Aula 06 – Instax OLED [Parte II]

Módulo 3

GOVERNADOR DO ESTADO DO PARANÁ

Carlos Massa Ratinho Júnior

SECRETÁRIO DE ESTADO DA EDUCAÇÃO

Roni Miranda Vieira

DIRETOR DE TECNOLOGIA E INOVAÇÃO

Claudio Aparecido de Oliveira

COORDENADOR DE TECNOLOGIAS EDUCACIONAIS

Marcelo Gasparin

Produção de Conteúdo

Darice Alessandra Deckmann Zanardini

Validação de Conteúdo

Cleiton Rosa

Revisão Textual

Kellen Pricila dos Santos Cochinski

Projeto Gráfico e Diagramação

Edna do Rocio Becker

2024

Introdução

Na **Aula 05 – Instax OLED [Parte I]**, começamos o preparo de uma imagem no formato bitmap para ser impressa no display OLED 128x64, como uma fotografia instantânea. Agora, chegou o momento de conectarmos o display OLED ao Arduino e fazermos as conversões dos dados da imagem para a programação. Vamos lá?

Objetivos desta aula

- Converter dados da imagem em matriz para exibição no display OLED;
- Utilizar a função **drawBitmap()**, recurso presente na biblioteca Adafruit GFX.

Lista de materiais

- Display OLED;
- Arduino Uno R3;
- Protoboard;
- 4 jumpers macho-macho;
- 4 jumpers macho-fêmea, caso não utilize protoboard;
- Computador e/ou notebook.

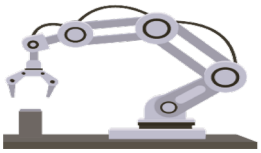
Roteiro da aula

Contextualização

Como vimos, o formato bitmap é um tipo de imagem que armazena os dados em pixels, ou seja, cada ponto da imagem tem uma cor e uma posição definidas, como em uma matriz. Este formato é ideal para o display OLED por permitir um controle preciso de cada pixel e alta qualidade da imagem.

Vimos, na última aula, que para converter uma fotografia para o formato bitmap podemos usar um software de edição de imagem, como os indicados na **Aula 04 – Softwares para Design**. Como nosso foco é a impressão de fotografia no display OLED, o primeiro passo foi recortar e redimensionar a imagem para o tamanho do display OLED presente no kit de Robótica. Depois, indicamos a conversão da imagem para escala de cinza, pois o seu display só pode mostrar dois tons, para você ter uma prévia da impressão.

A imagem que preparamos e está como exemplo na programação desta aula é do [braço robótico do B1N0](). Sinta-se à vontade para explorar outras imagens e fotografias, lembrando da proporção **2:1** para nosso display!

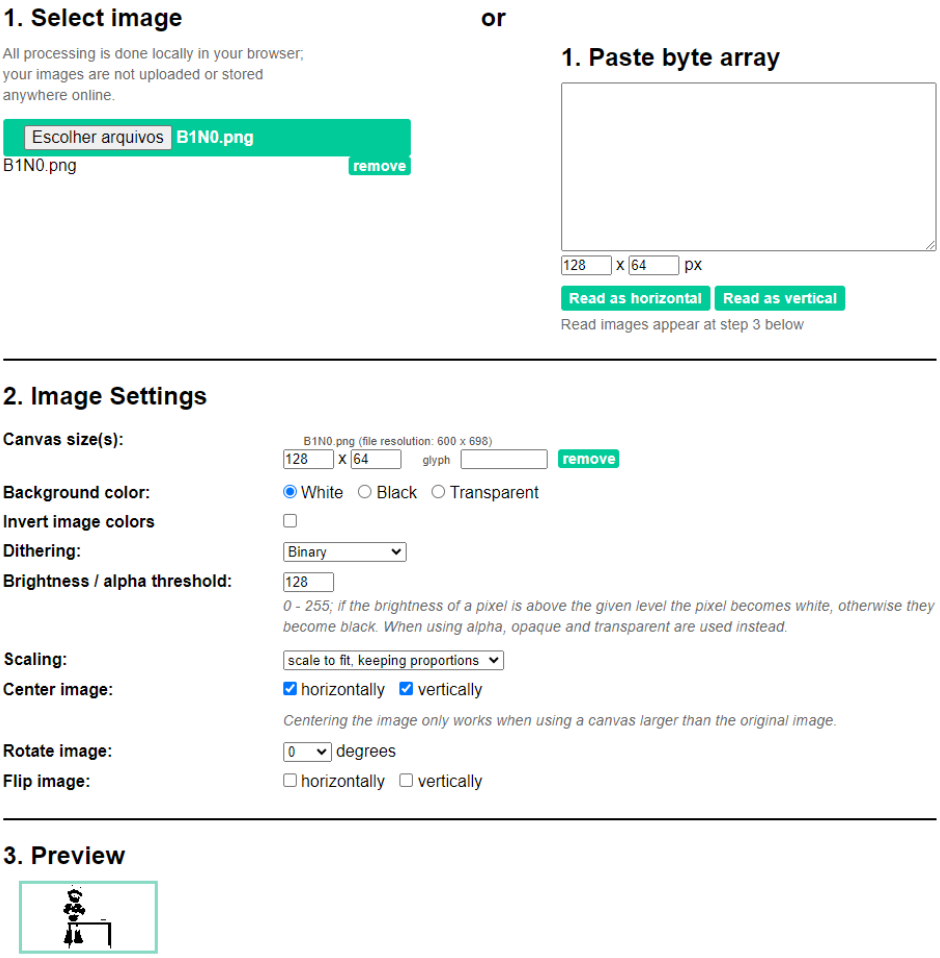
Imagem impressa no display OLED com 1 bit	Imagem recortada com proporção 2:1	Imagem original
		

Agora que temos a imagem em formato bitmap, precisamos convertê-la em um código que o Arduino possa entender, utilizando a ferramenta [Image2cpp](), e carregá-la ao Arduino para ser exibida, com o auxílio das bibliotecas **Adafruit SSD1306** e **Adafruit GFX Library**, no display OLED.

Como vimos na **Aula 05 – Instax OLED [Parte I]**, a ferramenta [Image2cpp]() pode ser utilizada online ou offline, caso você tenha descompactado no computador a pasta do projeto contendo arquivo HTML e demais complementos compartilhados no [GitHub do desenvolvedor]().

No exemplo da programação desta aula, indicamos os ajustes, por software de edição, que fizemos na imagem do B1N0 para imprimirmos apenas seu braço robótico. Porém, podemos utilizar o [Image2cpp]() para ajustes da imagem bitmap que queremos imprimir.

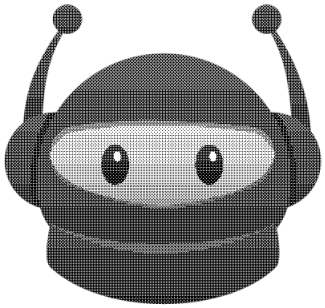
Por esta ferramenta, podemos fazer o upload (**1. Select image**) tanto da imagem original do [braço robótico do B1N0]() quanto de outra que você desejar e, nas configurações de imagem (**2. Image Settings**), definir a cor de fundo e escalar para o tamanho do display, verificando o resultado prévio (**3. Preview**).



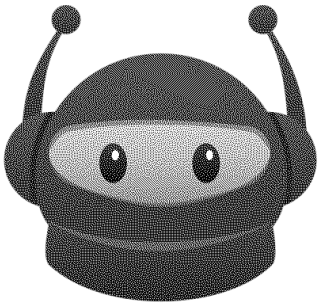
As opções de configuração da imagem (**Image Setting**s) no Image2cpp permitem personalizar a conversão da imagem em array de bytes. Vamos entender o que cada opção significa?

- **Canvas size(s)**: Define o tamanho do “quadro” ou “canvas” no qual a imagem será exibida e você pode escolher o tamanho adequado com base nas dimensões do display OLED.
- **Background color**: Aqui, você pode escolher a cor de fundo (branco, preto ou transparente) para a imagem.
- **Invert image colors**: Se ativada, essa opção inverte as cores da imagem. Pixels brancos se tornam pretos e vice-versa.
- **Dithering**: Suaviza ou mistura as cores da imagem original para representação monocromática dos pixels no display OLED, com efeito de pontilhamento e gradiente de pixels.

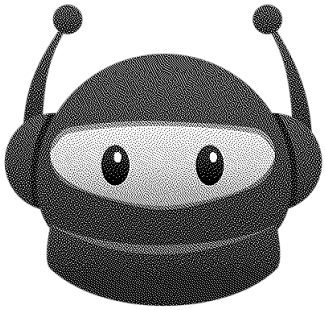
Binary – converte uma imagem colorida em preto e branco (1 bit) sem tons intermediários. Funciona aplicando um limiar: valores abaixo do limiar se tornam pretos, e valores acima se tornam brancos. É o método mais rápido e simples, mas pode perder muitos detalhes da imagem original.



Bayer – usa uma matriz infinitamente repetida que determina os valores de dither para cada pixel. É usado para converter imagens em tons de cinza para 1 bit (preto e branco).



Floyd-Steinberg – é o algoritmo de difusão de erro mais conhecido. Ele distribui o erro de quantização para os pixels vizinhos, criando um efeito de suavização.



Atkinson – também é um algoritmo de difusão de erro, mas usa pesos menores e ignora alguns pixels, resultando em um dithering mais suave e menos ruidoso.

- **Brightness / alpha threshold**: Ajusta o limiar de brilho ou transparência e seu valor varia de 0 a 255. Se o brilho de um pixel estiver acima do nível especificado, o pixel se torna branco. Caso contrário, o pixel se torna preto. Quando o canal alfa (transparência) é usado, os valores opaco e transparente são usados.
- **Scaling**: Dimensiona a imagem para o tamanho do quadro (**Canvas size**) definido.
    - *Original size* – mantém o tamanho original de upload da imagem.

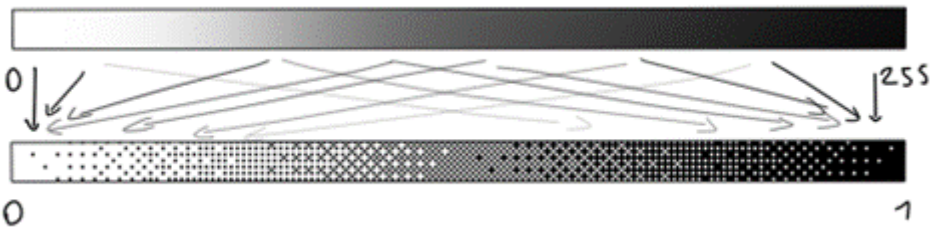
- *Scale to fit, keeping proportions* – dimensiona a imagem para caber no tamanho do quadro, mantendo as proporções.
- *Stretch to fill canvas* – estica a imagem para preencher a tela.
- *Stretch to fill canvas horizontally* – estica para preencher a tela horizontalmente.
- *Stretch to fill canvas vertically* – estica para preencher a tela verticalmente.

- **Center image**: Centraliza horizontalmente e/ou verticalmente a imagem no display se a proporção da tela for diferente.
- **Rotate image**: Rotaciona a imagem em ângulos de 0°, 90°, 180° ou 270°.
- **Flip image**: Vira a imagem no sentido horizontal e/ou vertical.

Saiba mais!

O **dithering** (pontilhamento) é uma técnica usada para simular tons de cinza ou cores em displays que só podem exibir preto e branco (ou poucas cores). Ele cria a ilusão de diferentes níveis de intensidade ou sombreamento, mesmo quando o display é binário (apenas preto e branco).

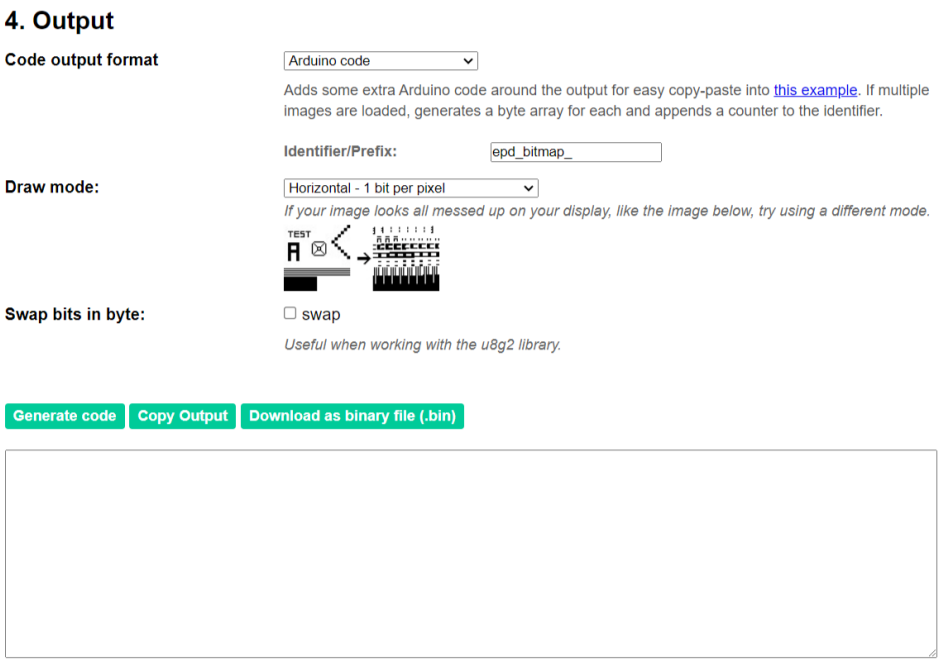
Ditherpunk – The article I wish I had about monochrome image dithering. Nesse artigo, em inglês, o desenvolvedor Surma traz exemplos desta técnica de pontilhamento de imagens monocromáticas sob uma ótica da arte, da programação e da matemática.



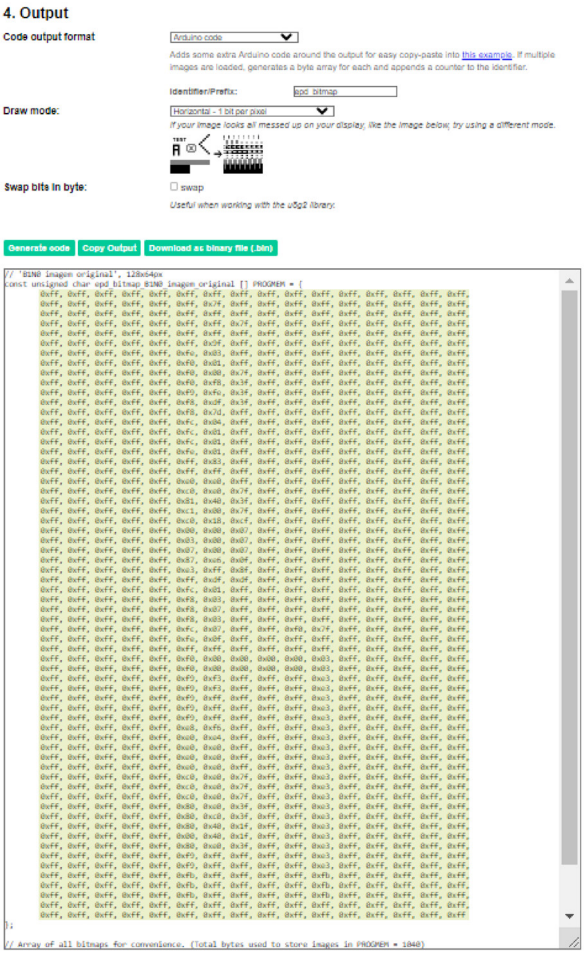
Dithering com valores para gradiente linear

Lembre-se de que essas configurações afetam como a imagem será representada no array de bytes gerado pelo **Image2cpp**. Experimente diferentes combinações desses parâmetros para obter o resultado desejado e conhecer outros, se atentando que, para melhor impressão no display OLED, você pode optar por fotografias com fundo mais limpo.

Estando ok o resultado prévio da imagem, confira se o formato está selecionado para Arduino (**4. Output**) e gere o código (**Generate code**).



Observe que, além do array de bytes, a ferramenta **Image2cpp** gera um complemento de linhas e atribui o nome do arquivo como nome da constante. Como nossa programação está pronta para você imprimir a imagem que desejar, copie apenas o conteúdo presente dentro das chaves da constante do tipo **unsigned char** para substituir na programação. Se você modificar o nome dessa constante, precisará corrigir o parâmetro da função **display.drawBitmap()**, a qual conheceremos mais adiante.

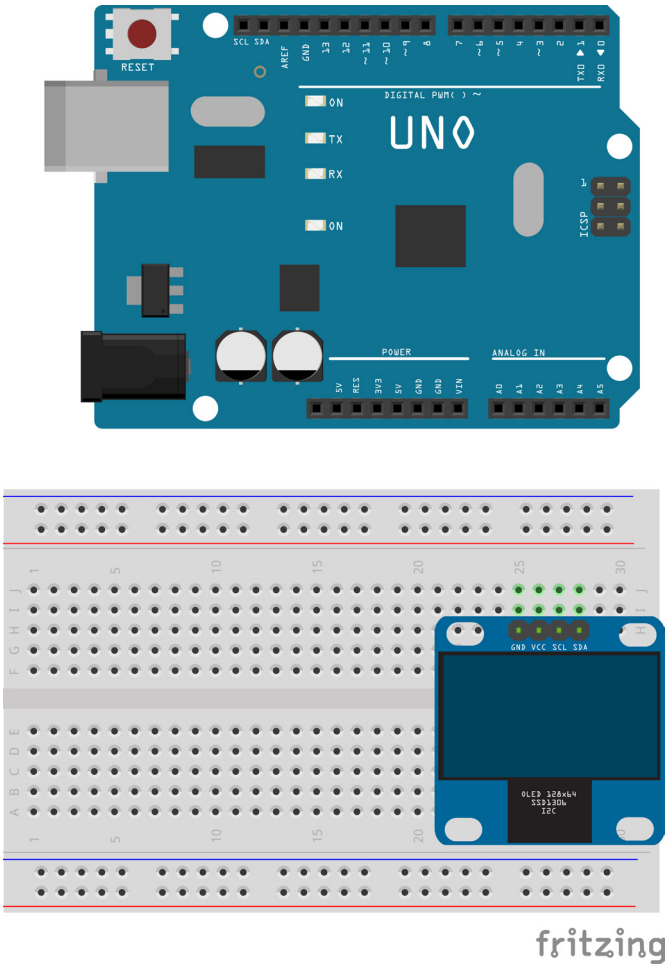


Antes de continuarmos com a programação, vamos à montagem do nosso protótipo Instax OLED para já podermos carregar o primeiro teste e imprimir o braço robótico do B1N0!

Montagem

Inicie a montagem dos componentes eletrônicos inserindo, na placa protoboard, o display OLED (figura 1) e observando a indicação da posição dos pinos a serem conectados: **GND** – **VCC** – **SCL** – **SDA**.

Figura 1 – Inserção do display OLED na protoboard



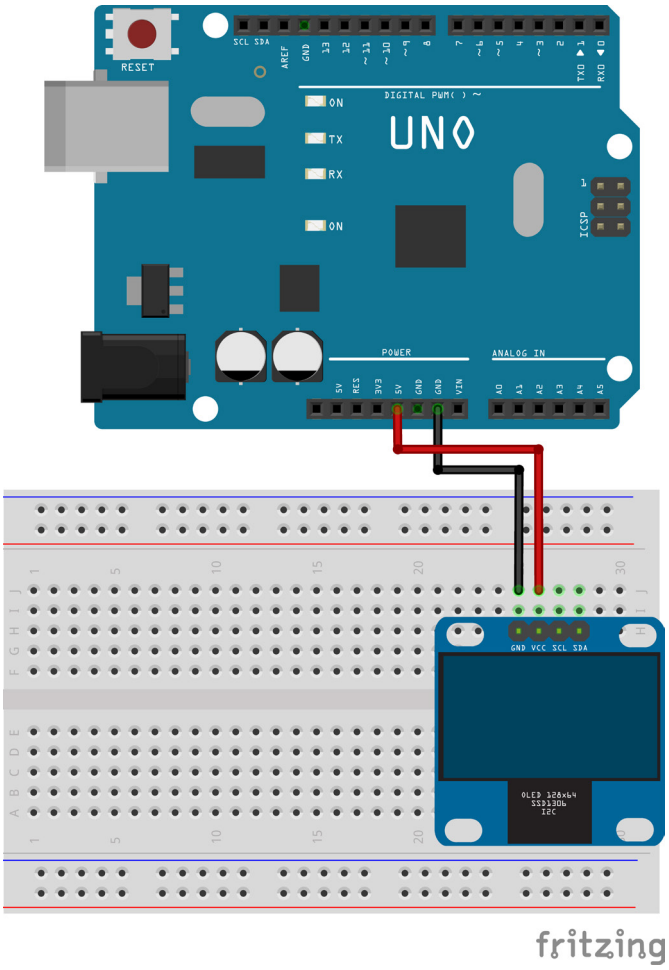
Fonte: SEED/DTI/CTE

Dica!

Como os fabricantes dos mais diversos componentes eletrônicos podem posicionar e soldar os pinos de forma diferenciada, certifique-se de seguir as conexões indicadas no componente que você possuir em mãos. As montagens propostas em aula são uma referência para as suas conexões.

Na sequência, inicie as **conexões de alimentação** do display OLED: conecte o pino GND à porta GND e o pino VCC do display à porta 5 V do Arduino (figura 2). **Tenha atenção que, conforme o modelo do display OLED, a ordem dos pinos pode ser outra. Então, antes de conectar os pinos ao Arduino, observe a indicação no display.**

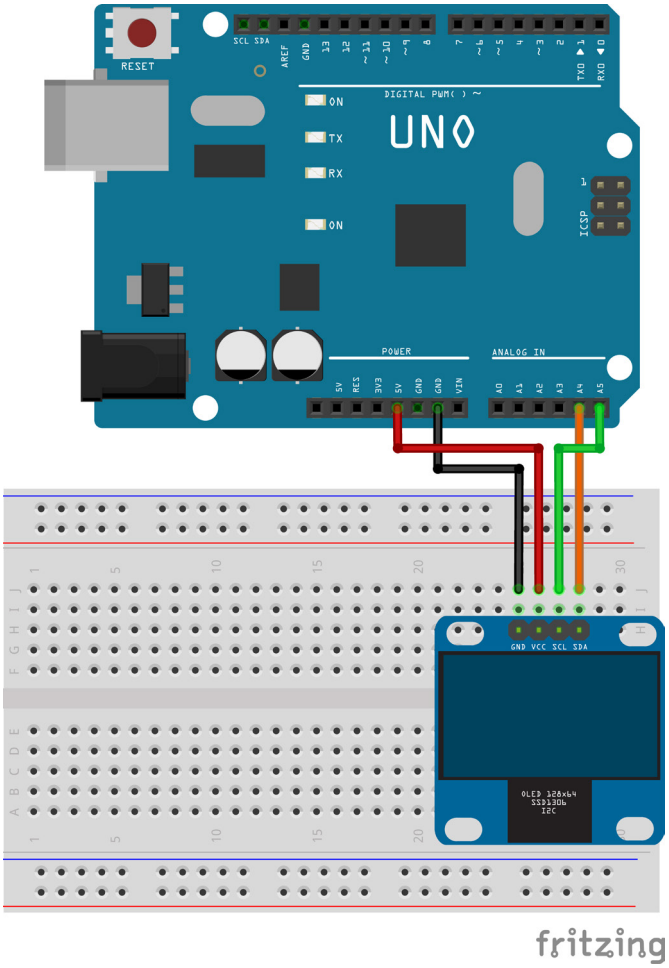
Figura 2 – Conexão dos pinos GND e VCC do display OLED ao Arduino



Fonte: SEED/DTI/CTE

Por fim, faremos as conexões de **transmissão de dados**: conecte o pino SCL (*serial clock*) do display OLED, referente ao sincronismo do display e do Arduino, à porta SCL do Arduino e o pino SDA *(serial data),* referente à troca de dados, à porta SDA do Arduino, se disponíveis. Caso seu Arduino não possua as portas exclusivas SCL e SDA, utilize as portas analógicas A5 e A4, respectivamente (figura 3).

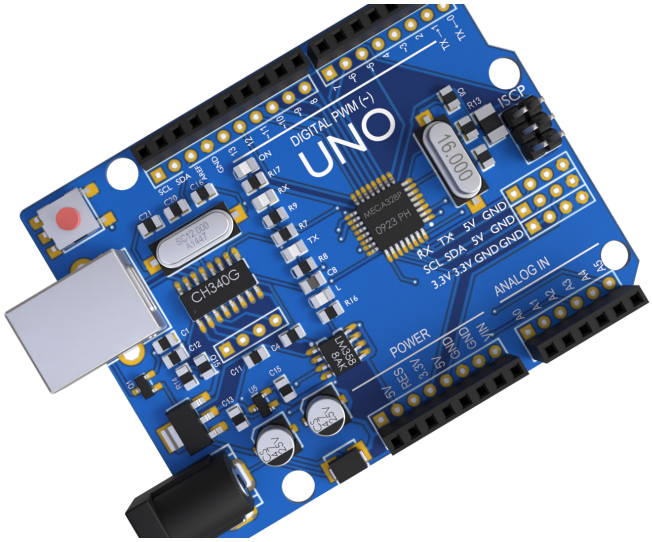
Figura 3 – Conexão dos pinos SCL e SDA do display OLED às portas A5 e A4 do Arduino



Fonte: SEED/DTI/CTE

As placas Arduino Uno possuem, em suas portas analógicas A4 e A5, os recursos SCL (*serial clock*) e SDA (*serial data*) para sincronização e comunicação serial entre dispositivos com conexões I2C, nas quais até 112 dispositivos são integrados com estas duas linhas de dados. Porém, algumas versões de Arduino Uno possuem uma extensão para essas portas (figura 4), que podem ser utilizadas para conexão do display OLED.

Figura 4 – Detalhe das portas SCL e SDA em versão do Arduino Uno R3



Fonte: SEED/DTI/CTE

As conexões do display OLED ao Arduino Uno, portanto, seguem uma tabela de conexões equivalentes (para outros modelos de Arduino, como Mega, as portas analógicas com os recursos SCL e SDA são outras) (tabela 1):

Tabela 1 - Conexão entre display OLED e Arduino

SDA	SCL	GND	VCC	**Pinos display OLED**
SDA ***ou*** A4	SCL ***ou*** A5	GND	5 V	**Portas Arduino**

Caso você prefira conectar o display OLED em outra superfície, personalizando mais ainda seu projeto, utilize jumpers macho-fêmea para as conexões, eliminando assim o uso da protoboard.

Finalizada a montagem, vamos à programação! Para iniciá-la, conecte a placa Arduino ao computador, através de um cabo USB, para que ocorra a comunicação entre a placa microcontroladora e o software Arduino IDE.

Agora, vamos programar por completo o nosso projeto Instax OLED!

Nesta programação, vamos imprimir em nossa Instax OLED a imagem do braço robótico do B1N0, exportado como bitmap com profundidade de 1 bit. Você pode experimentar gerar outras imagens e aplicar outras profundidades de bits para verificar como elas serão impressas.

Para programação do display OLED utilizando o software Arduino IDE ou o Arduino Editor Online, precisamos das bibliotecas **Adafruit SSD1306** e **Adafruit GFX Library**, desenvolvidas pela Adafruit, que permitem desenhar e imprimir imagens no display usando os dados do arquivo bitmap. Caso o Arduino IDE (versão Windows) indique que são necessárias outras bibliotecas para a utilização destas, clique na opção “instalar todas” (*Install all)*.

Na programação desta aula, o **void setup()** e o **void loop()** poderão permanecer inalterados. O que isso significa? Que novas imagens podem ser impressas apenas pela alteração dos dados da variável **const unsigned char**, no escopo da programação.

Sintaxe

const unsigned char var = val;

var: *nome da variável*

Parâmetros:

val: *valor a ser atribuído à variável*

*Na programação do projeto Instax OLED, a linha da constante **unsigned char** está acompanhada pela palavra-chave **PROGMEM**, que é um modificador de variáveis para que os dados sejam armazenados na memória flash do Arduino.*

const unsigned char B1N0 [] PROGMEM = { };

*Nas chaves, é onde colamos o array de bytes gerado pela ferramenta **Image2cpp**.*

Observe também que, como nossa proposta inicial é imprimir uma imagem no display, o **void loop()** está vazio e, no **void setup()**, temos apenas as funções para inicializar o display, limpar o buffer anterior e imprimir a imagem convertida em matriz e declarada no escopo da programação.

Incentivamos sempre que você analise as linhas dos códigos de programação compartilhados em nossas aulas para compreensão da sintaxe e autonomia sobre os projetos.

No software IDE, escreva ou copie e cole o código-fonte de programação (quadro 1), observando os comentários às linhas de programação para personalizar seu projeto quando desejar e imprimir outras fotografias.

Quadro 1 - Código-fonte da programação na linguagem do Arduino (Wiring)

```
/*****************************************/
/* Aula 06 – Instax OLED                 */
/*                                       */
/* Programação do projeto “Instax OLED” para */
/* exibição de imagens e fotografias.    */
/* Nesta programação, a imagem do braço robótico */
/* do B1N0 foi exportada como bitmap com */
/* profundidade de 1 bit.                */
/* Para gerar outras fotografias, exportar */
/* imagem bitmap e depois converter em   */
/* array de bytes pelo site              */
/* https://javl.github.io/image2cpp/     */
/*                                       */
/*****************************************/
/* Bibliotecas requeridas:               */
/*                                       */
/* - Adafruit SSD1306 by Adafruit        */
/* https://github.com/adafruit/Adafruit_SSD1306 */
/* http://librarymanager/All#SSD1306#Adafruit */
/*                                       */
/* - Adafruit GFX Library by Adafruit    */
/* https://github.com/adafruit/Adafruit-GFX-Library */
/* http://librarymanager/All#GFX#class#that */
/*                                       */
/* Caso o Arduino IDE (versão Windows) indique que */
/* são necessárias outras bibliotecas para utilizar a */
/* Adafruit SSD1306 e a Adafruit GFX, clique na */
/* opção “instalar todas” (Install all). */
/*****************************************/

// Inclusão das bibliotecas.
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Largura do display OLED, em pixels.
#define larguraDisplay 128

// Altura do display OLED, em pixels.
#define alturaDisplay 64

// Declaração SSD1306 display conectado para I2C (pinos SDA e SCL).
#define OLED_reset -1
// Reset pin # (ou -1 se compartilhar pino reset).

// Cria o objeto de controle “display” para o display OLED.
Adafruit_SSD1306 display(larguraDisplay, alturaDisplay, &Wire, OLED_reset);

// Array de bytes que contém a imagem 128x64px e está nomeada como B1N0.
// Para exibir outras imagens, substitua o array de bytes.
const unsigned char B1N0 [] PROGMEM = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x2e, 0xae, 0xa0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x02, 0x80, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x0a, 0xaa, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x01, 0x00, 0x54, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xa8, 0xaa, 0xaa, 0xaa, 0xab, 0xaa, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x05, 0x40, 0x40, 0x44, 0x44, 0x45, 0x45, 0x44, 0x54, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xe2, 0x2a, 0xaa, 0xaa, 0xaa, 0x2a, 0xaa, 0xea, 0x02, 0xa0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x05, 0x40, 0x00, 0x00, 0x04, 0x00, 0x40, 0x00, 0x50, 0x00, 0x00, 0x00, 0x00,
0x00, 0x02, 0x00, 0x2a, 0xaa, 0x00, 0x00, 0x0a, 0xaa, 0xaa, 0x80, 0x08, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x10, 0x10, 0x00, 0x00, 0x01, 0x01, 0x01, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0a, 0xaa, 0xaa, 0xaa, 0x8a, 0xaa, 0xaa, 0xab, 0xa2, 0xa8, 0x01, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x01, 0x04, 0x10, 0x11, 0x11, 0x10, 0x54, 0x00, 0x10, 0x00, 0x40, 0x00, 0x00, 0x00,
0x00, 0x2a, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0x8a, 0xaa, 0x80, 0xaa, 0x00, 0x20, 0x00, 0x00, 0x00,
0x00, 0x15, 0x44, 0x05, 0x44, 0x40, 0x00, 0x00, 0x04, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x2a, 0xa8, 0x0a, 0xaa, 0x00, 0x00, 0x00, 0x0a, 0xa0, 0x0a, 0xa0, 0x30, 0x00, 0x00, 0x00,
0x00, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0a, 0xa0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xa8, 0x02, 0xa8, 0x08, 0x00, 0x00, 0x00,
0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x10, 0x04, 0x00, 0x00, 0x00,
0x00, 0x0a, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x2a, 0x80, 0x2a, 0x88, 0x00, 0x00, 0x00,
0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x11, 0x04, 0x00, 0x00, 0x00,
0x00, 0x0a, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0xa0, 0x0a, 0xa2, 0x00, 0x00, 0x00,
0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x04, 0x00, 0x00, 0x00,
0x00, 0x2a, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xaa, 0x0a, 0xaa, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x2a, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x2a, 0xaa, 0xea, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x05, 0x14, 0x00, 0x00, 0x00,
0x02, 0xaa, 0xa8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0xaa, 0xaa, 0xa0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x01, 0x00, 0x00, 0x00,
0x0a, 0x80, 0x2a, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xaa, 0xab, 0xa0, 0x00, 0x00,
0x04, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x44, 0x45, 0x40, 0x00, 0x00,
0x2a, 0x00, 0x0a, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x2a, 0xae, 0xae, 0xa0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x54, 0x00, 0x00, 0x00,
0x08, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0xaa, 0xaa, 0xaa, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00,
0x08, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x2a, 0xaa, 0xaa, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x01, 0x10, 0x00, 0x00, 0x00, 0x00,
0x08, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0a, 0xaa, 0xaa, 0xa0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x40, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xaa, 0xea, 0xaa, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xaa, 0xaa, 0xa0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x14, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xaa, 0x80, 0x00, 0x00, 0x00, 0x00, 0x02, 0xaa, 0xba, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x55, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xaa, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0xaa, 0xaa, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x55, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x44, 0x44, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xaa, 0x80, 0x00, 0x00, 0x00, 0x00, 0x0a, 0xaa, 0xaa, 0xa0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x55, 0x40, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xaa, 0x80, 0x00, 0x00, 0x00, 0x00, 0x0e, 0xee, 0xee, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x55, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xaa, 0x80, 0x00, 0x00, 0x00, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0x80, 0x00, 0x00, 0x00, 0x00,
0x00, 0x55, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0x00, 0x00, 0x00,

```
  0x55, 0x55, 0x55, 0x55, 0x55, 0x54, 0x44, 0x44, 0x44, 0x44, 0x44, 0x55, 0x55, 0x00, 0x00, 0x00,
  0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xa8, 0x00, 0x00,
  0x44, 0x44, 0x44, 0x44, 0x44, 0x44, 0x44, 0x44, 0x44, 0x44, 0x44, 0x44, 0x44, 0x44, 0x00, 0x00,
  0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0x00, 0x00,
  0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x00, 0x00,
  0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0x00, 0x00
};

// Configurações do display e impressão da imagem.
void setup()
{
  // Inicializa o display pelo protocolo I2C e endereço 0x3C .
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  // Limpa o buffer anterior.
  display.clearDisplay();

   // Parâmetros da função drawBitmap(x, y, array de bytes, largura display, altura display, cor);
  display.drawBitmap(0, 0, B1N0, 128, 64, WHITE);

  // Exibe a imagem bitmap armazenada no buffer.
  display.display();
}

/* Para exibição de imagem estática, utilize as funções drawBitmap() e display() no setup da pro-
gramação. */
void loop() {
}
```

Com o código-fonte inserido no Arduino IDE, compile o programa e carregue-o ao Arduino.

A programação desta aula exibirá a fotografia do braço robótico do B1N0 e, para exibir outras fotografias, converta suas imagens bitmap para gerar o código que o Arduino entenderá, lembrando de copiar este código e colá-lo no sketch, dentro de um array de bytes e, pelas funções **display.drawBitmap(**x, y, bitmap, w, h, color**)** e **display.display()** da biblioteca **Adafruit SSD1306**, pedir para o Arduino desenhar a imagem e exibi-la no display.

A função **display.drawBitmap()** possui as seguintes descrições de parâmetros:

- **x** e **y**: coordenadas do canto superior esquerdo da imagem no display.
- **bitmap**: o nome do array de bytes, ou seja, da sequência de dados que contém a imagem. *No exemplo da nossa programação, demos o nome de B1N0.*
- **w** e **h**: largura e altura da imagem em pixels.
- **color**: cor dos pixels da imagem, identificada por WHITE ou BLACK.

A função **display.display()** não possui parâmetros e deve ser chamada após as funções de desenho (**draw**) para que as imagens sejam exibidas. A função não é necessária após cada comando de desenho, em outros projetos, a menos que se deseje, pois é possível agrupar várias operações de desenho e atualizar a tela de uma só vez chamando a função **display.display()**. Na programação do nosso projeto, utilizamos apenas uma função de desenho.

Após a transferência do programa para o Arduino, o display OLED exibirá a imagem! Observe que sua constituição é conforme a iluminação de pixels. Dependendo do modo como você exporta a imagem bitmap, indicando a quantidade de bits, a imagem poderá parecer mais ou menos preenchida. E o resultado também depende de como a conversão para o array de bytes é feita. Por isso, siga explorando a ferramenta **Image2cpp** para você personalizar ainda mais seus projetos e criar imagens incríveis para o seu projeto Instax OLED, usando suas próprias fotos ou qualquer outra imagem que você quiser.

Até a próxima!

Desafios:

Que tal converter outros estilos de imagens e fotografias para serem impressas no display OLED?

Que tal criar uma animação com as imagens impressas no display OLED? Para isso, crie no escopo duas ou mais variáveis **const unsigned char** para cada imagem e, no **void loop()**, peça para que o Arduino exiba cada uma por um intervalo de tempo, gerando assim o efeito animado.

Que tal desenhar e recortar em uma superfície uma imagem que se relacione às câmeras instantâneas com um orifício do tamanho do display OLED? Assim, você pode encaixá-lo para exibir suas fotografias impressas.

E se...

O projeto não funcionar - atente para alguns possíveis erros:

- Verifique se os jumpers estão nos pinos certos e na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique as conexões SCL e SDA do display OLED ao Arduino;
- Verifique se a programação está adequada ao display;
- Verifique a conversão da imagem bitmap em matriz e correta declaração no escopo da programação.

3. Feedback e finalização

Confira, compartilhando seu projeto com os demais colegas, se o objetivo de impressão de imagens e fotografias no display OLED foi alcançado. Compartilhem também suas experiências sobre todo o processo, desde a montagem do protótipo até as análises mais específicas sobre a ferramenta **Image2cpp** e as linhas de programação, refletindo em como isso pode contribuir para a criação de outros projetos.

REFERÊNCIAS

ARDUINO. Documentação de Referência da Linguagem do Arduino. Disponível em: https://www.arduino.cc/reference/pt/. Acesso em: 04 mar de 2024.

AUTOCORE ROBÓTICA. **Conhecendo o protocolo I2C com Arduino**. Disponível em: https:// autocorerobotica.blog.br/conhecendo-oprotocolo-i2c-com-arduino/. Acesso em: 15 ago de 2023.

CATARINA. Adair Santana. **Discretização de cores: Quantização**. Disponível em: https://www.inf.unioeste.br/~adair/PID/Notas%20Aula/Discretizacao%20de%20Cores%20-%20Quantizacao.pdf. Acesso em: 11 mar. 2024.

ELETROGATE. **Guia Completo do Display OLED (parte 1) – O que é? Como funciona?** Disponível em https://blog.eletrogate.com/guia-completo-do-display-oled-parte-1-o-que-e-como-funciona-2/. Acesso em: 14 ago de 2023.

ELETROGATE. **Guia Completo do Display OLED (parte 2) – Como programar?** Disponível em https://blog.eletrogate.com/guia-completo-do-display-oled-parte-2-como-programar-3/. Acesso em: 17 ago de 2023.

GITHUB. **Image2cpp, by Jasper van Loenen**. Disponível em: https://github.com/javl/image2cpp. Acesso em: 26 fev. 2024.

SURMA. **Ditherpunk – The article I wish I had about monochrome image dithering**. Disponível em: https://surma.dev/things/ditherpunk/. Acesso em: 11 mar. 2024.

DIRETORIA DE TECNOLOGIAS E INOVAÇÃO (DTI)

COORDENAÇÃO DE TECNOLOGIAS EDUCACIONAIS (CTE)

EQUIPE ROBÓTICA PARANÁ

Ailton Lopes

Andrea da Silva Castagini Padilha

Cleiton Rosa

Darice Alessandra Deckmann Zanardini

Edgar Cavalli Junior

Edna do Rocio Becker

José Feuser Meurer

Kellen Pricila dos Santos Cochinski

Marcelo Gasparin

Michele Serpe Fernandes

Michelle dos Santos

Roberto Carlos Rodrigues

Os materiais, aulas e projetos da “Robótica Paraná” foram produzidos pela Coordenação de Tecnologias Educacionais (CTE), da Diretoria de Tecnologia e Inovação (DTI), da Secretaria de Estado da Educação Paraná (SEED), com o objetivo de subsidiar as práticas docentes com os estudantes por meio da Robótica.

Este material foi produzido para uso didático-pedagógico exclusivo em sala de aula.