https://www.omniglot.com/conscripts/english.htm).

I have not, however, found any scripts simultaneously optimized for:

- low pixel resolution display
- human reading
- human hand writing

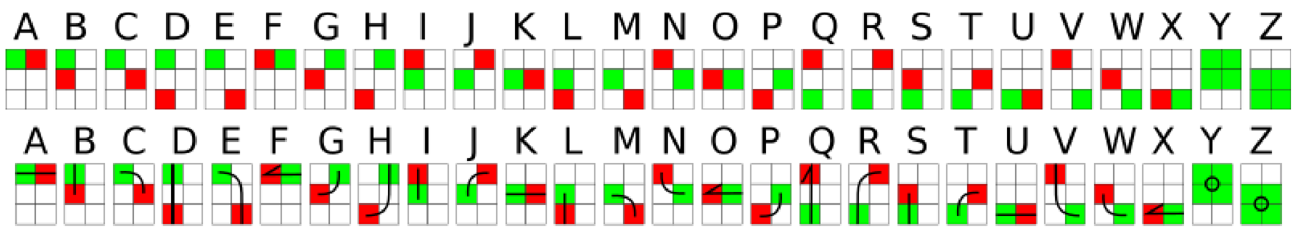
**** Many of the scripts mentioned were not designed with exactly the same goals in mind, so this is not a criticism of those scripts. I am coming from the perspective of these specific goals and evaluating them based on these criteria alone.**

Escript

Escript uses a 2x3 pixel grid. Each character is composed 2 different points, so there are 6x5=30 possible combinations. The 2 points are “start” and “end”, they represent a pen stroke.

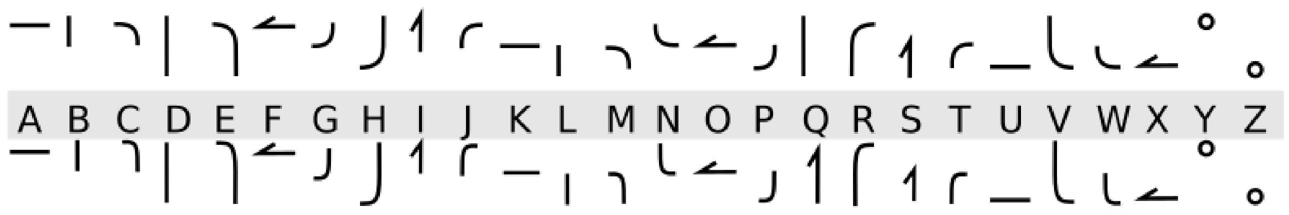
There would be several ways apply this “start” “end” 2x3 grid concept. I have tried to optimize it for human reading, human writing and pixel display.

Green : start   Red: end


A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

*Y and Z are special because it is actually (6*5)-6=24. 6 combinations are unused. Some vertical line combinations were removed to help the hand written form avoid ambiguity without overly complex letter forms or dramatic difference from the pixel form. Specifically vertical lines in the left column and vertical lines in the right column are impossible to distinguish without the grid for reference.

A start and end point are used so that they can also refer to pen strokes. Here is the written form.



First notice many letters only differ in swapping the start and end.

Rules (my arbitrary choices)

- Top → bottom
- Left → right
- Clockwise

If the start and end form a straight vertical or horizontal line, then the written form assumes top→bottom / left→ right.

If the direction of the letter is reverse, bottom → top or right → left then the line has a sharp arrow point to indicate direction.

If the start and end don’t form direct vertical or horizontal lines, then a curve or hook is drawn. The curve is clockwise (from “start” rotating clockwise to “end”)

Notice that between the letters G and H it skips the permutation of start-top-left → end-middle-left. This is for 2 reasons

1. In the written form it would not be distinct from the letter B
2. Skipping it allows B to be drawn with only 1x3 pixels

Escript Forms

Above you have already seen a basic written form of Escript and one possible way of display Escript with with 2 color LED pixels.

There are several other defined electronic display methods and more intricacies to hand writing.

Electronic Display

Escript was designed with several option for electronic display in mind.

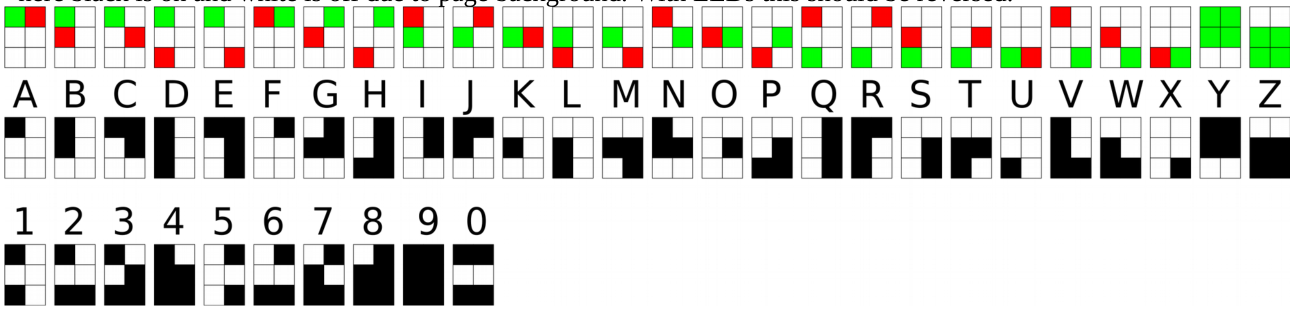
3 pixel tall matrix Display Can be done in

- Black / White (on / off)
- Black / White / Grey (on / off / 50% on)
- RGB (Color using 3 color channels)

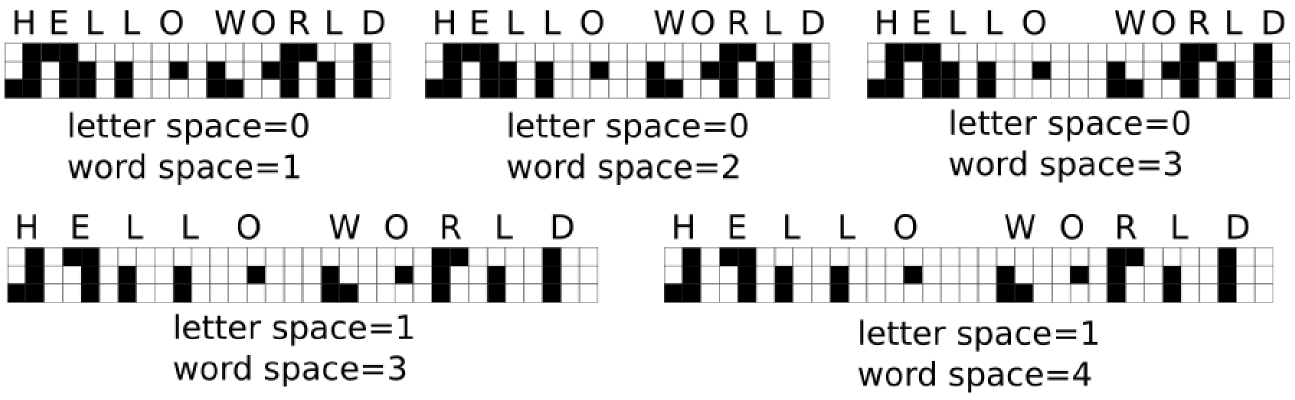
Black / White Matrix

Legibility : **LOW**

*here black is on and white is off due to page background. With LEDs this should be reversed.



A 2x3 matrix of LEDs, of any color, can be used to display a letter. While it would be possible to place each letter directly next to each other it would be far more difficult to read so I advise placing a spacer column between each letter. This of course means each word requires at least 2 columns of space as a "space character".



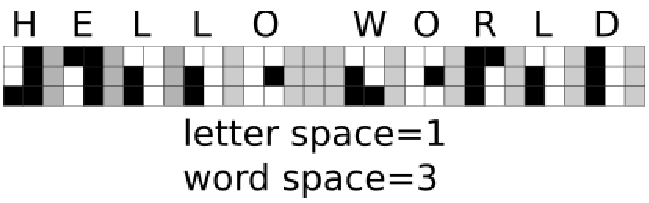
As you can see in the image above it is not very easy to read with only black and white pixels. The problem is primarily due to the fact that there can be empty space in either column. This means without any spacing there can be 2 empty columns (eg. L-O letter space=0).

It is hard to count pixel spaces, and not very suitable for human reading so I recommend if on/off must be used then the word spacer should be at bare minimum 2+letters space+1.

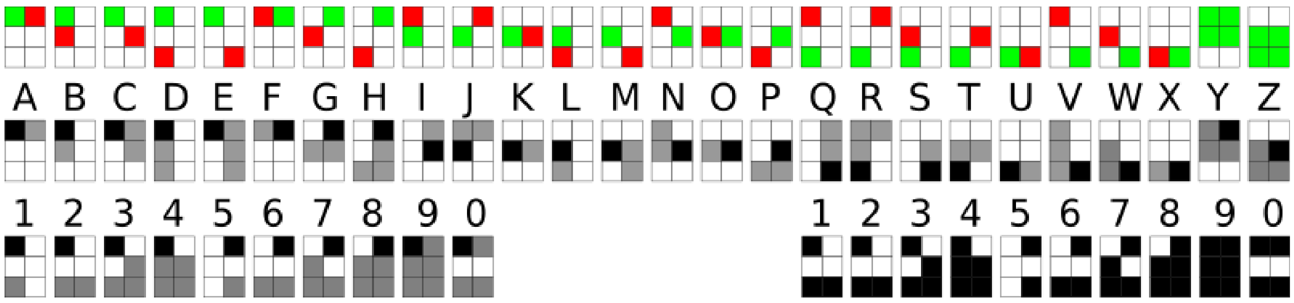
Black / White / Grey Matrix   Legibility : **MEDIUM**

*here black is on and white is off due to page background. With LEDs this should be reversed.

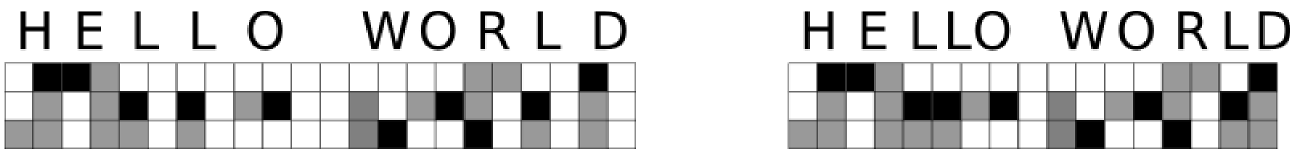
The first way to use a third color (half intensity on) is to use it for a spacing column in the black / white version



Using the third color in the letter allows the letter itself to be a bit easier to associate with the pen stroke. Full intensity can indicate where the pen stroke begins. Improving the association with the pen stroke motion allows it to be more easily read once the written form has been used.



Using a full intensity pixel to indicate the pen stroke start point and gray to indicate the the rest of the stroke allows the gap column between letters to be removed, and it also allows any empty columns within letters like B,D,I,L,Q,S, and X to be removed as well.

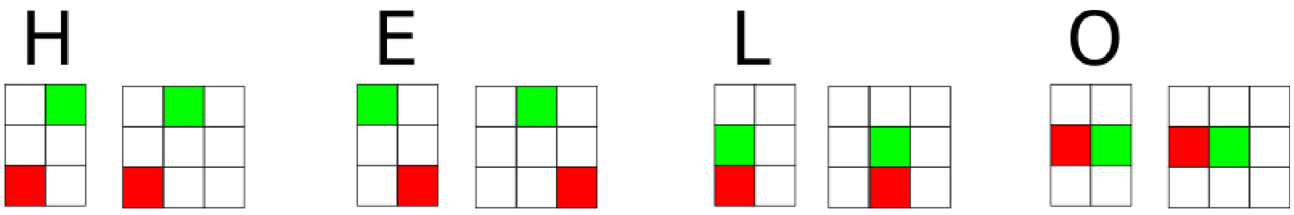


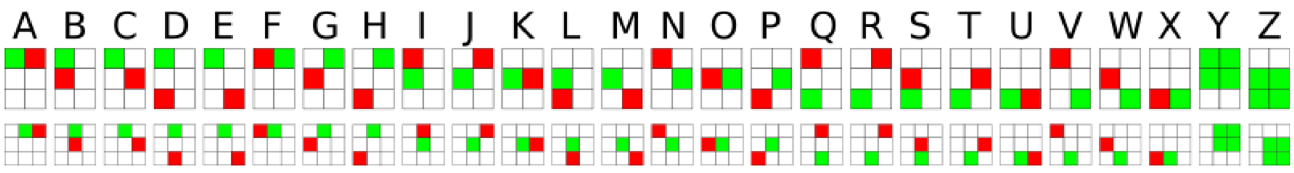
Reading with the letter space column and empty columns removed is quite simple:

1. Search the first column for a black start pixel (full intensity). If none is found check the next cloumn.
2. Follow the gray line(half intensity), if the previous column was skipped then follow it to the left.
3. Move to the next column and repeat

RGB Matrix   Legibility : **BEST**

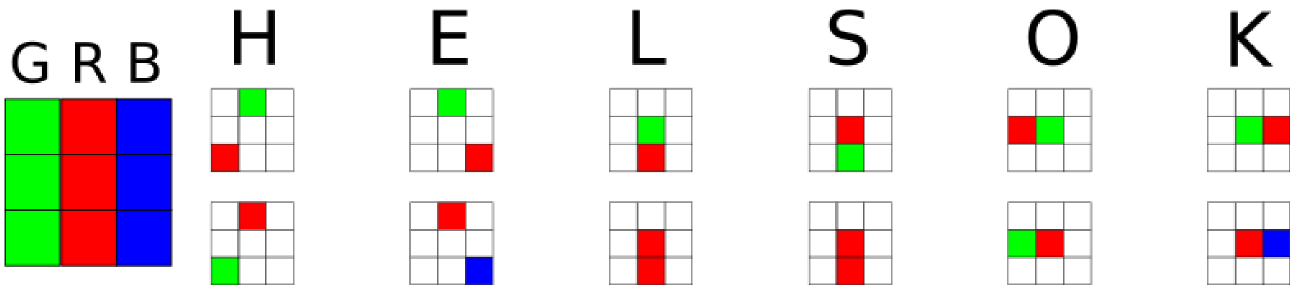
In order to build an RGB matrix the first step is to consider the starting pixel as being in the middle column of a 3x3 matrix.



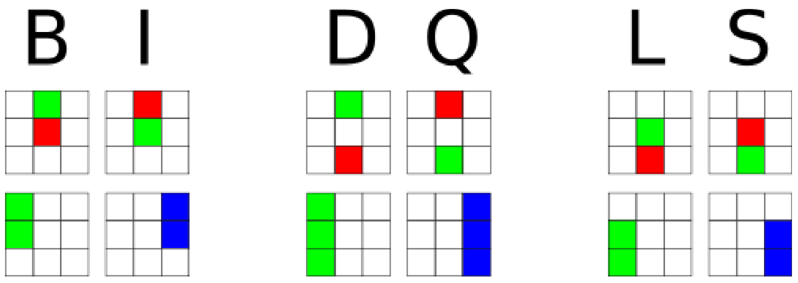


After increasing the grid to 3x3 the columns are coded with RGB colors.

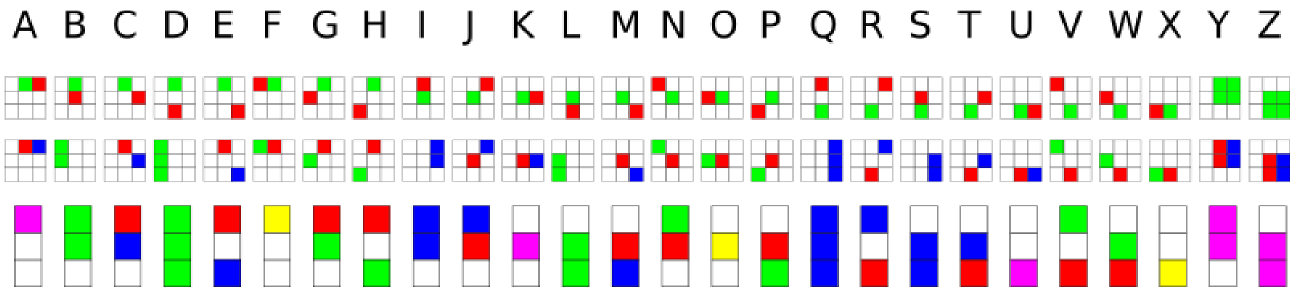
Originally the “start” pixel was green and the “end” pixel was red. This was just to match the simple “green for go, red for stop”. Here you will notice that red has been assigned to the center column. The reason for this will be explained shortly.



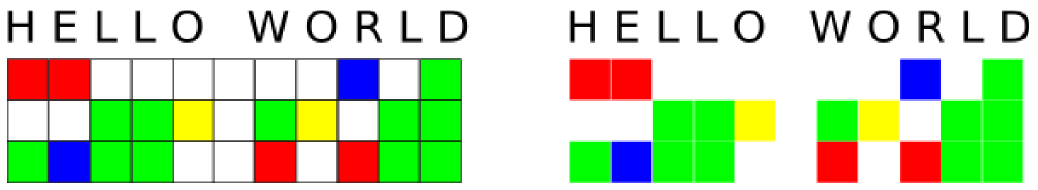
You will notice that this method makes the vertical down lines and vertical up lines identical. Above you can see that L and S become identical. This is solved by shifting them to the left or right. Top → bottom goes to the left (green), bottom → top goes to the right (blue).



Now we have a full 3x3 matrix of distinct characters. The final step is to merge the color columns.



This results in a 1x3 alphabet, so using RGB allows each letter to require only 3 vertical pixels.

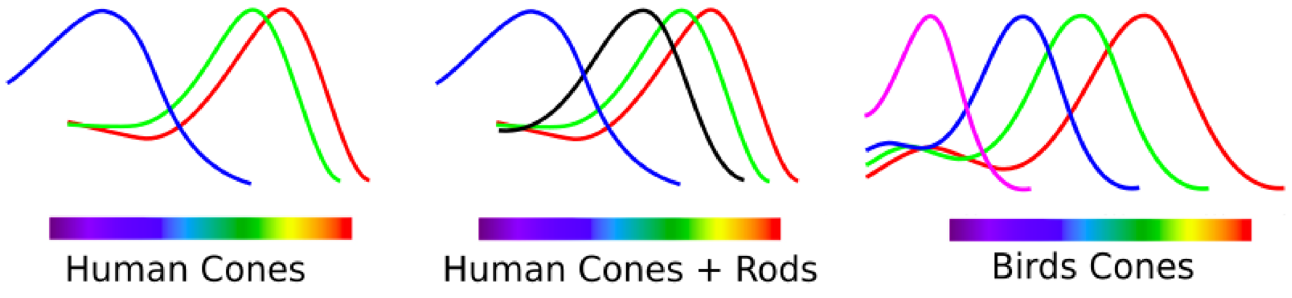


Learning to read this form is done with some simple rules:

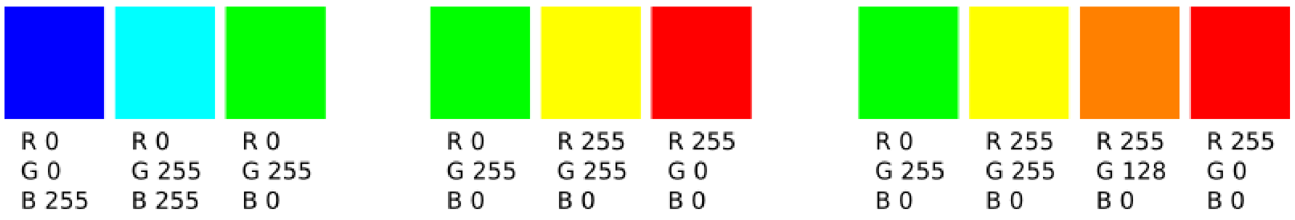
1. Search the column for red (red, purple or yellow)
    - If no red is found:
      Search for a green or blue line (green is an upwards line, blue is downwards)
2. find the second color : green(yellow) = left blue(purple) = right

Why GRB not RGB?

You may be wondering why red was put in the in center instead of preserving the RGB sequence. This was done due to the spectrum of human vision.



In the image above you can see a major gap between green and blue cones. This gap makes it difficult to identify colors between blue and green.



R 0 G 0 B 255	R 0 G 255 B 255	R 0 G 255 B 0

R 0 G 255 B 0	R 255 G 255 B 0	R 255 G 0 B 0

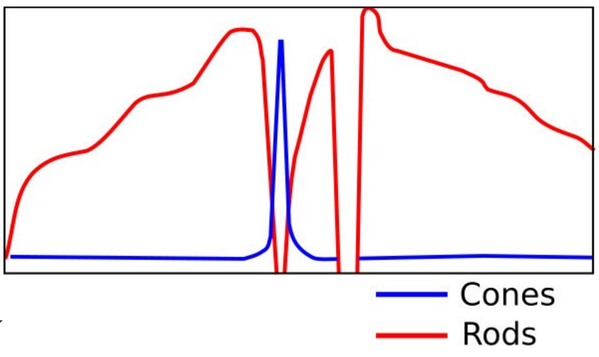
R 0 G 255 B 0	R 255 G 255 B 0	R 255 G 128 B 0	R 255 G 0 B 0

Between green and blue the color is what we commonly call “light blue”. Even with more green, I find my mind often jumps to guessing green or blue. Color experts may argue that I am just not well enough versed in colors to identify them instinctively.

Red and green, however, have not 1 but 2 obvious and distinct colors that even children can all identify.

Birds have more evenly spaced RGB cones so perhaps they would see only one clearly distinct red/green color but find a clearly distinct color between blue and green. They also have tetrachromacy (4 types of cones) with a 4th cone in the ultraviolet spectrum, which would probably be super cool and allow for way more options but we are stuck with just the 3.

You may notice that our rods do react most between the range of green and blue. But this does not help with color perception as one might hope. It does imply our black and white vision is biased towards blue/green colors, color vision(cones) is done with the center of the retina while black and white vision(rods) is done with peripheral vision.



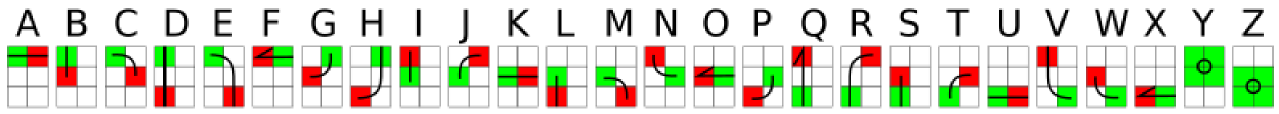
On the right you see the distribution of rods and cones in the human eye. Cones (colors) are almost exclusively in the center.

This is why the rods don’t help contribute to our color perception as one might hope.

*I use the more common wavelength sorted spectrum graph (BGR left to right) not the frequency sorted spectrum graph (RGB), that’s why green means left and blue means right in the RGB Escript.

Written Escript

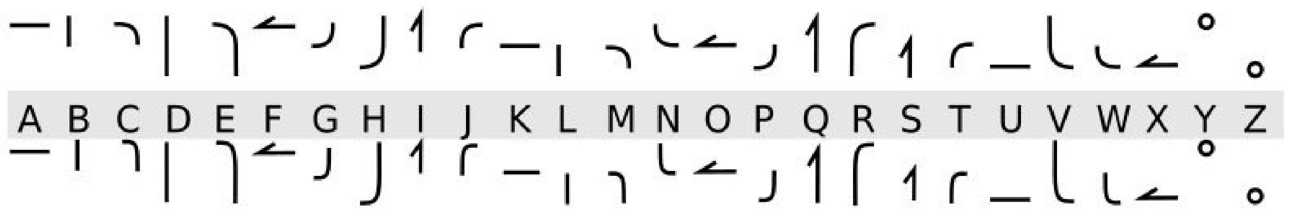
Hand written Escript has a basic form which also allows for some combination and compression.



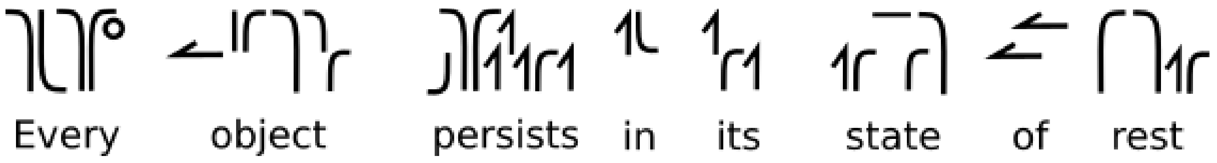
The image above shows how the 2 point code is turned into a pen stroke. The pen starts at on the green pixel and ends on the red pixel. There are simple rules to generate unique strokes for each letter

LINES : If the to pixels are in the same row or column it forms a line. If the line is **top** → **bottom** or **left**→ **right** it is a just a line. If the line is **bottom** → **top** or **right** → **left** then it is drawn with an arrow indicating the direction.

CURVES : if the two pixels do not form a vertical or horizontal line then it is a curve. The curve rotates clockwise from the start to the end.

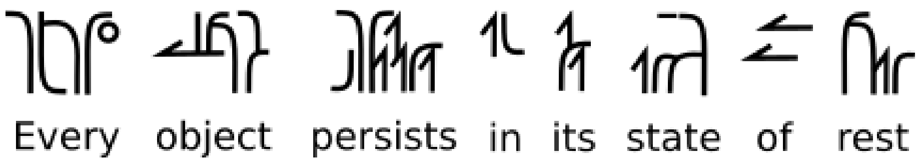


I prefer to compress the horizontal portion of the curved letters to save space. In the above picture you can see in the bottom version of the alphabet curved letters are slightly taller than they are wider. I do this because it saves a bit of space and because I feel it looks better.



Above you see the first few words of Newton’s First Law of motion in basic written form Escript.

Next it is possible to compress the hand written form by combining letters in ways that still keep it legible and unambiguous. Not all letters can be combined, some letters can be combined in multiple ways, and there are of course other possible ways of combining letters that I have not demonstrated.



I chose to demonstrate these combination methods because I find them to be the simplest. Escript was not designed and optimized towards this type of compression and combination. If you are interested in this type of compression please see Cscript (http://www.dscript.org/cscript.pdf)

The main goal of the written form is as a learning aide and association tool for the digital display version, learning these pen strokes allows you to see the digital version and quickly associate the pixels to physical movements.

Application

Applying the digital display version can be done in several ways.

1. Letter pulsing
2. Matrix strip

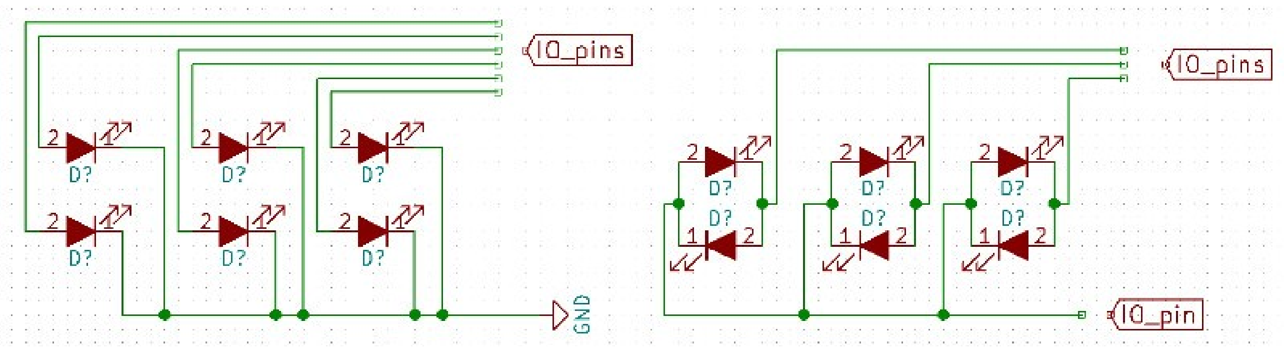
Letter pulsing

Many circuit builds and hobby dev board projects may want to control simple LEDs with minimal pin usage. By creating a matrix capable of displaying one letter at a time IO pin usage can be minimized. Single letter codes could be used as a abbreviations, and flashing pulsed letters can be used for strings of letters and words.

There are several wiring methods, depending on the matrix you choose to use.

2x3 black and white or 2x3 black / white / gray

There are 2 main ways to wire up 6 leds and control them with a simple MCU (like Arduino)



On the left you just connect each LED to an IO pin and have a shared ground (or voltage source)

On the right you can conserve 2 IO pins by multiplexing.

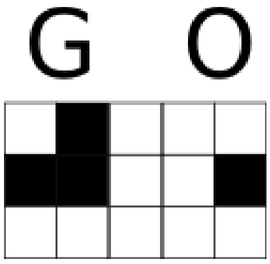
Further multiplexing could reduce it to 3 IO pins but that would require a complicated array of transistors/logic gates. Better would be to just add shift registers to the circuit.

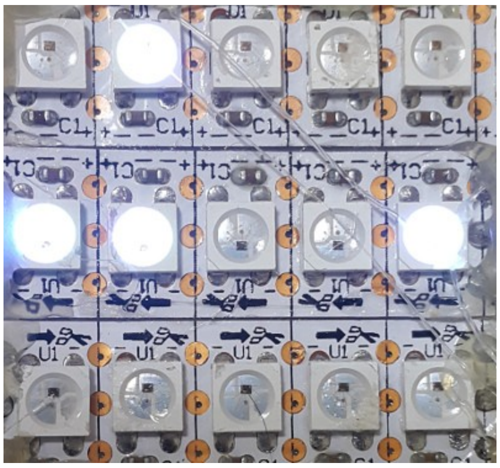
The downside to multiplexing is that it would require continuous driving, high speed switching is necessary to hold all 6 the LEDs in a state (except for a few letters that are only on one side of the matrix of course).

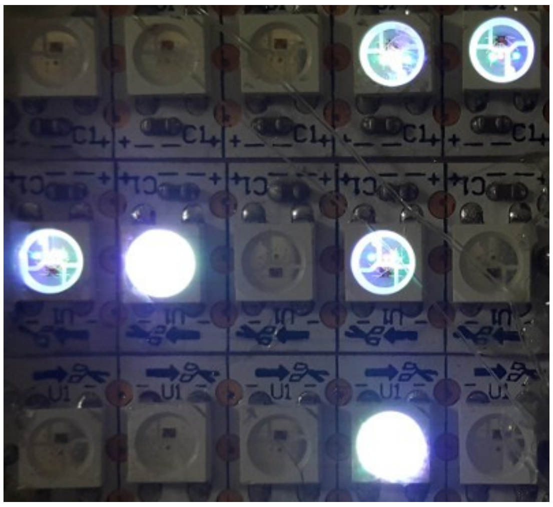
Continuous driving is already necessary for black/white/grey but that can be accomplished with PWM if you have access to 6 PWM channels. If there are not 6 PWM channels available then continuous driving must be used, most Arduinos have exactly 6 so if you need one for something else then this is not an option. When black/white/gray and continuous driving is used then it would not be much work to save a couple IO pins and multiplex with 4 pins, this would however limit you to 50% duty cycle. ****ok, ok.. no more electrical engineering.**

Letter pulsing only makes sense when you resort to super cheap layouts like this, once you upgrade to addressable LEDs or LED driver chip based display the number of controllable LEDs become only limited by price and circuit complexity. I had some addressable LED strips on hand so I built my demo with this.

On the right you can see the black and white version with the word GO written (with single column letter space)

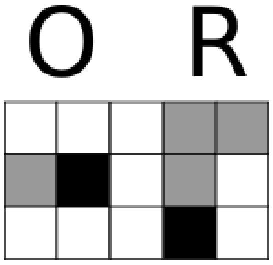






On the left there is the word OR with single column gap using the black / white / gray.

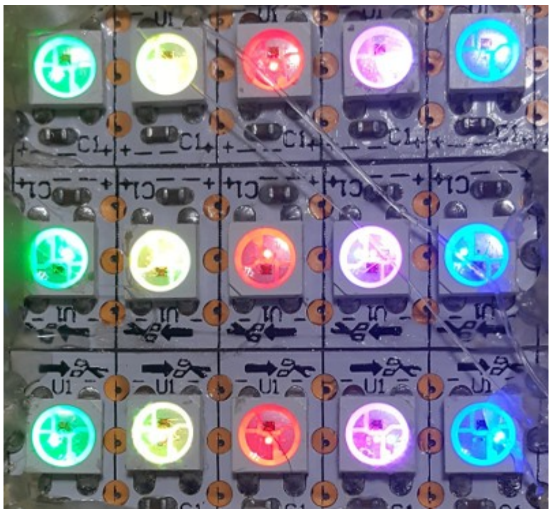
It is quite hard to photograph light intensity well but it is much easier to tell the difference between the light intensity in real life.



2x3 RGB / 1x3 RGB

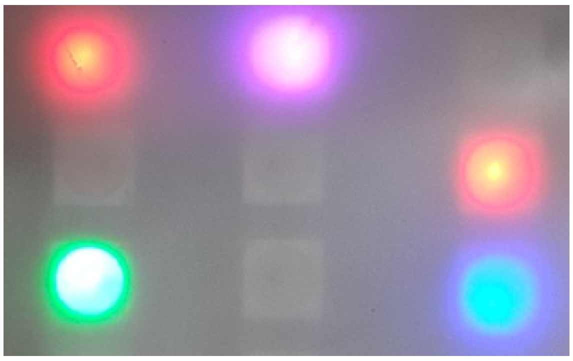
For RGB based display the number of LEDs (3 per pixel) is increased enough that it is best to upgrade to addressable LEDs. For this I use the ever so popular WS2812.
WS2812 LEDs allows for a large number of RGB LEDs to be controlled with a single IO pin.

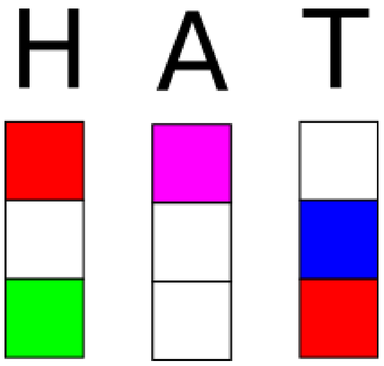
On the right you see a simple 5x3 grid of WS2812 LEDs I threw together. I cut 3 segments of 144 pixel per meter WS2812 LED strip, connected them in series and used an IO pin on an Arduino to drive them.



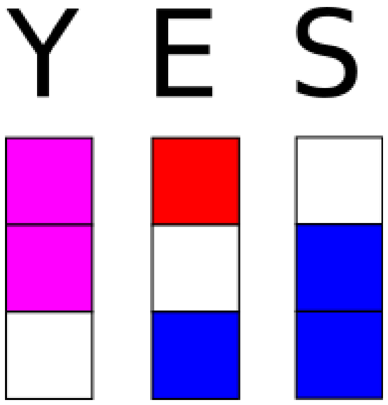
This grid can be sent a stream of values and it will hold that state until it is updated.

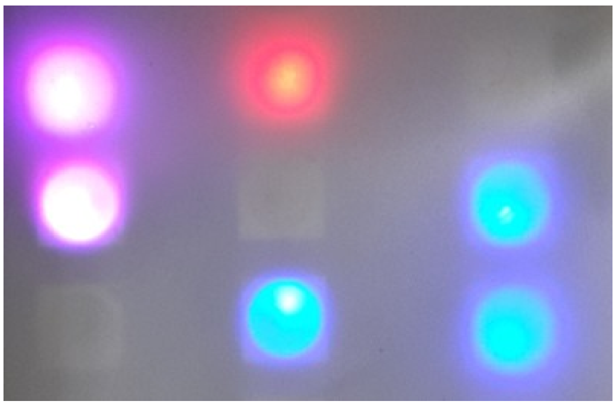
It requires only a single send of data, unlike multiplexing which would require constant driving to flip IO pins on and off at high speed just to hold a single set of values on the matrix. (A driver chip or shift registers also accomplish the same result)



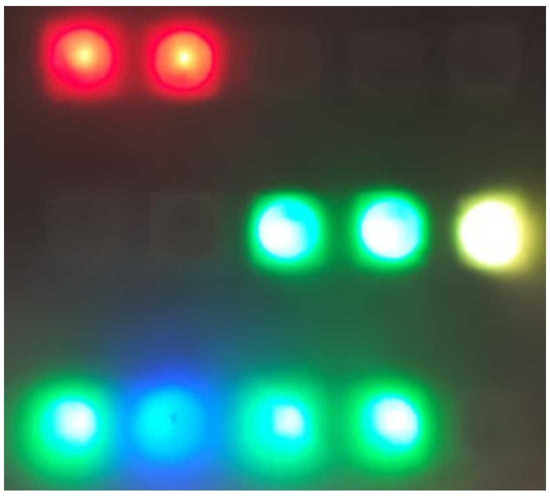


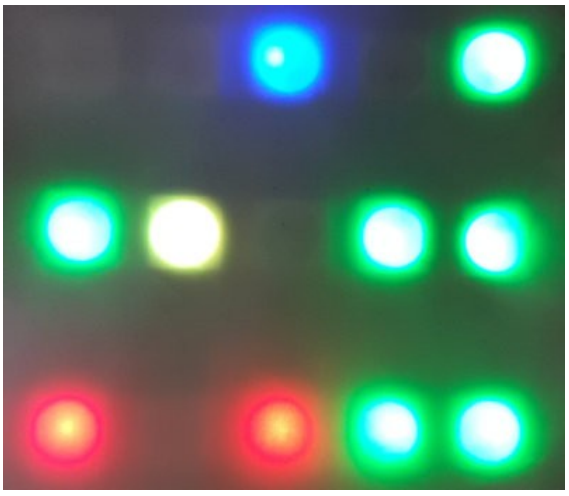
Above and below you see the words HAT and YES in 1x3 color matrix. The LED strips I used are more densely packed in one axis than another, I will use both possible alignments in this document as I have not decided which I prefer or if either has a legibility advantage. I also added LED film on top at this point to improve visibility in photographs.

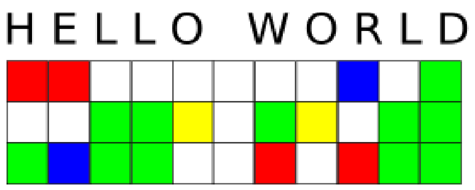


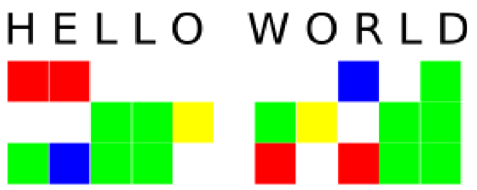


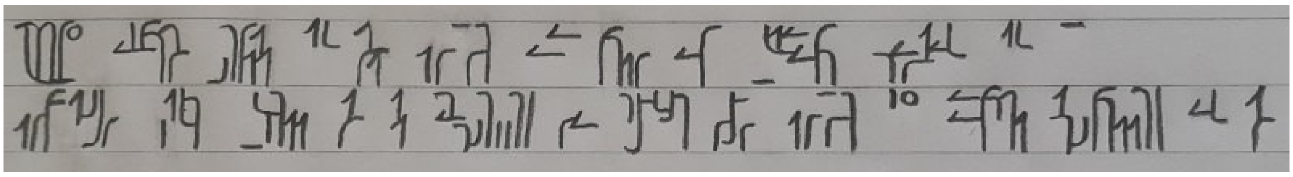
The demo matrix I built is 5x3 so rotation 90 degrees allows 5 letters.


HELLO


WORLD







Every object persists in its state of rest or uniform motion in a
straight line unless it is compelled to change that state by forces impressed on it
--Newton’s first law of motion