Proyecto Final:
Imprensión de logo con Arduino en Pantalla LCD 3310 de Pixeles

Rosa Isela Ortiz Guzmán
Informática Industrial
Universidad de Guanajuato Sede Yuriria
Guanajuato, México

María Guadalupe Espinosa Sánchez
Informática Industrial
Universidad de Guanajuato Sede Yuriria
Guanajuato, México

I. ABSTRACT

***Abstract*—El presente reporte mostrara el desarrollo que se siguió para lograr la Impresión de logos con la Pantalla LCD 3310 de Pixeles, basado en investigación e implementación en Arduino.**

Keywords—Arduino, Pixeles, Impresión, LCD, 3310, Industrial.

II. INTRODUCCIÓN

El píxel es la unidad más pequeña y diminuta de una imagen digital y está presente en un inmensurable número para formar una imagen completa. Cada píxel es una unidad homogénea de color que en suma y con una importante variación de colores dan como resultado una imagen más o menos compleja. Pueden contar con tres o cuatro elementos de color a elegir: rojo, verde y azul o magenta, amarillo y cian.

Todos los píxeles son cuadrados o rectangulares y pueden ser de color, blancos, negros o grises en diferentes tonalidades. Las combinaciones posibles de color son infinitas y han llegado a ser muy desarrolladas en comparación con las primeras imágenes digitales que carecían de suavidad y realidad.

La historia del píxel se remonta a principios de la década del 30 cuando empezó a ser utilizado el concepto para el cine. El término píxel hacía referencia a un elemento de imagen o "picture element". También es entendido por muchos como la célula menor que compone el complejo sistema que puede llegar a ser una imagen digital. Esta idea fue acuñada en la década del 70 y aplicada también a la televisión antes que a las computadoras.

En el presente reporte se desarrollará la implementación de impresión de Logos en la Pantalla LCD 3310 de Pixeles, la cual únicamente se habíamos usado utilizando con Microprocesador.

III. MARCO TEÓRICO

Materiales:

Arduino Uno:

El Arduino es una placa basada en un microcontrolador ATMEL. Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, las cuales las escribes con el lenguaje de programación que puedes utilizar en el entorno Arduino IDE. Estas instrucciones permiten crear programas que interactúan con los circuitos de la placa.

El microcontrolador de Arduino posee lo que se llama una interfaz de entrada, que es una conexión en la que podemos conectar en la placa diferentes tipos de periféricos. La información de estos periféricos que conectes se trasladará al microcontrolador, el cual se encargará de procesar los datos que le lleguen a través de ellos.

El tipo de periféricos que puedas utilizar para enviar datos al microcontrolador depende en gran medida de qué uso le estés pensando dar. Pueden ser cámaras para obtener imágenes, teclados para introducir datos, o diferentes tipos de sensores.

También cuenta con una interfaz de salida, que es la que se encarga de llevar la información que se ha procesado en el Arduino a otros periféricos. Estos periféricos pueden ser pantallas o altavoces en los que reproducir los datos procesados, pero también pueden ser otras placas o controladores.

Pantalla LCD 3310 pixeles:

Se trata de una pantalla gráfica útil en proyectos con microcontroladores como el Arduino. La presentación es en forma de módulo, que distribuye todas las señales de control en headers macho de 0.1″ fáciles de manejar que ayudan a conectar la pantalla a prototipos o circuitos definitivos sin necesidad de partes especiales

La pantalla tiene un controlador PCD8544 que es el mismo que se utilizó en la pantalla del Nokia 3310. El controlador de pantalla se comunica con el procesador principal a través de

una interfaz serie SPI, por lo que se requieren pocos pines para manejar el display.
Tiene una resolución de 84 x 48 pixeles monocromática lo que quiere decir que tiene 48 filas y 84 columnas

Herramientas:

IDE Arduino:

Dado que el Arduino es como un pequeño ordenador que ejecuta una serie de códigos que previamente le hemos introducido, necesitaremos un programa para poder meter estos códigos a la propia placa. Este programa se llama IDE, que significa "Integrated Development Environment" ("Entorno de Desarrollo Integrado"). Este IDE estará instalado en nuestro PC, es un entorno muy sencillo de usar y en él escribiremos el programa que queramos que el Arduino ejecute. Una vez escrito, lo cargaremos a través del USB y Arduino comenzará a trabajar de forma autónoma.

Proteus:

Proteus es una aplicación para la ejecución de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño del esquema electrónico, programación del software, construcción de la placa de circuito impreso, simulación de todo el conjunto, depuración de errores, documentación y construcción.

LCD Assistant:
Es una herramienta gratuita diseñada para convertir mapas de bits monocromáticos en matrices de datos para un uso fácil con programas para sistemas integrados con microcontroladores y pantallas LCD monocromáticas de gráficos. El programa crea archivos que se pueden usar con cualquier compilador de C para AVR, ARM, PIC, 8051 y otros microcontroladores.

IV. METODOLOGÍA

Para el desarrollo de la implementación para la Impresión de logos con Pixeles en la Pantalla LCD 3310 se realizó una pequeña investigación sobre el grafico de la Pantalla y sus conexiones, para ello nos basamos en lo siguiente. Vea la **Figura 1.**

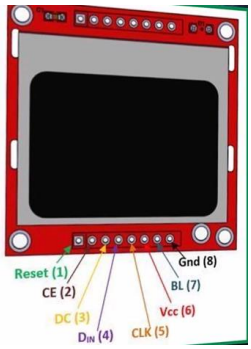

Figura 1. *Pines*

A continuación, se muestra para que es cada pin:
1. RST – Reset
2. CE - Chip Select
3. D/C-Data/Instruction Selection
4. DIN - Serial Data Line
5. CLK-Serial Clock Line
6. VCC - Power Input (3.3 V and 5 V available)
7. BL - Backlight Control
8. GND - Ground

Los pines del 1 al 5 pueden ir conectados a cualquier pin digital, el 6 a 3.3 v, con el 7 se puede encender o apagar la luz del fondo y el 8 a tierra.

Basándonos en ello construimos un circuito en Proteus el cual quedo de la siguiente manera:
Vea **Figura 2.**

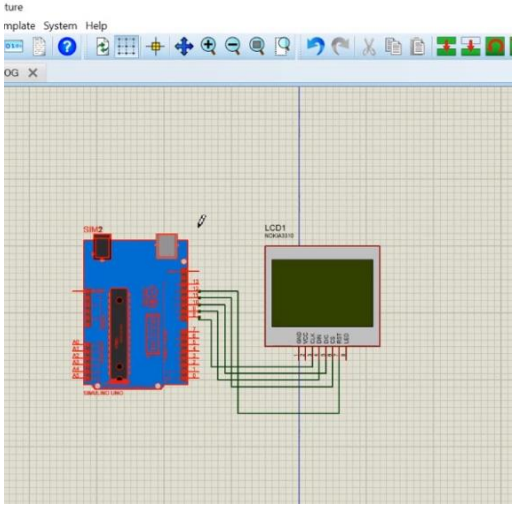

Figura 2. *Circuito.*

Posteriormente realizamos nuestro código en el cual tuvimos que descargar las librerías de **Adafruit de la LCD de Nokia 5110** la cual nos seria de ayuda ya que es la librería más completa y la compatible con varios LCD.
Posteriormente instalamos la Librería **Adafruit GFX** la cual nos ayudaría con la interfaz.

El código quedo de la siguiente manera:

```
#include <Nokia_LCD.h>

#define PIN_RESET 12
#define PIN_SCE   11
#define PIN_DC    10
#define PIN_SDIN  9
#define PIN_SCLK  8
#define LCD_C     LOW
#define LCD_D     HIGH
#define LCD_X     84
#define LCD_Y     48
```

Figura 3. *Librerias y definicion de pines.*

La **Figura 3** muestra la llamada a las librerías antes mencionadas, esto para poder manejar la pantalla y la interfaz de esta.
Posteriormente se muestra la declaración de los pines que se utilizaron y el tamaño que tiene la pantalla en horizontal (X) y en vertical (Y).
A continuación, se muestra la parte más compleja del trabajo ya que cuando se utilizo la pantalla en microprocesador todo se hizo a mano para poder llenar cada píxel de la pantalla, convirtiendo de números binarios a hexadecimales.
Sin embargo, para hacer esto más sencillo buscamos alguna herramienta que nos pudiese ayudar con las conversiones para ahorrar tiempo, la herramienta que se utilizo fue LCD Assistant.

```
int p[]={
0xF8, 0x00, 0x00, 0xF8, 0xF8, 0xF8, 0x00, 0xF8, 0xF8, 0xF8, 0x38, 0x18, 0x18, 0x08, 0x08, 0x08,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08,
0x08, 0x18, 0x18, 0x38, 0x78, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0x78, 0x38, 0xB8, 0x18,
0x58, 0x58, 0x58, 0x48, 0x48, 0x48, 0x40, 0x40, 0xC0, 0xD0, 0xD0, 0x90, 0xB0, 0x20, 0x60, 0x40,
0xC0, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0xFF, 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, 0x03, 0xFF, 0x07, 0x03, 0x01, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01,
0x03, 0x07, 0x3F, 0xFE, 0xFC, 0xF9, 0xC3, 0x0F, 0xFE, 0xFC, 0xF8, 0xE0, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0xFF,
0xFF, 0xFF, 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, 0xFF,
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, 0x01, 0x01, 0x01, 0x01, 0x00, 0x01, 0x01, 0x01, 0x01,
0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0xFF,
0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0xFF, 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, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x10, 0x70, 0xF0, 0xF0, 0x00, 0x00, 0xF0,
0xF0, 0xF0, 0x00, 0x00, 0xF0, 0xF0, 0xF0, 0x30, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x7F, 0xF8, 0xC0, 0x0F, 0x7F, 0xFF, 0xE0, 0x8F, 0x3F, 0xFF, 0xE0, 0x80, 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, 0x80, 0xE0, 0x3C, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xF0,
0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
```

Figura 4. *Funcion con el codigo en hexadecimal para la imagen.*

Posterioemnete tenemos la funcion LcdClear que es quien se encarga de hacer que nuesstro logo a imprimir comience desde el lado izquierdo hasta el derecho utilizando toda la pantalla.
Vea **Figura 5.**

```
void LcdClear(void)
{
  for (int index = 0; index < LCD_X * LCD_Y / 8; index++)
  {
    LcdWrite(LCD_D, 0x00);
  }
}
```

Figura 5. *Funcion para escibir en toda la pantalla.*

Después tenemos la función LcdInitialise la cual es encargada de hacer que la pantalla se inicie, es aquí donde se muestran los pines de entrada y salida, se ve después que pines van iluminados haciendo referencia a los estados de la **Figura 3.**

Vea **Figura 6.**

```
void LcdInitialise(void)
{
  pinMode(PIN_SCE, OUTPUT);
  pinMode(PIN_RESET, OUTPUT);
  pinMode(PIN_DC, OUTPUT);
  pinMode(PIN_SDIN, OUTPUT);
  pinMode(PIN_SCLK, OUTPUT);
  digitalWrite(PIN_RESET, LOW);
  digitalWrite(PIN_RESET, HIGH);
  LcdWrite(LCD_C, 0x21 );
  LcdWrite(LCD_C, 0xB1 );
  LcdWrite(LCD_C, 0x04 );
  LcdWrite(LCD_C, 0x14 );     //qu
  LcdWrite(LCD_C, 0x0C );
  LcdWrite(LCD_C, 0x20 );
  LcdWrite(LCD_C, 0x0C );
}
```

Figura 6. *Función para iniciar la pantalla.*

Posteriormente tenemos la función gotoXY la cual se encarga de recorrer la toda la pantalla. Vea la **Figura 7.**

```
void gotoXY(int x, int y)
{
  LcdWrite (0, 0x84 | x);
  LcdWrite (0, 0x48 | y);
}
```

Figura 7. *Recorrido de pantalla.*

Posteriormente tenemos las funciones de LcdWrite que como su nombre lo dice es para que la LCD escriba y tenemos la función LcdBitmap que hace referencia a las coordenadas del mapa de bits de la pantalla. Vea **Figura 8.**

```
void LcdWrite(byte dc, byte data)
{
  digitalWrite(PIN_DC, dc);
  digitalWrite(PIN_SCE, LOW);
  shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, data);
  digitalWrite(PIN_SCE, HIGH);
}

void LcdBitmap(int my_array[]){
  for (int index = 0; index < (LCD_X * LCD_Y/8); index++)
  LcdWrite(LCD_D, my_array[index]);
  }
```

Figura 8. *Escribir en pantalla y mapa de bits.*

Por último, tenemos dos funciones muy importantes ya que con ellas mandamos llamar a las otras funciones para iniciar la LCD, limpiarla si tiene algo cargado y que se inicie desde la primera coordenada (setup), la otra manda llenar el mapa de bits con la función de la **Figura 4**, será aquí donde se imprima el logo que hayamos elegido(loop).
Vea **Figura 9.**

```
void setup(void)
{
  LcdInitialise(); inicia el lcd
  LcdClear();     limpia la pantalla
  gotoXY(0,0);
}
void loop(void)
{
  LcdBitmap(p);    //mapa de bits de la pantalla
  while (p)
  {
  }
}
```

Figura 9. *Función setup y loop.*

Después de la implementación del código se realizó el circuito en físico el cual contiene las conexiones establecidas en la **Figura 2.**
El circuito quedo de la siguiente manera, para ello vea la **Figura 10.**

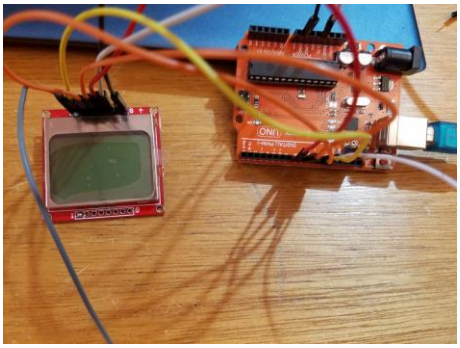

Figura 10. *Circuito Implementado.*

Después de tener el código funcional se cargó al Arduino.
Los resultados se verán la siguiente sección.

V. RESULTADOS

En esta sección veras el resultado de impresión de varios logos.

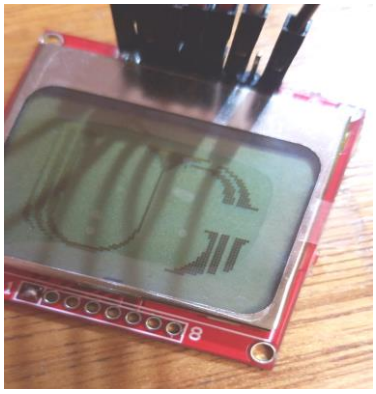

Figura 11. *Logo UG.*

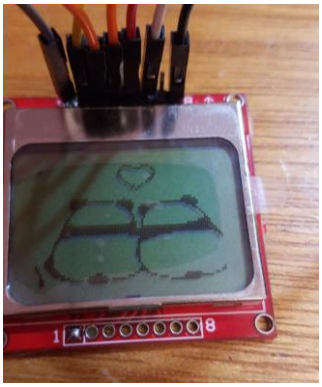

Figura 12. *Logo Pandas.*

Figura 13. *Logo Abeja.*

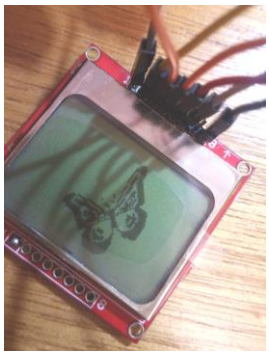

Figura 14. *Logo Mariposa..*

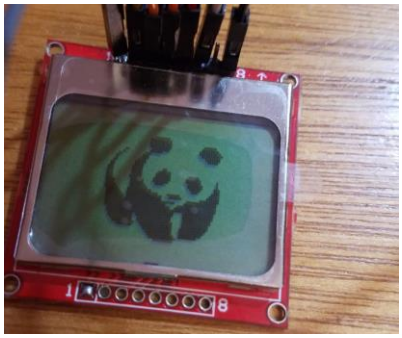

Figura 15. *Logo WFF.*

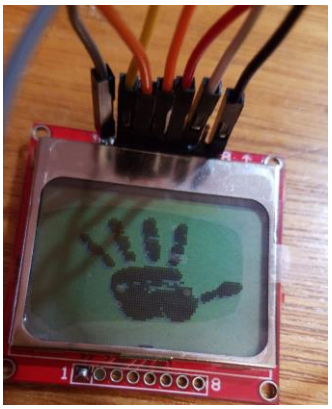

Figura 16. *Logo mano.*

VI. CONCLUSION

La realización de este proyecto tuvo algunas dificultades ya que al principio no encontrábamos mucha información de la pantalla y menos de cómo podríamos conectarla al Arduino, sin embargo, se logró mediante la implementación de una serie de información que encontramos en la red y de esta manera fuimos uniendo partes de la información.

Ya que nunca habíamos trabajado con Arduino otro de los problemas que se nos presento fue como agregar librerías y como subir el código a la placa ya que no la habíamos configurado, al igual que la implementación del circuito la implementación del código para la impresión de los logos fue resultado de investigar a fondo funciones que nos ayudaran a movernos en la pantalla y como pasarlas del proyecto realizado en microprocesadores a Arduino, lo único que nos fallo en este proyecto fue la funcionalidad del circuito en Proteus ya que esta herramienta nunca la habíamos utilizado y al meter la apantalla LCD nos marco errores que no pudimos solucionar.