TECNOLÓGICO
NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE ENERO JUNIO - 2024

INGENIERÍA EN SISTEMAS COMPUTACIONALES

LENGUAJES DE INTERFAZ

PROYECTO FINAL - DOCUMENTACIÓN

JULIO ALEJANDRO HERNÁNDEZ LEÓN - 21211963
SANTY FRANCISCO MARTINEZ CASTELLANOS - 21211989
LUIS ROBERTO LEAL LUA - 21211970

RENÉ SOLIS REYES

Índice

Código Utilizado y Descripción

```
// OV767X - Captura de bytes crudos de la cámara
// Team Debian
// Proyecto Final
// Martinez Castellanos Santy Francisco
// Leal Lua Luis Roberto
// Hernandez Leon Julio Hernandez

#include <Arduino_OV767X.h>  // Incluye la librería para controlar la cámara OV767X

int bytesPerFrame;  // Variable para almacenar el número de bytes por cuadro

// Array para almacenar los datos de la imagen capturada
// Tamaño QVGA (320x240) X 2 bytes por píxel (formato RGB565)
byte data[176 * 144 * 2];  // El tamaño se ajusta a QCIF (176x144)

void setup() {
  Serial.begin(460800 );  // Inicializa la comunicación serie a 460800 baudios
  while (!Serial);  // Espera a que se establezca la comunicación serie

  // Inicializa la cámara con resolución QCIF, formato RGB565 y sin escalado
  if (!Camera.begin(QCIF, RGB565, 1)) {
    Serial.println("Failed to initialize camera!");  // Mensaje de error si la cámara no se inicializa
    while (1);  // Detiene la ejecución en caso de error
  }

  // Calcula los bytes por cuadro: ancho x alto x bytes por píxel
  bytesPerFrame = Camera.width() * Camera.height() * Camera.bytesPerPixel();
```

```
  Camera.testPattern();  // Establece un patrón de prueba en la cámara para verificar su funcionamiento
}

void loop() {
  Camera.readFrame(data);  // Lee un cuadro de la cámara y lo almacena en el array data

  Serial.write(data, bytesPerFrame);  // Envía los datos de la imagen capturada a través del puerto serie
}
```

Código en Processing

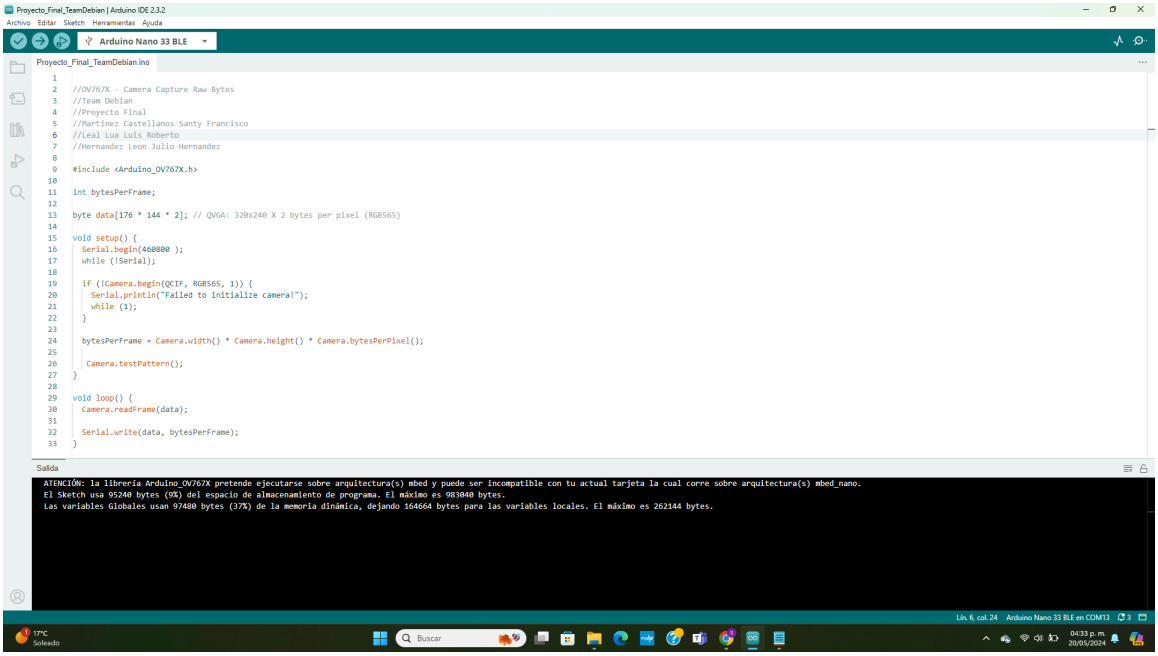
```
//Proyecto Final
//Team Debian
//Integrantes:
//Santy Francisco Martinez Castellanos - 21211989
//Julio Alejandro Hernández León -21211963
//Luis Roberto Leal Lua - 21211970

import processing.serial.*;  // Importa la biblioteca para la comunicación serial
import java.nio.ByteBuffer;  // Importa ByteBuffer para manejar los bytes de manera eficiente
import java.nio.ByteOrder;  // Importa ByteOrder para especificar el orden de los bytes

Serial myPort;  // Crea un objeto Serial para gestionar la comunicación serial

// Constantes para especificar las dimensiones y formato de los datos de la imagen
final int cameraWidth = 176;  // Ancho de la imagen capturada por la cámara
final int cameraHeight = 144;  // Alto de la imagen capturada por la cámara
final int cameraBytesPerPixel = 2;  // Bytes por píxel, RGB565 utiliza 2 bytes
final int bytesPerFrame = cameraWidth * cameraHeight * cameraBytesPerPixel;
// Total de bytes por imagen

PImage myImage;  // Imagen donde se mostrarán los datos
byte[] frameBuffer = new byte[bytesPerFrame];  // Buffer para almacenar los datos de un frame completo

void setup() {
  size(320, 240);  // Establece el tamaño de la ventana de la aplicación
```

```
  // Inicializa el puerto serial para comunicarse con la cámara
    myPort = new Serial(this, "COM13", 460800);  // Configura el puerto y la velocidad de transmisión

   // Configura el buffer de serial para que espere recibir un frame completo de bytes
  myPort.buffer(bytesPerFrame);

  // Crea un objeto PImage con el tamaño especificado y formato de color RGB
  myImage = createImage(cameraWidth, cameraHeight, RGB);
}

void draw() {
  image(myImage, 0, 0);  // Dibuja la imagen en la ventana de la aplicación
}

void serialEvent(Serial myPort) {
  // Lee los bytes recibidos por el puerto serial al buffer de frame
  myPort.readBytes(frameBuffer);

  // Prepara el ByteBuffer para manipular los bytes del frame
  ByteBuffer bb = ByteBuffer.wrap(frameBuffer);
     bb.order(ByteOrder.BIG_ENDIAN);     // Especifica que los bytes más significativos vienen primero

  int i = 0;  // Índice para iterar sobre los píxeles de la imagen

  // Mientras haya bytes restantes en el buffer
  while (bb.hasRemaining()) {
    // Lee un píxel en formato RGB565
    short p = bb.getShort();
```

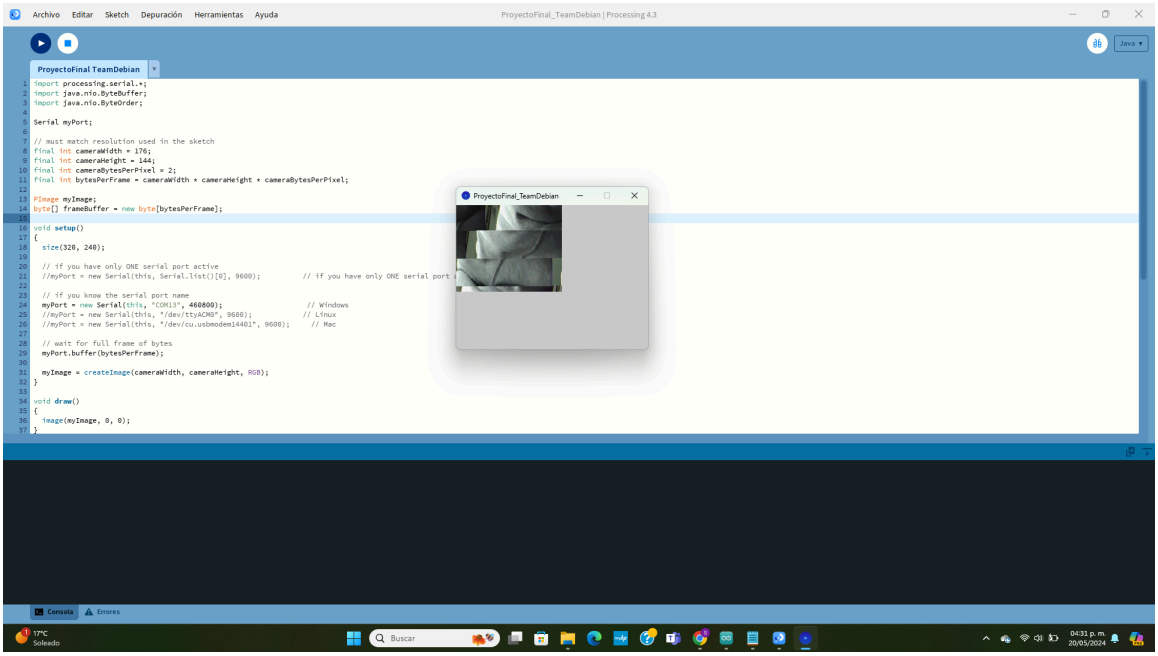
```
    // Convierte RGB565 a RGB888 para obtener los valores de color en 8 bits
    int r = ((p >> 11) & 0x1f) << 3;  // Extrae y convierte el componente rojo
    int g = ((p >> 5) & 0x3f) << 2;   // Extrae y convierte el componente verde
    int b = ((p & 0x1f) << 3);        // Extrae y convierte el componente azul

    // Asigna el color al píxel correspondiente en la imagen
    myImage.pixels[i++] = color(r, g, b);
  }
  // Actualiza la imagen con los nuevos píxeles procesados
  myImage.updatePixels();
}
```

Fotografías

Funcionalidad del arduino para el funcionamiento correcto del programa.

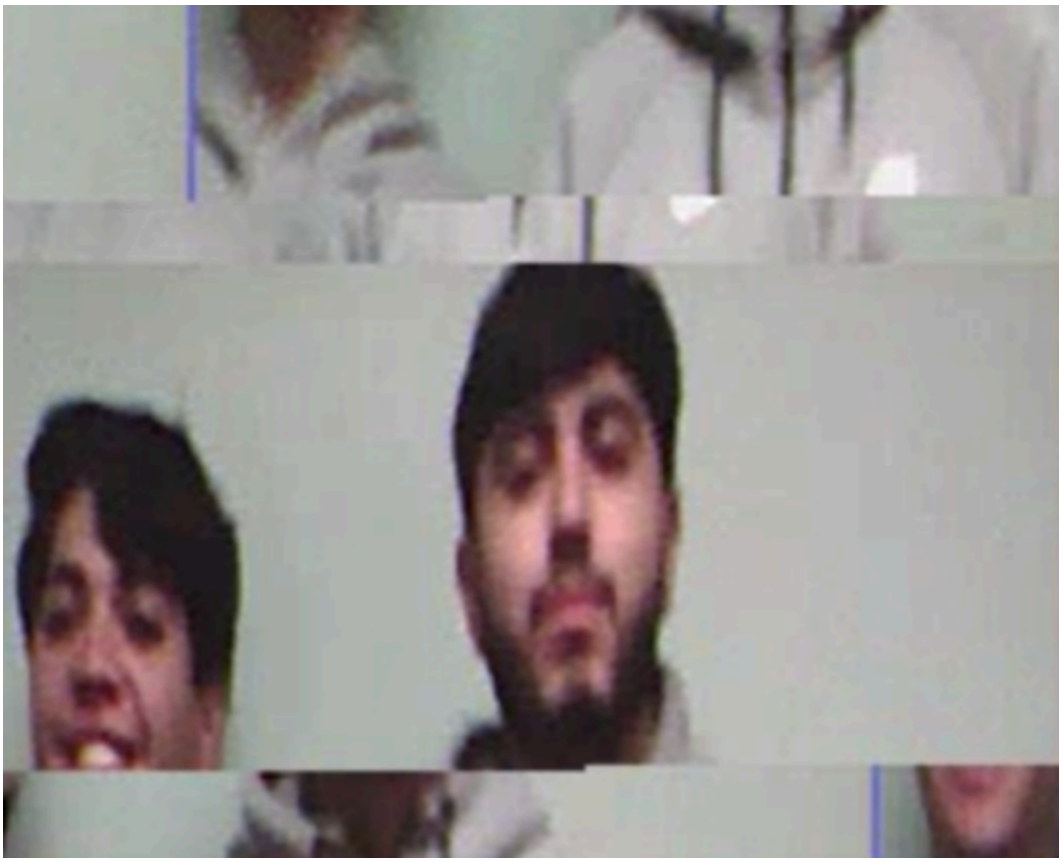


Captura de pantalla de programa Processing en el cuál se despliega otra ventana para mostrar la cámara y poder hacer la toma de la fotografía



Captura de fotografías

En la primer imágen se muestran los integrantes Julio Hernández (Izquierda) y Luis Lua (Derecha)



En la segunda imágen se muestran los integrantes Julio Hernández (Izquierda) y Luis Lua (Derecha)

En la tercer imágen se muestra al integrante Santy Martinez