# PROGRAMANDO LA MATRIZ DE LEDS RGB

https://thepihut.com/blogs/raspberry-pi-tutorials/coding-the-waveshare-rp2040-matrix

## Pines de LED (¡y una advertencia!)

Nuestra página de producto para esta placa proporciona enlaces a la wiki de Waveshare (https://www.waveshare.com/wiki/RP2040-Matrix) y una guía de configuración de pines (https://www.waveshare.com/wiki/RP2040-Matrix#Pinout). Esta wiki nos dice que los LED RGB direccionables están conectados a GPIO 16. Hay 25 LED, numerados del 0 al 24 y cada uno se puede configurar en una amplia variedad de colores proporcionando valores de rojo, verde y azul (RGB) en el rango 0. a 255.

Hemos descubierto que los valores en el rango de 0 a 20 son suficientes para la mayoría de los propósitos. Waveshare advierte contra el uso prolongado de valores elevados que podrían dañar el dispositivo al sobrecalentarse. ***¡Estás advertido!***

## Librería de Códigos

Afortunadamente, podemos usar la biblioteca Neopixel que ahora está integrada en MicroPython.

Esto hace que sea muy fácil de poner en marcha. El siguiente código importa la biblioteca e inicializa los LED:

```
# Waveshare RP2040-MATRIX with 5x5 grid of Neopixels
# NB colours (green, red, blue)
# Tony Goodhew for the pihut.com - 25 Aug 2023
import time
from machine import Pin
from neopixel import NeoPixel

# Set up 25 Neopixels on GPIO 16
np = NeoPixel(Pin(16), 25, bpp = 3) # bpp = bytes per pixel (GRB)
```

Estos LED son ligeramente diferentes de los Adafruit Neopixels y el orden de los colores es verde, rojo y azul (**GRB** en lugar de **RGB**). No hace mucha diferencia, sólo necesitamos agregar nuestros valores de color en los lugares correctos.

Podemos establecer el color de un píxel agregando la siguiente línea:

```
np[2] = (0, 10, 10) # Set the third pixel to MAGENTA
```

Rojo y azul (los valores '10') mezclados forman un color púrpura, llamado magenta, y los LED se cuentan de cero a 24.

Esto solo almacena los valores en un búfer: la instrucción np.write() envía el contenido del búfer a los LED que actualizan los colores. ¡Esto es muy rápido!

Intente ejecutar el código siguiente con las líneas agregadas. Deberías encontrar que el píxel central en la línea superior se ilumina en color violeta:

```
# Waveshare RP2040-MATRIX with 5x5 grid of Neopixels
# NB colours (green, red, blue)
# Tony Goodhew for the pihut.com - 25 Aug 2023
import time
from machine import Pin
from neopixel import NeoPixel

# Set up 25 Neopixels on GPIO 16
np = NeoPixel(Pin(16), 25, bpp = 3) # bpp = bytes per pixel (GRB)

np[2] = (0, 10, 10) # Set the third pixel to MAGENTA

np.write()
```

Ahora prueba el siguiente código y mira qué sucede. Tenga en cuenta los corchetes adicionales en las instrucciones de llenado. El color se define con una tupla y los corchetes son imprescindibles:

```
# Waveshare RP2040-MATRIX with 5x5 grid of Neopixels
# NB colours (green, red, blue)
# Tony Goodhew for the pihut.com - 25 Aug 2023
import time
from machine import Pin
from neopixel import NeoPixel

# Set up 25 Neopixels on GPIO 16
np = NeoPixel(Pin(16), 25, bpp = 3) # bpp = bytes per pixel (GRB)

time.sleep(2)

np.fill((0,10,0)) # Fill display with RED
np.write()
time.sleep(2)

np.fill((0,0,0)) # Fill display with BLACK
np.write()
```

Algunas cosas para probar:

- Establezca el píxel[0] verde, el píxel[21] azul y el píxel[24] amarillo
- Escriba un procedimiento para borrar la pantalla y probarla.
- Llena la pantalla con píxeles de colores aleatorios en orden aleatorio

# Usando coordenadas: valores (x, y) o (col, row)

¡Es hora de dar un paso más! Ahora usaremos coordenadas para controlar los LED, pero antes de explicar algo, copie el ejemplo de código a continuación y pruébelo:

```
# Waveshare RP2040-MATRIX with 5x5 grid of Neopixels
# NB colours (green, red, blue)
# Tony Goodhew for the pihut.com - 25 Aug 2023
import time
from machine import Pin
from neopixel import NeoPixel

# Set up 25 Neopixels on GPIO 16
np = NeoPixel(Pin(16), 25, bpp = 3) # bpp = bytes per pixel (GRB)

def clear():
    np.fill((0,0,0))
    np.write()


def wait(t):            # Delay t seconds
    time.sleep(t)


for i in range(25):  # 'Natural' order 0-24, as connected on board
    np[i] = (10,10,0)   # Yellow ON
    np.write()
    wait(0.15)
    np[i] = (0,0,0)     # Black OFF - Single blink
    wait(0.2)

clear()
wait(0.6)


for row in range(5): # Row at a time
    for col in range(5):
        np[5 * row + col] = (5,5,5) # White
        np.write()
        wait(0.15)
    wait(0.1)
clear()


for col in range(5): # Column at a time
    for row in range(5):
        np[5 * row + col] = (0,0,5) # Blue
        np.write()
        wait(0.15)
    wait(0.2)
clear()


def sq(x,y,n,r,g,b):
    for col in range(x,x+n):  # Top and Bottom
        np[5 * y + col] = (g,r,b)
        np[5 * (y+n-1) + col] = (g,r,b)

    for row in range(y,y+n):  # Sides
        np[5 * row + x] = (g,r,b)
        np[5 * row + x + n -1] = (g,r,b)
    np.write()
```

```
tt = 0.6
sq(0,0,5,4,0,0)
wait(tt)
sq(0,0,4,0,4,0)
wait(tt)
sq(0,0,3,0,0,4)
wait(tt)
sq (0,0,2,4,4,0)
wait(tt)
sq(0,0,1,0,4,4)
wait(tt)
clear()
```

Entonces ¿Qué es lo que hace? Revise el código junto con los puntos siguientes para comprender qué está haciendo y dónde:

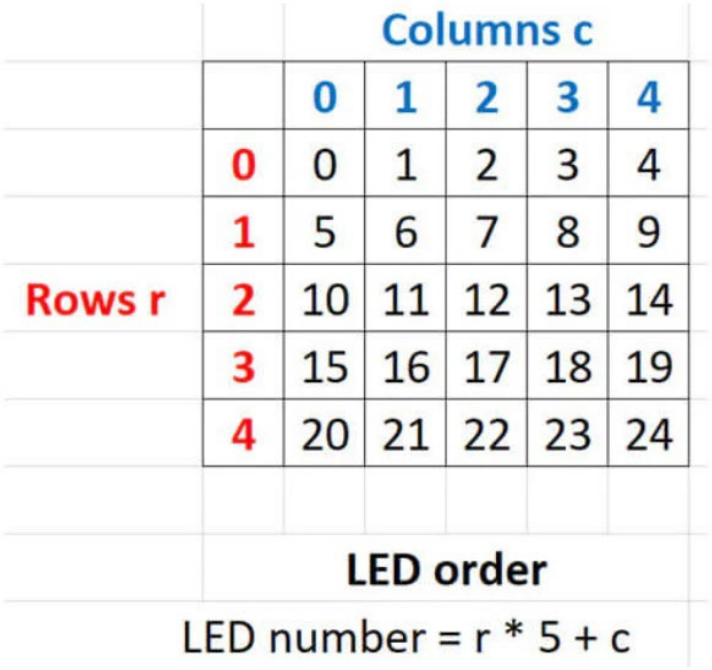
- Configura los LED RGB.
- Define procedimientos para borrar la pantalla y esperar un breve período.
- Parpadea cada uno de los LED por turno con amarillo en el "orden natural" 0 - 24
- Llena lentamente la pantalla fila por fila con blanco
- Llena lentamente la pantalla columna por columna en azul
- Define un procedimiento para dibujar cuadrados y luego dibuja una serie de cuadrados.

## Ejemplo de uso de coordenadas

Veamos un ejemplo sencillo:

```
col = 3
row = 1
np[5 * row + col] = (0,0,5) # Blue
```

Estas instrucciones harán que el LED 8 se vuelva azul, porque estamos haciendo referencia al LED en la columna 3, fila 1, como puede ver en la siguiente cuadrícula:



# Caracteres y Texto

Explicar cómo funcionan los caracteres y el desplazamiento probablemente merezca un artículo propio, por lo que no entraremos en detalles de "cómo" hoy, pero puedes descargar nuestro código de ejemplo (https://cdn.shopify.com/s/files/1/0176/3274/files/Matrix_DEMO_4.py), que está completamente comentado y es fácil de usar, y probarlo. por ti mismo.

No es necesario que entiendas cómo funciona para utilizar las instrucciones que siguen: ¡simplemente trata el código como una librería!

Hay dos instrucciones:

- `display(n,rr,gg,bb)  # (valor ASCII, rojo, verde, azul) = Carácter estacionario único`
- `scroll(s,rr,gg,bb)  # (cadena de mensaje, rojo, verde, azul) = mensaje desplazado`

Mira el vídeo para verlo en acción ( https://www.youtube.com/watch?v=GcY-zCfTieg )

### Cosas para Probar

- Desplaza tu nombre por la pantalla
- Utilice un potenciómetro de 10k como dispositivo de entrada (rango de 0 a 9) y muestre el valor en los LED
- Dibuja una forma de corazón rojo sobre un fondo azul.
- Utilice 3 potenciómetros para controlar los valores rojo, verde y azul de los píxeles en tres esquinas de la pantalla y su color mezclado en un cuadrado en el centro.

# Juego de Bonificación: Zap it

¡Divirtámonos un poco! Este es un juego de coordinación de manos y ojos y solo necesita un simple botón/interruptor (tipo momentáneo) que conecte GPIO 14 a GND.

### Cómo jugar

- Un objetivo blanco recorre el borde de la pantalla.
- Un jugador tiene tres intentos en un juego.
- El jugador comienza con 100 puntos e intenta atacar al objetivo mientras está a la derecha del marcador de objetivo rojo presionando el botón
- Si el jugador presiona el botón demasiado pronto, demasiado tarde o falla una rotación más allá de la posición objetivo, se pierde un punto.
- Se registra un impacto y el objetivo se vuelve amarillo y verde, contador de marcha, el píxel se ilumina
- Un punto perdido por un golpe fallido se indica mediante un breve destello azul.
- La puntuación se muestra al final del juego.

¿Puedes conseguir 100?

## Código Zap it

Aquí está el código del juego. Como de costumbre, cópielo en Thonny (use el botón copiar a la derecha, ¡es más fácil!) y ejecútelo como cualquier otro código:

```
# WS RP2040 'Zap it' Game
# Needs a button switch between GPIO 14 and GND
# Tony Goodhew 28th Aug 2023 (Author)
# How to play:
#   A white target runs round the edge of the display.
#   The player starts with 100 points and tries zap the target while it is
#   to the right of the red target marker.
#   If the player hits the button too early, too late or misses a rotation
#   past the target position a point is lost.
#   A hit is registered with the target turning yellow and a green pixel is lit.
#   A point lost is indicated by a short blue flash.
#   The score is shown at the end of the go.

import time
from machine import Pin
from neopixel import NeoPixel

# Set up 25 Neopixels on GPIO 16
np = NeoPixel(Pin(16), 25, bpp = 3) # bpp = bytes per pixel (GRB)

def clear():
    np.fill((0,0,0))
    np.write()

def wait(t):            # Delay t seconds
    time.sleep(t)

# ========== START OF FONT LIBRARY ==========
# Instructions:
# display(n,rr,gg,bb) # (ASCII value, red, green, blue)  = stationary character
# scroll(s,rr,gg,bb)  # (string, red, green, blue) = Scrolled message

# Character codes ASCII 32 - 127 5x5 pixels
# 5x5 font - Tony Goodhew 26th August 2023 (author)
# 5 columns of 5 bits = LSBs [@ == 00000]
q = [
    "@@@@@", #    32
    "@@]@@", #  ! 33
    "@X@X@", #  " 34
    "J_J_J", #  # 35
    "IU_UR", #  $ 36
    "QBDHQ", #  % 37
    "JUJA@", #  & 38
    "@@X@@", #  ' 39
    "@NQ@@", #  ( 40
    "@@QN@", #  ) 41
    "@J~J@", #  * 42
    "@DND@", #  + 43
    "@@BC@", #  , 44
    "@DDD@", #  - 45
    "@@A@@", #  . 46
    "ABDHP", #  / 47
    "NQQN@", #  0 48
    "@I_A@", #  1 49
```

```
    "IS_I@", #  2 50
    "UUUJ@", #  3 51
    "^BGB@", #  4 52
    "]UUR@", #  5 53
    "NUUF@", #  6 54
    "PSTX@", #  7 55
    "JUUJ@", #  8 56
    "LUUN@", #  9 57
    "@@E@@", #  : 58
    "@@BC@", #  ; 59
    "DJQ@@", #  < 60
    "@JJJ@", #  = 61
    "@QJD@", #  > 62
    "HPUH@", #  ? 63
    "_QUU]", #  @ 64
    "OTTO@", #  A 65
    "_UUJ@", #  B 66
    "NQQQ@", #  C 67
    "_QQN@", #  D 68
    "_UUQ@", #  E 69
    "_TTP@", #  F 70
    "NQUV@", #  G 71
    "_DD_@", #  H 72
    "@@_@@", #  I 73
    "BAA^@", #  J 74
    "_DJQ@", #  K 75
    "_AAA@", #  L 76
    "_HDH_", #  M 77
    "_HD_@", #  N 78
    "NQQN@", #  O 79
    "_TTH@", #  P 80
    "NQSO@", #  Q 81
    "_TVI@", #  R 82
    "IUUR@", #  S 83
    "PP_PP", #  T 84
    "^AA^@", #  U 85
    "XFAFX", #  V 86
    "~C~C~", #  W 87
    "QJDJQ", #  X 88
    "PHGHP", #  Y 89
    "QSUY@", #  Z 90
    "@_Q@@", #  [ 91
    "PHDBA", #  \ 92
    "@Q_@@", #  ] 93
    "@HPH@", #  ^ 94
    "AAAA@", #  _ 95
    "@@X@@", #  ` 96
    "BUUO@", #  a 97
    "_EEB@", #  b 98
    "FIII@", #  c 99
    "BEE_@", #  d 100
    "NUUH@", #  e 101
    "@_TP@", #  f 102
    "HUUN@", #  g 103
    "@_DG@", #  h 104
    "@@W@@", #  i 105
    "@AAV@", #  j 106
    "_DJA@", #  k 107
    "@_A@@", #  l 108
```

```
    "OHDHO", #  m 109
    "@GDG@", #  n 110
    "FIIF@", #  o 111
    "OJJD@", #  p 112
    "DJJG@", #  q 113
    "OHHD@", #  r 114
    "IUUR@", #  s 115
    "@H_I@", #  t 116
    "NAAN@", #  u 117
    "HFAFH", #  v 118
    "LCLCL", #  w 119
    "@JDJ@", #  x 120
    "@LEO@", #  y 121
    "@KMI@", #  z 122
    "@DNQ@", #  { 123
    "@_@@@", #  | 124
    "@QND@", #  } 125
    "HPHP@", #  ~ 126
    "@HTH@", #  □ 127
    ]

# This procedure converts the font codes to a string
# of 25 ones and zeros characters - 5 cols of 5 rows
def decode(qq):
    powers = [16,8,4,2,1]
    str25 = ""
    for p in range(5):
        n = ord(qq[p])
        if n == 126:
            n = 92 # replace ~ with \
        str5 = ""
        for c in range(5):
            if ((powers[c] & n)/powers[c]) == 1:
                str5 = str5 + "1"
            else:
                str5 = str5 + "0"
        str25 = str25 + str5
    return str25

# Display a single character
def display(n,rr,gg,bb): # ASCII value, red, green blue
    qq = decode(q[n])    # get coded character pattern
    clear()              # Clear the display to BLACK
    for c in range(5):       # Loop through the 5 columns
        for r in range(5):  # Loop through the rows
            cc = qq[r + 5 * c] # current 'bit' at (c, r) coordinate
            if cc == "1":      # Is current 'bit' a one?
                np[5 * r + c] = (gg,rr,bb) # If so, set to colour
    np.write()  # Update the display

# Display a 'frame' of 25 bits - Used in scroll
def display_frame(ss,rr,gg,bb): # Message string of chars and colour
    clear()
    for c in range(5):          # Columns
        for r in range(5):      # Rows
            cc = ss[r + 5 * c]  # Get the 'bit' at (c,r)
            if cc == "1":       # Is it a one?
                np[5 * r + c] = (gg,rr,bb) # If so, set to colour
    np.write()          # Update display
```

```
def scroll(s,rr,gg,bb):
    long = ""
    s = " " + s + "  "           # Add a space character at the end
    l = len(s)                   # Length of string
    for p in range(l):           # Loop through the characters
        i = ord(s[p])-32         # Adjusted ASCII No of character
        temp = decode(q[i])      # get coded character code
        long = long + temp       # Concatenate char to long string
        last5 = temp[-5:]        # Get last 5 characters
        if last5 != "00000":     # Is there a "1" in last column?
            long = long +"00000" #   If so, add an extra space column (W and M are
wider)

    # We now have a very long string of ones and zeros to send to
    # the display in small 'frames' of 25 bits each to fill the pixels

    p = 0
#    print(s," Buffer: ",len(long))
    while p < len(long)-25:          # Have we reached the end?
        frame = long[p:p + 25]       # Extract the 'frame' of 25 bits
        display_frame(frame,rr,gg,bb) # Show the frame on the Neopixels
        p = p + 5                    # Move pointer forward by 1 column
        wait(0.17)                   # Short delay - reduce to speed up scrolling
# ========== END OF FONT LIBRARY =============

# ======= MAIN ===================
sw = Pin(14, Pin.IN,Pin.PULL_UP)      # Set up button - Zero when pressed
# Title
scroll("Zap it",5,5,0)
clear()
np.write()
score = 0            # Targets zapped!
opps = 0             # Zap opportunities = pass the target area
p = 0                # Position of target on circuit (0-15)
errors = 0
err = 0
running = True       # Loop control variable
loops = 4000         # Delay loop variable - reduce to increase speed - harder!
route = [10,5,0,1,2,3,4,9,14,19,24,23,22,21,20,15] # By pixel numbers
p = 0
err = 0
while score < 3:     # You have 3 goes
    running = True
    display(17 + score,0,0,5) # Display the 'Go' number - 1 to 3
    np.write()
    time.sleep(0.3)
    clear()
    np[13] = (0,5,0)     # Display Red 'Target marker'
    if score > 0:
        np[6] = (3,0,0)
    if score == 2:
        np[7] =(3,0,0)
    np.write
    while running:
        np[route[p]] = (3,3,3) # White target
        np.write()
        for w in range(loops):
            if sw.value() == 0:            # Read switch
```

```
                if p != 8 :                 # Off target ?
                    # Missed! - too early or too late
                    err = 1                 # Increment errors
                    np[16] = (0,0,5)        # Err Flag set - blue
                    np.write()
                if (p == 8) and (err == 0):# On target?
                    # Zapped!
                    w = loops+10 #          # Halt w loop
                    running = False         # Halt outer loop
                    np[14] = (5,5,0)        # Hit flag set  = yellow
                    score = score + 1           # Increment score
                    np[score + 5] = (3,0,0)     # Hit score incremented - green
                    np.write()
                    time.sleep(1)
        if p == 8:
            opps = opps + 1                 # Increment opportunities counter
        np[route[p]] = (0,0,0)              # Clear current target position
        np.write()
        p = (p + 1) % 16                    # Increment loop pointer
        if p == 0:
            errors = errors + err           # Update errors
            err = 0
            np[16] = (0,0,0)                # Turn off blue err flag

    np.write()

print(score, opps, errors)
result = 103 - opps - errors
print(result)
scroll("Score: " + str(result),5,5,0)
```

## Sobre el Autor

Este artículo fue escrito por Tony Goodhew. Tony es un profesor de informática jubilado que comenzó a escribir código en 1968, cuando se llamaba programación: ¡comenzó con FORTRAN IV en un IBM 1130! Miembro activo de la comunidad Raspberry Pi, sus principales intereses ahora son la codificación en MicroPython, los viajes y la fotografía.