Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Organización Computacional
Sección B
Ing. Otto Rene Escobar Leiva
Auxiliar: Carlos Rangel

PROYECTO 1

Plotter Serial

Integrantes

Grupo 3

Nombre	Carné
Oswaldo Antonio Choc Cuteres	201901844
Jencer Hamilton Hernández Alonzo	202002141
Cristian Raúl Vega Rodríguez	202010942
Javier Andrés Monjes Solórzano	202100081
Angel Isaías Mendoza Martínez	202180003
Estephanie Alejandra Ruíz Pérez	202201318
Edin Rafael Santizo Barrera	202202072
Juan Pascual Itzep Coguox	202202161
Rocío Samaí López Vásquez	201709035
Diego Andrés Dubón Samayoa	202202429
Juan José Almengor Tizol	202212209

Guatemala, 26 de abril 2024

Índice

Introducción

Un plotter es un periférico de computadora que permite dibujar o representar diagramas y gráficos. Existen plotters monocromáticos y de cuatro, ocho o doce colores. El plotter funciona mediante el movimiento de plumas sobre el papel. Cuando la máquina debe realizar un trazo complejo, hace el dibujo muy lentamente debido al movimiento mecánico de las plumas. Esta lógica de funcionamiento hace que los plotters no sean adecuados para pintar superficies, ya que deben pasar las plumas en numerosas ocasiones. En cambio son útiles para la delineación. Las plumas se encuentran dentro de un tambor. El plotter dispone de dos motores paso a paso, que se mueven por el eje X (a lo ancho del papel) y por el eje Y (con movimiento vertical de las plumas o generando el movimiento del papel). Los motores stepper o paso a paso son motores los cuales están hechos para producir movimientos más precisos que los motores DC, estos existen de 2 tipos los cuales son: unipolares y bipolares, a los cuales se les tiene que ingresar una secuencia determinada para moverlos.

Por medio de la lógica del funcionamiento del plotter, se desarrollará una impresora innovadora y un software especializado que permita realizar impresiones de alta calidad con diseños personalizados, satisfaciendo así las necesidades de la universidad y proporcionando una solución tecnológica avanzada para la comunidad educativa.

Descripción del Problema

La Escuela de Ciencias y Sistemas está organizando una demostración de proyectos de innovación en la cual nosotros como alumnos del curso de Organización Computacional, para dicho fin como equipo desarrollamos un proyecto, un nuevo tipo de impresora nada tradicional, dicha impresora será controlada por un software especial diseñado por su nosotros y la cual será controlada desde un PC por medio del puerto (“Serial/Paralelo”).

Lo que se busca como finalidad es que se tenga un sitio web con el juego de totito el cual contendrá 4 figuras y 4 colores. El Tipo de impresora a desarrollar será “Cuadri-Color” la cual será descrita a continuación:

Requerimientos

Se elaboraron varios circuitos combinacionales y secuenciales, los cuales serán capaces de manipular un sistema de ejes “X” y “Y” para el prototipado de un “Plotter” el cual replicará en una Hoja de Papel Bond mediante un tipo de lápiz lo modelado en una aplicación de escritorio.

Lógica del Sistema

Aplicación

Se desarrolló una aplicación que cuenta con una interfaz gráfica y un lienzo en el cual se puedan realizar los dibujos utilizando el mouse del pc a modo de PixelArt en una matriz de 3 x 3, la aplicación cuenta con las opciones generales como “Archivo” (“abrir”, “nuevo” y “guardar”), “Ayuda”, “Analizar Entrada” e “Imprimir Matriz”, la extensión del archivo debe ser “.orga”. La aplicación también cuenta con una serie de figuras predefinidas.

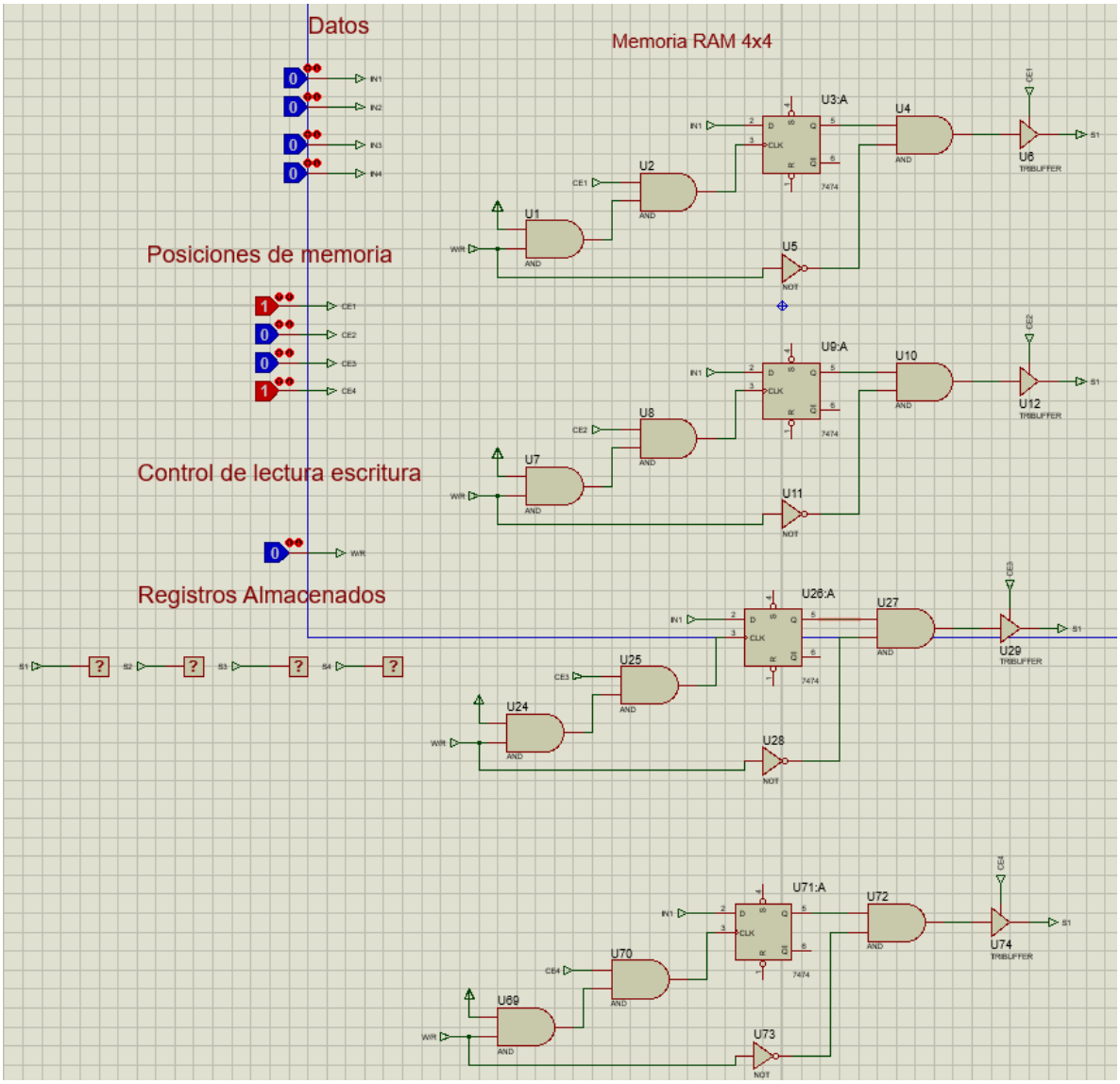
La conexión que se implementó por medio del puerto serial que se utilizó con la interfaz de envío y recepción de datos en forma serial RS232, este se comunica con el fronted mediante el pyserial quien enviará la información a los registros elaborados con flip-flops. La transmisión de datos se detallará con el envío desde la PC hacia el controlador de los motores del Plotter.

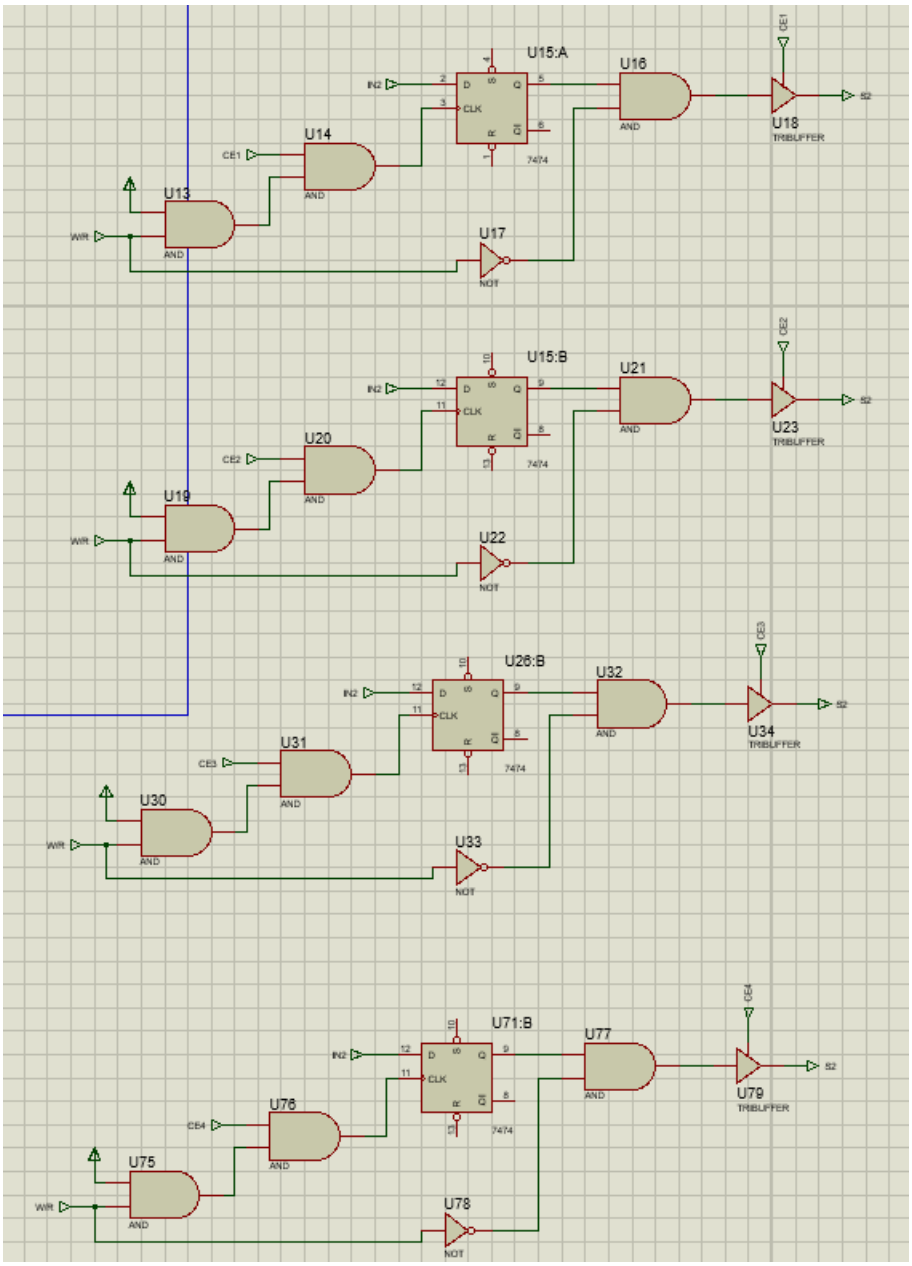
La conexión TX y RX; conecta el pin TX (transmisión) del módulo RS232 al pin RX1 (pin 19) del Arduino Mega y el RX (recepción) del módulo al pin TX1 (pin 18) del Arduino Mega. Esto permite la comunicación bidireccional. Se conectaron los pines de alimentación del módulo RS232 a las salidas correspondientes del Arduino Mega, generalmente 5V y GND. Y para la conexión de la led, esta se conecta al pin 13 y GND del Arduino, con una resistencia adecuada en serie para limitar la corriente (utilizada entre 220 y 470 ohmios).

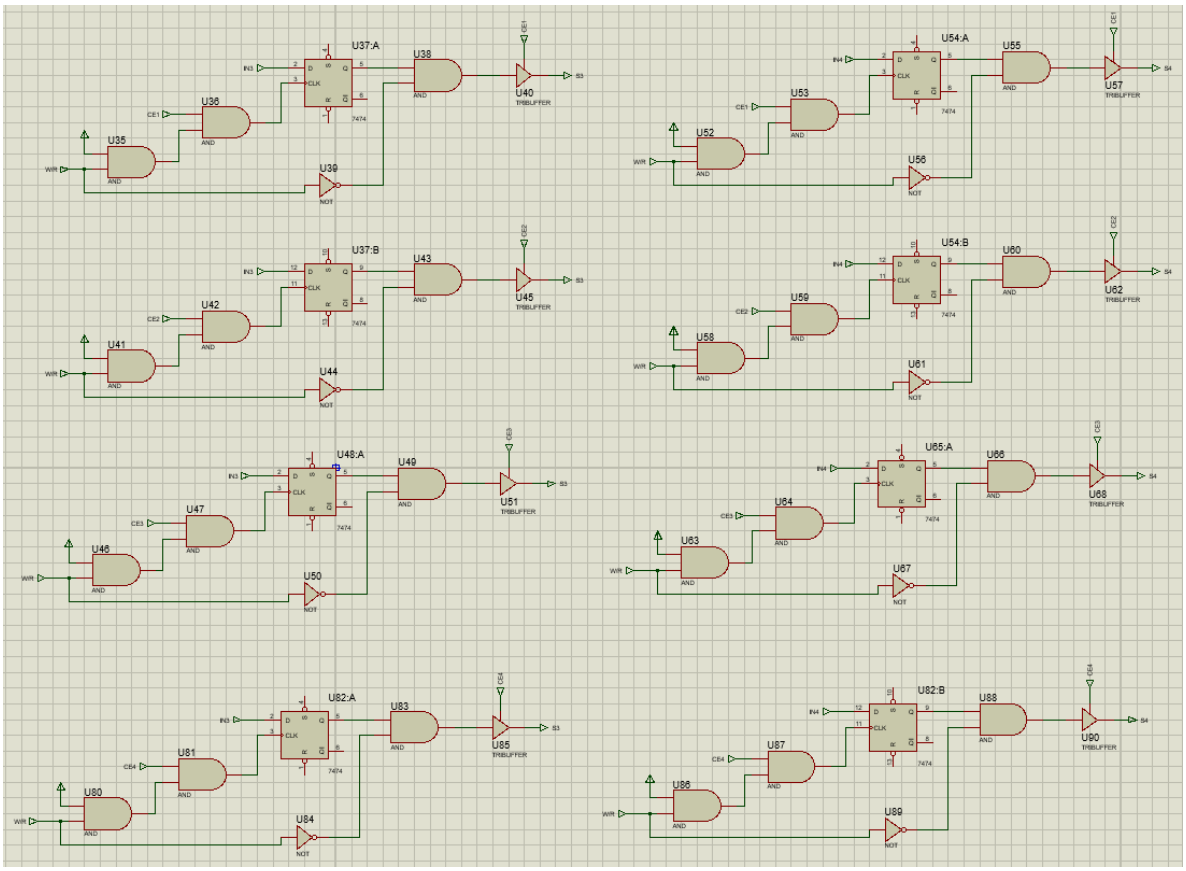
La impresora que se realizó tiene un cabezal de impresión que se desplaza por el área de impresión, esta es capaz de almacenar 3 coordenadas a pintar por medio de la matriz de flip-flop simulada por la memoria RAM, la cual será programada por la PC para luego empezar a imprimir por medio de 2 lectores de CD, también para los controladores del disco se usó un módulo puente H for stepper motor, 2ª, 25w.

Diagramas del Diseño del Circuito

Memoria de Acceso Aleatoria.




U15:A
U16
U18
TRIBUFFER
U14
U13
U17
U15:B
U21
U23
TRIBUFFER
U20
U19
U22
U26:B
U32
U34
TRIBUFFER
U31
U30
U33
U71:B
U77
U79
TRIBUFFER
U76
U75
U78
AND
NOT
7474
CLK
IN2
WR
CE1
CE2
CE3
CE4
OE1
OE2
OE3
OE4
S2


U37:A
U38
U40
TRBUFFER
U36
U35
U39
NOT
7474
AND
U54:A
U55
U57
U53
U52
U56
U37:B
U43
U45
U42
U41
U44
U54:B
U60
U62
U59
U58
U61
U48:A
U49
U51
U47
U46
U50
U65:A
U66
U68
U64
U63
U67
U82:A
U83
U85
U81
U80
U84
U82:B
U88
U90
U87
U86
U89

Código comentado

BACKEND

Main.py

proyecto > Backend > Main.py > ...

Estephanie Alejandra Ruiz, 3 days ago | 1 author (Estephanie Alejandra Ruiz)

```python
import serial
import time

# Configura el puerto serial para que coincida con la configuración mostrada en tu monitor serial.
ser = serial.Serial('COM9', 9600)
time.sleep(2)  # Espera a que se establezca la conexión serial

def toggle_led(command):
    """
    Envia el comando al Arduino para controlar el LED.
    'A' para encender, 'a' para apagar.
    """
    ser.write(command.encode())  # Envía el comando al Arduino

try:
    while True:
        cmd = input("Ingrese 'A' para encender el LED o 'a' para apagarlo (Ingrese 'salir' para terminar): ")
        if cmd in ['a', 'A']:
            toggle_led(cmd)
        elif cmd == 'salir':
            print("Saliendo...")
            break
        else:
            print("Comando no reconocido. Intente nuevamente.")
finally:
    ser.close()  # Cierre de conexion con el arduino

```

Api.py

proyecto > Backend > Api.py > serve_frontend

Estephanie Alejandra Ruiz, 3 days ago | 1 author (Estephanie Alejandra Ruiz)

```python
from flask import Flask, request, jsonify, send_from_directory
from flask_cors import CORS

import Interprete.Gramatica as Gm
import json
import os

#import pyserial

app = Flask(__name__)
CORS(app)  # Habilitar CORS para la aplicación Flask

# Ruta para servir los archivos estáticos del frontend
@app.route('/', defaults={'path': ''})      Estephanie Alejandra Ruiz, 3 days ago • lectura-carga de archivos ...
@app.route('/<path:path>')
def serve_frontend(path):
    if path == "":
        return send_from_directory('../Frontend', 'Page.html')
    elif os.path.exists(os.path.join('../Frontend', path)):
        return send_from_directory('../Frontend', path)
    else:
        return send_from_directory('../Frontend', 'Page.html')

@app.route('/plotterserial/analizar', methods=['POST'])
def analizar():

    datos = request.json
    codigo_recibido = datos.get('texto', '')

    if codigo_recibido == '':
        return jsonify({'error': 'No se recibió código'})

    parse = Gm.parse(codigo_recibido)

    if not parse:
        return jsonify({
```

Api.py

proyecto > Backend > Api.py > analizar

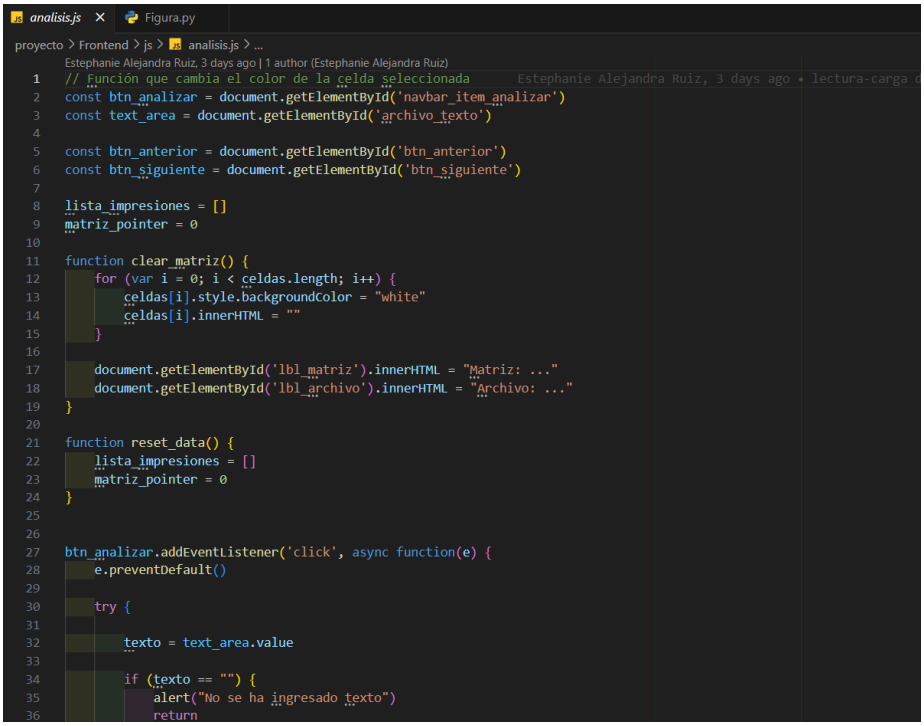
```
            'mensaje': 'Error en el análisis',
            'impresiones': []
        })

    sentencias = []
    for sent in parse:
        sentencias.append(json.dumps(sent.to_dict()))

    try:
        return jsonify({
            'mensaje': 'Análisis terminado',
            'impresiones': sentencias
        })

    except Exception as e:
        print(e)
        return jsonify({
            'mensaje': 'Error en el análisis',
            'impresiones': []
        })

@app.route('/plotterserial/graficar', methods=['POST'])
def graficar():

    datos = request.json
    matriz_recibido = datos.get('matriz', '')

    if not matriz_recibido:
        return jsonify({'error': 'No se recibió matriz'})


    for item in matriz_recibido:
        fila = item['pos']['fila']
        columna = item['pos']['columna']
        color = item['color']
        figura = item['figura']
        print(f"Fila: {fila}, Columna: {columna}, Color: {color}, Figura: {figura}")
```

Api.py

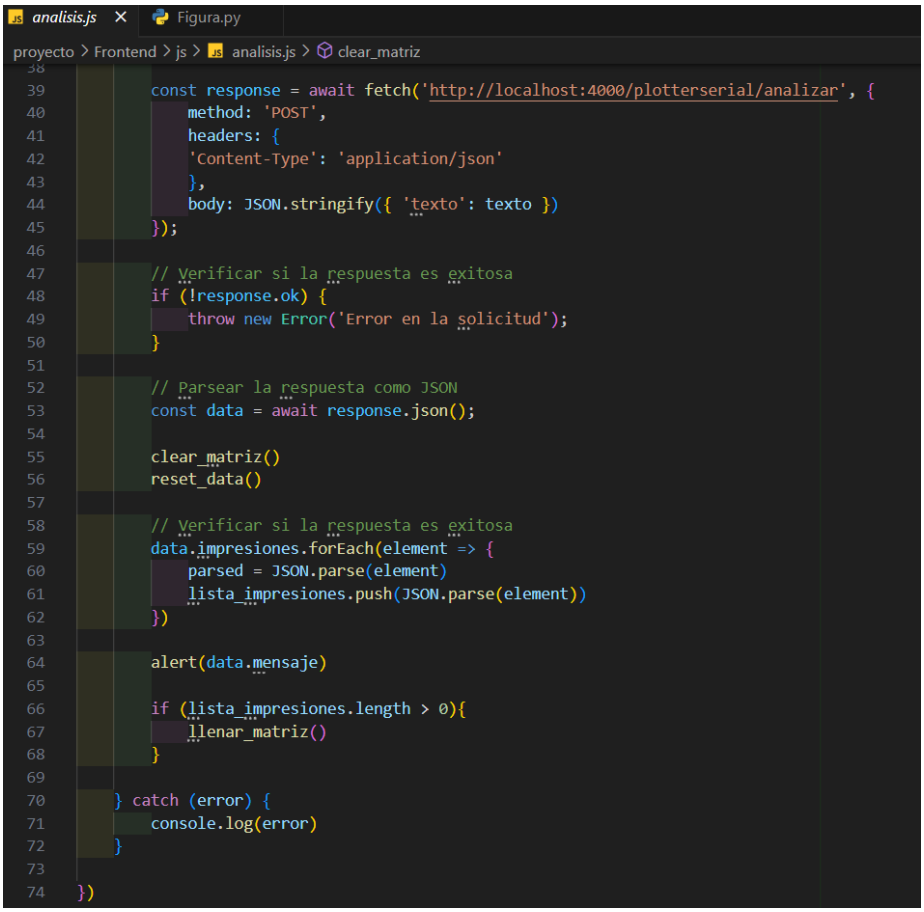
proyecto > Backend > Api.py > ...

```
        fila = item['pos']['fila']
        columna = item['pos']['columna']
        color = item['color']
        figura = item['figura']
        print(f"Fila: {fila}, Columna: {columna}, Color: {color}, Figura: {figura}")
        print("")

        # TODO: Implementar logica impresora con pyserial


    return jsonify({'mensaje': 'Graficación terminada'})

if __name__ == '__main__':

    # Ejecutar la aplicación Flask
    app.run(debug=True, port=4000)      Estephanie Alejandra Ruiz, 3 days ago • lectura-carga
```

FRONTEND



```
// Función que cambia el color de la celda seleccionada
const btn_analizar = document.getElementById('navbar_item_analizar')
const text_area = document.getElementById('archivo_texto')

const btn_anterior = document.getElementById('btn_anterior')
const btn_siguiente = document.getElementById('btn_siguiente')

lista_impresiones = []
matriz_pointer = 0

function clear_matriz() {
    for (var i = 0; i < celdas.length; i++) {
        celdas[i].style.backgroundColor = "white"
        celdas[i].innerHTML = ""
    }

    document.getElementById('lbl_matriz').innerHTML = "Matriz: ..."
    document.getElementById('lbl_archivo').innerHTML = "Archivo: ..."
}

function reset_data() {
    lista_impresiones = []
    matriz_pointer = 0
}


btn_analizar.addEventListener('click', async function(e) {
    e.preventDefault()

    try {

        texto = text_area.value

        if (texto == "") {
            alert("No se ha ingresado texto")
            return
```



```
        const response = await fetch('http://localhost:4000/plotterserial/analizar', {
            method: 'POST',
            headers: {
            'Content-Type': 'application/json'
            },
            body: JSON.stringify({ 'texto': texto })
        });

        // Verificar si la respuesta es exitosa
        if (!response.ok) {
            throw new Error('Error en la solicitud');
        }

        // Parsear la respuesta como JSON
        const data = await response.json();

        clear_matriz()
        reset_data()

        // Verificar si la respuesta es exitosa
        data.impresiones.forEach(element => {
            parsed = JSON.parse(element)
            lista_impresiones.push(JSON.parse(element))
        })

        alert(data.mensaje)

        if (lista_impresiones.length > 0){
            llenar_matriz()
        }

    } catch (error) {
        console.log(error)
    }

})
```

archivos.js Figura.py

proyecto > Frontend > js > archivos.js > ...

Estephanie Alejandra Ruiz, 3 days ago | 1 author (Estephanie Alejandra Ruiz)

```js
const navbar_item_abrir = document.getElementById('navbar_item_abrir')
const navbar_item_nuevo = document.getElementById('navbar_item_nuevo')
const navbar_item_guardar = document.getElementById('navbar_item_guardar')
const file_input = document.getElementById('file_input')
const archivo_texto = document.getElementById('archivo_texto')


navbar_item_abrir.addEventListener('click', (e) => {
    e.preventDefault()
    console.log("abriendo...")

    file_input.click()
    //


})

file_input.addEventListener('change', (e) => { //cuando se selecciona un archivo

    const file = e.target.files[0]

    if (file) {
        const reader = new FileReader()

        const fileName = file.name
        // Verificar que el archivo sea .olc
        const extension = fileName.split('.').pop()
        if (extension !== 'orga' && extension !== 'txt') {
            alert('El archivo seleccionado no es un archivo .orga')
            return
        }

        reader.onload = (event) => {
            const fileContent = event.target.result
            archivo_texto.value = (fileContent)
            document.getElementById('lbl_archivo').innerHTML = "Archivo: " + file.name
        }
```

archivos.js Figura.py

proyecto > Frontend > js > archivos.js > ...

```js
            reader.readAsText(file)
        }

        file_input.value = null
    })

    navbar_item_nuevo.addEventListener('click', (e) => {
        e.preventDefault()
        console.log("nuevo...")
        archivo_texto.value = ''
        clear_matriz()
        reset_data()
    })

    navbar_item_guardar.addEventListener('click', (e) => {
        e.preventDefault()

        // Obtener el nombre del sessionStorage
        const nombreArchivo = 'archivo.orga'

        const contenido = archivo_texto.value
        if (contenido === '') {
            alert('No hay código para guardar')
            return
        }

        console.log("guardando...")
        guardarArchivo(nombreArchivo, contenido)
    })

    function guardarArchivo(nombre, contenido) {
        const blob = new Blob([contenido], { type: 'text/plain' })
        const url = URL.createObjectURL(blob)

        const a = document.createElement('a')
        a.href = url
        a.download = nombre
```

matriz.js Figura.py

proyecto > Frontend > js > matriz.js > ...

Estephanie Alejandra Ruiz, 3 days ago | 1 author (Estephanie Alejandra Ruiz)

```
1   Estephanie Alejandra Ruiz, 3 days ago • lectura-carga de archivos ...
2   let color = "cyan" // Color de la celda seleccionada
3   let celda = "celda00" // Celda seleccionada por defecto
4   let figura = "circulo" // Imagen de la celda seleccionada
5
6   // Función que cambia el color de la celda seleccionada
7   const celdas = document.getElementsByClassName("celda_mat")
8   const lbl_color_selec = document.getElementById("color_seleccionado")
9   const lbl_figura_selec = document.getElementById("figura_seleccionado")
10
11
12  // Cuando se hace click en una celda, se llama a la función que la cambia y se pasa su id
13  for (var i = 0; i < celdas.length; i++) {
14      celdas[i].addEventListener("click", function() {
15          celda = this.id
16          cambiar_color()
17          cambiar_imagen()
18      });
19  }
20
21  // Función que cambia el color de la celda seleccionada
22  function cambiar_color() {
23      document.getElementById(celda).style.backgroundColor = color
24  }
25
26
27  // Función que cambia la imagen de la celda seleccionada
28  function cambiar_imagen() {
29      document.getElementById(celda).innerHTML = "";
30
31      if (figura == "") {
32          return
33      }
34
35      var img = document.createElement("img")
36      img.src = "/images/" + figura + ".png"
```

matriz.js Figura.py

proyecto > Frontend > js > matriz.js > ...

```
37      document.getElementById(celda).appendChild(img)
38  }
39
40  // Funciones que cambian el color de la celda seleccionada
41  document.getElementById("col_cyan").addEventListener('click', function(e) {
42      e.preventDefault()
43      color = "cyan"
44      lbl_color_selec.innerHTML = "Color: Cyan"
45  })
46  document.getElementById("col_yellow").addEventListener('click', function(e) {
47      e.preventDefault()
48      color = "yellow"
49      lbl_color_selec.innerHTML = "Color: Amarillo"
50  })
51  document.getElementById("col_magenta").addEventListener('click', function(e) {
52      e.preventDefault()
53      color = "magenta"
54      lbl_color_selec.innerHTML = "Color: Magenta"
55  })
56  document.getElementById("col_black").addEventListener('click', function(e) {
57      e.preventDefault()
58      color = "black"
59      lbl_color_selec.innerHTML = "Color: Negro"
60  })
61
62  // Funciones que cambian la imagen de la celda seleccionada
63  document.getElementById("fig_circulo").addEventListener('click', function(e) {
64      e.preventDefault()
65      figura = "circulo"
66      lbl_figura_selec.innerHTML = "Figura: Circulo"
67  })
68  document.getElementById("fig_triangulo").addEventListener('click', function(e) {
69      e.preventDefault()
70      figura = "triangulo"
71      lbl_figura_selec.innerHTML = "Figura: Triángulo"
72  })
73  document.getElementById("fig_equis").addEventListener('click', function(e) {
```

matriz.js Figura.py

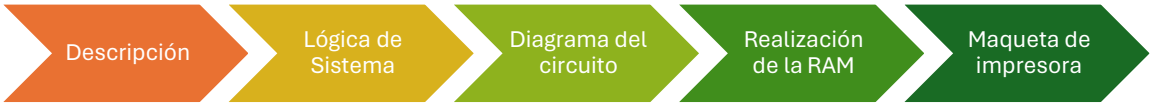
proyecto > Frontend > js > matriz.js > ...

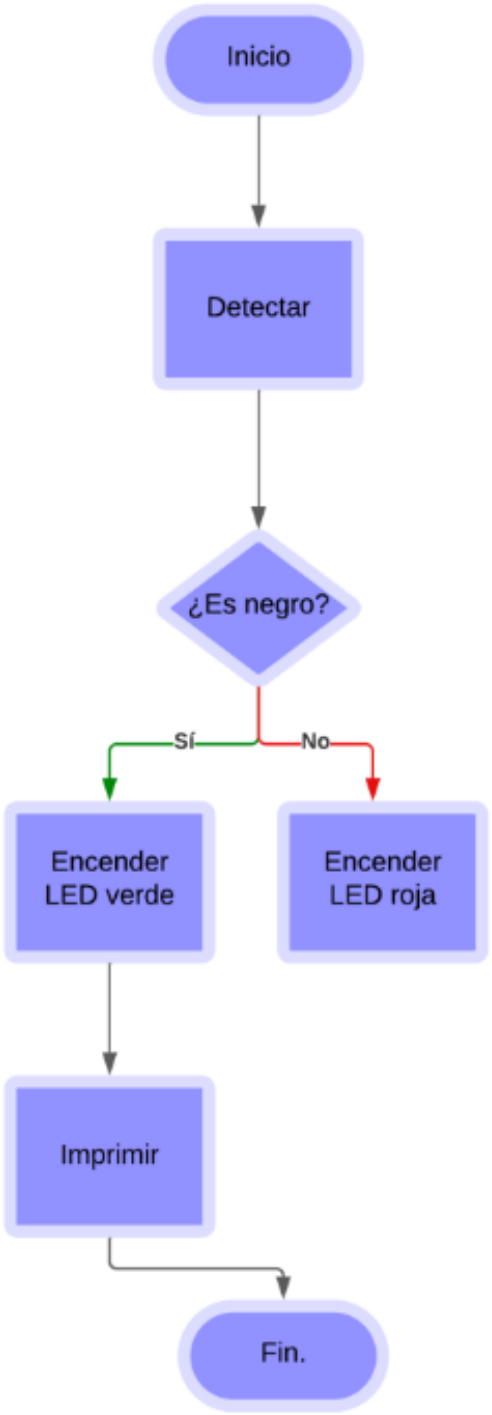
```
92  // MANEJO APIS
93
94  const btn_impimir = document.getElementById('navbar_item_imprimir')
95
96  btn_impimir.addEventListener('click', async function(e) {
97      e.preventDefault()
98      console.log("Imprimiendo...")
99      //obtener informacion de todas las celdas de la matriz
100     let matriz = []
101     for (var i = 0; i < celdas.length; i++) {
102         let celda = celdas[i]
103         let pos = { "fila": celda.id[5], "columna": celda.id[6] }
104         let color = celda.style.backgroundColor
105         let figura = celda.innerHTML.includes("img") ? celda.innerHTML.split("/")[3].split(".")[0] : ""
106
107         // verificar si la celda tiene una imagen
108         if (figura != "") {
109             matriz.push({ pos, color, figura })
110         }
111     }
112
113     // Verificar si la matriz esta vacia
114     if (matriz.length == 0) {
115         alert("Matriz vacía")
116         return
117     }
118
119     console.log(matriz)
120
121     try {
122         // Llamada a la api
123         const response = await fetch('http://localhost:4000/plotterserial/graficar', {
124             method: 'POST',
125             headers: {
126                 'Content-Type': 'application/json'
127             },
128             body: JSON.stringify({ 'matriz': matriz })
```

Equipo Utilizado

- Arduino Mega
- Cable USB a serial
- Modulo RS232
- Led
- Resistencia
- Protoboard
- Compuertas 7404 y 7408
- Flip Flop D 74174
- 74LS126 TTL Buffers de Bus
- Modulo Sensor de Color TCS230 TCS3200
- MD – L298N 15uente 15uente H for stepper motor, 2A, 25W
- Servomotor
- Material reciclado para la impresora

Diagramas con Explicación

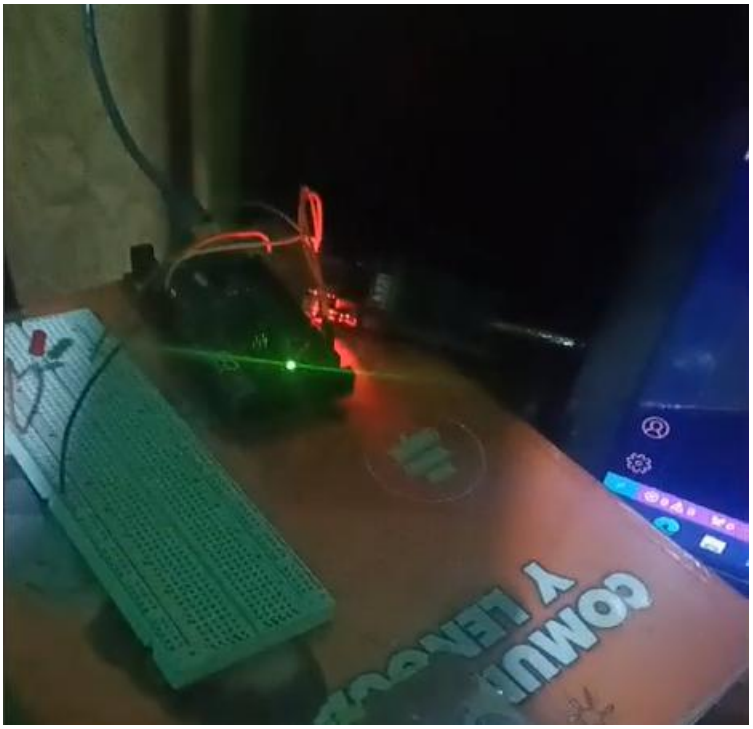



Inicio
Detectar
¿Es negro?
Sí
No
Encender LED verde
Encender LED roja
Imprimir
Fin.

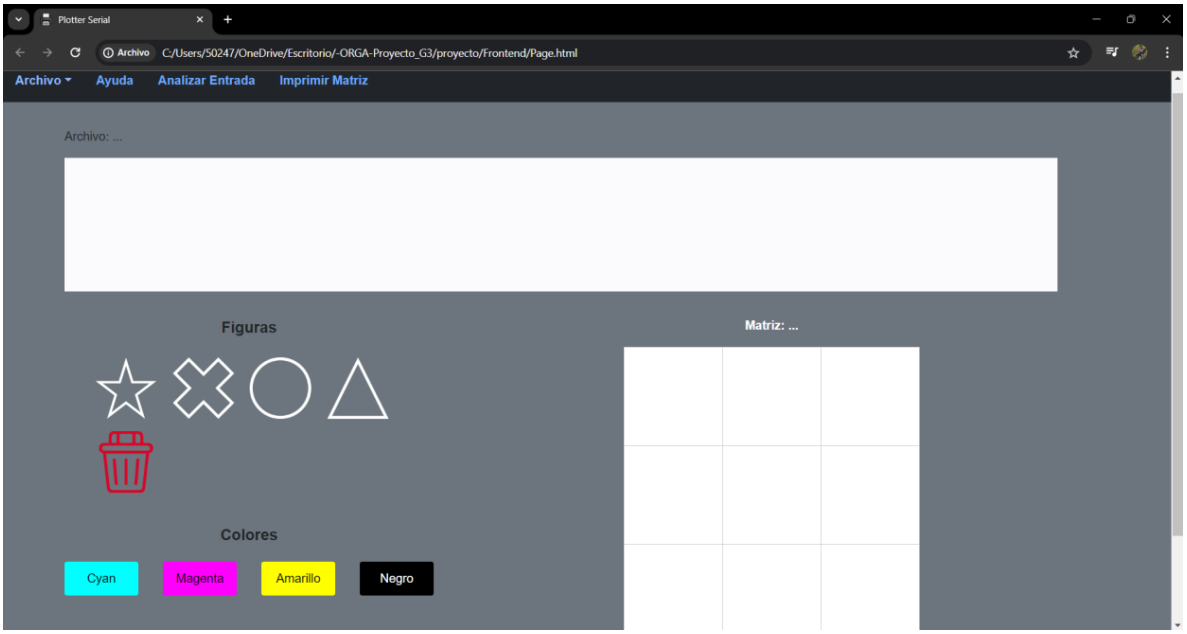
Manual de Usuario

Plotter Serial

Primero realizaremos las debidas conexiones de la memoria RAM a la PC que se utilizará y conexión con la impresora que se hizo.



Ingresaremos a nuestra página donde se encuentra la interfaz del juego, con cada una de las figuras que se pueden utilizar dentro la aplicación.

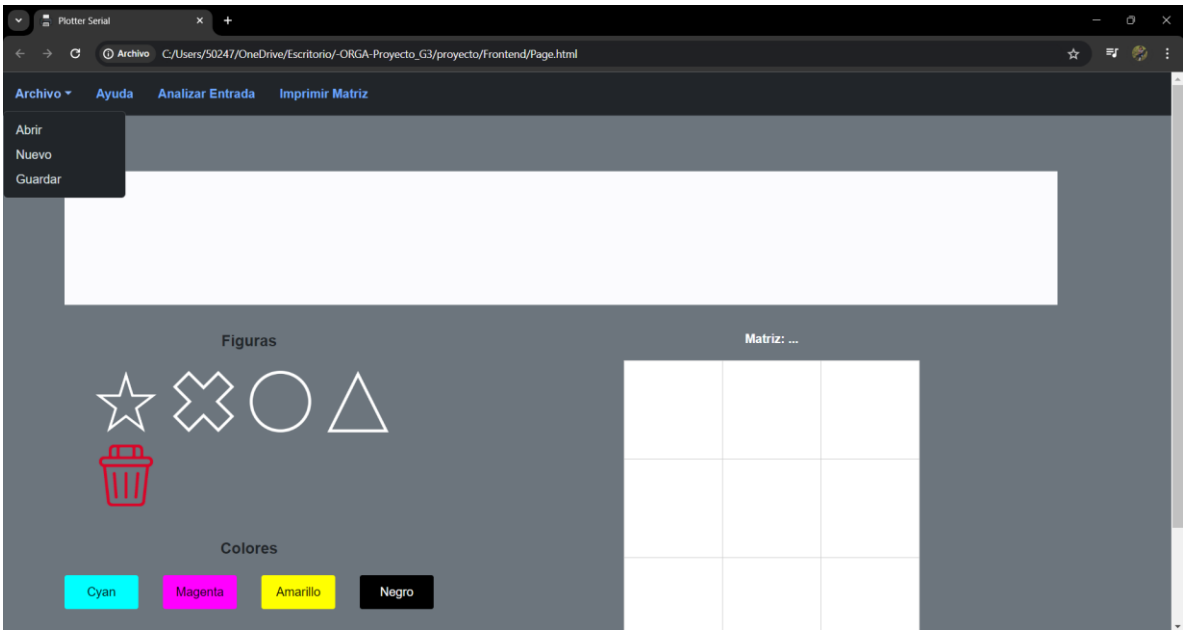


Nuestro menú principal contiene las opciones de “abrir” la cual nos servirá para poder escoger el archivo que la aplicación va a procesar para empezar a ver la colocación de cada figura en la matriz.

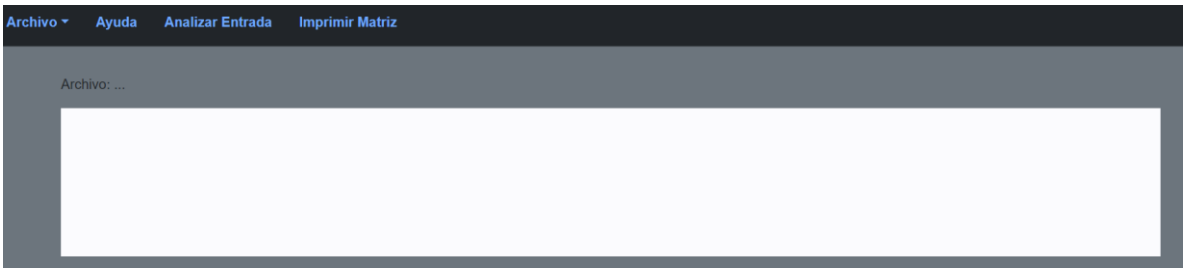
```
# Esto es un comentario

new_print PrimeraImpresion;

# set_print_x(fila, columna, color);
# set_print_o(fila, columna, color);
# set_print_triangulo(fila, columna, color);
# set_print_estrella(fila, columna, color);

# Coordenadas para el jugador x
set_print_x(1, 1, cyan);
set_print_x(2, 1, negro);
set_print_x(2, 2, cyan);
set_print_x(3, 2, amarillo);
set_print_x(3, 3, magenta);


# Coordenadas para el jugador o
set_print_o(3, 1, magenta);
set_print_o(1, 2, cyan);
set_print_o(1, 3, amarillo);
set_print_o(2, 3, negro);

end_print;

```

La opción “nuevo” lograra borrar el archivo anterior y poder escoger uno nuevo para iniciar un nuevo juego.

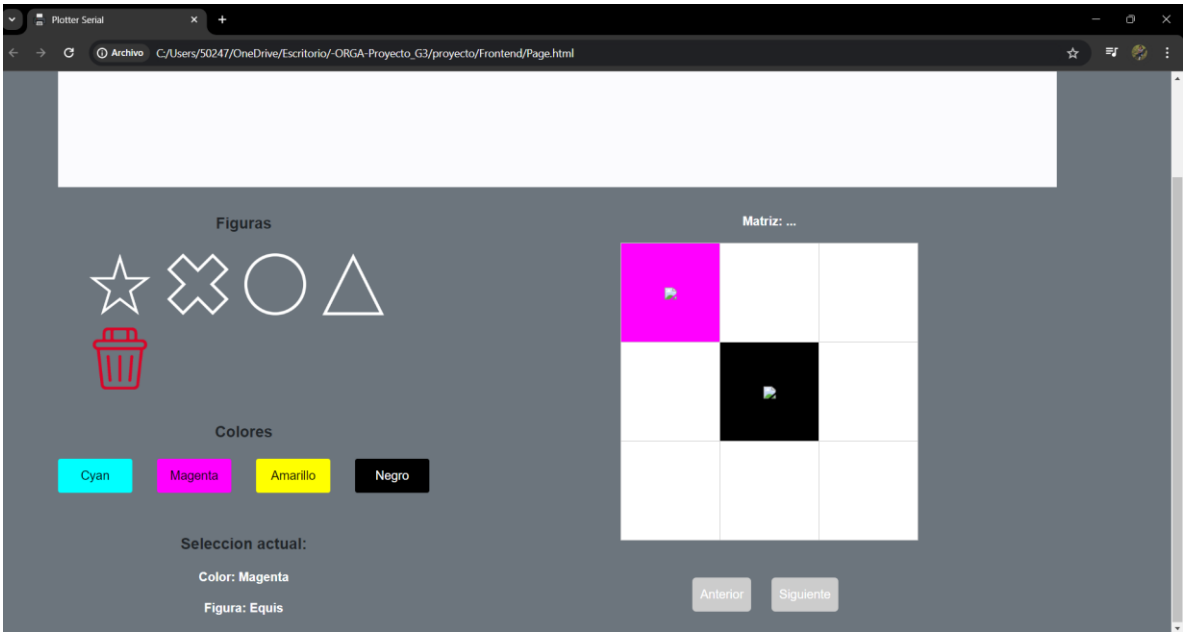
La opción “guardar” creará un archivo para guardar todo el proceso del juego que se realizó.

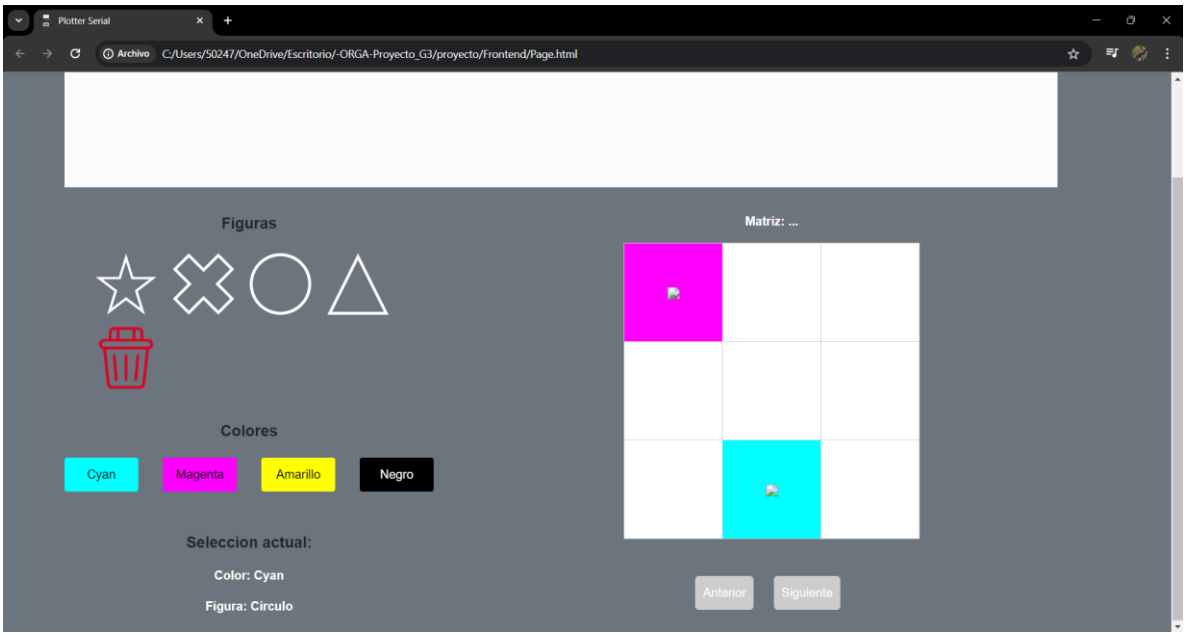


Acá se podrá visualizar y editar en vivo el archivo de entrada, esta funcionalidad permite no solo ver el contenido actual del archivo en cuestión, sino también modificarlo directamente en el mismo entorno.

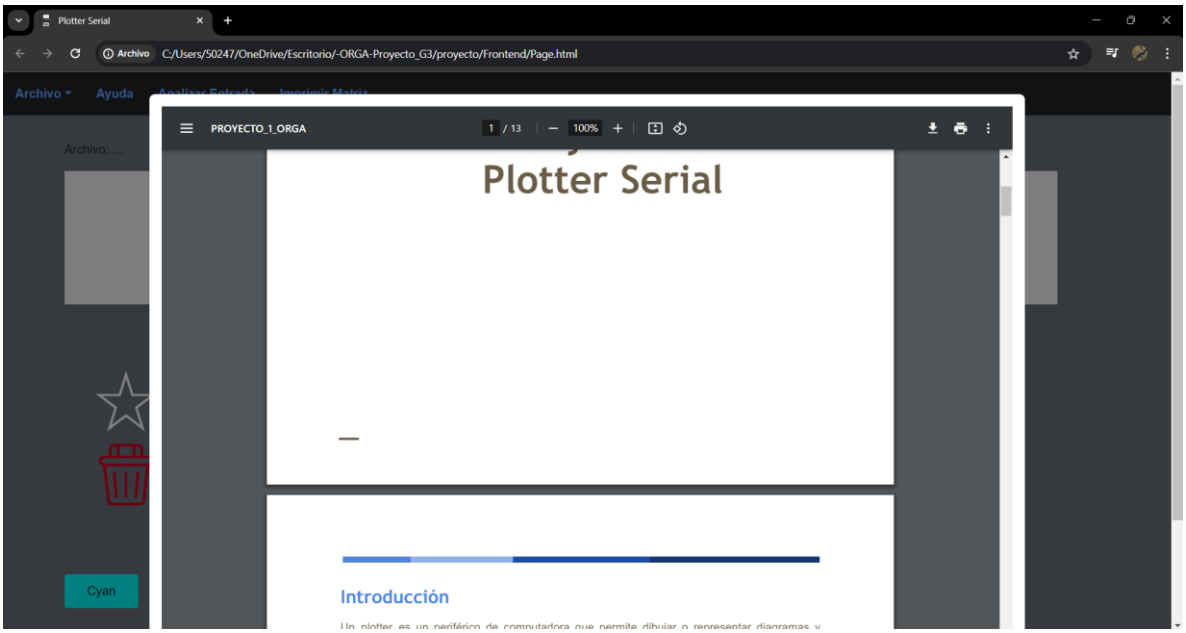


Podrás realizar tu propio archivo, primero tendrás que escoger un color y luego la figura que deseas colocar dentro de la matriz, la extensión del archivo debe ser “.orga”. Y si alguna de las figuras no es la que se deseaba colocar, ésta se podrá eliminar y realizar el mismo procedimiento anterior para colocar la nueva figura.





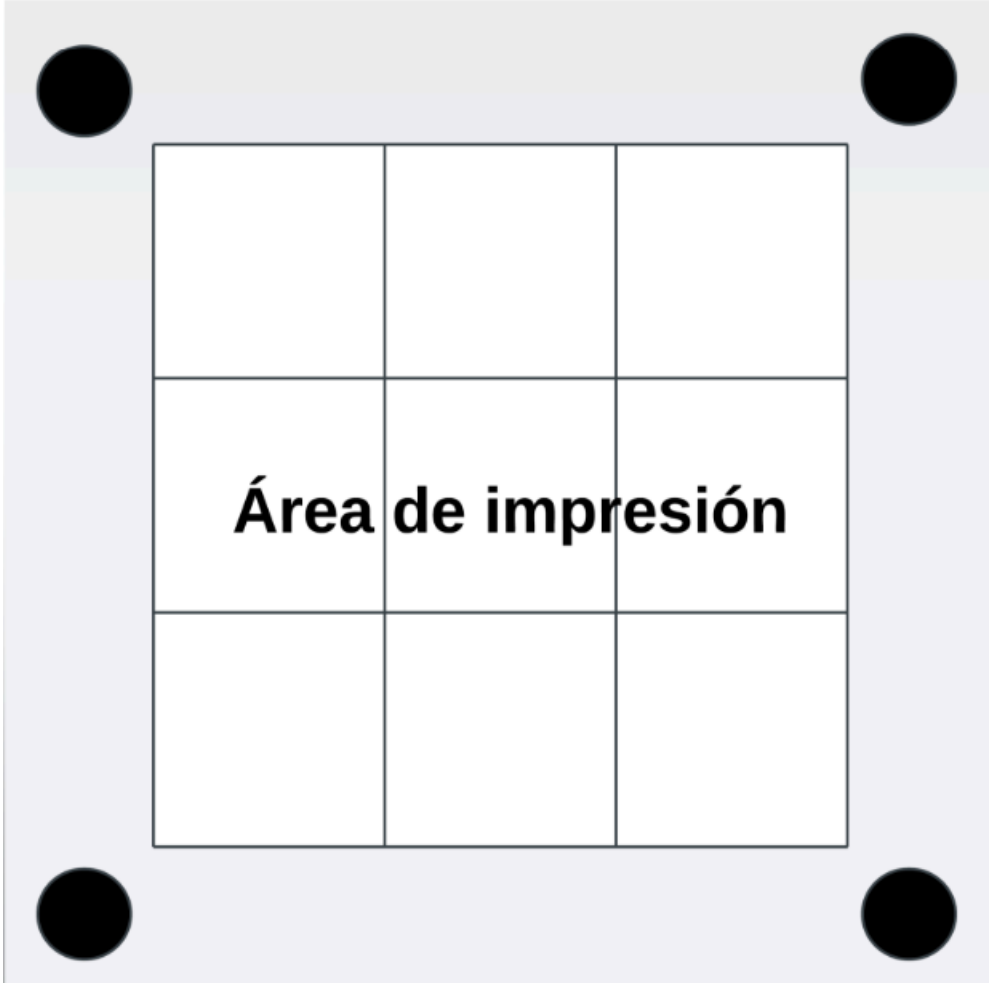
En nuestra opción “Ayuda” se podrá visualizar los manuales necesarios para poder utilizar mejor las opciones de la aplicación, y sea más fácil para el usuario.

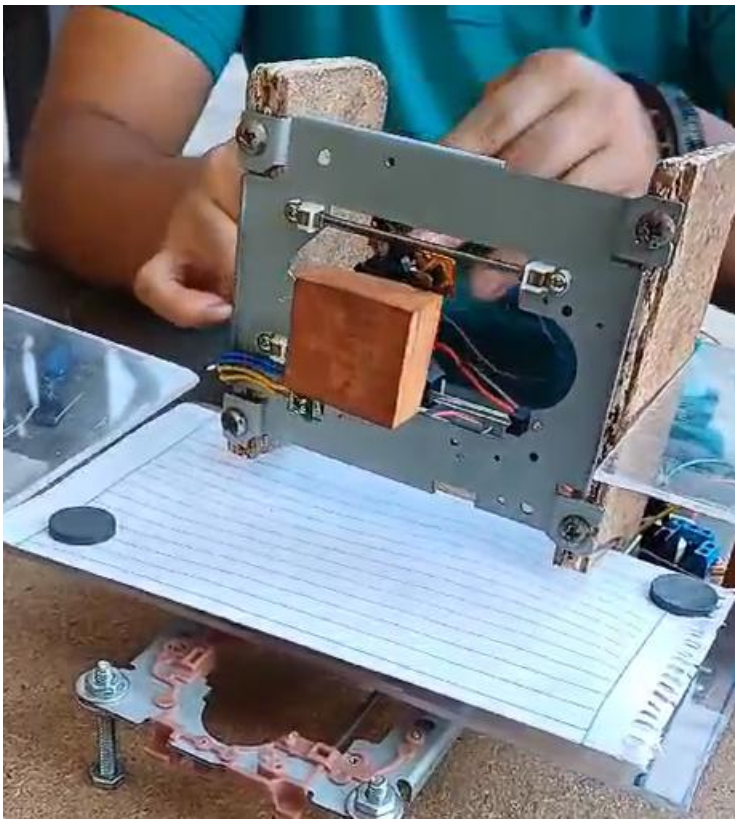


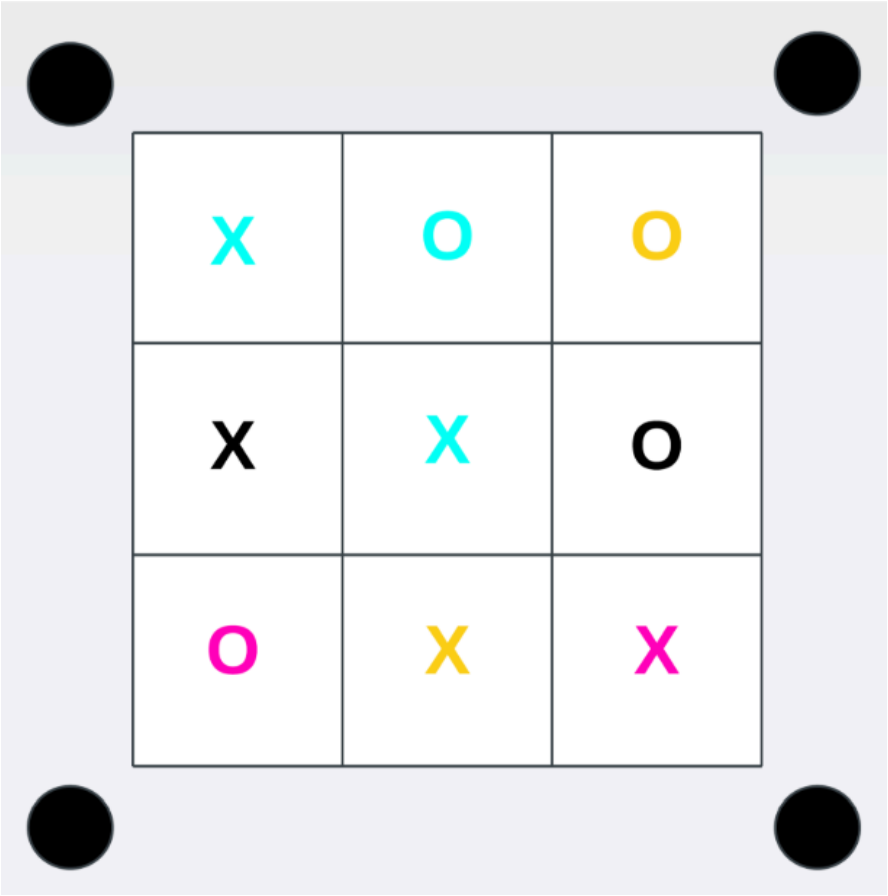
“Analizar Entrada” podrá verificar cada coordenada que contiene el archivo que se cargó a la aplicación y ésta poder ingresar cada figura en la matriz.

Luego al terminar de agregar todas las figuras necesarias a la matriz, este archivo se guardará y se podrá presionar la opción de “imprimir matriz”, esta opción realizará la conexión con la impresora que se realizó y empezará a imprimir la matriz terminada que se creó en la aplicación.

El área de impresión será una hoja de papel bond (120 gramos) tamaño carta, la cual en cada esquina tendrá 1 indicador, dichos indicadores serán utilizados por sensores de color, el cual permitirá la impresión una vez que los 4 sensores detecten que el área de impresión está correctamente alineada. Se debe indicar por medio de un LED de color “Azul” si todos los sensores poseen una alineación correcta y uno “Amarillo” si más de alguno tienen una alineación incorrecta.






X O O
X X O
O X X

Manual Técnico

INTRODUCCIÓN

Este manual describe los pasos necesarios para cualquier persona que tenga ciertas bases de sistemas pueda realizar la instalación del aplicativo creado para la poder utilizar la aplicación del Plotter Serial. Es importante tener en cuenta que en el presente manual se menciona las especificaciones mínimas de hardware y software para la correcta instalación del aplicativo.

OBJETIVOS

- ❏ Mejorar el uso de los distintos tipos de Flip Flops y contadores.
- ❏ Implementar una transmisión serial a través de los puertos de una PC.
- ❏ Desarrollar una estructura mecánica funcional.
- ❏ Aplicar e Implementar conocimientos de registros.
- ❏ Implementar software para el control de puertos.
- ❏ Aplicar e Implementar conocimientos de memorias.
- ❏ Aprender conocimientos de lógica secuencial y simplificación de estados.
- ❏ Aprender el funcionamiento y uso de motores paso a paso (Stepper).
- ❏ Aprender el uso de distintos tipos de sensores.
- ❏ Integrar circuitos eléctricos y producir movimientos mecánicos.
- ❏ Aplicar todos los conocimientos aprendidos en el curso
- ❏ Aplicar los conocimientos adquiridos de Arduino.
- ❏ Aprender la aplicación de una memoria RAM.

REQUERIMIENTOS TÉCNICOS

REQUERIMIENTOS MÍNIMOS DE HARDWARE

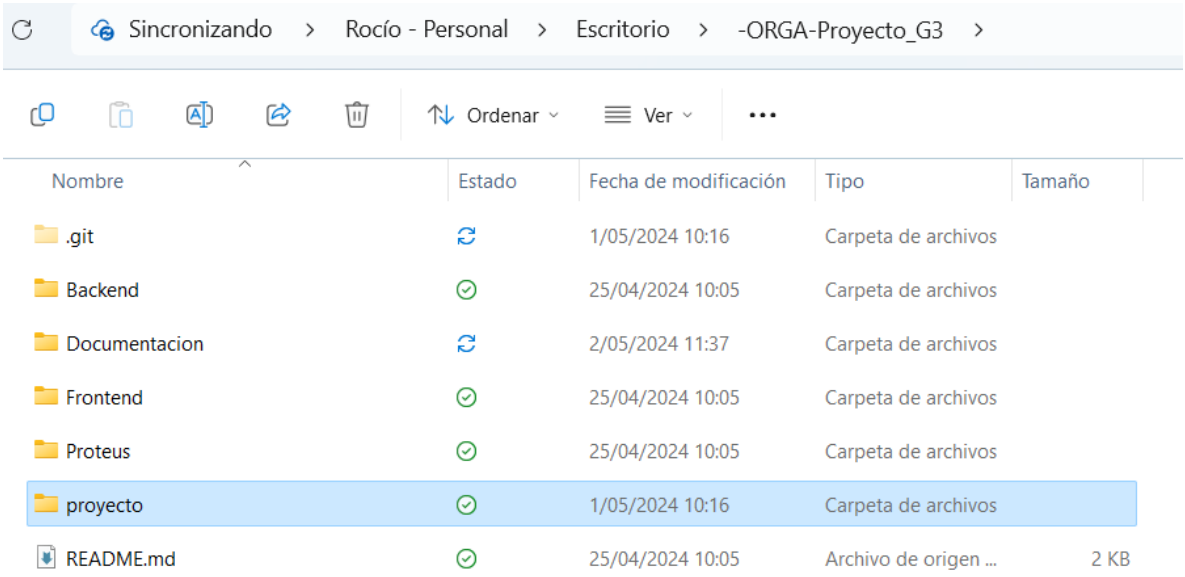
- Procesador
- Memoria RAM. Componentes: 2 sensor color tcs230, 2 motor stepper, 1 servomotor, 74-126, 74-08, 74-04 y 74-174.
- Disco Duro

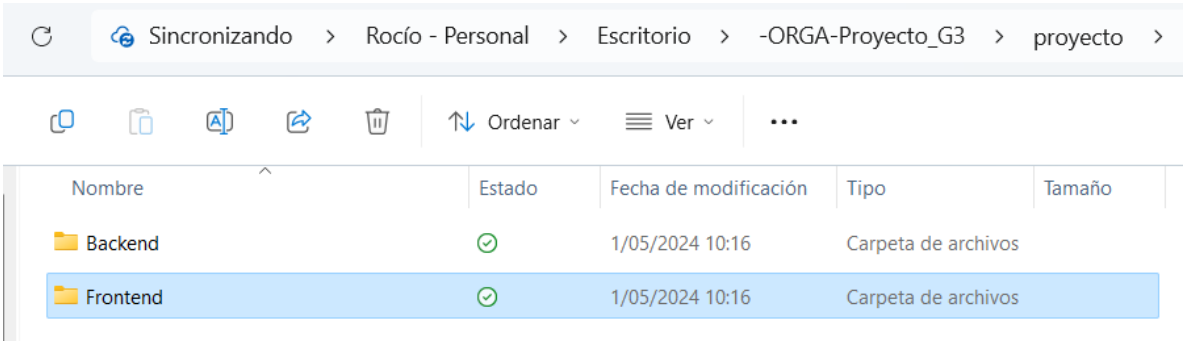
REQUERIMIENTOS MÍNIMOS DE SOFTWARE

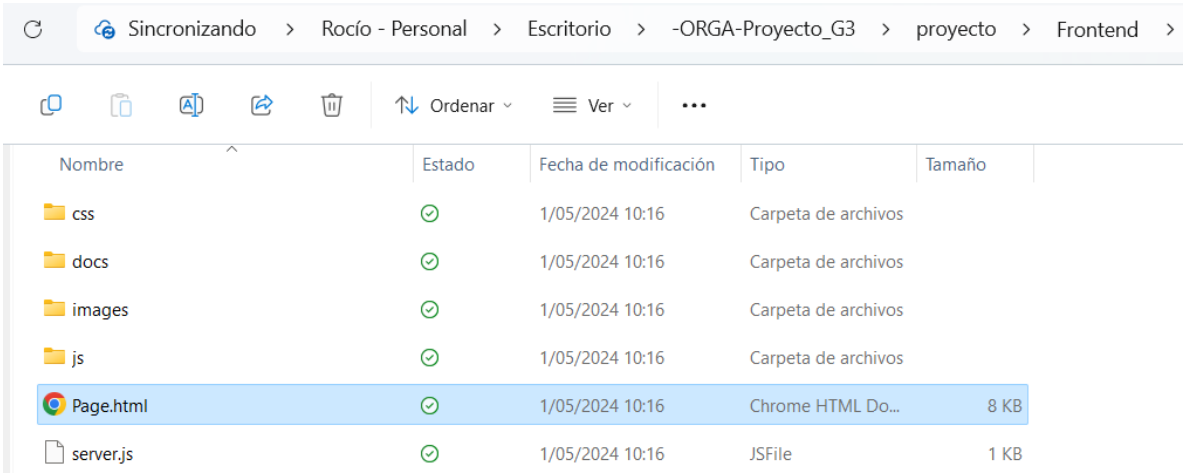
- Privilegios de administrador
- Sistema Operativo

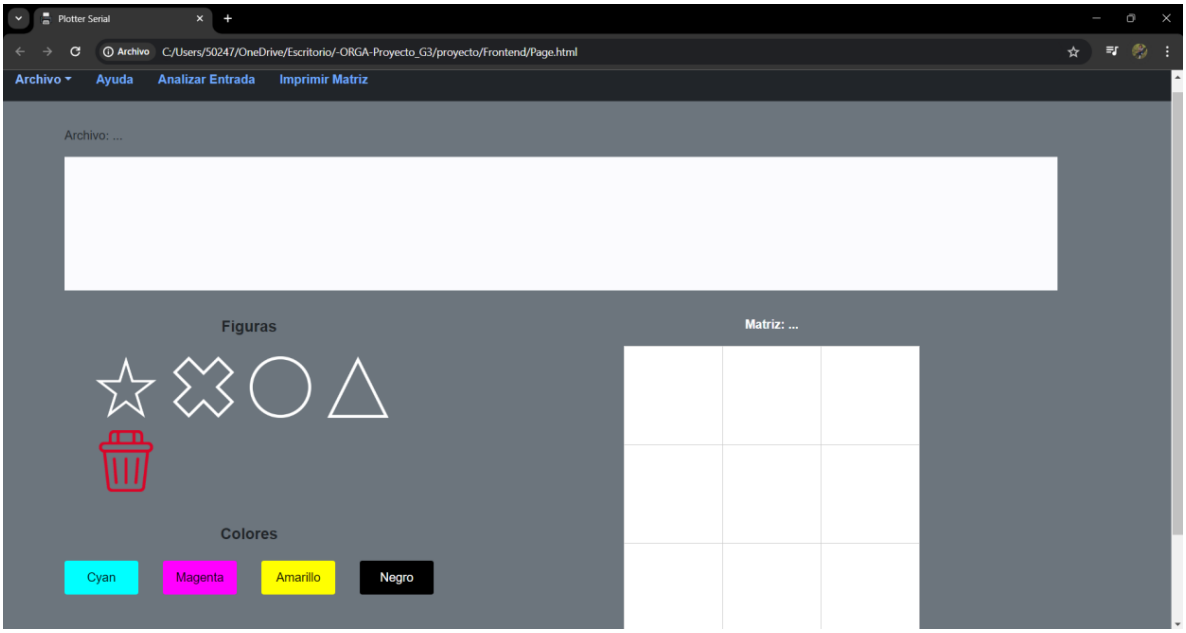
INTALACIÓN

Se ingresará en la carpeta de proyecto y luego en la de fronted y se podrá abrir la página llamada page.html para poder empezar a utilizar la aplicación.






Sincronizando
Rocío - Personal
Escritorio
-ORGA-Proyecto_G3
proyecto
Frontend
Ordenar
Ver
Nombre
Estado
Fecha de modificación
Tipo
Tamaño
css
1/05/2024 10:16
Carpeta de archivos
docs
1/05/2024 10:16
Carpeta de archivos
images
1/05/2024 10:16
Carpeta de archivos
js
1/05/2024 10:16
Carpeta de archivos
Page.html
1/05/2024 10:16
Chrome HTML Do...
8 KB
server.js
1/05/2024 10:16
JSFile
1 KB


Plotter Serial
Archivo
C:/Users/50247/OneDrive/Escritorio/-ORGA-Proyecto_G3/proyecto/Frontend/Page.html
Archivo
Ayuda
Analizar Entrada
Imprimir Matriz
Archivo: ...
Figuras
Matriz: ...
Colores
Cyan
Magenta
Amarillo
Negro

- Importar pyserial y CORS para la aplicación.

```
#import pyserial

app = Flask(__name__)
CORS(app)  # Habilitar CORS para la aplicación Flask
```

- Rutas para ingresar archivos.

```
@app.route('/', defaults={'path': ''})
@app.route('/<path:path>')
def serve_frontend(path):
    if path == "":
        return send_from_directory('../Frontend', 'Page.html')
    elif os.path.exists(os.path.join('../Frontend', path)):
        return send_from_directory('../Frontend', path)
    else:
        return send_from_directory('../Frontend', 'Page.html')
```

- Analizar archivo.

```
@app.route('/plotterserial/analizar', methods=['POST'])
def analizar():

    datos = request.json
    codigo_recibido = datos.get('texto', '')

    if codigo_recibido == '':
        return jsonify({'error': 'No se recibió código'})

    parse = Gm.parse(codigo_recibido)

    if not parse:
        return jsonify({
            'mensaje': 'Error en el análisis',
            'impresiones': []
        })

    sentencias = []
    for sent in parse:
        sentencias.append(json.dumps(sent.to_dict()))

    try:
        return jsonify({
            'mensaje': 'Análisis terminado',
            'impresiones': sentencias
```

```
        })

    except Exception as e:
        print(e)
        return jsonify({
            'mensaje': 'Error en el análisis',
            'impresiones': []
        })
```

- Graficar en la matriz

```
@app.route('/plotterserial/graficar', methods=['POST'])
def graficar():

    datos = request.json
    matriz_recibido = datos.get('matriz', '')

    if not matriz_recibido:
        return jsonify({'error': 'No se recibió matriz'})


    for item in matriz_recibido:
        fila = item['pos']['fila']
        columna = item['pos']['columna']
        color = item['color']
        figura = item['figura']
        print(f"Fila: {fila}, Columna: {columna}, Color: {color}, Figura:
{figura}")
        print("")

        # TODO: Implementar logica impresora con pyserial


    return jsonify({'mensaje': 'Graficación terminada'})
```

- Configuaración de puerto serial y conexión de LED.

```
ser = serial.Serial('COM9', 9600)
time.sleep(2)  # Espera a que se establezca la conexión serial

def toggle_led(command):
    """
    Envia el comando al Arduino para controlar el LED.
    'A' para encender, 'a' para apagar.
    """
    ser.write(command.encode())  # Envía el comando al Arduino
```

```
try:
    while True:
        cmd = input("Ingrese 'A' para encender el LED o 'a' para apagarlo
(Ingrese 'salir' para terminar): ")
        if cmd in ['a', 'A']:
            toggle_led(cmd)
        elif cmd == 'salir':
            print("Saliendo...")
            break
        else:
            print("Comando no reconocido. Intente nuevamente.")
finally:
    ser.close()  # Cierre de conexion con el arduino
```

- Combinaciones de los colores para las figuras.

```
class Figura:
    def __init__(self, forma, color, vacio):
        self.forma = forma
        self.color = color
        self.vacio = vacio

    def binario(self):
        if self.forma == 'estrella':
            if self.color == 'cyan':
                return '00001b'
            if self.color == 'magenta':
                return '00011b'
            if self.color == 'yellow':
                return '00101b'
            if self.color == 'black':
                return '00111b'
        if self.forma == "equis":
            if self.color == 'cyan':
                return '01001b'
            if self.color == 'magenta':
                return '01011b'
            if self.color == 'yellow':
                return '01101b'
            if self.color == 'black':
                return '01111b'
        if self.forma == "circulo":
            if self.color == 'cyan':
                return '10001b'
            if self.color == 'magenta':
```

```
                return '10011b'
            if self.color == 'yellow':
                return '10101b'
            if self.color == 'black':
                return '10111b'
        if self.forma == "triangulo":
            if self.color == 'cyan':
                return '11001b'
            if self.color == 'magenta':
                return '11011b'
            if self.color == 'yellow':
                return '11101b'
            if self.color == 'black':
                return '11111b'

        if self.vacio:
            return '00000b'
```

- Creación de la matriz y funciones que son las que hacen a que las figuras se puedan cambiar, o cambiarles el color y la posición,

```
let color = "cyan" // Color de la celda seleccionada
let celda = "celda00" // Celda seleccionada por defecto
let figura = "circulo" // Imagen de la celda seleccionada

// Función que cambia el color de la celda seleccionada
const celdas = document.getElementsByClassName("celda_mat")
const lbl_color_selec = document.getElementById("color_seleccionado")
const lbl_figura_selec = document.getElementById("figura_seleccionado")


// Cuando se hace click en una celda, se llama a la función que la cambia y
se pasa su id
for (var i = 0; i < celdas.length; i++) {
    celdas[i].addEventListener("click", function() {
        celda = this.id
        cambiar_color()
        cambiar_imagen()
    });
}

// Función que cambia el color de la celda seleccionada
function cambiar_color() {
    document.getElementById(celda).style.backgroundColor = color
```

```
}


// Función que cambia la imagen de la celda seleccionada
function cambiar_imagen() {
    document.getElementById(celda).innerHTML = "";

    if (figura == "") {
        return
    }

    var img = document.createElement("img")
    img.src = "/images/" + figura + ".png"
    document.getElementById(celda).appendChild(img)
}

// Funciones que cambian el color de la celda seleccionada
document.getElementById("col_cyan").addEventListener('click', function(e) {
    e.preventDefault()
    color = "cyan"
    lbl_color_selec.innerHTML = "Color: Cyan"
})
document.getElementById("col_yellow").addEventListener('click', function(e)
{
    e.preventDefault()
    color = "yellow"
    lbl_color_selec.innerHTML = "Color: Amarillo"
})
document.getElementById("col_magenta").addEventListener('click', function(e)
{
    e.preventDefault()
    color = "magenta"
    lbl_color_selec.innerHTML = "Color: Magenta"
})
document.getElementById("col_black").addEventListener('click', function(e) {
    e.preventDefault()
    color = "black"
    lbl_color_selec.innerHTML = "Color: Negro"
})

// Funciones que cambian la imagen de la celda seleccionada
document.getElementById("fig_circulo").addEventListener('click', function(e)
{
    e.preventDefault()
    figura = "circulo"
```

```
    lbl_figura_selec.innerHTML = "Figura: Circulo"
})
document.getElementById("fig_triangulo").addEventListener('click',
function(e) {
    e.preventDefault()
    figura = "triangulo"
    lbl_figura_selec.innerHTML = "Figura: Triángulo"
})
document.getElementById("fig_equis").addEventListener('click', function(e) {
    e.preventDefault()
    figura = "equis"
    lbl_figura_selec.innerHTML = "Figura: Equis"
})
document.getElementById("fig_estrella").addEventListener('click',
function(e) {
    e.preventDefault()
    figura = "estrella"
    lbl_figura_selec.innerHTML = "Figura: Estrella"
})
document.getElementById("fig_eliminar").addEventListener('click',
function(e) {
    e.preventDefault()
    figura = ""
    color = "white"
    lbl_color_selec.innerHTML = "Color: ..."
    lbl_figura_selec.innerHTML = "Figura: Eliminar"
})


// MANEJO APIS

const btn_impimir = document.getElementById('navbar_item_imprimir')

btn_impimir.addEventListener('click', async function(e) {
    e.preventDefault()
    console.log("Imprimiendo...")
    //obtener informacion de todas las celdas de la matriz
    let matriz = []
    for (var i = 0; i < celdas.length; i++) {
        let celda = celdas[i]
        let pos = { "fila": celda.id[5], "columna": celda.id[6] }
        let color = celda.style.backgroundColor
        let figura = celda.innerHTML.includes("img") ?
celda.innerHTML.split("/")[3].split(".")[0] : ""
```

```
        // verificar si la celda tiene una imagen
        if (figura != "") {
            matriz.push({ pos, color, figura })
        }
    }

    // Verificar si la matriz esta vacia
    if (matriz.length == 0) {
        alert("Matriz vacía")
        return
    }

    console.log(matriz)

    try {
        // Llamada a la api
        const response = await
fetch('http://localhost:4000/plotterserial/graficar', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ 'matriz': matriz })
        })

        const data = await response.json()
        alert(data.mensaje)

    } catch (error) {
        console.log(error)

    }
})
```

Presupuesto

Facturas	Monto Total	Monto Individual (c/u)
Factura 1	Q224.00	Q20.36
Factura 2	Q35.00	Q3.18
Factura 3	Q68.00	Q6.18
Sin factura	Q35.00	Q3.18
Factura 4	Q6.00	Q0.55
Factura 6	Q43.00	Q3.91
Factura 7	Q65.00	Q5.91
TOTAL	**Q476.00**	

Aporte

Nombre	Aporte
Oswaldo Antonio Choc Cuteres	Maqueta
Jencer Hamilton Hernández Alonzo	RAM
Cristian Raúl Vega Rodríguez	Código Back y Arduino
Javier Andrés Monjes Solórzano	Maqueta
Angel Isaías Mendoza Martínez	RAM
Estephanie Alejandra Ruíz Pérez	Código Front y Back
Edin Rafael Santizo Barrera	Código Front y Back
Juan Pascual Itzep Coguox	RAM
Rocío Samaí López Vásquez	Documentación
Diego Andrés Dubón Samayoa	RAM
Juan José Almengor Tizol	Código Front y Back

Conclusiones

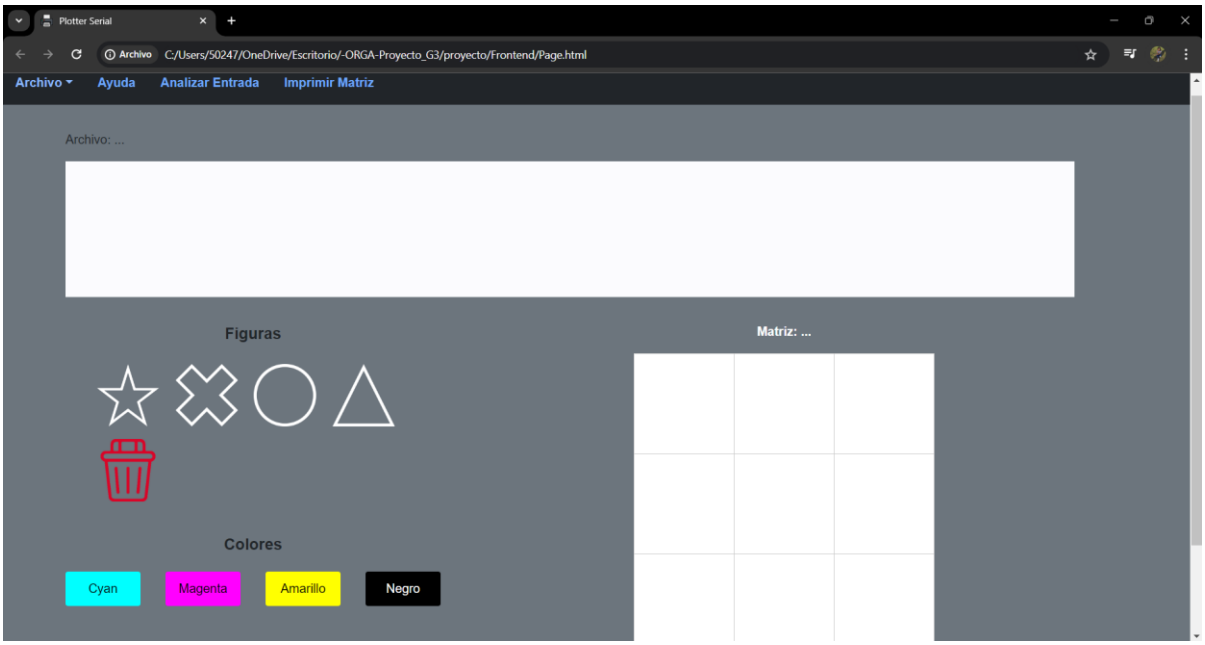
- Por medio de la aplicación y práctica de los Flip-Flops y contadores, nos permitió comprender mejor su funcionamiento y aplicaciones en circuitos secuenciales, lo que facilitó el diseño del sistema para que fuera más eficiente.
- Al principio se encontraron varios obstáculos para la conexión serial pero por medio de varias pruebas se logró obtener la transmisión serial tanto con el dominio del hardware como con el software necesario.
- La capacidad de diseñar y construir estructuras mecánicas funcionales demuestra una compresión integral de los principios de la ingeniería y su aplicación práctica en este proyecto.
- El uso de registros y memorias es fundamental en el almacenamiento y procesamiento de datos en sistemas digitales, lo que permite la creación de sistemas más complejos con una solución vial.

Recomendaciones

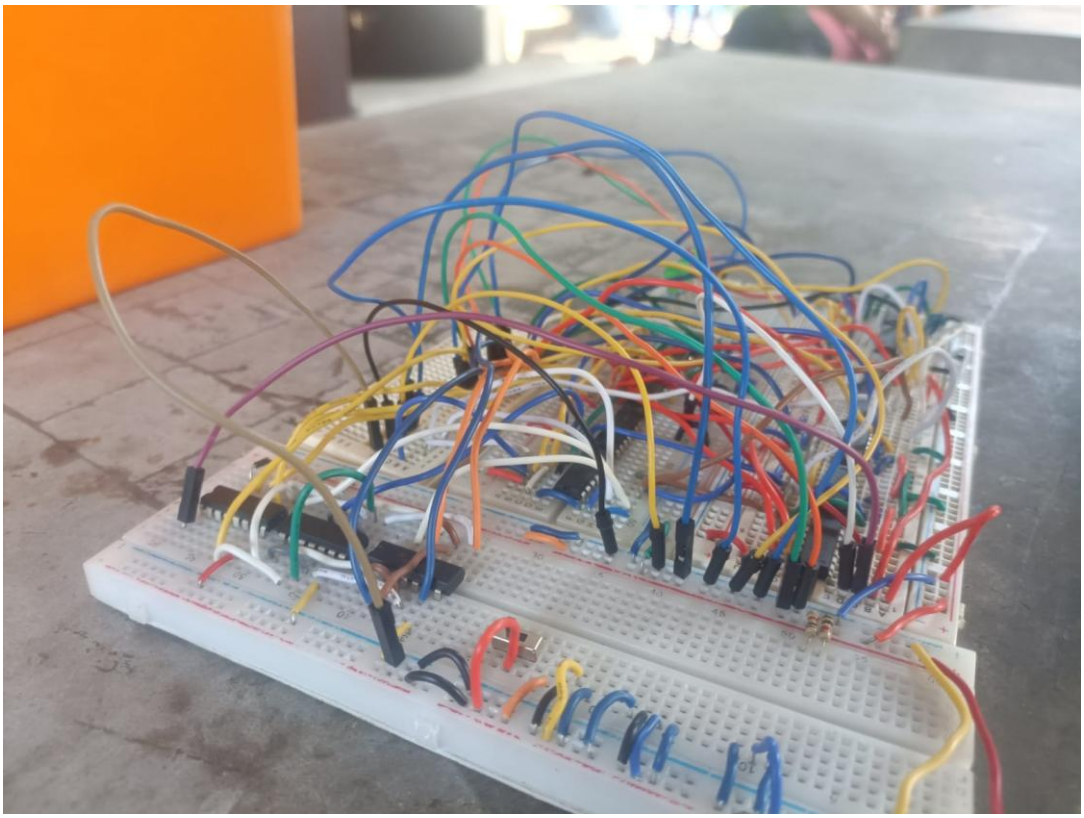
- Seguir explorando y experimentando con diferentes Flip-Flops y contadores, ya que esto puede llegar a profundizar la compresión y descubrir nuevas aplicaciones en circuitos secuenciales.
- Considerar cuidadosamente las herramientas y tecnologías que se utilizaran para el desarrollo del software y el control de la impresora. Asegurarse de elegir las adecuadas para el tipo de aplicación que se desarrollo y permitir el cumplimiento de los requisitos establecidos.
- Realizar pruebas exhaustivas en todas las etapas del desarrollo para garantizar que el software y el hardware funcionen correctamente y cumplas los requisitos. Corregir problemas que surjan durante las pruebas y realizar pruebas de integración para asegurarse de que cada uno de los componentes funcionen correctamente.

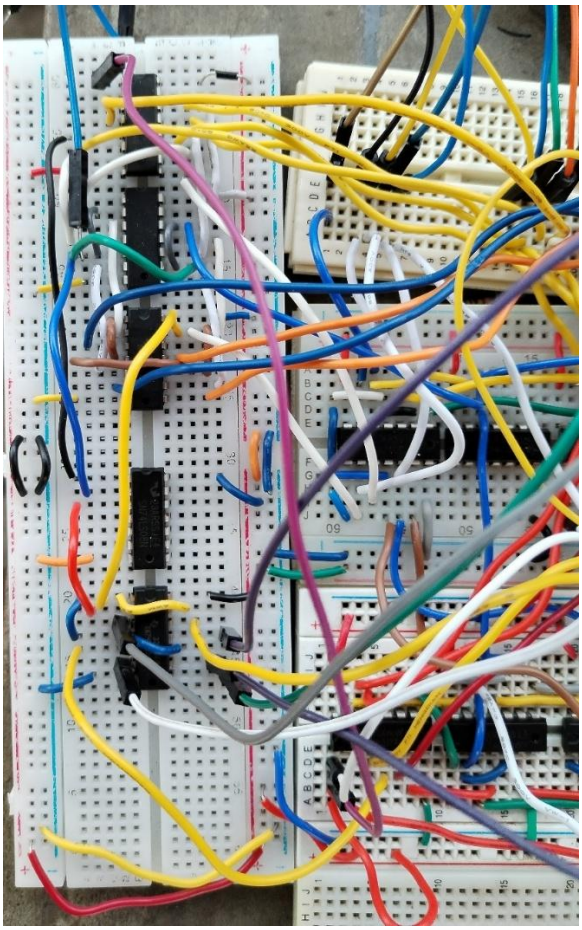
Anexos

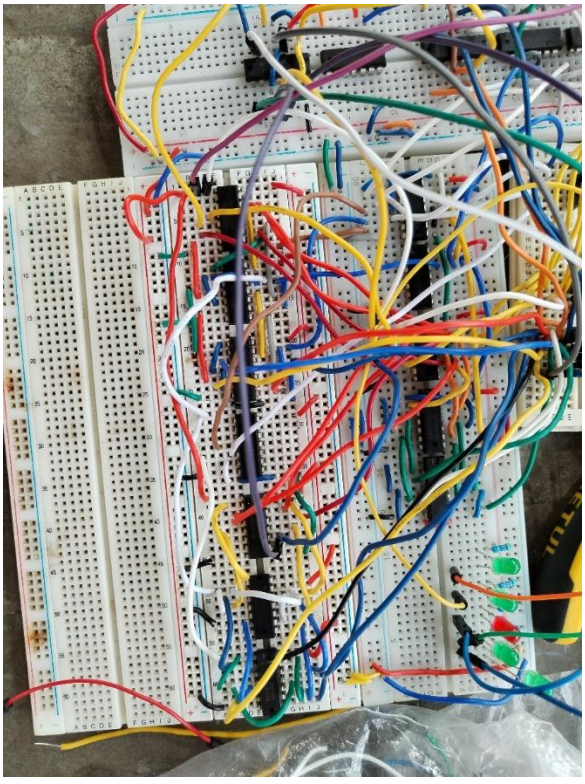
Interfaz



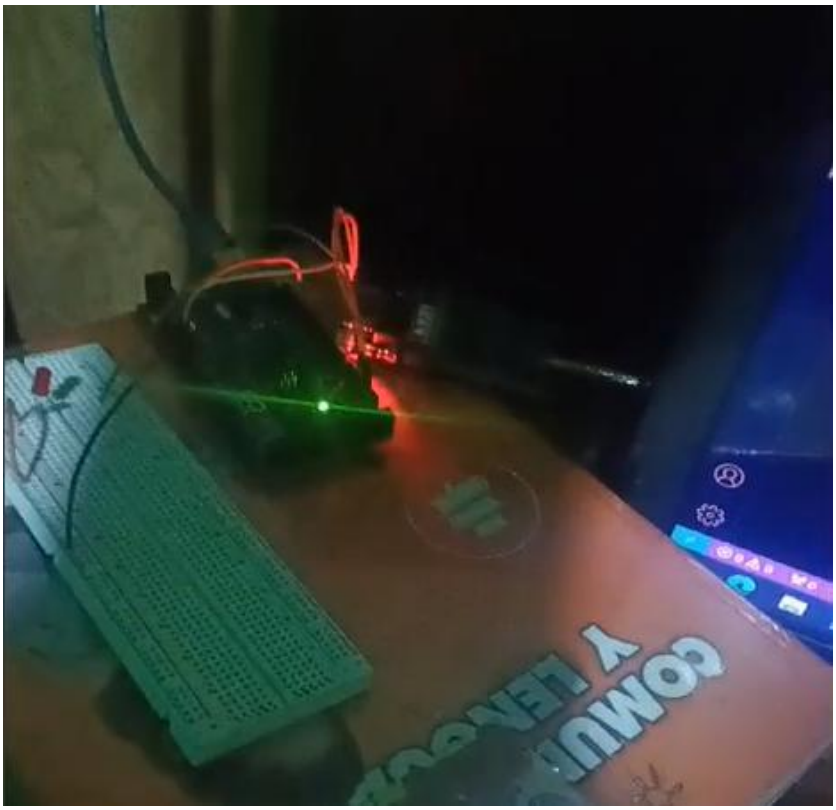
RAM

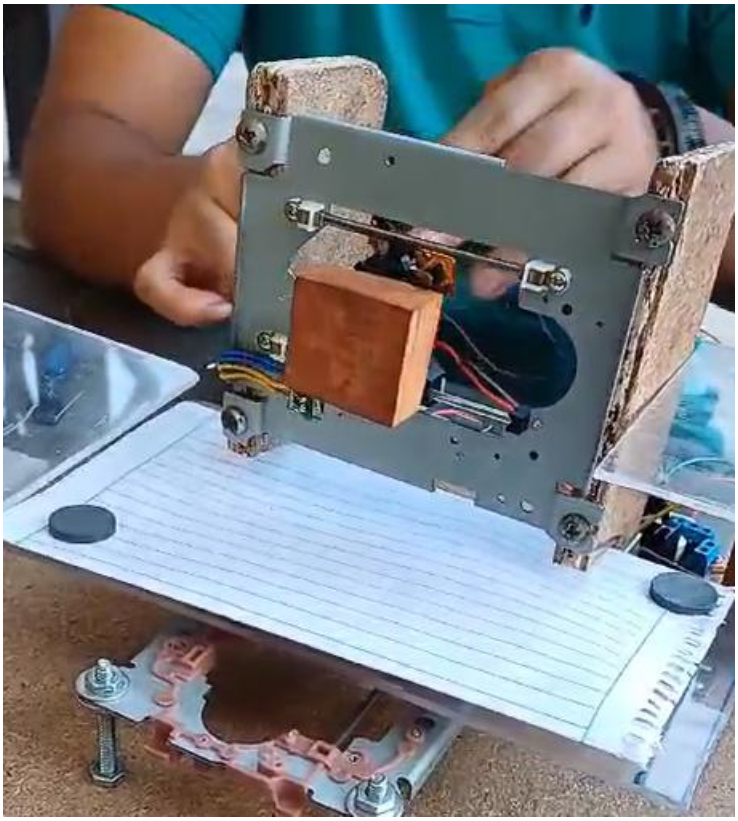






Impresora





Maqueta final de la Impresora