UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

PROYECTOS DIGITALES AVANZADOS

PRÁCTICA N°13:

GRAFICADORA DE CORAZÓN

Profesor:
Pedro Rene
Cabrera

Bachiller:
Claudia Rodríguez
C.I: 27.943.668

Barcelona, febrero de 2025

1. OBJETIVOS DE LA PRÁCTICA

1.1. Graficar un corazón en la pantalla OLED utilizando la fórmula: $x^2 + (y - \sqrt[3]{x^2})^2 = 1$.

1.2. Graficar punto por punto el corazón como si se estuviera graficando en un plano cartesiano, por lo tanto, se tiene que obtener el punto de origen en el eje x (abscisas) y en el eje y (ordenadas) con respecto a la pantalla OLED.

1.3. Realizarle un escalamiento a la gráfica, es decir, poderla graficar de distintos tamaños, para ello, se le pide al usuario los valores de ancho y alto del corazón.

1.4. Establecer un rango con valores mínimos y máximos para que al momento de graficar el corazón no se deforme o se grafique de manera incorrecta.

1.5. Verificar y manejar errores de entrada del usuario al ingresar valores desde el teclado matricial, asegurando que solo se acepten números dentro del rango permitido.

1.6. Realizar pruebas empíricas para determinar que coeficiente o exponente modifica la curvatura del corazón en su parte superior.

1.7. Agregado: Los valores de las dimensiones del corazón se introducen mediante el uso del keypad matricial y como se mencionó anteriormente, estos valores se validarán.

2. DESARROLLO

2.1. Planteamiento del Problema.

Para la presente práctica, se plantea graficar un corazón en una pantalla OLED de dimensiones 128 x 64 píxeles. Dado el tamaño limitado de la pantalla, es necesario establecer restricciones en las dimensiones del corazón para evitar deformaciones o que se salga del área visible. El programa permitirá que el usuario ingrese las dimensiones del corazón utilizando un teclado matricial 4x4. Los valores permitidos deben estar en los siguientes rangos:

- Ancho: entre 18 y 30 píxeles.
- Alto: entre 11 y 23 píxeles.

El sistema deberá verificar y validar la entrada del usuario. Cuando se presione la tecla *, el programa comprobará si los valores ingresados están dentro del rango permitido. Si no lo están, se solicitará al usuario que los reingrese hasta que sean correctos.

Una vez validados los valores, el corazón se graficará en la pantalla OLED de acuerdo con la fórmula matemática establecida: $x^2 + (y - \sqrt[3]{x^2})^2 = 1$.

Además, el código deberá incluir manejo de errores para prevenir entradas inválidas.

Uno de los retos principales en esta práctica es el escalamiento del corazón, ya que se busca que pueda representarse en distintos tamaños sin deformarse. Para ello, es necesario analizar la fórmula y realizar pruebas empíricas, determinando qué coeficientes o exponentes afectan la curvatura de la figura y cómo influyen en la representación gráfica.

En resumen, el propósito de esta práctica es desarrollar un programa que permita graficar un corazón con dimensiones ajustables mediante un teclado matricial, garantizando que la figura sea correcta y se ajuste al área visible de la pantalla OLED.

2.2. Solución para el problema.

La solución para este problema es la siguiente (se presenta una sola solución dado que el código se ejecuta en un solo Raspberry Pi Pico W a diferencia de prácticas anteriores que se tenían dos o hasta tres soluciones porque se tenían dos Picos W interactuando entre sí.):

1. **Configuración del Keypad y pantalla OLED:** Permite configurar el teclado matricial para poder leer las teclas apretadas. Las funciones que interceden para el manejo del keypad matricial son **init_keypad(), scan_keypad().** Con respecto a la pantalla OLED, que se ha manejado en todas las prácticas, se tienen las mismas funciones **crear_oled()**, y **mostrar_oled(oled, message, n),** donde oled es la instancia para manipular la pantalla, message representa la cadena que se mostrará y por último n, representa un número real para el tiempo en el que el mensaje se mostrará en la pantalla.

2. **Entrada de Datos**: Usando la función **obtener_valor(message, min_val, max_val, oled)**, donde sus parámetros representan: message (que es un str, para mostrar que datos se están solicitando), min_val y max_val (representa el valor mínimo y máximo para el ancho/alto del corazón) y por último, oled, que es para controlar la pantalla OLED y mandar y mostrar mensajes. En esta función, se le pide al usuario que ingrese el ancho y el alto, por medio del teclado matricial. Si el valor está fuera de rango permitido, muestra un mensaje error y el usuario pueda ingresar de nuevo los datos.

3. **Implementación de la función dibujar_corazon(oled, tam_ancho, tam_alto):** Esta función tiene tres parámetros, una oled para el manejo de la pantalla, tam_ancho y tam_alto que fueron solicitados al usuario. Solo se pasa a la función **dibujar_corazon** cuando los datos de ancho y alto son válidos. En esta función se dibuja punto por punto el corazón, se definieron dos variables (centro_x = 64) y (centro_y = 37), esto para saber dónde está el punto de centro en la pantalla OLED y asi mismo, sería el punto de origen en el plano cartesiano.

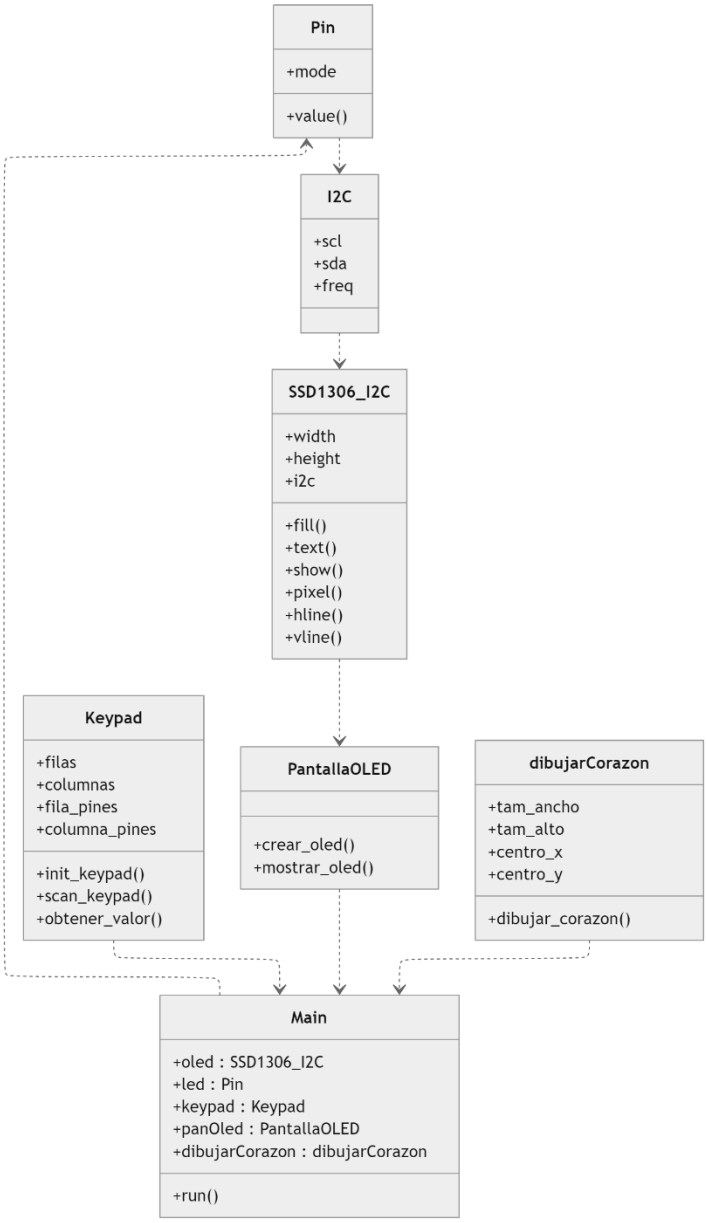
Definido lo anterior, se pasa a dibujar punto por punto el corazón para finalizar dibujando las líneas correspondientes al plano cartesiano tanto horizontal como verticalmente. Se recuerda que se está dibujando son los bordes del corazón como si fuera en el plano cartesiano en papel y lápiz. Con respecto a la curvatura, es importante señalar cual es el exponente que modifica y que se encarga de realizar dicha tarea:

```
pos_y = int(abs((pow((pos_x - centro_x) / tam_ancho, 0.5) + pow(-pow((pos_x - centro_x) / tam_ancho, 2) + 1, 1 / 2)) * -1 * tam_alto + centro_y))
oled.pixel(pos_x, pos_y, 1)
oled.pixel(pos_x - (pos_x - centro_x) * 2, pos_y, 1)

pos_y = int(abs((pow((pos_x - centro_x) / tam_ancho, 0.5) - pow(-pow((pos_x - centro_x) / tam_ancho, 2) + 1, 1 / 2)) * -1 * tam_alto + centro_y))
oled.pixel(pos_x, pos_y, 1)
oled.pixel(pos_x - (pos_x - centro_x) * 2, pos_y, 1)
oled.show()
```

Antes de hacer las pruebas en clase, dicho exponente era 2/3, esta potencia determina la forma del corazón, o su curvatura característica en la parte superior, por lo tanto, si se deseaba estudiar su comportamiento, se tenia que empezar a variar este exponente, por lo tanto, si el valor se sube a 0.8 o a 1, se tiene una curvatura más ancha y pronunciada, mientras que si se mantiene entre valores de 0.45 a 0.67, se mantiene la forma original. El punto medio para mejorar dicha curvatura fue 0.5, de igual manera, al final del informe se muestra cuales fueron los resultados obtenidos al variar dicha potencia.

2.3. Diagrama UML



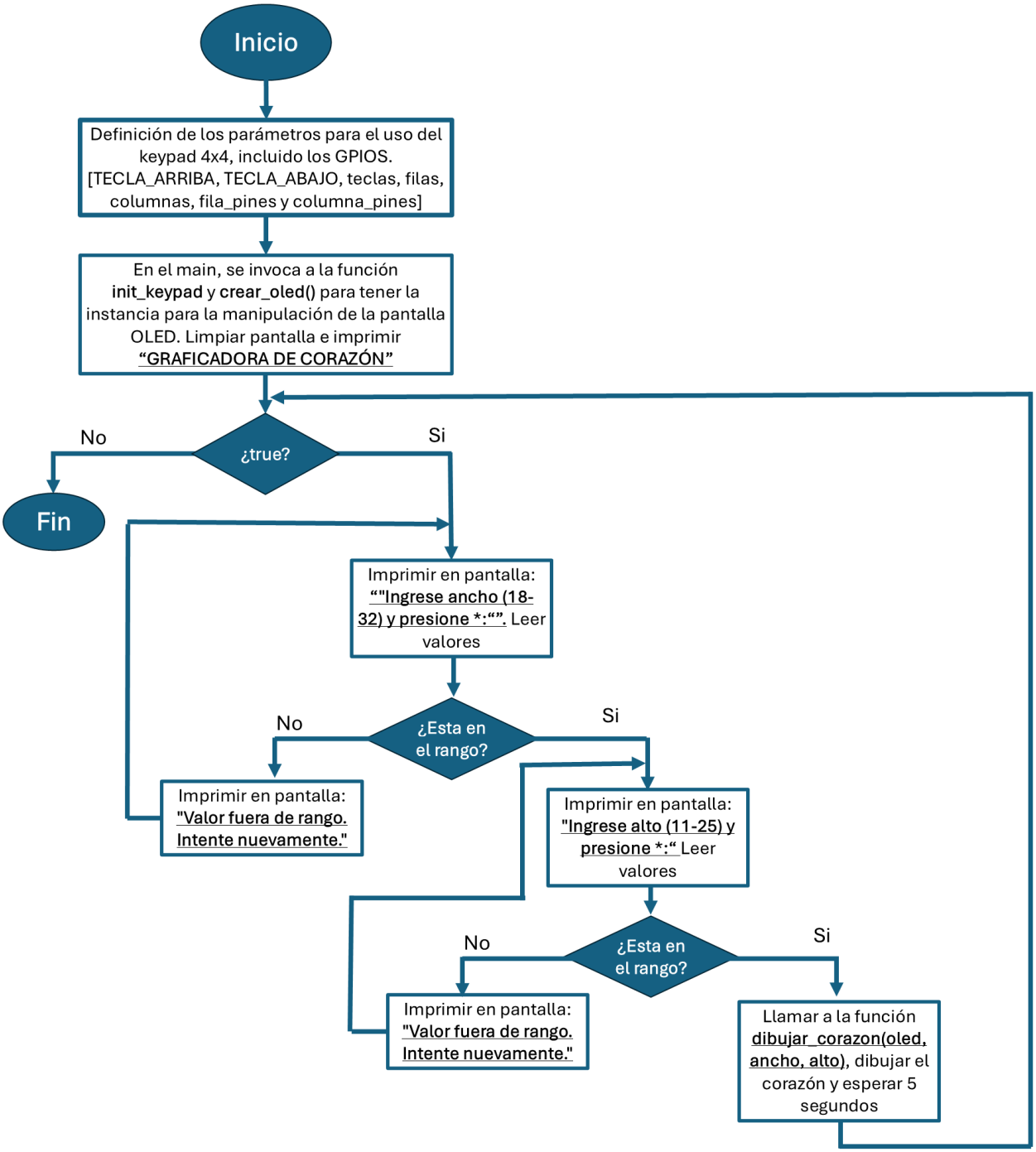
El siguiente diagrama representa como se relacionan cada una de estas clases, asi mismo, se puede observar que en la clase SSD1306_I2C (que es la encargada de la manipulación de la pantalla OLED) cuenta con distintos métodos que ya se han usado en prácticas anteriores, como es el fill (para limpiar pantalla), text (para escribir un texto), show (para mostrar algo en pantalla, ya sea limpiarla o una cadena), pixel (para dibujar un punto indicando sus coordenadas), y por último, hline y vline (para dibujar líneas horizontal y verticalmente).

Asi mismo, se señala que dibujarCorazon contiene cuatro atributos que hacen referencia a las dimensiones con que se dibujará el corazón y las coordenadas del centro (punto de origen) en la pantalla OLED.

La clase keypad, compuesta de los atributos filas, columnas, fila_pines, columna_pines, está compuesta por los métodos init_keypad(), scan_keypad() y obtener_valor(), este último para que el usuario introduzca los valores a través del keypad.

Por último, se tiene la clase main donde se hace llamado a lo anteriormente mencionado para asi poder realizar la manipulación y concluir el objetivo de graficar el corazón de acuerdo con los requerimientos del usuario.

2.4. Diagrama de flujo.



3. ANEXOS.

3.1. Código.

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import utime
import math
from time import sleep

# --- definicion de parametros para el uso del keypad matricial --- #
TECLA_ARRIBA  = 0
TECLA_ABAJO = 1

teclas = [['1', '2', '3', 'A'], ['4', '5', '6', 'B'], ['7', '8', '9', 'C'], ['*', '0', '#', 'D']]

filas = [2,3,4,5]
columnas = [6,7,8,9]

fila_pines = [Pin(nombre_pin, mode=Pin.OUT) for nombre_pin in filas]
columna_pines = [Pin(nombre_pin, mode=Pin.IN, pull=Pin.PULL_DOWN) for nombre_pin in columnas]

# --- inicializacion del keypad --- #
def init_keypad():
  for fila in fila_pines:
    fila.low()

# --- funcion para determinar si la tecla esta siendo presionada --- #
# --- esta se invoca en la funcion obtener_valor --- #
def scan_keypad():
  for fila in range(4):
    fila_pines[fila].high()
    for columna in range(4):
      if columna_pines[columna].value() == TECLA_ABAJO:
        utime.sleep(0.3)
        fila_pines[fila].low()
        return teclas[fila][columna]
    fila_pines[fila].low()
  return None

# --- funcion para determinar ancho y alto ingresado por keypad --- #
# --- se utiliza la tecla asterisco del keypad para finalizar la entrada --- #
def obtener_valor(mensaje, min_val, max_val, oled):
  valor = ""
  print(mensaje)
  while True:
    tecla = scan_keypad()
    if tecla:
      if tecla == '*':
        if valor and min_val <= int(valor) <= max_val:
          return int(valor)
        else:
          mensaje = "Valor fuera de rango. Intente nuevamente."
          print(mensaje)
          mostrar_oled(oled, mensaje, 3)
          valor = ""
      elif tecla.isdigit():
        valor += tecla
        cadena = f"Ingresado: {valor}"
        print(cadena)
        mostrar_oled(oled, cadena, 0.5)

# --- creacion de la instancia oled --- #
def crear_oled():
  i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=400000)
  return SSD1306_I2C(128, 64, i2c)

# -- funcion para mostrar mensajes en la pantalla OLED -- #
def mostrar_oled(oled, message, n):
  oled.fill(0)
  ancho_caracter = 7 # <-- tamaño en pixeles de un caracter
  max_columna = 120   # <-- long max de la pantalla
  fila = 0          # <-- primera linea
  columna = 0     # <-- primera columna
```

```
    palabras = message.split() # <-- string a lista
    for palabra in palabras:
        ancho_palabra = len(palabra) *
ancho_caracter
        if columna + ancho_palabra >
max_columna:
            fila += 16
            columna = 0
        if fila >= 50:
            oled.show()
            sleep(n)
            oled.fill(0)
            fila = 0
            columna = 0
        oled.text(palabra, columna, fila)
        columna = columna + 7 # --> espacio
entre palabras porsia
        columna += ancho_palabra +
ancho_caracter
    oled.show()
    sleep(n)

# --- funcion para dibujar corazon, param:
oled, ancho y alto --- #
def dibujar_corazon(oled, tam_ancho,
tam_alto):
    centro_x = 64
    centro_y = 37
    oled.fill(0)

    # dibujar cora punto x punto
    for pos_x in range(128):
        if pos_x < centro_x:
            continue
        if pos_x - centro_x - 1 > tam_ancho:
            continue

        pos_y = int(abs((pow((pos_x - centro_x) /
tam_ancho, 2 / 3) + pow(-pow((pos_x -
centro_x) / tam_ancho, 2) + 1, 1 / 2)) * -1 *
tam_alto + centro_y))
        oled.pixel(pos_x, pos_y, 1)
        oled.pixel(pos_x - (pos_x - centro_x) * 2,
pos_y, 1)
        pos_y = int(abs((pow((pos_x - centro_x) /
tam_ancho, 2 / 3) - pow(-pow((pos_x -
centro_x) / tam_ancho, 2) + 1, 1 / 2)) * -1 *
tam_alto + centro_y))
        oled.pixel(pos_x, pos_y, 1)
        oled.pixel(pos_x - (pos_x - centro_x) * 2,
pos_y, 1)
        oled.show()

    # plano cartesiano
    oled.hline(0, 37, 128, 1)
    oled.vline(64, 0, 64, 1)
    oled.show()


if __name__ == "__main__":
    init_keypad()
    led = Pin(14, Pin.OUT)
    led.value(0)

    oled = crear_oled()
    oled.fill(0)
    oled.show()
    mostrar_oled(oled, "GRAFICADORA DE
CORAZON", 3)

    while True:
        # llamar para mostrar pantalla oled la
cadena
        led.value(1)
        mostrar_oled(oled, "Ingrese ancho (18-30)
y presione * para confirmar:", 2)
        led.value(0)
        # invocar funcion para solicitar
parametros, (cadena, valor min, valor max,
instancia oled)


        ancho = obtener_valor("Ingrese ancho (18-
30) y presione * para confirmar:", 18, 30, oled)

        led.value(1)
        # llamar para mostrar pantalla oled la
cadena
        mostrar_oled(oled, "Ingrese alto (11-23) y
presione * para confirmar:", 2)
```

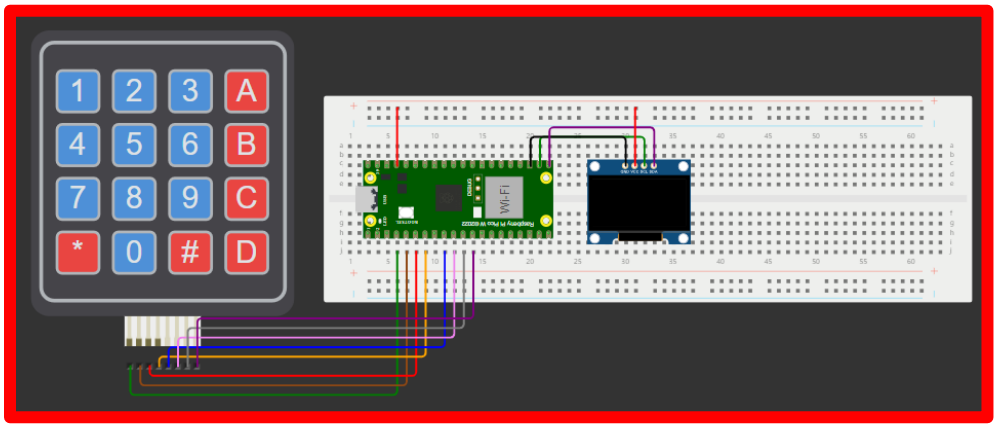
```
    led.value(0)

    # invocar funcion para solicitar parametros, (cadena, valor min, valor max, instancia oled)
    alto = obtener_valor("Ingrese alto (11-23) y presione * para confirmar:", 11, 23, oled)

    # dibujar con los datos capturados
    led.value(1)
    dibujar_corazon(oled, ancho, alto)

    # tiempo de espera para no borrar tan rapido el cora
    sleep(5)
    led.value(0)
    mostrar_oled(oled, "Reiniciando para graficar con otros datos...", 3)
```

3.2. Diagrama circuital



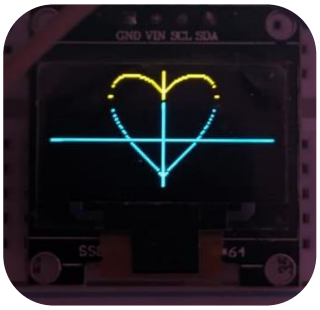
3.3. Diccionario de comandos utilizados:

Comando	Descripción
oled.fill(0)	Para limpiar pantalla. Posteriormente a esta línea, se tiene que hacer llamado de oled.show()
oled.text(str)	Para escribir un texto en pantalla. Al igual que pasa con oled.fill(0), se tiene que hacer llamado de oled.show()
oled.show()	Para poder mostrar algo en pantalla, ya sea limpiar, una cadena, una línea, etc.
oled.vline(int, int, int, int)	Para dibujar una línea vertical. Sus parámetros indican lo siguiente: columna donde se va a dibujar, fila a donde se va a dibujar, largo de la línea, color.
oled.hline(int, int, int, int)	Para dibujar una línea horizontal. Sus parámetros indican lo siguiente: columna donde se va a dibujar, fila a donde se va a dibujar, largo de la línea, color.
oled.pixel(int, int, int)	Para dibujar en un píxel especifico de la pantalla. Sus parámetros indican lo siguiente: fila, columna y color.

3.4. Pruebas empíricas

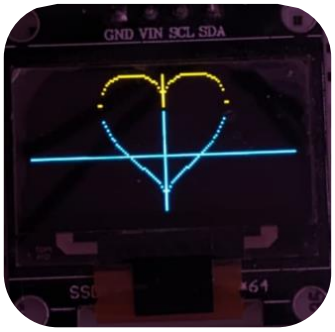
3.4.1. Ancho = 30, alto = 23, exponente = 1

3.4.2. Ancho = 30, alto = 23, exponente = 2/3 (0.67)



3.4.3. Ancho = 30, alto = 23, exponente = ½ (0.5)

3.4.4. Ancho = 30, alto = 23, exponente = ½ (0.4)



3.4.5. Ancho = 30, alto = 23, exponente = ½ (0.35)