Taller de Programación y Robótica en CMM BML **2023T2 – CL23 – 22y29 Noviembre 2023**

[R-HW] Usar **Displays #2 OLED I2C 128x64** – 120’

1. Librería: Pros y Contras
2. Montaje: Pico ((W) + display SSD1306
3. Pantalla disposición de pixeles
4. Comandos de la librería + programa de muestra con menu
5. HW Tests x6 : básicos y avanzados
6. Avanzado : Conversión de imágenes – 20’
7. Avanzado: “Tripas” de la librería – OOP /POO Herencia y sobrecarga -20’

[R] Potenciómetro, ver lectura grafica en OLED – 45’

1. Idea y montaje
2. Un ejemplo que funciona, pero con codigo mal escrito
3. SOLUCION: seudocódigo y explicación

Voluntario : J.C. Santamaria

Clase 23.1.0 – [R-HW] Displays Presentación de la Serie

Presentacion Serie Displays

Los displays son la forma más elaborada de mostrar información en robótica. Mas simple y menos versátil que los displays tenemos: LED 1 / 2 / 3 colores, neopixels, 10 bar leds, 7 Segments Display, 8x8 LED Matrix, 4 Digit LED, .. ver tutoriales de MicroPython for Kids para estos 4 últimos tipos (que no hemos visto y solo veremos si queréis).

En esta serie vamos a ver

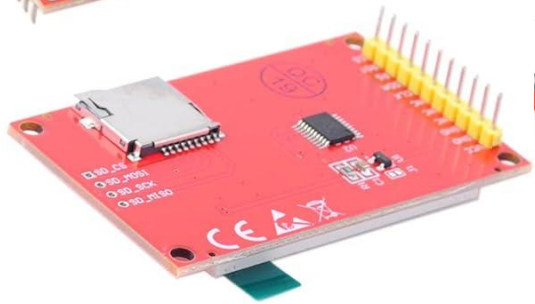
#1 - LCD 16x2 o 20x4 i2C : solo caracteres

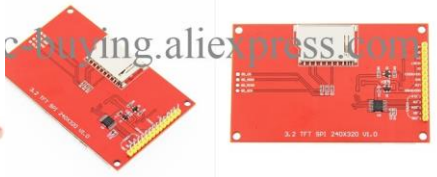
#2 - SSD1306 I2C - Graphic : 0.96” 128 x64 1 color

#3 - TFT ST7735r SPI Graph: 1.8” 128x160 RGB 64K

#4- TFT SPI- ST7789VW Graph 1.14” 240 × 135, RGB 64K

#5 – TFT SPI ILI9341 Graph 2.8”/3.2· 320 x 240, RGB 64K + touch screen XPT2046 + SD card





Clase 23.1.1–[R-HW] Display OLED I2C –Librería, Pros&Cons

Ventajas

- Gráfico y buen contrates
- Barato
- Bajo consumo
- Conexión simple
- Muchos tutoriales / 1 sola librería

Contras

- Solo 128 x 64 pixeles (máximo)
- Tamaño pequeño 0,96” o 1.3”
- Letra pequeña / 1 sola fuente 8x8

Librería SSD1306 I2c

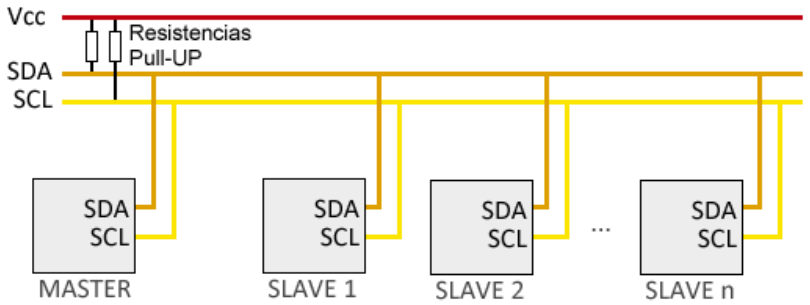
- Hay una librería standard en upython -> ver en Thony
- Usa una copia “pizarra” (por usar librería framebuffer)
- Hay varios tamaños con diferentes resoluciones h x v -> hay que inicializar con la resolución
  - **128 hor x 64 vert** -> 0,96” o 1.3 “
  - 128 x 32 -> 0,69”
  - A veces hay un pin RESET que debe estar a+3.3

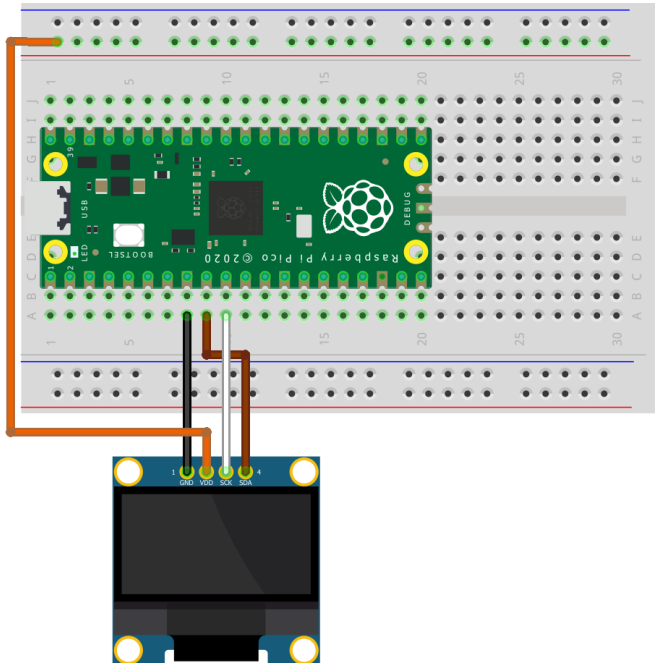
Clase 23.1.2 – [R-HW] Display OLED I2C – Montaje

I2C – Información del protocolo

El bus I2C requiere únicamente **dos cables Reloj-SCL y Datos-SDA**. Funciona en modo **maestro-esclavo sincróno**: el uC es el maestro y los sensores o display los esclavos: el dispositivo **maestro inicia la comunicación** con los esclavos, y puede mandar o recibir datos de los esclavos. Los esclavos no pueden iniciar la comunicación

Cada dispositivo conectado al bus debe tener una **dirección única(7bits),** que, junto a la **frecuencia** del bus, y los **pines SCL y SDA** es lo **único que configuramos en SW**





En las prácticas de CL23

Se puede usar la Pico o la PICO W

Pines: Se recomienda usar los I2C por defecto

I2C0 I2C(0, scl=Pin(5), sda=Pin(4), freq=400_000)

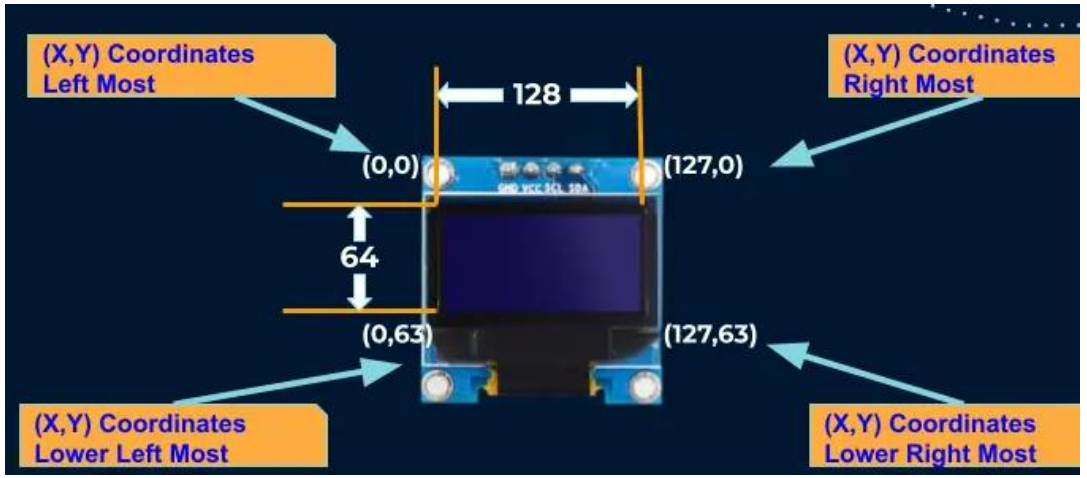
 # I2C1 I2C(1, scl=Pin(7), sda=Pin(6), freq=400_000)

Usamos el I2C1 para poder seguir usando el I2C0 con LCD, si hiciera falta

Ejemplo ‘’’i2c = machine.I2C(1, scl=machine.Pin(7), sda=machine.Pin(6), freq=400_000)’’

En ocasiones tienen un pin de RESET que normalmente debe ponerse a +3.3

Clase 23.1.3 – [R-HW] Display oled ssd1306 - Pantalla

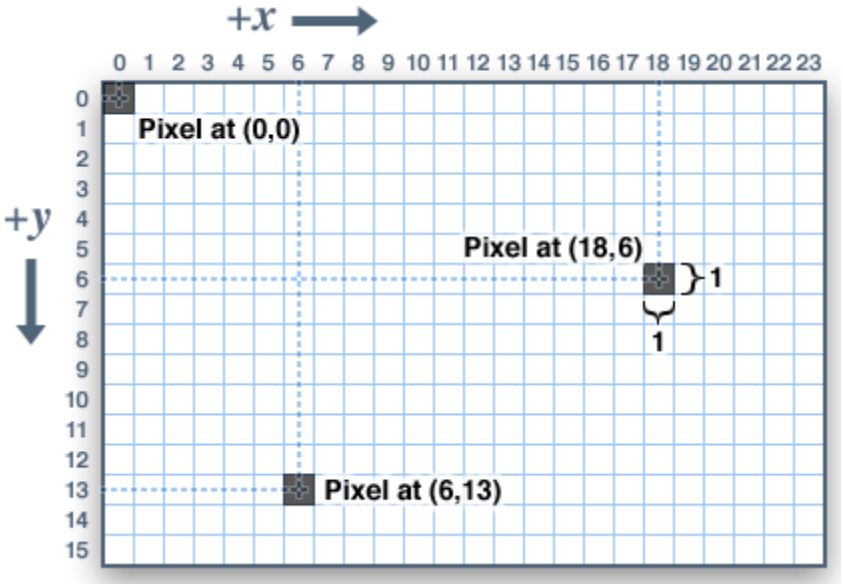


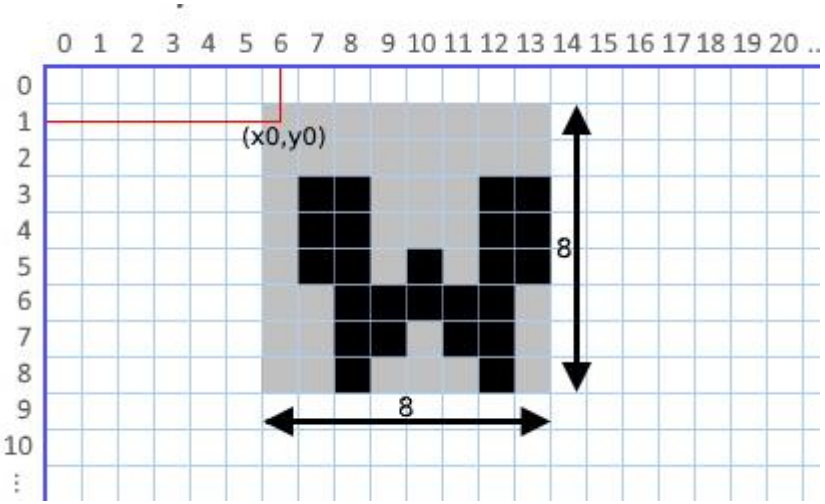
Pantalla: <u>128 horizontal x 64 vertical</u>

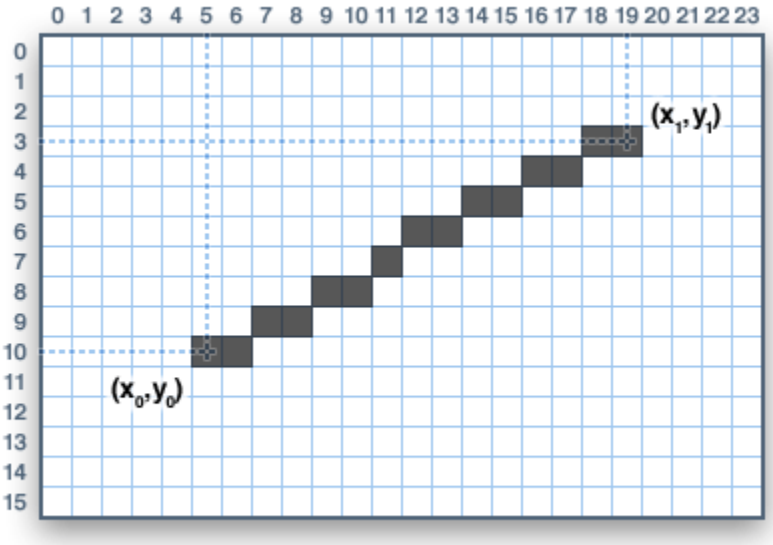
Coordenadas referidas a ESQUINA SUPERIOR IZQUIERDA

X : crece de izq -> derecha

Y : crece de Arriba -> Abajo (algo confuso cuidado)







Clase 23.1.4a – [R-HW] Display oled ssd1306 – Comandos detalle
Ref https://www.esploradores.com/oled_ssd1306/

BMMR_bhwt_ssd1306_cmd_graph_3_0.py

Programa con menú para probar comandos gráficos. Estudiar además:

- Estudiar como esta hecho el menú
- Estudiar como trata a las funciones como objetos

Método	Que hace	Parámetros / ejemplo	Herencia de
fill(state)	Rellena toda la pantalla de 1 color	State=black (0) or white(1)	FrameBuffer
rect(x0, y0, width, height, c, f)	Dibuja un rectángulo vacío	x0,y0 origen esq sup izq	FrameBuffer
fill_rect(x, y, width, height, c)	Dibuja un rectángulo lleno	Equivalente a rect(……, True)	FrameBuffer
hline(x, x, length, state)	Dibuja línea horizontal	oled.hline(0, 0, 127, 1)	FrameBuffer
vline(x,y,length, color)	Dibuja línea vertical	oled.vline(width - 1, 0, height - 1, 1)	FrameBuffer
line(x1,y1,x2,y2)	Dibuja línea con coor. origen y fin	oled.line(0,0, 127, 63, 1)	FrameBuffer
pixel(x,y, color)	Dibuja un pixel		FrameBuffer
ellipse(*x*, *y*, *xr*, *yr*, *c*[, *f*, *m*])	Dibuja un círculo o elipse	X,y centro / xr,yr radios, c color 0/1 'f' = True relleno, 'm' cuadrante	FrameBuffer
poli(*x*, *y*, *coords*, *c*[, *f*])	Avanzado	Ver en prog cmd_graph	FrameBuffer

Clase 23.1.4b – [R-HW] Display oled ssd1306 – Comandos detalle Ref https://www.esploradores.com/oled_ssd1306/

BMMR_bhwt_ssd1306_cmd_graph_3_0.py

Método	Que hace	Parámetros / ejemplo	Herencia de
text(string, x0,y0,color)	Escribe un texto	X0,y0 esquine sup izq, color 0 /1 se puede omitir será 1 en ese caso	FrameBuffer
scroll(x,y)	Scroll el display	X = pix der o (-) izq, Y= pix arr, (-) abajo	FrameBuffer
show()	Muestra el buffer en oled	SIN parámetros	SSD1306
invert(i)	Invierte el display	i= 1 invierte, i= 0 normal	SSD1306
poweroff(), poweron()	La pantalla se puede **apagar** y **encender** utilizando *.poweron()* y *poweroff()*	SIN parámetros CUIDADO si dejas apagada, y luego nov es nada	SSD1306
contrast()	Regula el **contraste** de la pantalla	0-255, 0 apagada y 255 completamente iluminada.	SSD1306
blit(fbuf, x, y, color)	Avanzado, dibuja un sub-buffer		FrameBuffer

Clase 23.1.5– [R-HW] Display oled ssd1306 – HW Tests

BMMR_bhwt_ssd1306_all_lines_1_0.py

Test super Básico 1

Muestra las 8 líneas posibles en el display

Fuente por defecto de 8x8 es de framebuffer

BMMR_bhwt_ssd1306_blit_1_0.py

(Avz) Test Blit

Muestra cómo usar un sub-buffer, para dibujar un icono o similar

BMMR_bhwt_ssd1306_img_en_lista_3_0.py

(Avz) Test de Imagen logo

Muestra el logo de Raspberry como imagen guardada en una lista de bits. Se ha usado un conversor de ficheros de imagen. Usa blit

BMMR_bhwt_ssd1306_text_logo_lin_1_5.py

Test Básico 2

Muestra un mensaje genérico (fuente 8x8) + el logo de micropython hecho con rectángulos y líneas

BMMR_bhwt_ssd1306_icon_en_matriz_1_0.py

Test Matriz

Muestra cómo usar un sub-buffer, para dibujar un icono o similar + lo muestra repetido en una diagonal de pantalla

BMMR_bhwt_ssd1306_img_en_file_2_0.py

(Avz) Test de Imagen File

Muestra una imagen, escalada a 128x64 mono en formato PBM, leyendo un fichero

Requiere uso COMPLEJO de conversores Usa blit

Clase 23.1.6– [R-HW] Display oled ssd1306 – Avanzado Conversión de imágenes

caballo_de_web.jpg

Con Paint, recorto, giro 90ª, escalo a 128 x64 y guardo en monocolor .bmp

Cab90gmono.bmp

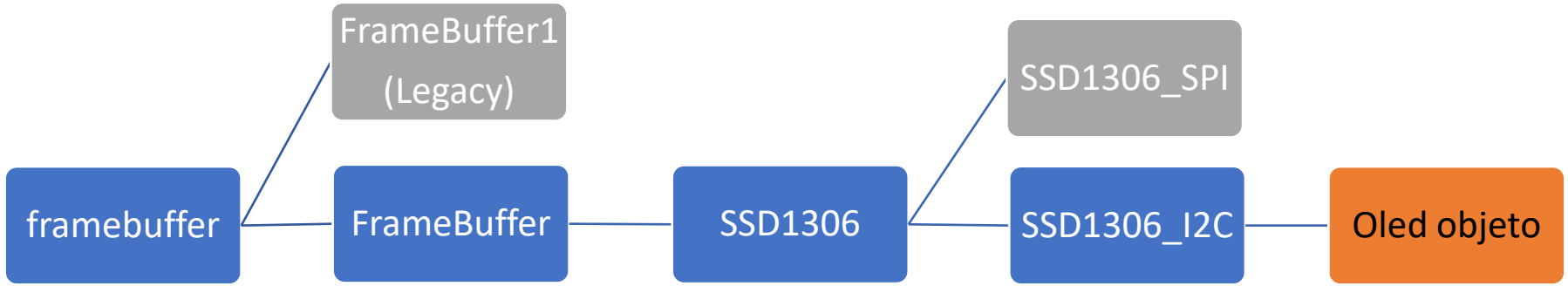
Uso https://convertio.co/es
Para convertir a pbm "raw". (NO asscii)

cab90gmono2.pbm

La conversión de imágenes para visualizar en displays, es compleja por los muchos formatos de imagen existentes, y porque se necesita conocer el HW de cómo se representan las imágenes en el display. Pero es muy satisfactorio
Ver con más detalle en
https://www.esploradores.com/oled_ssd1306/

```
#fichero PBM , 2 líneas tipo y tamaño
P4
128 64
                    8          <          ~
> à          ÿî          ÿž?€    ÿ€
```

Clase 23.1.7 – [R-HW] Display oled ssd1306 - Avanzado: Librería “tripas” – OOP/ POO Herencia y sobrecarga



Define constantes
'GS2_HMSB’,
'GS4_HMSB’,
'GS8',
'MONO_HLSB’,
'MONO_HMSB’,
'MONO_VLSB’,
'MVLSB’,
'RGB565’
+ clases
FrameBuffer y
FrameBuffer1

+ Métodos :
'blit’,
'ellipse’,
'fill’,
'fill_rect’, 'rect',
'hline’, 'vline'
'line’,
'pixel', 'poly',
'scroll’,
'text',

+ Métodos :
'init_display’,
'show’,
'poweroff’,
'poweron’,
'contrast’
+ Constantes HW ssd1306
Usa, pero no define
write_cmd
Write_data

SSD1306_I2C
añade/modifica los métodos

1. write_cmd
2. write_data

Oled **hereda** métodos y constantes de:

1. SSD1306_I2C
2. SSD1306
3. FrameBuffer
4. framebuffer

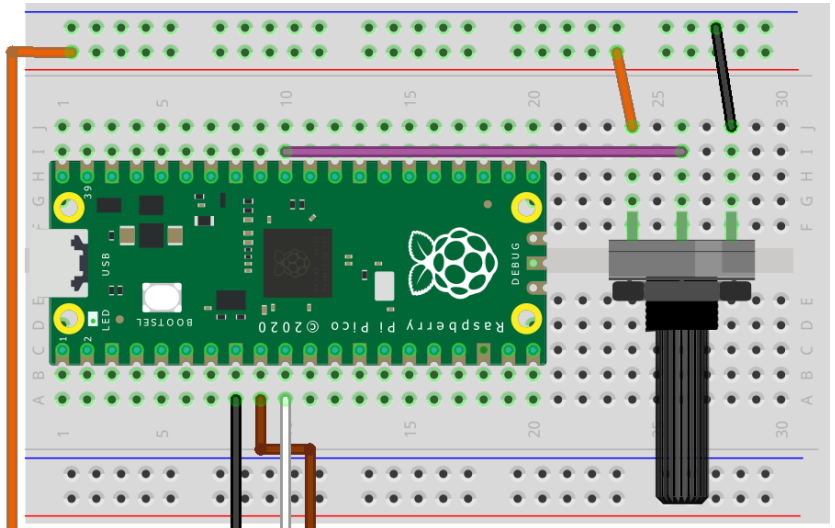
La **subrecarga de métodos** permite modificar el comportamiento de los métodos heredados de la clase padre en la clase hija. Cuando se llama al método en la clase hija, el intérprete de Python buscará la definición del método en la clase hija primero y, si no lo buscará en la clase padre. SSD1306 cambia los métodos write_cmd y write_date, para adaptarlos a comunicación I2C o SPI dependiendo

Clase 23.2.1 [R] Potenciómetro, ver lectura grafica en OLED Idea y montaje

1ra idea: hacer proyecto donde se usen intensivamente las capacidades gráficas del display.

Objetivo: **Representar gráficamente un valor analógico en unos ejes de coordenadas standard Y= valor analógico, X=tiempo**.

Para el valor analógico elegimos la lectura por uno de los ADC de la PICO, del voltaje del pin intermedio de un potenciómetro cuyos otros 2 pines estan a 0 y +3.3 voltios respectivamente, de forma que el voltaje en este pin esté en este rango que es el que puede leer el ADC directamente



Clase 23.2.1 [R] Potenciómetro, ver lectura grafica en OLED
Un ejemplo que funciona, pero con codigo mal escrito

BMMR_ssd1306_graf_pot_malcode_3_0.py

Encontramos un ejemplo que funciona, pero cuyo codigo es confuso, mal organizado y probablemente una "traducción" de C a Python, es decir poco "*pytonico*".

¿ Por qué el codigo es malo?

- Mal comentado, difícil de seguir (la versión aquí esta comentada por mi)
- Algunas constantes , no estan explicadas
- Mezcla codigo de distintos niveles de abstracción ……
- Lo peor, desde mi punto de vista, es que en la función **plot_time** hay demasiadas cosas, está casi todo el programa --> mala estructura, es innecesariamente compleja. Se supone que se puede usar en otros programas, pero está mal encapsulada

Lo bueno es que fijamos mejor el objetivo:

- Dejamos la línea 0 para mostrar en texto el voltaje => Zona Top
- Eje Y a la izquierda con los valores máximo y minimo del voltaje : anchura en horizontal 25 pix = 3 char => Zona Left
- Eje X última línea horizontal = 63
- Gráfico dibujo continuo con 2 fases
  - Bucle 1 - hasta llegar al final a la derecha del display
  - Bucle 2- Scroll horizontal SOLO de la parte grafica del valor de voltaje

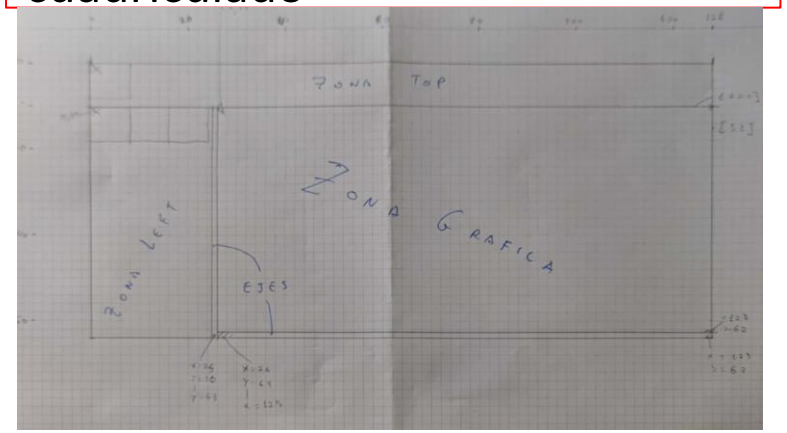
    Como no se puede hacer en esta librería un scroll parcial, haremos uno total y redibujaremos zona top y left

Clase 23.2.3 [R] Potenciómetro, ver lectura grafica en OLED - Seudocódigo

BMMR_ssd1306_graf_pot_4_1.py

SOLUCION : Con lo aprendido del codigo “mal codificado” , empiezo en base cero a codificar.

Lo más laborioso es la definición de zonas, y revisar los valores frontera de los pixeles, ver si en al ejecución no "desbordamos" zonas o el propio display, etc. Recomiendo usar papel cuadriculado




```
# 0- Constantes y variables globales + las que definen las zonas del display
# F.1 Dibuja los ejes X e Y con etiquetas F.2 Borra zona top, zona izquierda y ejes
# 1 - Creación de objetos , i2c, oled y potenciómetro sobre ADC
# 2 y 3- Limpia pantalla y Presentación inicial
# 4- Bucle inicial hasta llenar la zona grafica
#  4.1 - Dibuja los ejes y etiquetas Y  + inicializa variable dibujo del grafico x e y
#4.3 Bucle inicial propiamente: for de anchura eje X, en pixeles
    # 4.4-5 Leer voltios y escribe en zona top
    # 4.6 Mapea y dibuja valor voltios en zona gráfica, como línea desde anterior
    # 4.7-8 Show + swap valor anterior por actual y espera
# 5. Bucle 2do con scroll : while True
    # 5.1-2 borramos zona top, zona izquierda y ejes  scroll1 en horizontal
    # 5.3-4 leo voltios, escribo en zona top+ Dibuja los ejes y etiquetas de eje Y
    # 5.5 Mapea y dibuja valor voltios
    # 5.6-7 Show + swap valor anterior por actual y espera
```

Archivo de transparencias relevantes

Taller personalizado de Programación y Robótica en CMM BML 2023 -2024

[Redes] Socket Bind : un ordenador en modo servidor puede tener 2 direcciones de red

```
C:\Users\josec>ipconfig

Configuración IP de Windows


Adaptador de Ethernet Ethernet 4:

   Sufijo DNS específico para la conexión. . :
   Vínculo: dirección IPv6 local. . . : fe80::7631:d510:cd2c:cbf3%11
   Dirección IPv4. . . . . . . . . . . . . . : 192.168.1.55
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . : 192.168.1.1

Adaptador de LAN inalámbrica Conexión de área local* 1:

   Estado de los medios. . . . . . . . . . . : medios desconectados
   Sufijo DNS específico para la conexión. . :

Adaptador de LAN inalámbrica Conexión de área local* 2:

   Estado de los medios. . . . . . . . . . . : medios desconectados
   Sufijo DNS específico para la conexión. . :

Adaptador de LAN inalámbrica Wi-Fi:

   Sufijo DNS específico para la conexión. . :
   Vínculo: dirección IPv6 local. . . : fe80::aa1f:6c6d:27f6:7f29%5
   Dirección IPv4. . . . . . . . . . . . . . : 192.168.1.58
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . : 192.168.1.1

Adaptador de Ethernet Conexión de red Bluetooth:

   Estado de los medios. . . . . . . . . . . : medios desconectados
   Sufijo DNS específico para la conexión. . :

C:\Users\josec>
```

Un PC con cable Ethernet y wifi ambos activados tiene 2 direcciones de red, en el ejemplo tiene

192.168.1.55 -> Ethernet 4

192.168.1.58 -> Wifi

Para escuchar por todas, se configura bind como

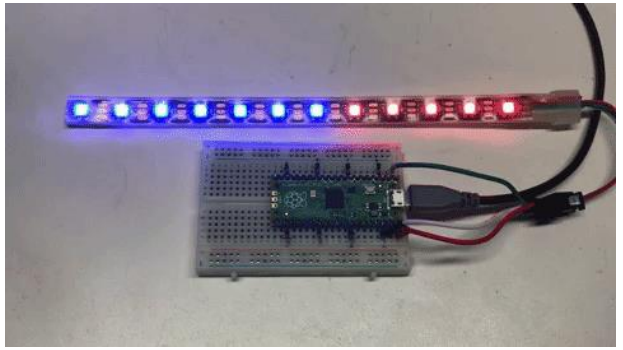
socket.bind((0.0.0.0), port) or

socket.bind(‘’, port)

usa la dirección especial INADDR_ANY para escuchar en todas las direcciones al mismo tiempo

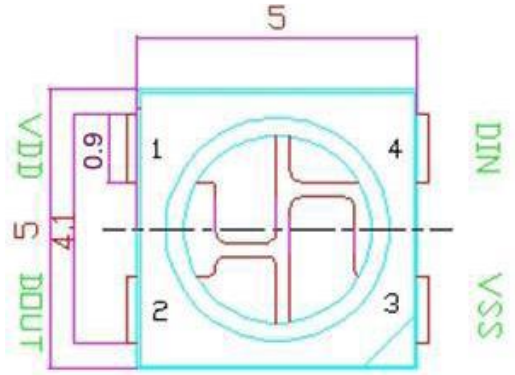
[Robotica] **Neopixel** ¿Qué son y para que se usan? Seguiremos fuente # **3 - MicroPython for Kids**

https://www.coderdojotc.org/micropython/basics/05-neopixel/



Los **neopixeles** son

- Dispositivos de Salida,
- Tipo led que combinan los 3 colores básicos Red Green Blue, con intensidades programables en cada color = 8bits x3 => 16 millones de colores
- Se pueden conectar “encadenados” y se puede programar el color de cada pieza de la cadena
- Solo requieren 1 hilo para su control + 2 para alimentación
  - El hilo de control “entra” y “sale” de cada neopixel
- El mas común es el compuesto por unidades de **WS2812B**



NO.	Symbol	Function description
1	VDD	Power supply LED
2	DOUT	Control data signal output
3	VSS	Ground
4	DIN	Control data signal input

- Usaremos la librería integrada en MicroPython: desde 1.18 hay soporte integrado para NeoPixels para el microcontrolador Raspberry Pi Pico y Pico W (RP2040)
- Alimentación de módulos a 5volt con uControlador a 3.3volt -> Ver Uso Basico
- Cómo mezclar lógicas de 5volt y de 3.3 volt

[Orden] #1 Que hemos hecho hasta ahora y que sugiero hacer en los próximos meses

Clases	Fechas	Super Resumen
0,1,2,3	Feb-M	Primeros pasos , 1ros programas
4	Marzo	Entrega **PicoW**, **Led**
5,6 y 7	Ma-Ab	**Servos**, POO, **hacer librería Servo**
8,9	Abril	**Pulsadores**, rebotes, **interrupciones** Prototipado progresivo : cifra cesar Buscar librerías
10-11 12	Mayo	3 en ralla – bloques juego, estrategias **Neopixel**, volts-niveles lógicos / Trama Bucles **for** / **Funciones / Ficheros->Log** Entradas analógicas **ADC, Sensor Temp**
13+14	Mayo Junio	**Multímetro**: uso básico **Diccionarios , List comprehension** Sonido **Altavoz** con **transistor NPN**
15+16	Junio	**Proyecto Completo Termómetro (NTC -> DHT) con Display LCD I2C**

Objetivos sugeridos para 4T2023- 2024

1. 'Internet no es difícil' – haremos IoT
   - Vamos a usar las capacidades Wifi del PICO W para "cacharear" con Internet de las cosas (IoT)
   - En Python del PC haremos programas que lean API´s de internet
2. **Seguiremos profundizando en conceptos de Python para poder hacer mejores programas, tanto en PC como en PICOW**
   - Ejemplo: profundizar en diccionarios y datos JSON (que se usan intensivamente en API´s)
3. Opcional – Avanzar aún más en Python
   - Hacer juegos con interfaz gráfica **PyGame**, con MAKING GAMES WITH PYTHON & PYGAME
   - Hacer **programas prácticos** con AUTOMATE THE BORING STUFF WITH PYTHON

[Orden y Para casa] #2 Internet para makers

Ejemplo **Hecho** / **Pendiente**

¿Qué es internet? Selección Tutos

How does the Internet work? 1

How does the Internet work? 2

Guía Buena algo Antigua(ingles)
mucho detalle

Video

How the internet Works in 5 minutes: A 5-minute by Aaron Titus

¿Cómo funciona Internet? 10 minutos en castellano

Tipos de proyectos maker – IoT y ejemplos

1. Leer/acceder a recursos a través de API´s
   - Ej. Cheerligth
   - Ej Datos meteo Openweather / AEMET
2. Publicar datos de sensores propios
   1. POST : Thinkspeak
   2. MQTT: Thinkspeak
   3. En modo AP : ej valor temperatura PICO W
3. Avisos de sensores propios a email, Telegram o WA : ej. IFTTT
4. Controlar HW propio
   1. Desde Dentro de TU red local
      - Web server LEDx3 + NTC / Control de relé
   2. Desde Fuera de la red local (es más complejo)
      1. API´s o MQTT:
      2. APP Movil : ej Anvil
      3. Firebase (google) …..(abrir puertos en router NO RECOMENDABLE)