Roteiro de funcionamento Display L1-12864A-AZ 128x64 com Backlight Azul, segundo o teste efetuado

O Display LCD Gráfico L1-12864A-AZ 128x64 é um excelente display voltado para implementações gráficas, que necessitam de uma exibição personalizada com maior riqueza de detalhes, possibilitando desde a exibição dos simples caracteres alfanuméricos, até a exibição de imagens em estilo monocromático.

Essa versão 128x64 possui 8192 pixels distribuídos em 128 linhas e 64 colunas, proporcionando um alto desempenho, podendo ser utilizado tanto em simples aplicações como em aplicações industriais.

Ele atua em modo STN, e possui como controlador o SBN0064G ou equivalentes, sendo o L1-12864A-AZ, controlado por um KS0107 e dois KS0108. A interface de comunicação utilizada por esse display é do tipo 6800 MPU Serial de 8 Bits, e por possuir luz de fundo permite uma fácil visualização dos caracteres.

Descritivo da Pinagem do Display L1-12864A-AZ

Pino	Símbolo	Nível	Função
1	VSS	0V	Terra
2	VDD	5V/3.3V	Fonte de alimentação
3	V0	—	Ajuste de contraste
4	RS	H/L	H: Dados / L: Comandos
5	RW	H/L	H: Leitura / L: Escrita
6	E	H, H>L	Habilitação do sinal
7	DB0	H/L	Barramento de Dados
8	DB1	H/L	Barramento de Dados
9	DB2	H/L	Barramento de Dados
10	DB3	H/L	Barramento de Dados
11	DB4	H/L	Barramento de Dados
12	DB5	H/L	Barramento de Dados
13	DB6	H/L	Barramento de Dados
14	DB7	H/L	Barramento de Dados
15	CS1	L	Seletor Chip A
16	CS2	L	Seletor Chip B
17	RST	L	Reset
18	Vout	—	Saída de energia negativa para condução de lcd
19	LED A	3,3V à 10V	Fonte de alimentação para iluminação do Backlight
20	LED K	0V	Terra

Os **pinos 1 e 2** são referentes a alimentação do Display, sendo o pino 1 VSS Zero Volts (GND), e o pino 2 VDD, o pino de alimentação positiva de 3.3 à 5V;

O **pino 3** VO é referente ao ajuste de contraste do display. No teste efetuado utilizamos um potenciômetro de 10KOhms.

O **pino 4** RS é referente a seleção do registrador, sendo nível baixo (0) para seleção de comandos, ou nível alto (1) para dados.

O **pino 5** RW é referente a leitura (Read) ou escrita (Write), sendo nível alto (1) para leitura, e nível alto (0) para escrita.

O **pino 6** E, é referente a ativação de leitura e escrita no Display proporcionando a habilitação e desabilitação do sinal.

Os **pinos 7,8,9,10,11,12,13,14** são pinos referentes ao barramento de dados.

O **pino 15** CS1 é referente a seletor do Chip 1, ele seleciona a primeira metade ou o primeiro controlador KS0108 do lcd.

O **pino 16** CS2 é referente a seletor do Chip 2, ele seleciona a segunda metade ou o segundo controlador KS0108 do lcd.

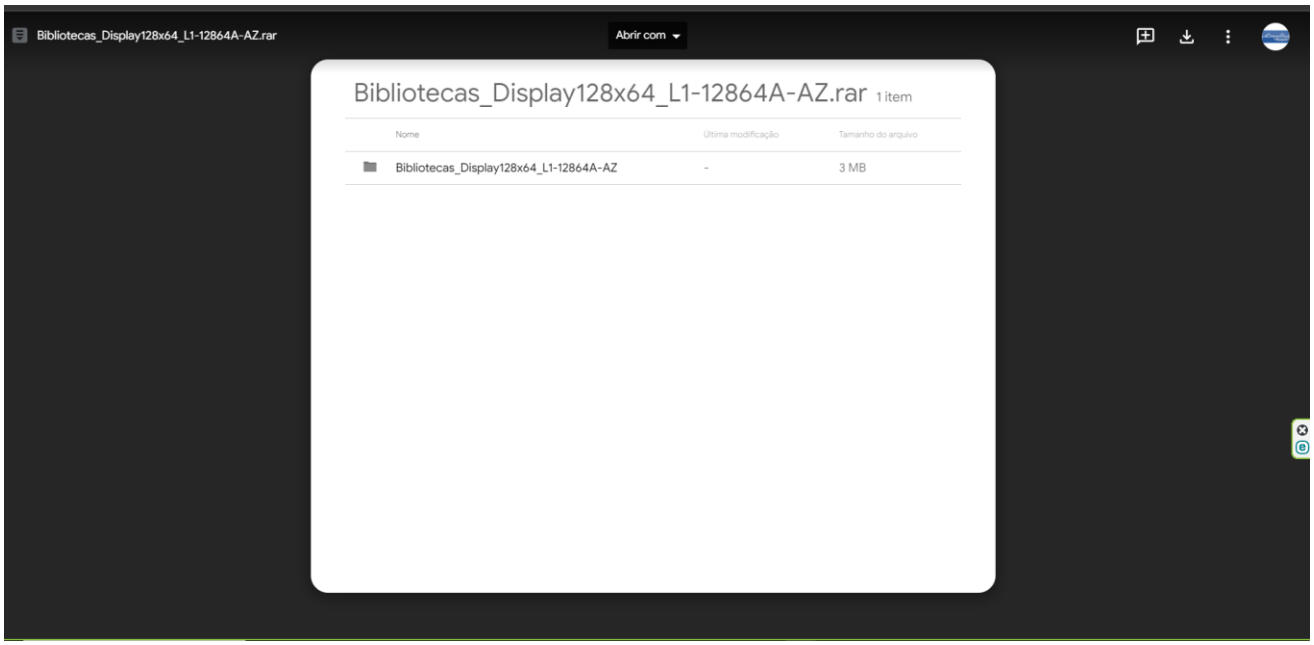
O **pino 17** RST é referente ao Reset do sinal.

O **pino 18** VOUT é referente a saída de energia negativa para o ajuste do contraste do LCD.

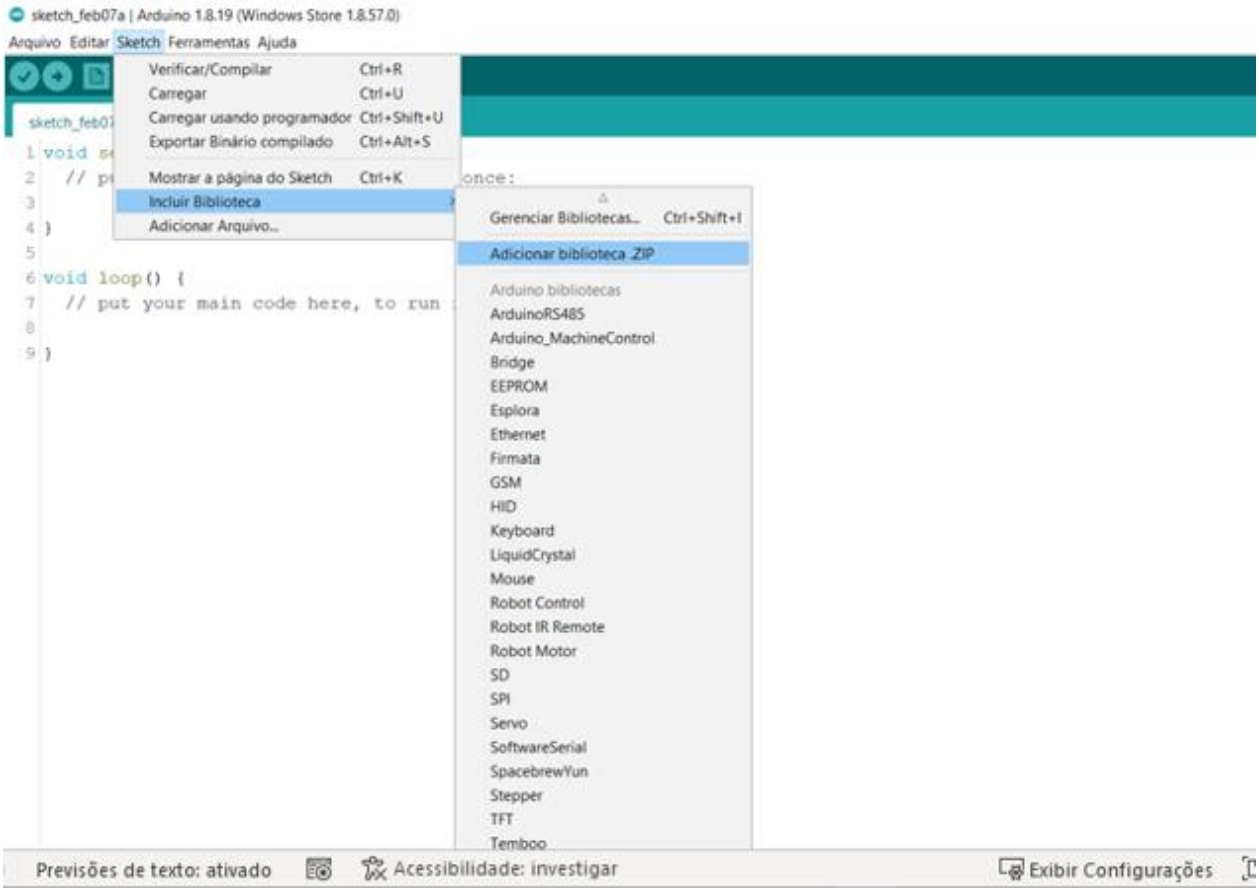
O **pino 19** é referente ao anodo do led de iluminação (Backlight) 3,3V à +10V. (Recomenda-se a utilização de uma fonte independente).

O **pino 20** é referente ao catodo do led de iluminação (Backlight) GND.

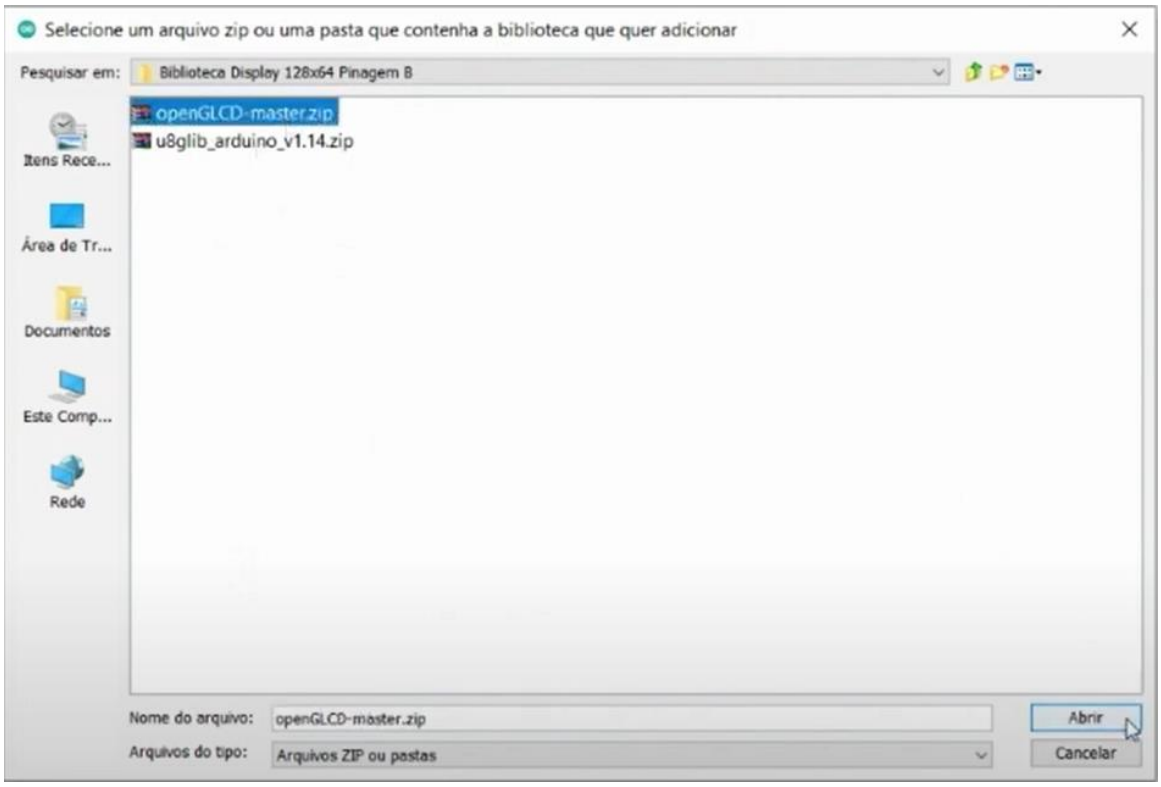
Agora que já conhecemos o Display e a função de seus pinos, vamos instalar as bibliotecas compatíveis com o Display, O link para download está disponível **clicando aqui**, na descrição do anúncio, e na descrição do vídeo.



Após efetuado o download da biblioteca, abra a IDE do Arduino. Clique na aba Sketch, Incluir Biblioteca, e em Adicionar Biblioteca ZIP.



Busque pelo arquivo baixado, selecione-o e clique em Abrir.



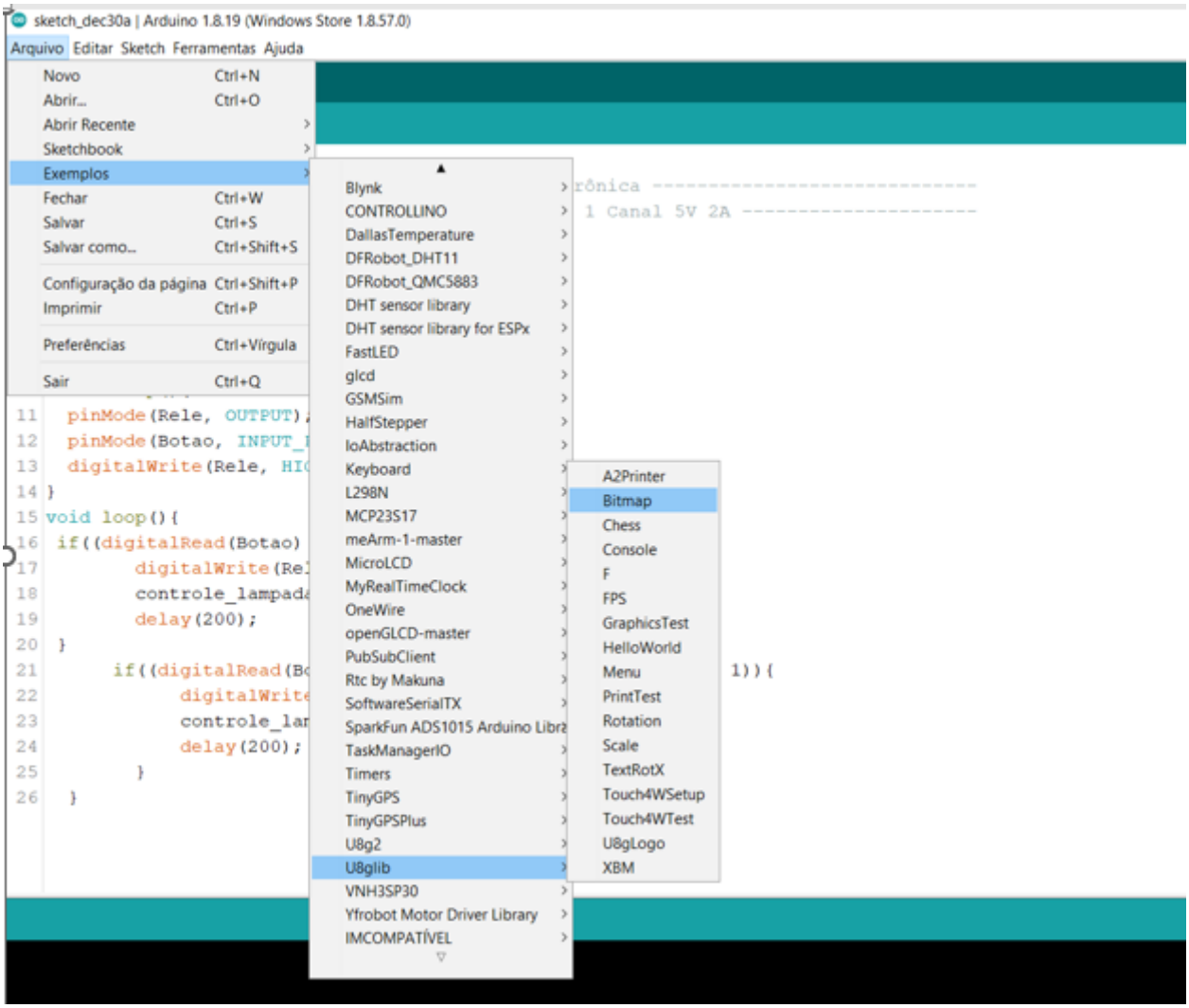
Após a importação da biblioteca, temos que nos atentar a conexão dos pinos segundo as configurações ditadas pela biblioteca.

Conexões para Display GLCD, segundo as bibliotecas OpenGLCD e U8glib

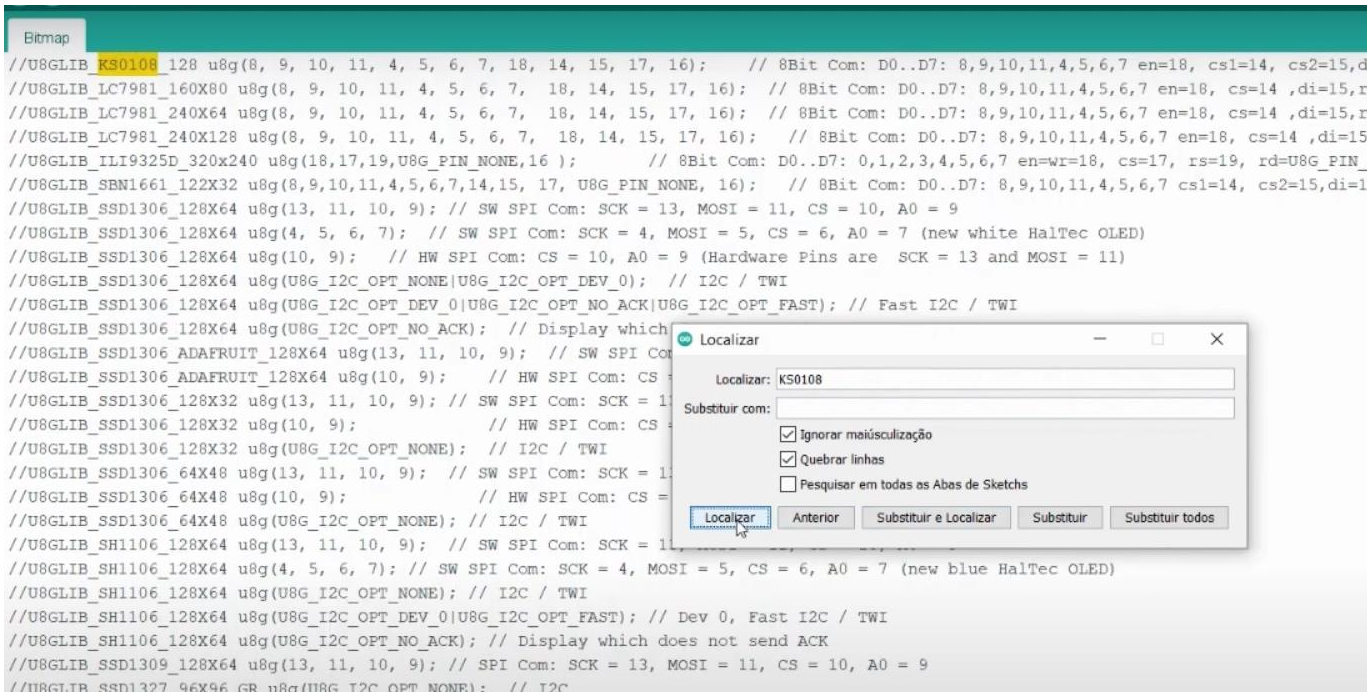
Para o correto funcionamento do Display, deve-se atentar as especificações impostas pela biblioteca a ser utilizada. Ao utilizarmos as bibliotecas GLCD disponíveis, utilizaremos a tabela disponibilizada pela IDE do Arduino **(clique aqui para ter acesso direto a tabela pela IDE)** para mapeamento dos pinos de conexão do display ao arduino, garantindo assim a o correto funcionamento do Display. Atente-se ao modelo do seu display, o modelo do display aqui testado é Pinagem B, como selecionado na tabela, mais como pode ser visto, existem quatro pinagens padrão para esse tipo de display.

Arduino 168	Arduino mega	Função	Pinagem A	Pinagem B	Pinagem C	Pinagem D	Comentários
5V	5V	+5 volts	1	!2!	!2!	4	
Gnd	Gnd	GND	2	!1!	!1!	3	
n / D	n / D	VO (contraste)	3	3	3	5	Conecte ao potenciômetro no pino de ajuste de contraste (pino do meio)
8	22	D0	4	7	7	9	
9	23	D1	5	8	8	10	
10	24	D2	6	9	9	11	
11	25	D3	7	10	10	12	
4	26	D4	8	11	11	13	
5	27	D5	9	12	12	14	
6	28	D6	10	13	13	15	
7	29	D7	11	14	14	16	
14 (alog0)	33	CSEL1	12	15	16	1	Seleção de chip 1
15 (alog1)	34	CSEL2	13	16	15	2	Seleção de chip 2
		Redefinir	14	17	17		(consulte a nota do pino de redefinição abaixo)
16 (alog2)	35	R_W	15	5	5	7	Ler escrever
17 (alog3)	36	D_I	16	4	4	6	Dados/ Instrução (RS)
18 (alog4)	37	PT	17	6	6	8	Habilitar
n / D	n / D	Vee/Vout (saída de contraste)	18	18	18		Conecte a uma perna do potenciômetro de 10k ou 20k
n / D	n / D	Luz de fundo +5	19	19	19		Resistor de 100 à 330 ohms para +5v
GND	GND	Retroiluminação Gnd	20	20	20		
n / D	n / D	n / D	n / D	n / D	n / D	n / D	Conecte a outra perna do potenciômetro de contraste ao Gnd

Para efetuar o mapeamento correto dos pinos do display com o Arduino segundo a biblioteca, basta abrir um exemplo qualquer, identificar o modelo do seu display, e refazer a pinagem. Para isso, vá em Exemplos, U8glib e em Bitmap (Bitmap foi o exemplo que utilizamos para nossos testes).



Com o código aberto, clique em Ctrl+f para abrir a janela localizar. Em localizar digite o nome do controlador do seu display. Olhando o datasheet do display utilizado para teste, vimos que o controlador é o KS0108.



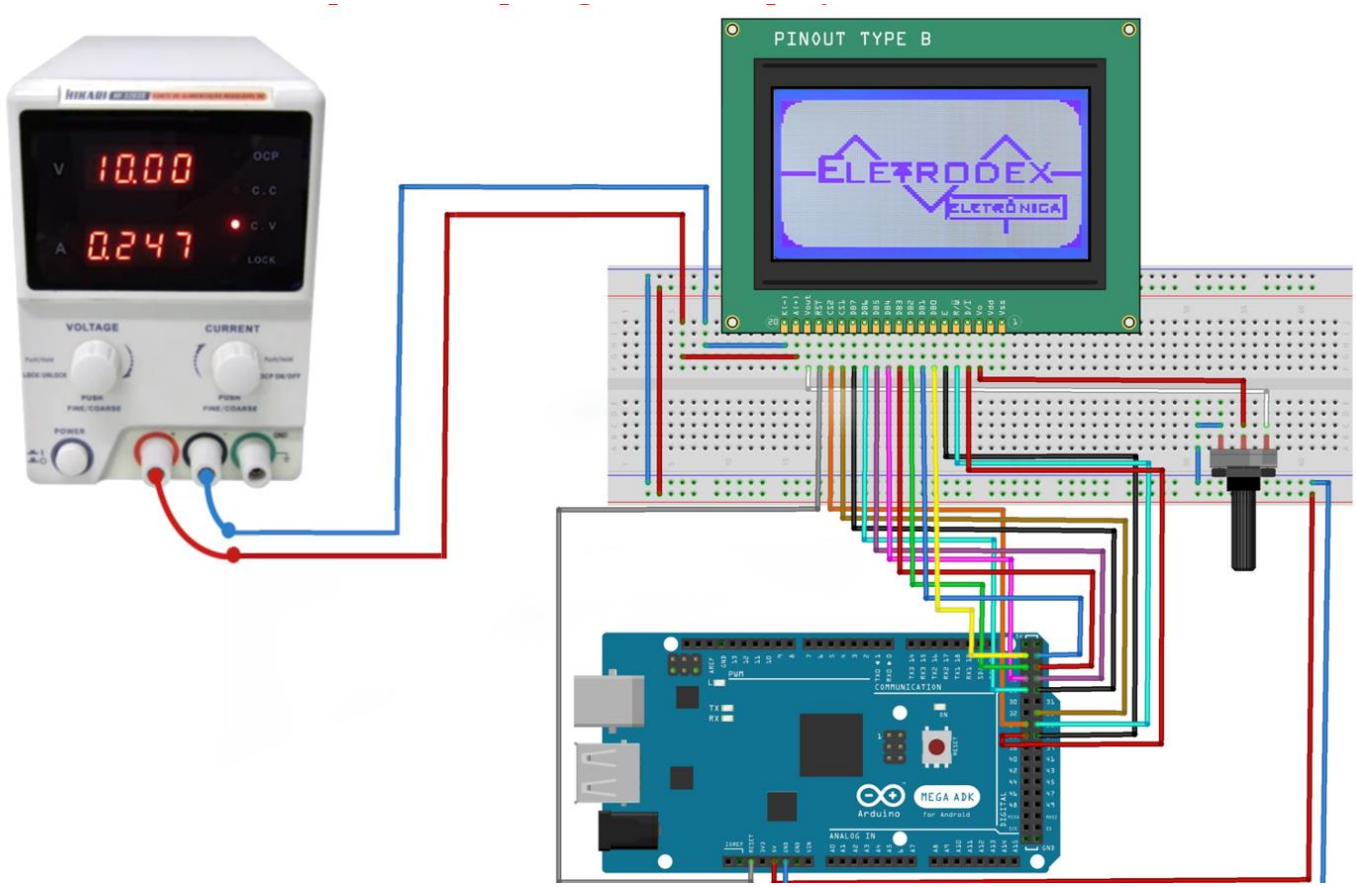
Remova as barras para tornar a linha válida, modifique os pinos de 8,9,10,11,4,5,6,7,18,14, 15, 17,16 para 22,23,24,25,26,27,28,29,37,33,34,36,25 respectivamente

```
Bitmap §
//U8GLIB_T6963_240X128 u8g(8, 9, 10, 11, 4, 5, 6, 7, 14, 15, 17, 18, 16); // 8Bit Com: D0..D7: 8,9,10,11,4,5,6,7, cs=14, a0=15, wr=17, rd=18, reset=16
//U8GLIB_T6963_128X128 u8g(8, 9, 10, 11, 4, 5, 6, 7, 14, 15, 17, 18, 16); // 8Bit Com: D0..D7: 8,9,10,11,4,5,6,7, cs=14, a0=15, wr=17, rd=18, reset=16
//U8GLIB_T6963_240X64 u8g(8, 9, 10, 11, 4, 5, 6, 7, 14, 15, 17, 18, 16); // 8Bit Com: D0..D7: 8,9,10,11,4,5,6,7, cs=14, a0=15, wr=17, rd=18, reset=16
//U8GLIB_T6963_128X64 u8g(8, 9, 10, 11, 4, 5, 6, 7, 14, 15, 17, 18, 16); // 8Bit Com: D0..D7: 8,9,10,11,4,5,6,7, cs=14, a0=15, wr=17, rd=18, reset=16
//U8GLIB_HT1632_24X16 u8g(3, 2, 4);   // WR = 3, DATA = 2, CS = 4
//U8GLIB_SSD1351_128X128_332 u8g(13, 11, 8, 9, 7); // Arduino UNO: SW SPI Com: SCK = 13, MOSI = 11, CS = 8, A0 = 9, RESET = 7 (http://electronics.ilsoft.co.uk/A
//U8GLIB_SSD1351_128X128_332 u8g(76, 75, 8, 9, 7); // Arduino DUE: SW SPI Com: SCK = 13, MOSI = 11, CS = 8, A0 = 9, RESET = 7 (http://electronics.ilsoft.co.uk/A
//U8GLIB_SSD1351_128X128_332 u8g(8, 9, 7); // Arduino: HW SPI Com: SCK = 13, MOSI = 11, CS = 8, A0 = 9, RESET = 7 (http://electronics.ilsoft.co.uk/ArduinoShield
//U8GLIB_SSD1351_128X128_HICOLOR u8g(76, 75, 8, 9, 7); // Arduino DUE, SW SPI Com: SCK = 76, MOSI = 75, CS = 8, A0 = 9, RESET = 7 (http://electronics.ilsoft.co.
//U8GLIB_SSD1351_128X128_HICOLOR u8g(8, 9, 7); // Arduino, HW SPI Com: SCK = 76, MOSI = 75, CS = 8, A0 = 9, RESET = 7 (http://electronics.ilsoft.co.uk/ArduinoSh
//U8GLIB_SSD1351_128X128GH_332 u8g(8, 9, 7); // Arduino, HW SPI Com: SCK = 76, MOSI = 75, CS = 8, A0 = 9, RESET = 7 (Freetronics OLED)
//U8GLIB_SSD1351_128X128GH_HICOLOR u8g(8, 9, 7); // Arduino, HW SPI Com: SCK = 76, MOSI = 75, CS = 8, A0 = 9, RESET = 7 (Freetronics OLED)

U8GLIB_KS0108_128 u8g(22, 23, 24, 25, 26, 27, 28, 29, 37, 33, 34, 36, 35);     // 8Bit Com: D0..D7: 22,23,24,25,26,27,28,29 en=37, cs1=33, cs2=34, rs=36, rw=35
```

Agora, para utilizar todos os exemplos das bibliotecas disponíveis, ou em um novo código, basta colar essa linha com as alterações abaixo da declaração da biblioteca (#include "U8glib.h"). Após efetuada a redefinição dos pinos, deve-se efetuar a montagem física

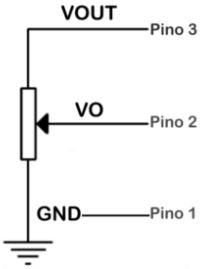
Conexão da pinagem do Display L1-12864A-AZ, ao Arduino Mega



O **pino 1** VSS deve ser conectado ao Zero Volts (GND) do Arduino Mega

O **pino 2** VDD, deve ser conectado a alimentação positiva de 5V do Arduino Mega

O **pino 3** VO deve ser conectado ao pino 2 de sinal do potenciômetro, o potenciômetro utilizado para teste foi de 10 KOhms



O **pino 4 RS** deve ser conectado ao pino 36 do Arduino Mega.

O **pino 5** RW deve ser conectado ao pino 35 do Arduino Mega.

O **pino 6** E, deve ser conectado ao pino 37 do Arduino Mega.

O **pino 7,8,9,10,11,12,13,14 (DB0 à DB7)** devem ser conectados aos pinos 22,23,24,25,26,27,28,29 do Arduino Mega respectivamente.

O **pino 15** CS1 deve ser conectado ao pino 33 do Arduino Mega.

O **pino 16** CS2 deve ser conectado ao pino 34 do Arduino Mega.

O **pino 17** RST deve ser conectado ao pino RESET do Arduino Mega (caso haja problema ao compilar o software para o arduino, recomenda-se remover essa conexão).

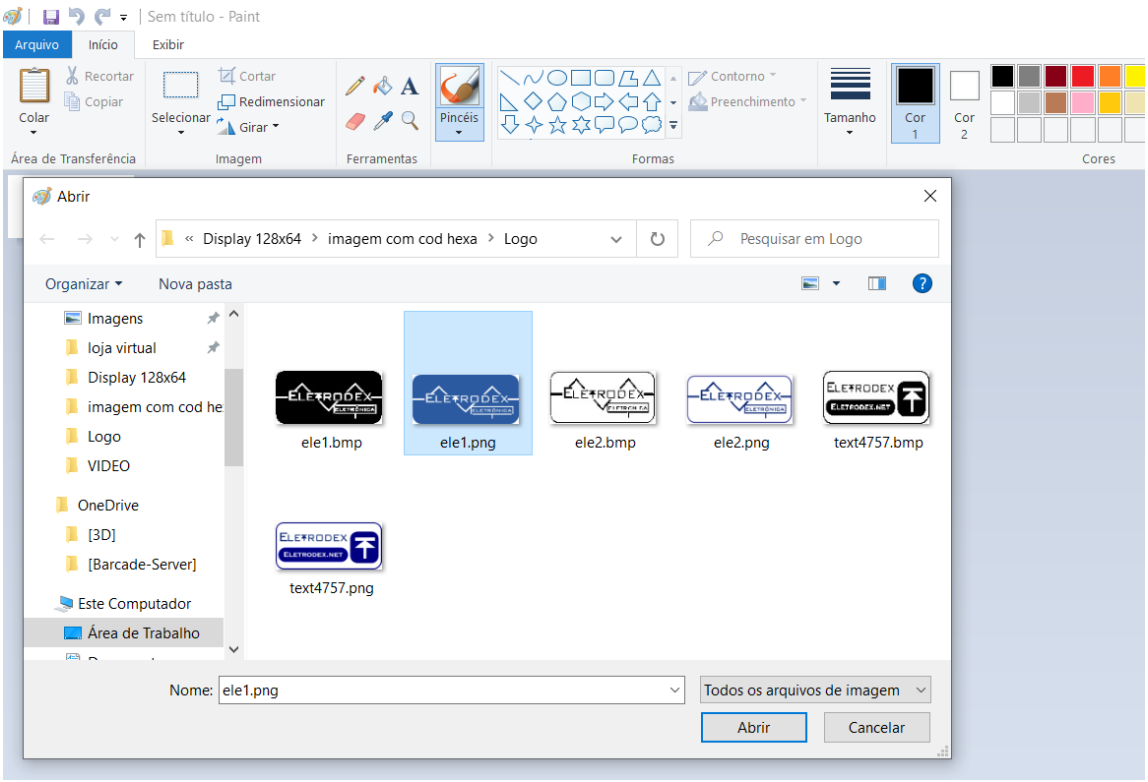
O **pino 18** VOUT deve ser conectado ao pino 3 do potenciômetro como demonstrado na figura 2, e o pino 1 do potenciômetro deve ser conectado ao GND do Arduino Mega.

O **pino 19** A deve ser conectado ao VCC 3,3V à 5V da fonte independente.

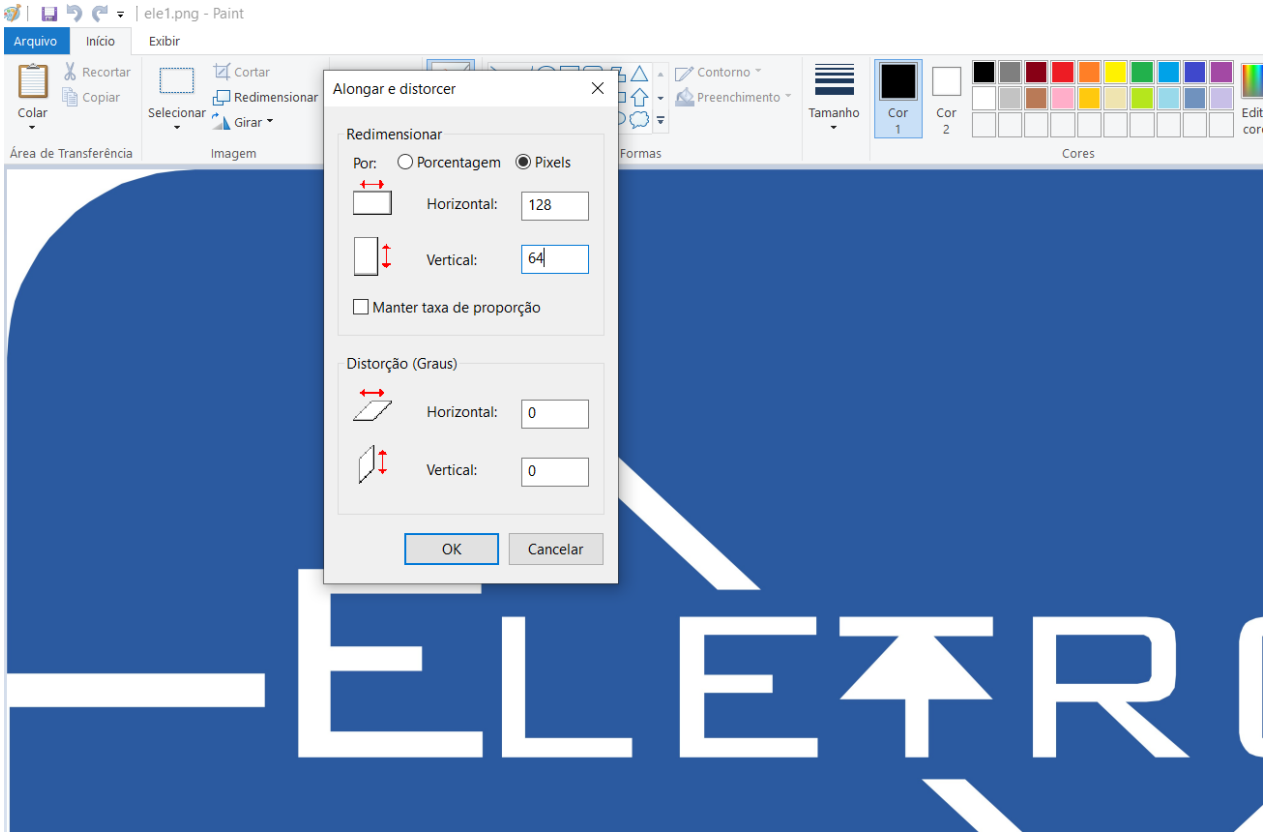
O **pino 20** K deve ser conectado ao GND da fonte independente.

Após efetuadas as conexões entre o display e o Arduino, vamos converter a imagem escolhida em bitmap monocromático e em seguida converter o bitmap monocromático em hexadecimal para utilizarmos o código hexa reconhecível pelo microcontrolador.

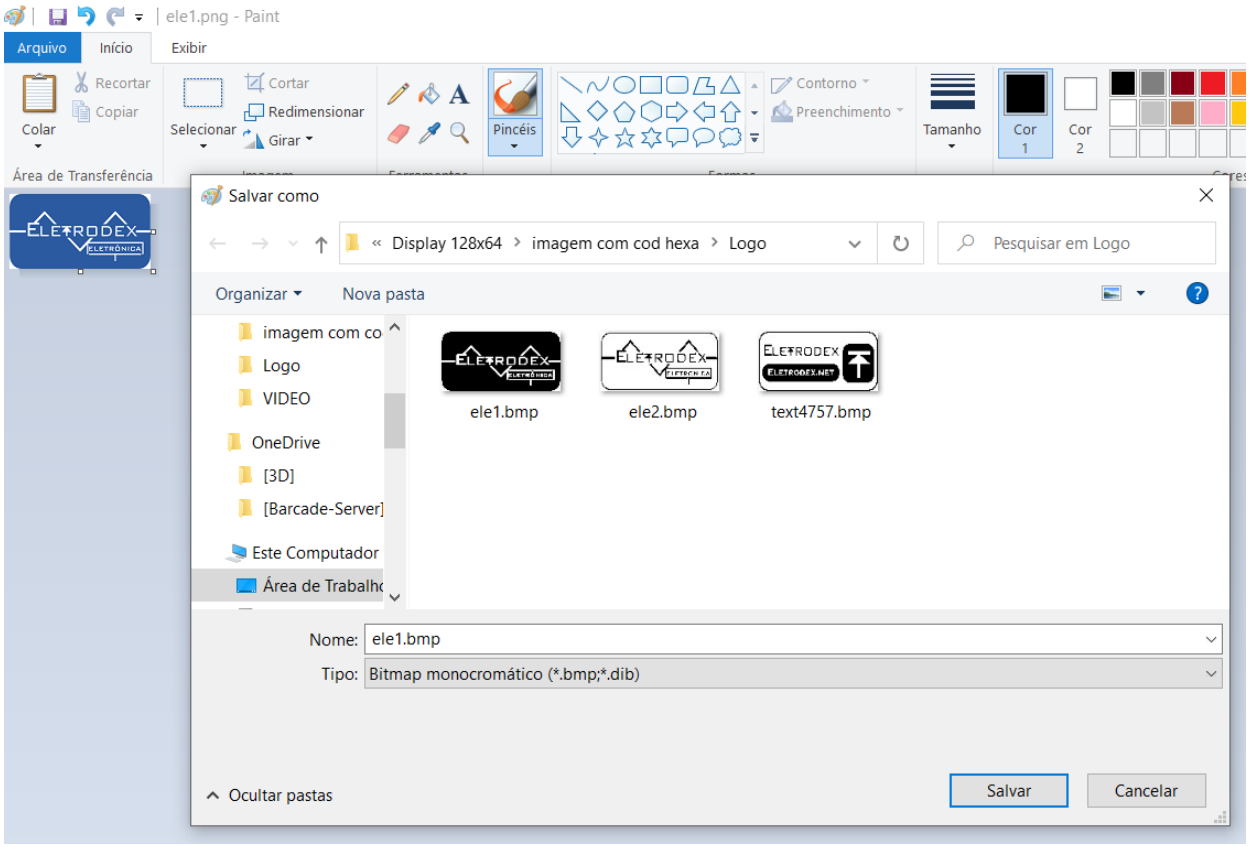
Inicialmente abra o Paint disponível no próprio Windows. Vá na aba Arquivo – Abrir, e selecione a imagem. Escolhemos a logo da Eletrodex.



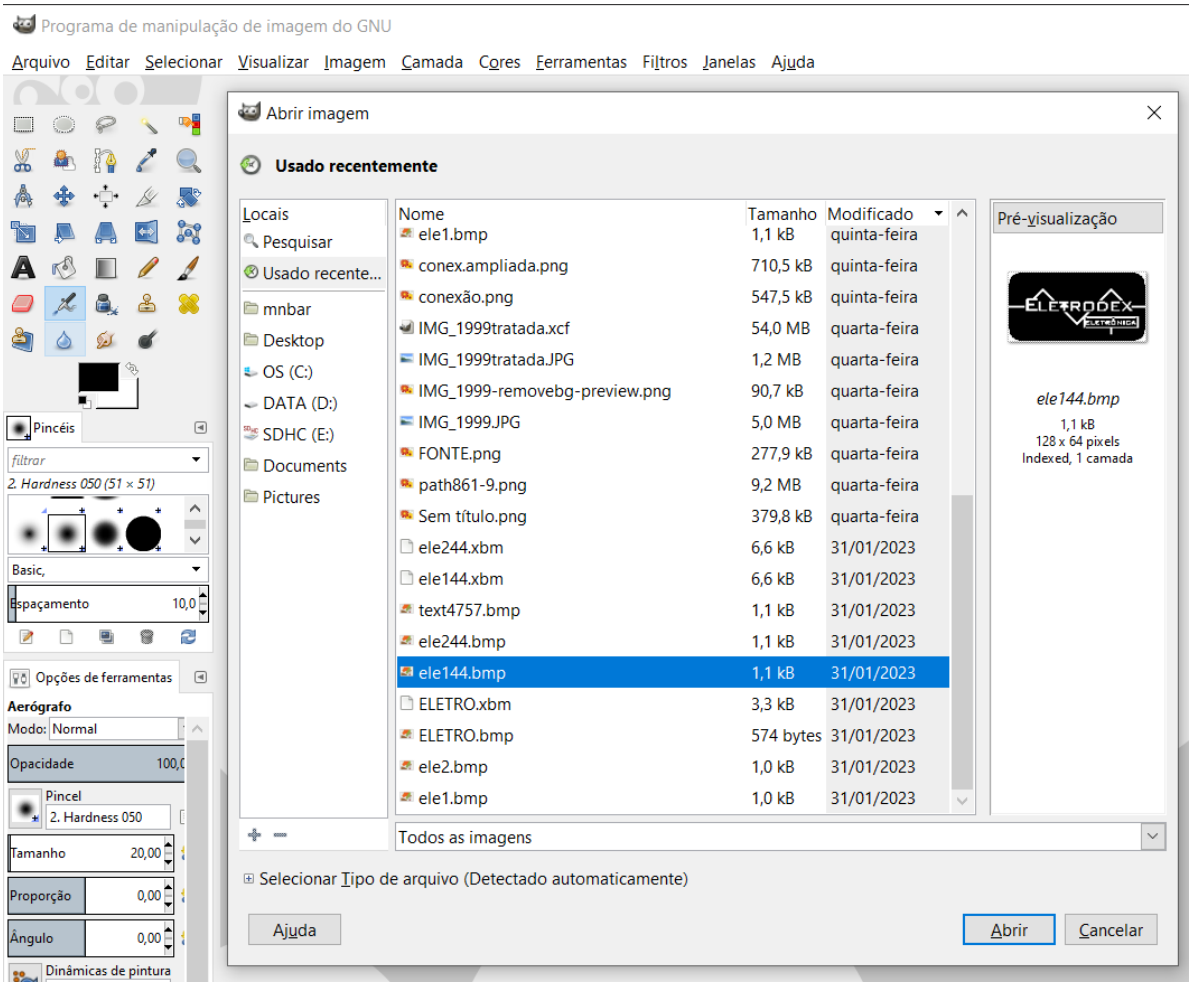
Com a imagem escolhida aberta, clique em Redimensionar. Na caixa que se abre desmarque “Manter taxa de proporção”, marque a caixa “Pixels”, na caixa de valores referente a Horizontal digite 128, e na caixa de valores referente a vertical digite 64



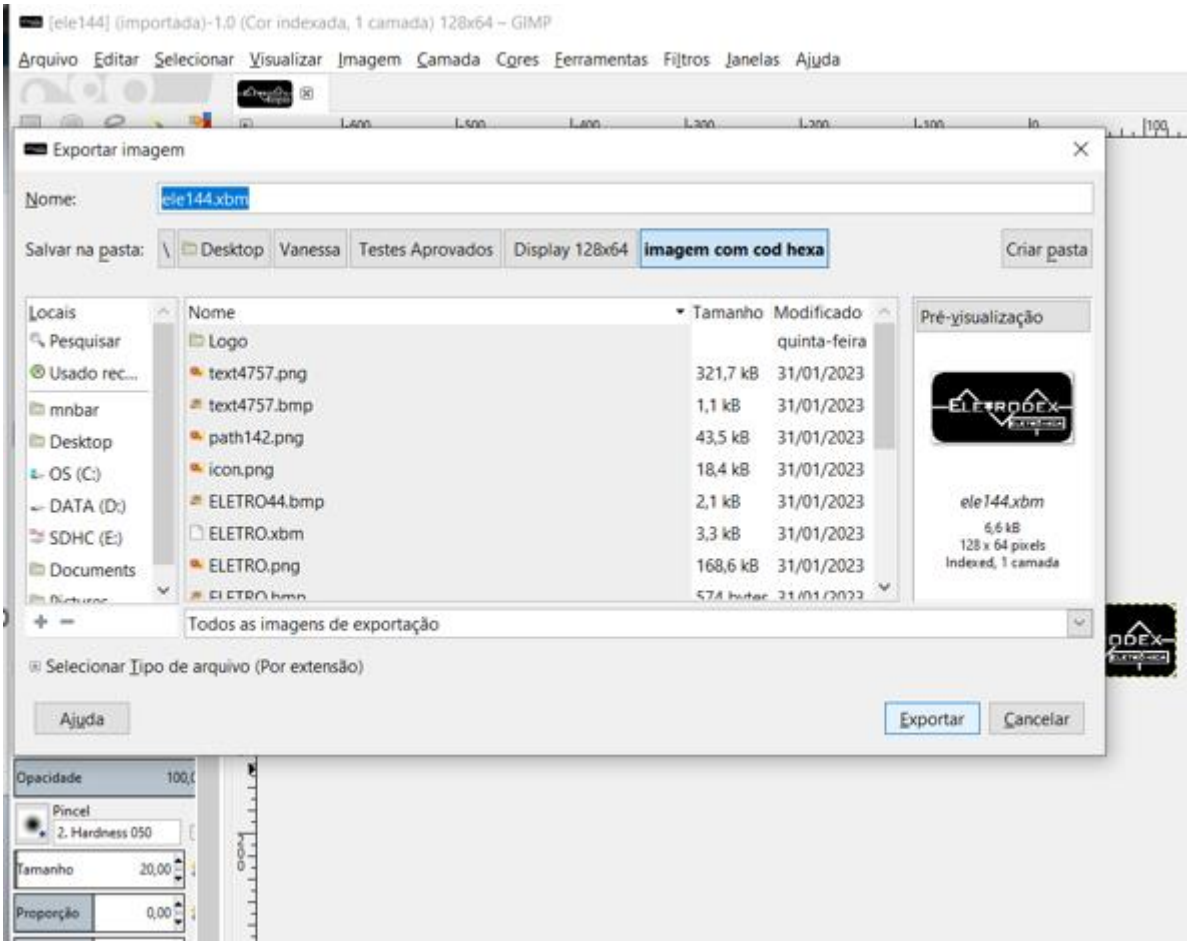
Depois de redimensionar a imagem salve a imagem, Clique em Arquivo, salvar como, na janela que se abre em Tipo selecione “Bitmap monocromático (*.bmp; *dib)”, dê um nome a imagem e salve.



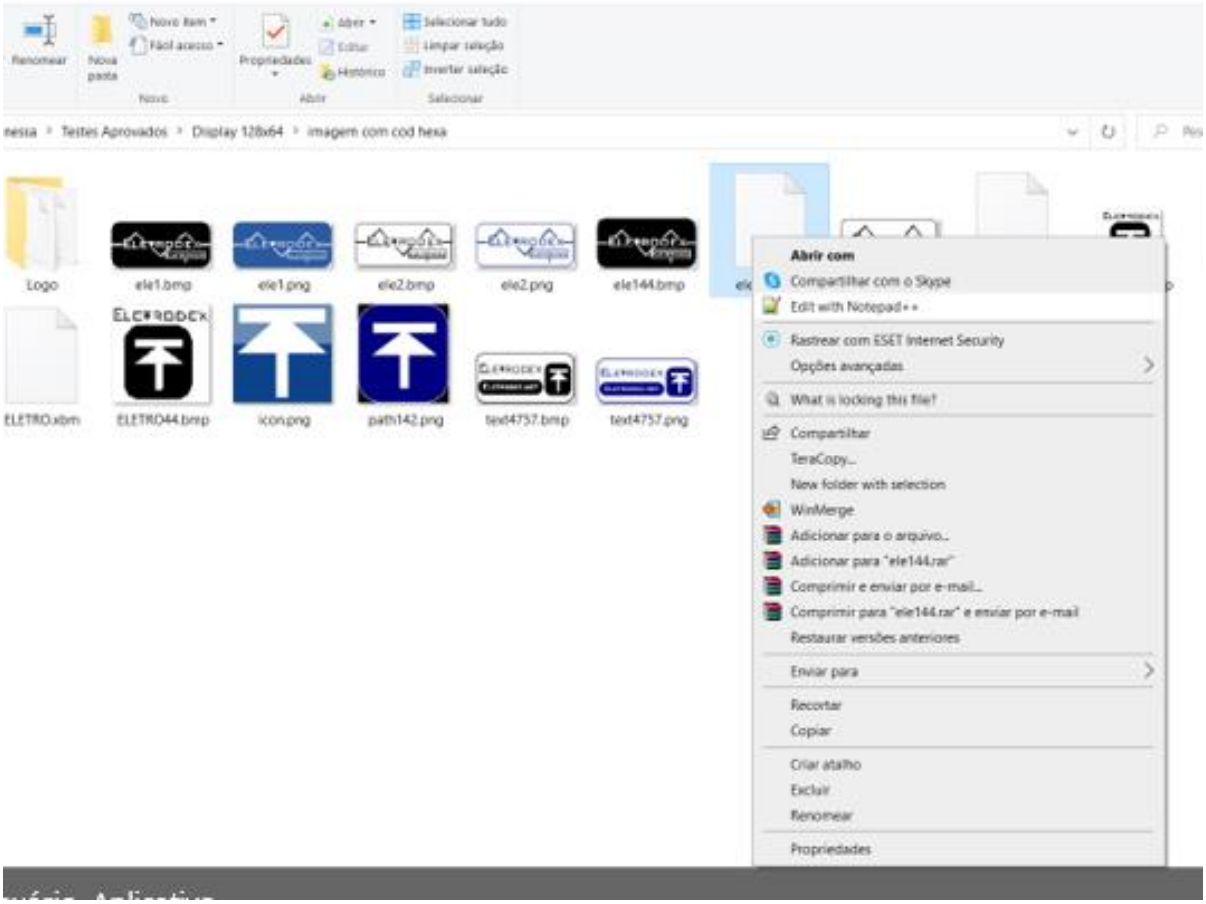
Com a imagem salva em formato Bitmap com extensão .bmp, abra o GIMP (software gratuito de edição de imagens), clique em Arquivo, Abrir, e busque pela imagem salva no formato bitmap monocromático.



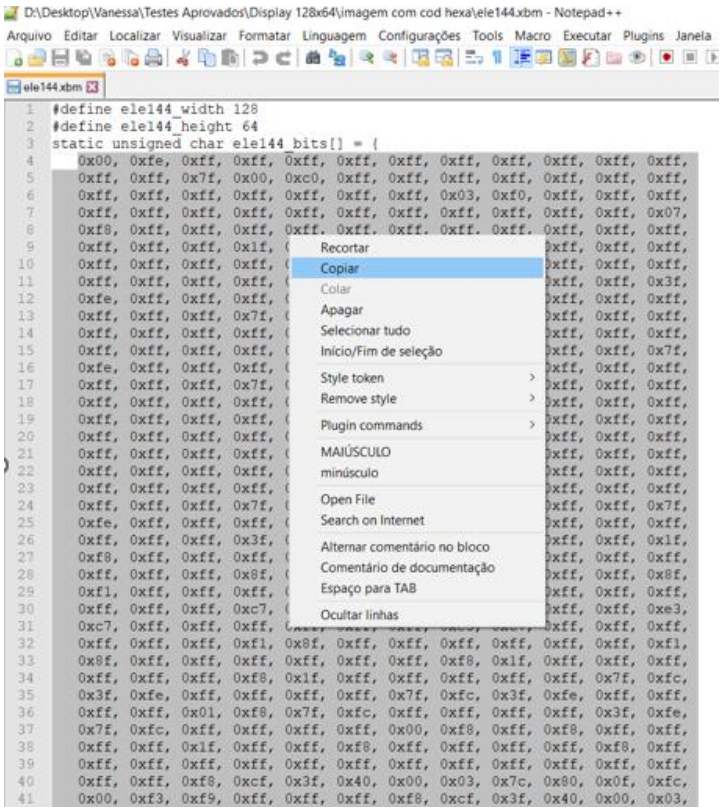
Abra a imagem, clique em Arquivo - Exportar como, e na janela que se abre modifique a extensão da imagem para. xbm e exporte clicando em Exportar.



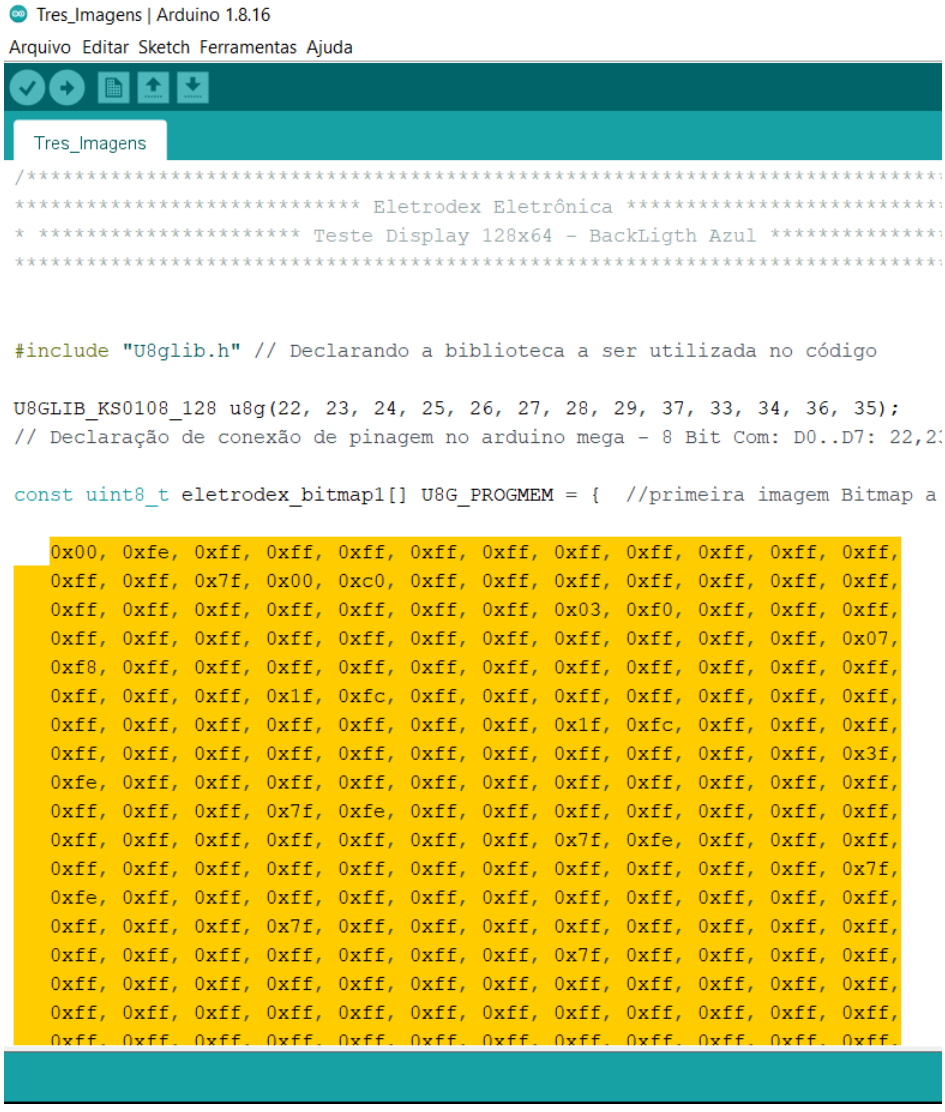
Vá na pasta onde foi salva a imagem em formato xbm, clique em cima do arquivo com o botão direito e abra com um editor de texto, utilizamos o Notepad++



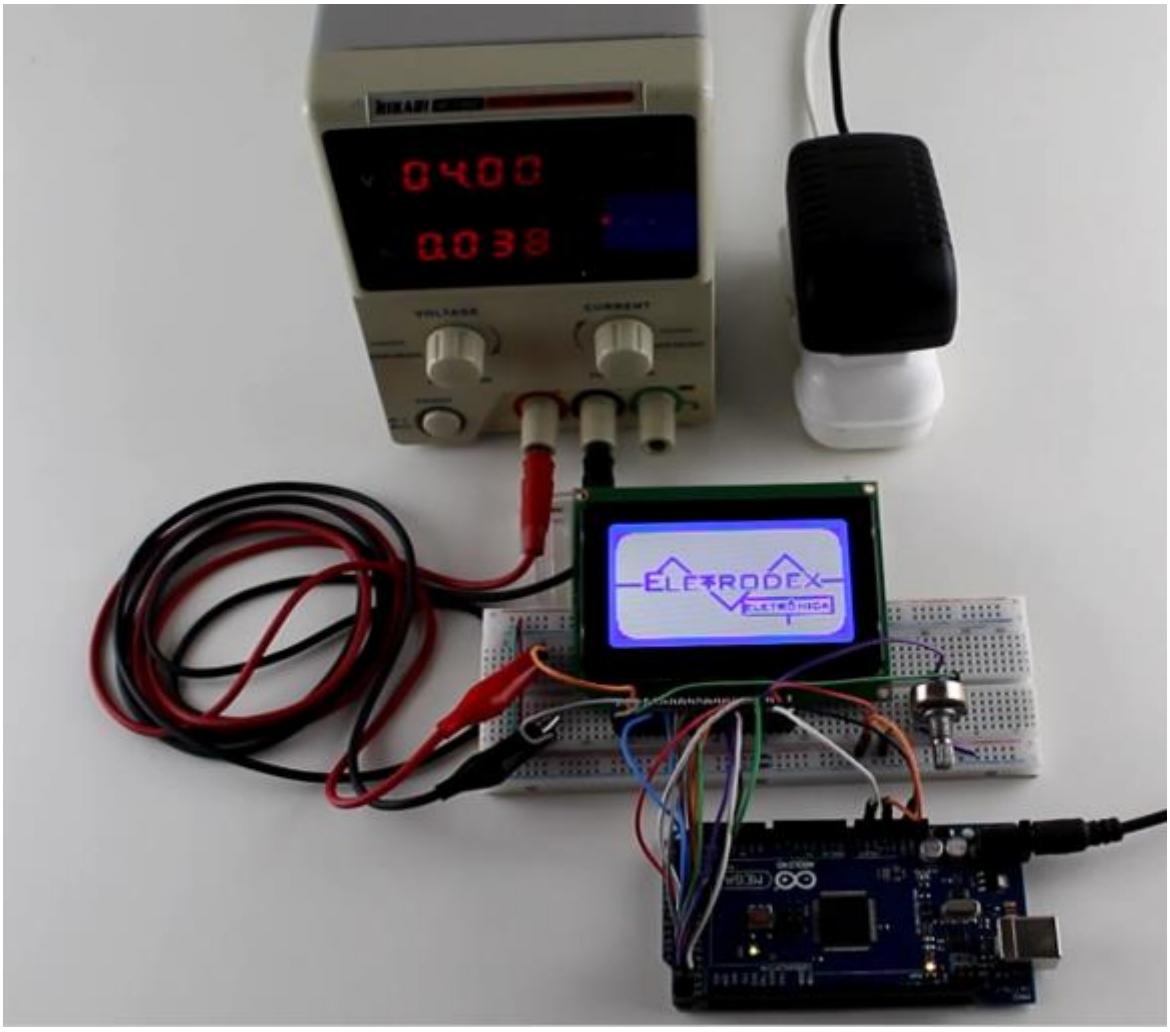
Copie o código em hexadecimal aberto no notepad++.



Cole no código disponibilizado no arquivo software abaixo de const uint8_t eletrodex_bitmap1[] U8G_PROGMEM = {



Compile o Arquivo para o microcontrolador, e se todas as conexões estiverem corretas a imagem será exibida no display como se segue:



Faça o mesmo procedimento com as demais imagens, e adicione o código hexadecimal no sketch

Observação: A fonte utilizada foi uma fonte variável, a fim de controlar a tensão de iluminação da luz de fundo para obtermos a melhor resposta visual do display. Vale ressaltar que se pode alimentar a luz de fundo Anodo e Catodo com até 10V.

Confira o vídeo do teste de funcionamento do display clicando aqui.