image2cpp

image2cpp is a simple tool to change images into byte arrays (or your array back into an image) for use with Arduino and (monochrome) displays such as OLEDs. It was originally made to work with the Adafruit OLED library. An example sketch for Arduino and this library can be found here.

More info (and credits) can be found in the Github repository. This is also where you can report any issues you might come across.

This tool also works offline. Simply save this page to your computer and open the file in your browser.

1. Select image

Sélect. fichiers

or

1. Paste byte array

128 x 64 px

Read as horizontal Read as vertical

2. Image Settings

Canvas size(s): Home-Server-icon.png (file resolution: 128 x 128)
128 x 64 glyph remove

Background color: ● White ○ Black ○ Transparent

Invert image colors ☐

Brightness / alpha threshold: 128

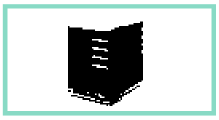
0 - 255; if the brightness of a pixel is above the given level the pixel becomes white, otherwise they become black. When using alpha, opaque and transparent are used instead.

Scaling scale to fit, keeping proportions ▼

Center: ☑ horizontally ☐ vertically

Note: centering the image only works when using a canvas larger than the original image.

3. Preview



4. Output

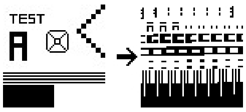
Code output format Arduino code, single bitmap ▼

Adds some extra Arduino code around the output for easy copy-paste. If multiple images are loaded, generates a single byte array.

Identifier: myBitmap

Draw mode: Vertical - 1 bit per pixel ▼

If your image looks all messed up on your display, like the image below, try using a different mode.



Generate code

```
const unsigned char NaN [] PROGMEM = {
	// 'Home-Server-icon, 128x64px
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
	0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x07, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x1f,
	0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
```