Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Organización Computacional

Ing. Otto René Escobar Leiva

Auxiliar Javier Gutiérrez

PROYECTO

Plotter Serial

Nombre	Carné	Participación
Bismarck Estuardo Romero Lemus	201708880	100%
Josué Nabí Hurtarte Pinto	202202481	100%
Naomi Rashel Yos Cujcuj	202001814	100%
Angela Paulina Rodriguez López	201503569	100%
Rodrigo Alejandro Tahuite Soria	202202854	100%
Nathan Antonio Valdez Valdez	202001568	100%
Gonzalo Fernando Pérez Cazún	202211515	100%
Rony Omar Miguel López	201905750	100%
Kevin Eduardo Castañeda Hernández	201901801	100%

Introducción

El proyecto se centra en el desarrollo de un Plotter Serial, un periférico de computadora diseñado para dibujar o representar gráficos de manera precisa. Utilizando motores paso a paso y circuitos combinacionales y secuenciales, el Plotter Serial busca mejorar el uso de Flip-Flops y contadores, así como implementar una transmisión serial a través de los puertos de una PC. Este proyecto es un desafío para nosotros los estudiantes porque tenemos que, a aplicar conocimientos de registros, memorias, lógica secuencial y simplificación de estados, además de aprender sobre el funcionamiento de motores paso a paso y el uso de sensores.

Descripción del Problema

Por parte de la Universidad de San Carlos de Guatemala, a través de su Facultad de Ingeniería, requiere el desarrollo de un Plotter Serial para una demostración de proyectos de innovación. El objetivo es crear una impresora no tradicional controlada por un software especial diseñado por el equipo estudiantil. Este Plotter Serial estará conectado a una PC a través de puertos seriales o paralelos, permitiendo la impresión de figuras predefinidas en una hoja de papel bond mediante un lápiz o herramienta de impresión similar. Para lograrlo, se deberán elaborar circuitos combinacionales y secuenciales para manipular los ejes X y Y del Plotter, así como una aplicación con interfaz gráfica que podrá cargar archivos de entrada. El proyecto también incluye la implementación de sensores de color para asegurar la alineación correcta del área de impresión. La fecha límite para la entrega y calificación es el viernes 26 de abril de 2024.

Objetivos

Objetivo General:

Diseñar e implementar un Plotter Serial altamente funcional y preciso, capaz de dibujar figuras predefinidas en una hoja de papel bond mediante una conexión serial desde una PC. Demostrar la eficiencia y viabilidad del sistema mediante una presentación clara y ordenada del proyecto.

Objetivos específicos:

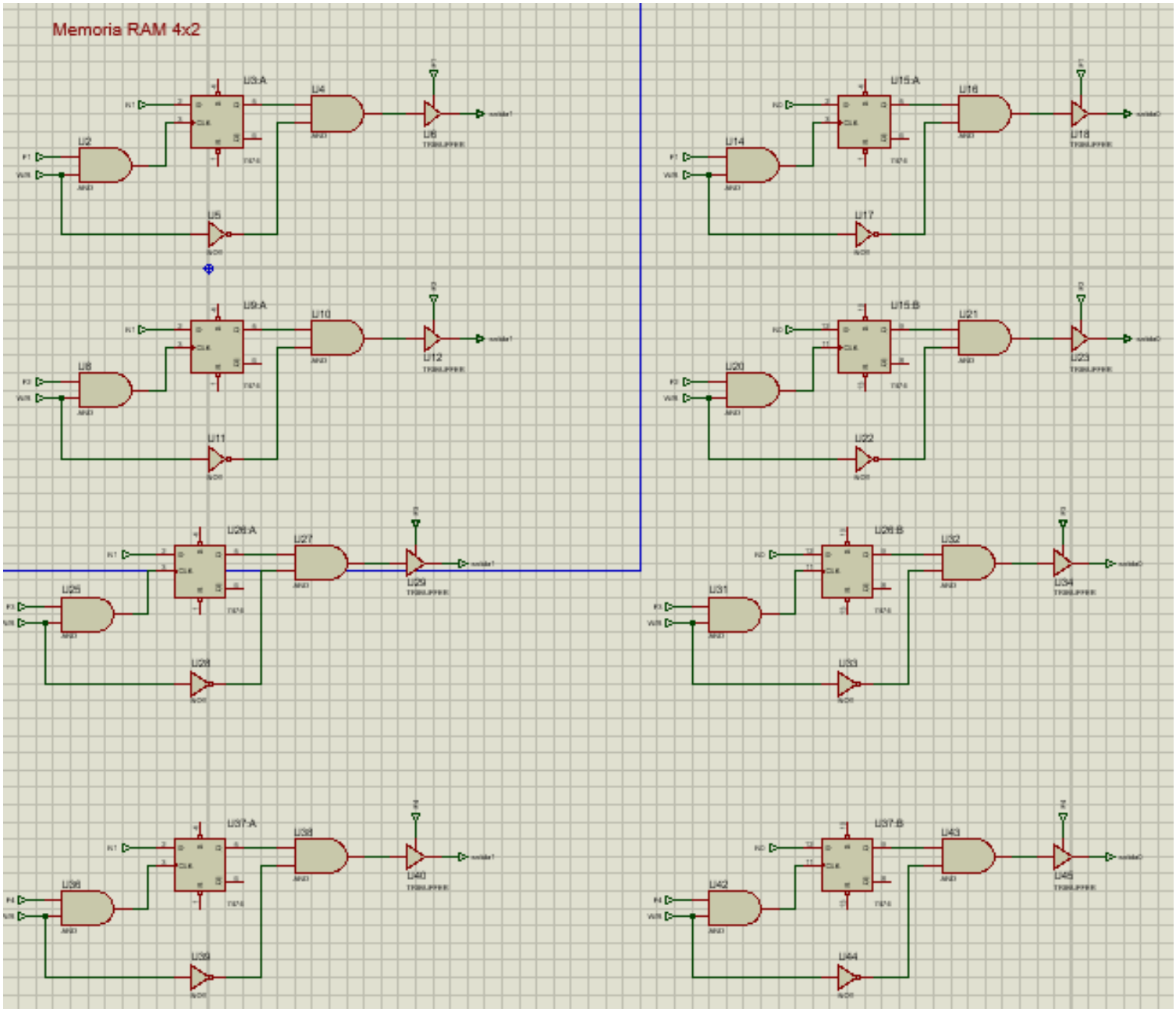
- Construir una estructura mecánica robusta y precisa que permita el movimiento suave y controlado del cabezal de impresión sobre el papel bond, asegurando la calidad de los dibujos realizados.
- Utilizar técnicas de Lógica Combinacional y Mapas de Karnaugh para optimizar el diseño de los circuitos y garantizar un funcionamiento preciso del Plotter Serial.
- Aprender en detalle el funcionamiento de los motores paso a paso (Stepper) y otros elementos electromecánicos relevantes para la construcción y operación del Plotter.
- Dominar el uso de sensores de color para la correcta alineación del área de impresión del Plotter, asegurando la calidad de los dibujos realizados.
- Integrar de manera efectiva el software necesario para el control de los puertos de la PC y la comunicación serial con el Plotter, garantizando una operación fluida y confiable del sistema.

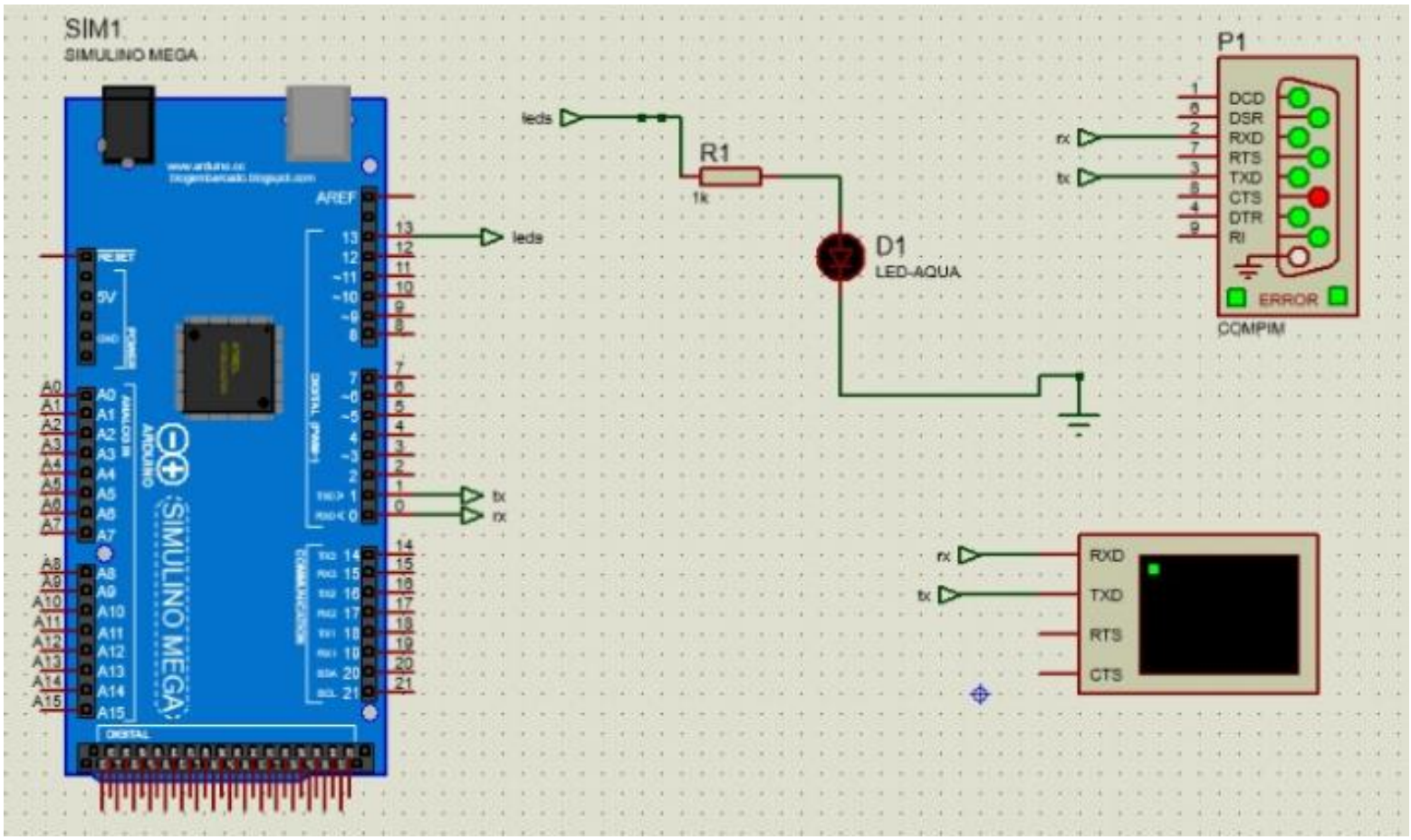
Lógica del Sistema

1. Interfaz de Usuario:
    - El usuario interactúa con la interfaz de usuario para cargar un archivo que contiene la descripción de las figuras a dibujar y sus respectivas ubicaciones en el lienzo.
    - El software de la interfaz lee el archivo y extrae la información necesaria, como las figuras, sus coordenadas y los colores asociados.
2. Comunicación con Arduino:
    - Una vez que se ha cargado el archivo y se ha procesado la información, el software establece una conexión serial con el dispositivo Arduino.
    - Se envían los datos al Arduino, incluyendo las coordenadas de las figuras y los colores correspondientes.
3. Escritura en la Memoria RAM:
    - El Arduino recibe los datos y los almacena temporalmente en una memoria RAM de 4x2, construida en una placa.
    - La memoria RAM retiene la información necesaria para el proceso de impresión, incluyendo las coordenadas de las figuras y los colores asociados.
4. Control de Movimientos del Plotter Serial:
    - Una vez que la información está almacenada en la memoria RAM, el Arduino procesa los datos y controla los movimientos del Plotter Serial.
    - El Arduino envía las señales necesarias para que el Plotter Serial mueva el lápiz de manera precisa y dibuje las figuras en las ubicaciones especificadas en el lienzo.
    - El Plotter Serial realiza los movimientos requeridos para dibujar las figuras seleccionadas en el papel, utilizando los colores indicados.

Diagramas del Diseño del Circuito

MEMORIA RAM 4X2




SIM1
SIMULINO MEGA
leds
R1
1k
D1
LED-AQUA
P1
DCD
DSR
RXD
RTS
TXD
CTS
DTR
RI
ERROR
COMPIM
rx
tx
AREF
RESET
5V
GND
POWER
ANALOG IN
DIGITAL (PWM~)
COMMUNICATION
DIGITAL
SIMULINO MEGA
ARDUINO
RXD
TXD
RTS
CTS

Código comentado

```
#include <Separador.h> // Incluye la librería Separador.h

// Definiciones de pines
#define P0 22
#define P1 24
#define N0 26
#define N1 28
#define RW 30
#define RW 32 // ¡Esta definición está duplicada!
#define P0L 23
#define P1L 25
#define N0L 27
#define N1L 29
Separador s; // Instancia un objeto de la clase Separador
void setup() {
  Serial.begin(9600); // Inicia la comunicación serial a 9600 baudios
  // Configura los pines como entrada o salida
  pinMode(P0, OUTPUT);
  pinMode(P1, OUTPUT);
  pinMode(N0, OUTPUT);
  pinMode(N1, OUTPUT);
```

```
  pinMode(RW, OUTPUT);
  pinMode(RW, OUTPUT); // ¡Este pin se configura dos veces como salida!
  pinMode(P0L, INPUT);
  pinMode(P1L, INPUT);
  pinMode(N0L, INPUT);
  pinMode(N1L, INPUT);
  pinMode(RWL, INPUT);
  pinMode(RWL, INPUT); // ¡Este pin se configura dos veces como entrada!
  Serial.println("Recibió"); // Imprime un mensaje por el puerto serial
}
void loop() {
  while (Serial.available()) {
    serialEvent(); // Llama a la función serialEvent() cuando hay datos disponibles en el puerto serial
  }
}
void serialEvent() {
  String datosrecibidos = Serial.readStringUntil('\n'); // Lee los datos recibidos hasta que se encuentra un salto de línea

  // Separa los datos recibidos en figura, posición X, posición Y y color
```

```
  String figura = s.separa(datosrecibidos, ';', 0);
  String posx = s.separa(datosrecibidos, ';', 1);
  String posy = s.separa(datosrecibidos, ';', 2);
  String color = s.separa(datosrecibidos, ';', 3);


  // Llama a las funciones para configurar la figura, color y posiciones
X e Y
  setfig(figura);
  setcolor(color);
  Posx(posx);
  Posy(posy);
}
// Configura la figura según el dato recibido
void setfig(String fig) {
  digitalWrite(P0, LOW);
  digitalWrite(P1, LOW);
  if (fig == "O") {
    digitalWrite(N0, LOW);
    digitalWrite(N1, LOW);
    Serial.println("La figura recibida en Arduino es O");
  } else if (fig == "X") {
    digitalWrite(N0, HIGH);
```

```
    digitalWrite(N1, LOW);
    Serial.println("La figura recibida en Arduino es X");
  } else if (fig == "estrella") {
    digitalWrite(N0, LOW);
    digitalWrite(N1, HIGH);
    Serial.println("La figura recibida en Arduino es estrella");
  } else if (fig == "triangulo") {
    digitalWrite(N0, HIGH);
    digitalWrite(N1, HIGH);
    Serial.println("La figura recibida en Arduino es triángulo");
  }
  digitalWrite(RW, HIGH);
  digitalWrite(RW, LOW);
}
// Configura el color según el dato recibido
void setcolor(String colour) {
  digitalWrite(P0, HIGH);
  digitalWrite(P1, HIGH);

  if (colour == "magenta") {
    digitalWrite(N0, LOW);
    digitalWrite(N1, LOW);
```

```
    Serial.println("El color recibido en Arduino es magenta");
  } else if (colour == "amarillo") {
    digitalWrite(N0, HIGH);
    digitalWrite(N1, LOW);
    Serial.println("El color recibido en Arduino es amarillo");
  } else if (colour == "negro") {
    digitalWrite(N0, HIGH);
    digitalWrite(N1, LOW);
    Serial.println("El color recibido en Arduino es negro");
  } else if (colour == "cyan") {
    digitalWrite(N0, HIGH);
    digitalWrite(N1, HIGH);
    Serial.println("El color recibido en Arduino es cyan");
  }
  digitalWrite(RW, HIGH);
  digitalWrite(RW, LOW);
}

// Configura la posición X según el dato recibido
void Posx(String x) {
  digitalWrite(P0, HIGH);
  digitalWrite(P1, LOW);
```

```
  if (x == "1") {
    digitalWrite(N0, HIGH);
    digitalWrite(N1, LOW);
    Serial.println("La posición X recibida en Arduino es 1");
  } else if (x == "2") {
    digitalWrite(N0, LOW);
    digitalWrite(N1, HIGH);
    Serial.println("La posición X recibida en Arduino es 2");
  } else if (x == "3") {
    digitalWrite(N0, HIGH);
    digitalWrite(N1, HIGH);
    Serial.println("La posición X recibida en Arduino es 3");
  }
  digitalWrite(RW, HIGH);
  digitalWrite(RW, LOW);
}

// Configura la posición Y según el dato recibido
void Posy(String y) {
  digitalWrite(P0, LOW);
  digitalWrite(P1, HIGH);
```

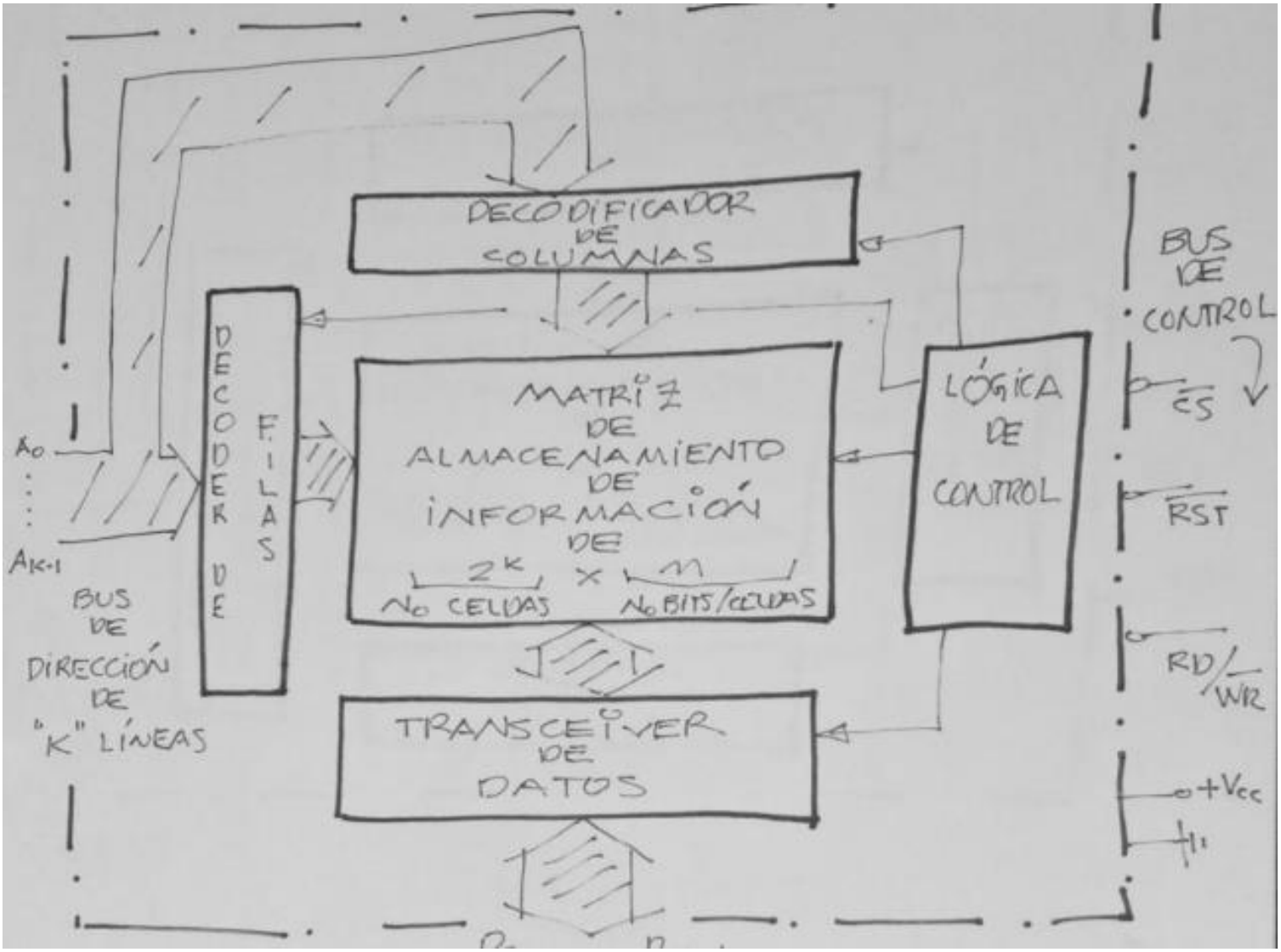
```
  if (y == "1") {
    digitalWrite(N0, HIGH);
    digitalWrite(N1, HIGH);
    Serial.println("La posición Y recibida en Arduino es 1");
  } else if (y == "2") {
    digitalWrite(N0, LOW);
    digitalWrite(N1, HIGH);
    Serial.println("La posición Y recibida en Arduino es 2");
  } else if (y == "3") {
    digitalWrite(N0, HIGH);
    digitalWrite(N1, HIGH);
    Serial.println("La posición Y recibida en Arduino es 3");
  }
  digitalWrite(RW, HIGH);
  digitalWrite(RW, LOW);
}

void lectura() {
  // Esta función está vacía y no se utiliza en el código principal
}
```

Equipo Utilizado

El equipo utilizado en la siguiente practica fue el siguiente:

EQUIPO UTILIZADO
Protoboard
motor steper
Cable para protoboard
ARDUINO MEGA
Sensor de color
Flip-Flop RS
LEDS
Resistencias de diferente kilo ohmios
batería de 9v
multímetro
Placa fenólica
Cloruro Férrico
Compuertas Lógicas 7432, 7404, 7408, 7486
Brocas para PCB
Barreno
Papel termotransferible
Marcador permanente negro
Cautín
Buffer
Contadores
Decoder

Diagramas con Explicación

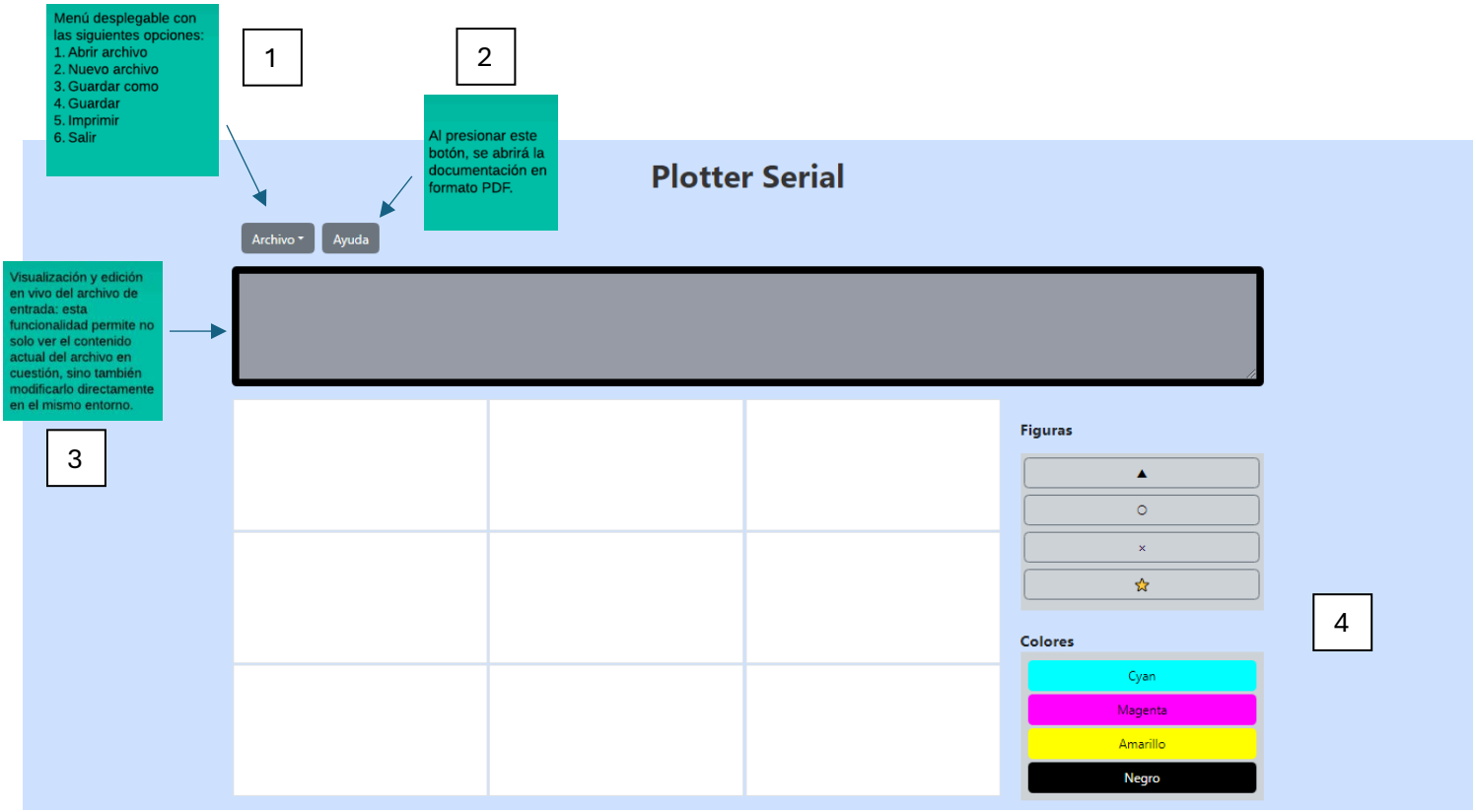


Manual de Usuario

¡Bienvenido al programa de Plotter Serial! Este software ha sido diseñado para brindarte una experiencia intuitiva y eficiente en la creación y edición de dibujos, utilizando un Plotter Serial para plasmar tus ideas en papel con precisión y calidad.

Con este programa, podrás cargar archivos existentes, crear nuevos diseños desde cero, modificarlos según tus necesidades y finalmente imprimirlos físicamente utilizando el Plotter Serial.

Descripción de los pasos para el uso de las opciones más importantes:



1. Archivo: Este botón tiene las opciones de Abrir un archivo(cargarlo), Nuevo archivo, Guardar, Imprimir y salir.
2. Ayuda: Este botón sirve para mostrar la documentación.
3. Visualización y edición: Aquí se ve reflejado el archivo que se cargó y se podrá modificar.
4. Figuras y Colores: Estos botones sirven para elegir la figura y color para luego ponerlo en el tablero.

Luego de cargar nuestro archivo de entrada. Le damos a “imprimir” esto envía al backend la Figura, Fila, Columna y Color y procede a empezar imprimirlo físicamente.

IMAGEN DE IMPRESORA (FISICO)

Manual Técnico

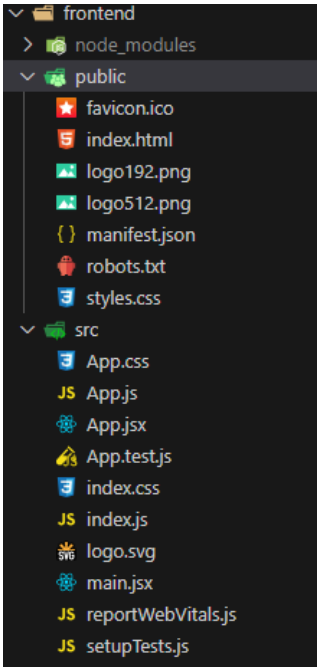
La solución se compone de un software de Plotter Serial que combina un frontend desarrollado con React y JavaScript, junto con HTML y CSS para la interfaz de usuario. Para la gestión de archivos de entrada, se utilizó Python en conjunto con la biblioteca PyLex. Esta solución ofrece una plataforma fácil de usar para cargar, crear y editar dibujos, permitiendo al usuario seleccionar figuras y colores para su impresión.

Requerimientos mínimos del sistema

- Tener instalado Proteus, Node.js, Python y librerías varias.
- IDE: en nuestro Visual Studio Code y Arduino IDE
- Controlador de versiones: GIT

Estructura del programa:

Frontend:



funciones y código utilizado:

```
const handleCellClick = (rowIndex, colIndex) => {
  setBoard((prevBoard) => {
    const newBoard = [...prevBoard];
    const cell = newBoard[rowIndex][colIndex];
    if (cell.shape === null) {
      // Only update if the cell is empty
      newBoard[rowIndex][colIndex] = {
        color: selectedColor,
        shape: selectedShape,
      };
    }
    return newBoard;
  });
};
```

```
const handleColorSelection = (color) => {
  setSelectedColor(color);
};

const handleShapeSelection = (shape) => {
  setSelectedShape(shape);
};

const toggleFileMenu = () => {
  setIsFileMenuOpen(!isFileMenuOpen);
};

const NewFile= () => {
  setFileContent('');
};

const handleOpenFile = (event) => {
  const file = event.target.files[0];
  const reader = new FileReader();
  reader.onload = function (event) {
    setFileContent(event.target.result);
  };
  reader.readAsText(file);
};
```

```
const handleSaveFile = () => {
  const fileName = prompt("Please enter the file name");
  if (fileName) {
    const element = document.createElement("a");
    const file = new Blob([fileContent], {type: 'text/plain'});
    element.href = URL.createObjectURL(file);
    element.download = `${fileName}.orga`; // Always use .orga extension
    document.body.appendChild(element);
    element.click();
  }
};
```

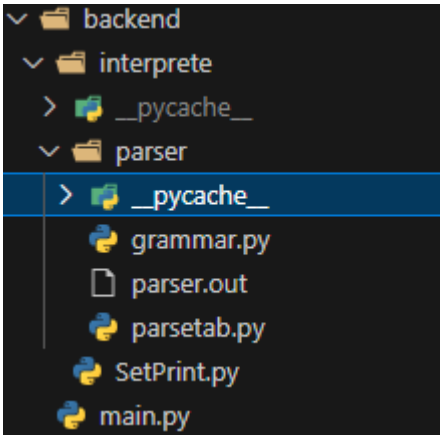
```
const handleSubmit = () => {
  fetch("http://localhost:5000/sendData", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ code_in: fileContent }),
  })
    .then((response) => {
      if (!response.ok) {
        throw new Error("Error al enviar los datos al servidor");
      }
      return response.json();
    })
    .then((data) => {
      console.log("Respuesta del servidor:", data);
      // Aquí puedes manejar la respuesta del servidor según tus necesidades
    })
    .catch((error) => {
      console.error("Error:", error);
      // Aquí puedes manejar errores de solicitud o del servidor
    });
};
```

```
test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fu
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <link rel="stylesheet" href="style.css">
    <title>React App</title>
  </head>
  <body style="background-color: #cde1ff;">

    <h1 class="page-title">Plotter Serial</h1>

    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.j
```

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;

}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;

}

.page-title {
  text-align: center;
  padding: 20px;
  font-size: 2.5rem;
  font-weight: bold;
  color: #333;
}
```

Backend:



grammar.py

```
reserved = {
    "new_print" : "NEW_PRINT",
    "set_print_x" : "SET_PRINT_X",
    "set_print_y" : "SET_PRINT_Y",
    "set_print_o" : "SET_PRINT_O",
    "set_print_triangulo" : "SET_PRINT_TRIANGULO",
    "set_print_estrella" : "SET_PRINT_ESTRELLA",
    "cyan" : "CYAN",
    "magenta" : "MAGENTA",
    "amarillo" : "AMARILLO",
    "negro" : "NEGRO",
    "end_print" : "END_PRINT"
}

tokens = [
    'PARA', 'PARC', 'COMA', 'PUNTOYCOMA',
    'NUMBER', 'ID',
] + list(reserved.values())

t_PARA = r'\('
t_PARC = r'\)'
t_COMA = r'\,'
t_PUNTOYCOMA = r'\;'

def t_NUMBER(t):
    r'\d+'
    try:
        t.value = int(t.value)
    except ValueError:
        print(f'error al parsear el valor: {t.value} \n column: {t.lexpos} line: {t.lineno}')
        t.value = None
    return t

def t_ID(t):
    r'[a-zA-Z_][a-zA-Z_0-9]*'
    t.type = reserved.get(t.value,'ID')
```

```
def t_newline(t):
    t.lexer.lineno += t.value.count("\n")

def t_error(t):
    try:
        print(f'Error lexico {t.value} \n column: {t.lexpos} line: {t.lineno}')
    except Exception as e:
        print(f"Ocurrió un error: {str(e)}")
    t.lexer.skip(1) # recuperacion del error

def p_start(p):
    '''start    : NEW_PRINT ID PUNTOYCOMA instrucciones END_PRINT PUNTOYCOMA'''
    p[0] = p[4]
    return p[0]

def p_instrucciones(p):
    '''instrucciones : instrucciones instruccion
                     | instruccion'''
    if len(p) > 2:
        p[1].append(p[2])
        p[0] = p[1]
    else:
        p[0] = [p[1]]

def p_instruccion(p):
    '''instruccion : SET_PRINT_X PARA NUMBER COMA NUMBER COMA color PARC PUNTOYCOMA
                   | SET_PRINT_Y PARA NUMBER COMA NUMBER COMA color PARC PUNTOYCOMA
                   | SET_PRINT_O PARA NUMBER COMA NUMBER COMA color PARC PUNTOYCOMA
                   | SET_PRINT_TRIANGULO PARA NUMBER COMA NUMBER COMA color PARC PUNTOYCOMA
                   | SET_PRINT_ESTRELLA PARA NUMBER COMA NUMBER COMA color PARC PUNTOYCOMA'''

    if p[1] == 'set_print_x':
        p[0] = SetPrint('X', p[3], p[5], p[7])
    elif p[1] == 'set_print_y':
        p[0] = SetPrint('Y', p[3], p[5], p[7])
    elif p[1] == 'set_print_o':
        p[0] = SetPrint('O', p[3], p[5], p[7])
    elif p[1] == 'set_print_triangulo':
        p[0] = SetPrint('T', p[3], p[5], p[7])
    elif p[1] == 'set_print_estrella':
        p[0] = SetPrint('E', p[3], p[5], p[7])

    return p[0]
```

```
def p_color(p):
    '''color    : CYAN
                | MAGENTA
                | AMARILLO
                | NEGRO '''

    p[0] = p[1]
    return p[0]


def p_error(p):
    try:
        if p:
            print(f'Error sintactico linea:{p.lineno}, col:{p.lexpos} Token: {p.value}')
        else:
            print(f'Error de sintaxis  \n column: {p.lexpos} line: {p.lineno}')
    except Exception as e:
        print(f"Ocurrió un error: {str(e)}")

class codeParams:
    def __init__(self, line, column):
        self.line = line
        self.column = column

def get_params(t):
    line = t.lexer.lineno  # Obtener la línea actual desde el lexer
    lexpos = t.lexpos if isinstance(t.lexpos, int) else 0  # Verificar si lexpos es un entero
    column = lexpos - t.lexer.lexdata.rfind('\n', 0, lexpos)
    return codeParams(line, column)


def parse(input_text):
    lex.lex() #lexico
    parser = yacc.yacc() #sintactico
    result = parser.parse(input_text)
    return result
```

Main.py

```python
app = Flask(__name__)
CORS(app)

@app.route('/', methods=['GET'])
def index():
    return jsonify({'status': 200, 'message': 'Hello, world!'})


@app.route('/sendData', methods=['POST'])
def sendData():
    if not request.json or 'code_in' not in request.json:
        return jsonify({"error": "La solicitud debe ser un JSON y contener el campo 'code_in'"}), 400

    code_in = request.json['code_in']
    print(code_in)

    ast = grammar.parse(code_in)

    try:
        print("\nformato de instrucciones:  Figura;Fila;Columna;Color")
        for instruction in ast:
            tmp = instruction.execute()
            print("codigo enviado: "+ str(tmp.encode('ascii')))

            # serialArduino.write(tmp.encode('ascii'))
            # print("codigo en consola serial de arduino: " + serialArduino.readline())
    except Exception as e:
        print(f"Ocurrió un error: {str(e)}")
        return jsonify({"error": "Ocurrió un error al ejecutar el código"}), 500

    result = {
        "status": 200,
        "message": "Código ejecutado correctamente",
    }
    return jsonify(result)

if __name__ == '__main__':
    app.run(port=5000)

if __name__ == "__main__":
    app.run(debug=True)
```

```
#define P0 22
#define P1 24
#define N0 26
#define N1 28
#define RW 30
#define RW 32
#define P0L 23
#define P1L 25
#define N0L 27
#define N1L 29

Separador s;

void setup() {
 Serial.begin(9600);

  pinMode(P0, OUTPUT);
  pinMode(P1, OUTPUT);
  pinMode(N0, OUTPUT);
  pinMode(N1, OUTPUT);
  pinMode(RW, OUTPUT);
  pinMode(RW, OUTPUT);
  pinMode(P0L, INPUT);
  pinMode(P1L, INPUT);
  pinMode(N0L, INPUT);
  pinMode(N1L, INPUT);
  pinMode(RWL, INPUT);
  pinMode(RWL, INPUT);
 Serial.println("recibio");
```

```
  if(fig=="O"){
   digitalWrite(N0,LOW);
   digitalWrite(N1,LOW);
   Serial.println("La figura recibida en ard es O");
  }
  else if(fig=="X"){
   digitalWrite(N0,HIGH);
   digitalWrite(N1,LOW);
   Serial.println("La figura recibida en ard es X");
  }
  else if(fig=="estrella"){
   digitalWrite(N0,LOW);
   digitalWrite(N1,HIGH);
   Serial.println("La figura recibida en ard es estrella");
  }
  else if(fig=="triangulo"){
   digitalWrite(N0,HIGH);
   digitalWrite(N1,HIGH);
   Serial.println("La figura recibida en ard es triangulo");
  }
   digitalWrite(RW,HIGH);
   digitalWrite(RW,LOW);
}
 void setcolor(String colour){
  digitalWrite(P0,HIGH);
   digitalWrite(P1,HIGH);
```

```
}
 void setcolor(String colour){
  digitalWrite(P0,HIGH);
   digitalWrite(P1,HIGH);



  if(colour=="magenta"){
   digitalWrite(N0,LOW);
   digitalWrite(N1,LOW);
   Serial.println("El color recibida en ard magenta "
  }
  else if(colour=="amarillo"){
   digitalWrite(N0,HIGH);
   digitalWrite(N1,LOW);

   Serial.println("El color recibida en ard amrll ");
  }
   else if(colour=="negro"){
   digitalWrite(N0,HIGH);
   digitalWrite(N1,LOW);

   Serial.println("El color recibida en ard negro");

  }
   else if(colour=="cyan"){
   digitalWrite(N0,HIGH);
   digitalWrite(N1,HIGH);

   Serial.println("El color recibida en ard cyan");
```

Presupuesto

Gastos

	Nombre	Cantidad	Precio	Precio total
1	Arduino Mega	1	210	210
2	Ledd de colores	10	1	10
3	Flip-Flop R-S	8	15	120
4	Compuerta AND	8	7.50	60
5	Compuerta NOT	4	6	24
6	Buffer	8	13	104
7	Motores steper	3	6	18
8	Protoboards	2	36	72
9	Decoder 74LS48	2	15	30
10	Sensor de Color	1	100	100
11	Piezas con impresora 3d	1	100	100
12	Cargador de 5v	1	0	0
13	Resistencias de 1k ohm	15	0.75	11.25
14	Placa de cobre	3	12	36
15	Brocas 1/32	2	3	6
16	Brocas 1/16	2	3	6
	TOTAL			907.25

Aporte de Cada Integrante

INTEGRANTES	APORTE (Q)
Bismarck Estuardo Romero Lemus	100.83
Naomi Rashel Yos Cujcuj	100.83
Josué Nabí Hurtarte Pinto	100.61
Angela Paulina Rodriguez López	100.83
Rodrigo Alejandro Tahuite Soria	100.83
Nathan Antonio Valdez Valdez	100.83
Gonzalo Fernando Pérez Cazún	100.83
Rony Omar Miguel López	100.83
Kevin Eduardo Castañeda Hernández	100.83
TOTAL	907.25

Conclusiones

- La implementación exitosa de la comunicación entre el software de la interfaz de usuario y el dispositivo Arduino, así como la integración de una memoria RAM, demuestra la eficacia del sistema en el control y la coordinación de los movimientos del Plotter Serial para la impresión precisa de figuras.
- La combinación de tecnologías como React, JavaScript, Python, PyLex y Arduino ha permitido desarrollar un sistema completo y funcional que ofrece una experiencia fluida y eficiente al usuario, facilitando la creación y edición de dibujos para su impresión con el Plotter Serial.
- La utilización de una memoria RAM de 4x2 ha contribuido significativamente a la optimización del proceso de impresión, permitiendo el almacenamiento temporal de los datos necesarios para el dibujo, lo que ha mejorado la velocidad y precisión del sistema en la producción de figuras en el papel.
- La exitosa implementación del sistema de Plotter Serial no solo ha demostrado la capacidad del equipo para aplicar los conocimientos teóricos adquiridos, sino también su habilidad para integrar diferentes tecnologías y componentes hardware y software en un proyecto coherente y funcional.

Recomendaciones

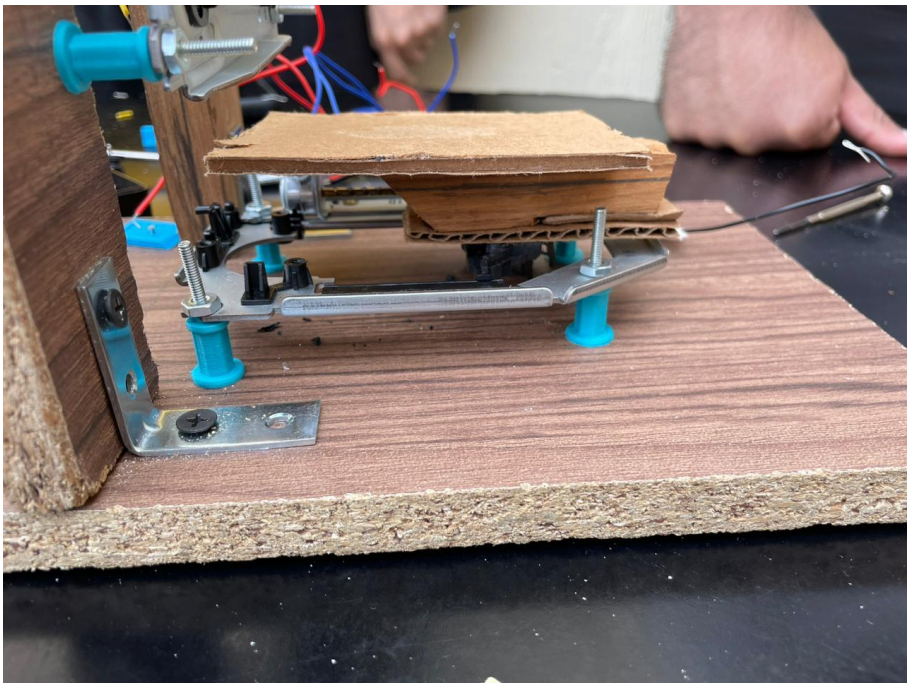
- Se recomienda seleccionar cuidadosamente los componentes mecánicos del Plotter Serial, asegurándose de que sean de alta calidad y estén diseñados para soportar el uso continuo y el movimiento preciso del lápiz.
- Para el dispositivo Arduino, se sugiere seguir buenas prácticas de programación y realizar pruebas exhaustivas antes de implementar cambios en el código, con el fin de evitar errores y asegurar un rendimiento estable del sistema.
- En el caso del Arduino, se aconseja seguir las especificaciones del fabricante y utilizar componentes compatibles y de buena calidad para garantizar la estabilidad y fiabilidad del sistema en su conjunto.
- Imprimir en 3 piezas para armar la base física de nuestro brazo o impresora a realizar.

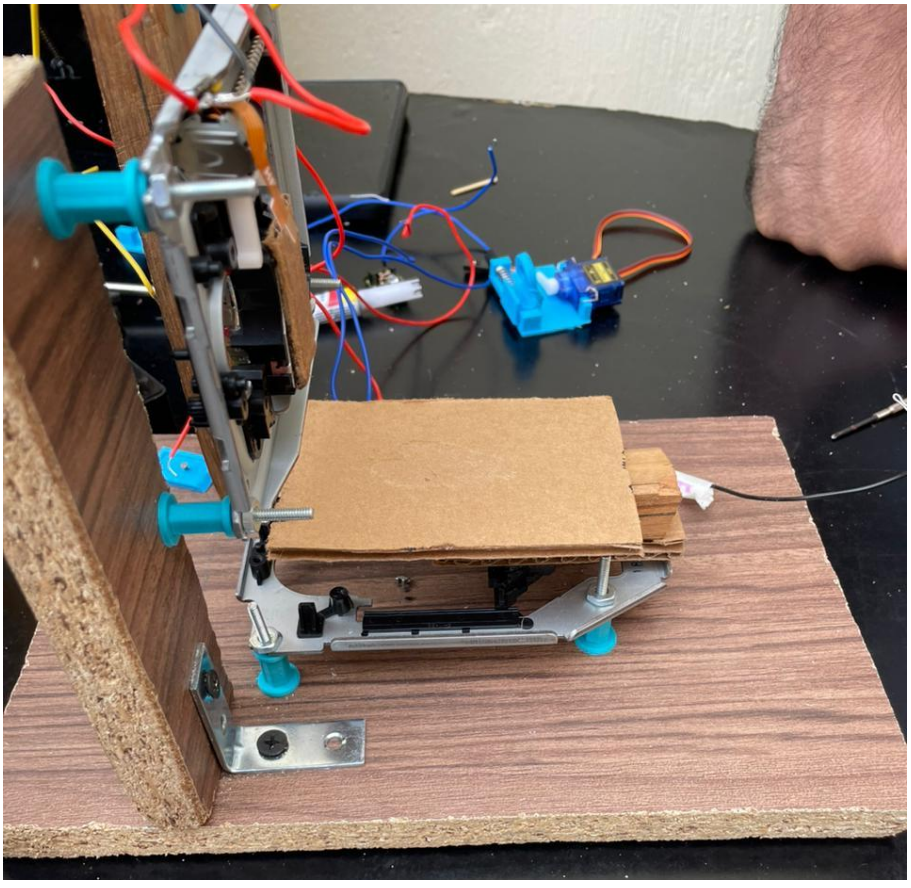
  .

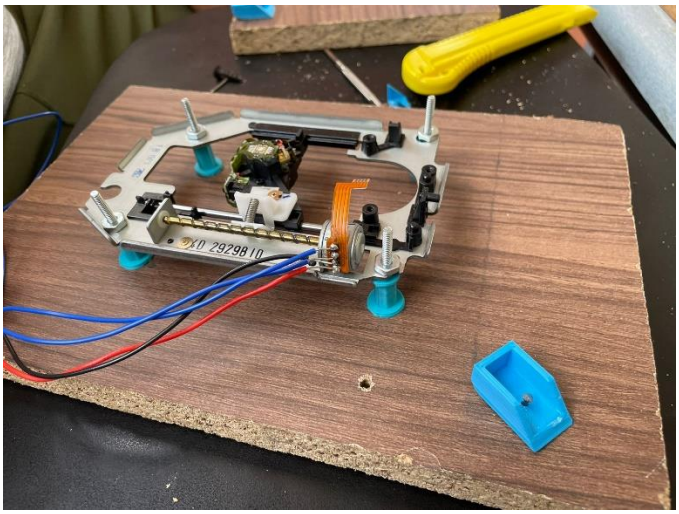
Tabla de las combinaciones para las figuras y sus colores

	P1	P2	N1	N0	
FIGURAS	0	0	0	0	o
FIGURAS	0	0	0	1	X
FIGURAS	0	0	1	0	★
FIGURAS	0	0	1	1	▲
POS X	0	1	0	1	1
POS X	0	1	1	0	2
POS X	0	1	1	1	3
POS Y	1	0	0	1	1
POS Y	1	0	1	0	2
POS Y	1	0	1	1	3
COLOR	1	1	0	0	MAGENTA
COLOR	1	1	0	1	AMARILLO
COLOR	1	1	1	0	NEGRO
COLOR	1	1	1	1	CYAN

Anexos








Si funciona profe, pero solo cuando le pongo el dedo encima, si se lo quito deja de funcionar