**Laboratorio 7 – Conexión del registro de desplazamiento 74HC595N para controlar más salidas**

Hasta ahora, hemos estado controlando los LED con pines GPIO individuales.

No solo es un uso bastante ineficiente de la Raspberry Pi sino que incluso si utilizáramos todos los pines disponibles para un LED, eso sería un espectáculo de luces bastante aburrido.

En cambio, al agregar un controlador de LED como el económico 74HC595N, puede obtener fácilmente la capacidad de aprovechar varios LED o múltiples pantallas de números de segmento de 4x7.

Usaremos la interfaz SPI para controlar una matriz de ocho LEDs.

**Preparación**

Esto es lo que necesitarás para empezar. Utilicé el 74HC595N, que es un chip comúnmente disponible localmente.

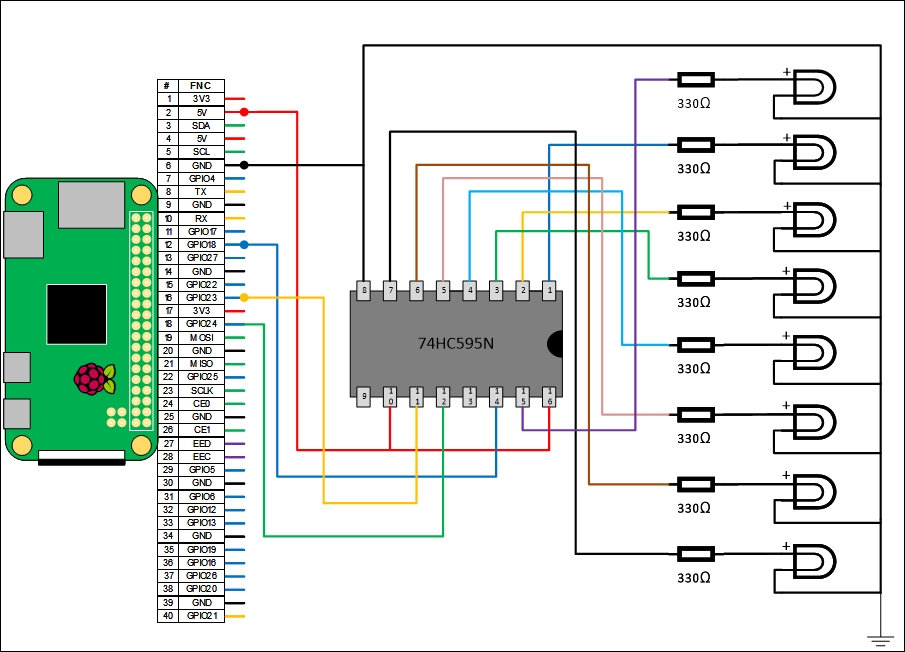
* Ocho resistencias de 330 ohmios.
* Un controlador 74HC595N LED
* Ocho LEDs, cualquier color que quieras.
* Cables de puente, tablero de pan, zapatero
* También usaremos la biblioteca PiShiftPy, disponible en pip:

       sudo pip instalar PiShiftPy

**Cómo hacerlo...**

Conectar el 74HC595N a la Raspberry Pi Zero es tan fácil como cualquier conexión SPI, con las salidas del controlador ejecutando su matriz de LED. Aquí está el esquema de cableado:





El código para probar esto es bastante simple. Cree el archivo shiftregled.py en la carpeta ch6 e ingrese el siguiente código:

import PiShiftPy as shift

import time

# Controlar salidas por el registro serial 74HC595

# Se debe probar con un 7-segmentos o a manera de prueba, con LEDs

# Si no posee todos, utilice uno solamente.

def main():

shift.init()

shift.write\_all(0x00)

while True:

try:

ledseq(1,0,0,0,0,0,0,0)

ledseq(1,1,0,0,0,0,0,0)

ledseq(1,1,1,0,0,0,0,0)

ledseq(0,1,1,1,0,0,0,0)

ledseq(0,0,1,1,1,0,0,0)

ledseq(0,0,0,1,1,1,0,0)

ledseq(0,0,0,0,1,1,1,0)

ledseq(0,0,0,0,0,1,1,1)

ledseq(0,0,0,0,0,0,1,1)

ledseq(0,0,0,0,0,0,0,1)

ledseq(0,0,0,0,0,0,1,1)

ledseq(0,0,0,0,0,1,1,1)

ledseq(0,0,0,0,1,1,1,0)

ledseq(0,0,0,1,1,1,0,0)

ledseq(0,0,1,1,1,0,0,0)

ledseq(0,1,1,1,0,0,0,0)

ledseq(1,1,1,0,0,0,0,0)

ledseq(1,1,0,0,0,0,0,0)

finally:

ledseq(0,0,0,0,0,0,0,0)

def ledseq(a,b,c,d,e,f,g,h):

shift.push\_bit(a)

shift.push\_bit(b)

shift.push\_bit(c)

shift.push\_bit(d)

shift.push\_bit(e)

shift.push\_bit(f)

shift.push\_bit(g)

shift.push\_bit(h)

shift.write\_latch()

time.sleep(0.1)

if \_\_name\_\_ == "\_\_main\_\_":

main()

La biblioteca PiShiftPy hace que la operación de uno o más registros de turnos sea muy fácil. Los pines GPIO predeterminados utilizados son GPIO 18 para datos, GPIO 23 para el reloj y GPIO 24 para el pestillo, por lo que no tuvimos que poner nada en init (). Si queremos usar pines diferentes o estamos utilizando más de un registro de desplazamiento, podemos definir explícitamente nuestra configuración:

shift.init(data\_pin=18, clock\_pin=23, latch\_pin=24, chain\_number=1)

Los pines corresponden a los números GPIO de datos, reloj y chip select corresponden al bus SPI y el chain\_number indica cuántos registros de desplazamiento están conectados.

La función ledseq () simplemente identifica los ocho bits asociados con las ocho conexiones de LED, y las activa o desactiva. La función write\_latch () enclava los bits empujados a los pines de salida en el 74HC595N. Esto evita que los LED parpadeen a medida que empujamos los bits uno a la vez.

def ledseq(a,b,c,d,e,f,g,h):

shift.push\_bit(a)

shift.push\_bit(b)

shift.push\_bit(c)

shift.push\_bit(d)

shift.push\_bit(e)

shift.push\_bit(f)

shift.push\_bit(g)

shift.push\_bit(h)

shift.write\_latch()

time.sleep(0.1)

En la función main (), solo llamamos a la función ledseq en un bucle con la configuración de encendido / apagado que estamos buscando. Es sencillo activar cualquier configuración de LED, ya que ciertos LED le notifican eventos diferentes o ejecutan diferentes patrones.

Además, puede extenderlo a chips 74HC595N adicionales y ejecutar incluso más LED utilizando los mismos tres pines GPIO en la Raspberry Pi. Esto se hace conectando el pin 9 de este 74HC595N al pin 14 del siguiente. Los pines 11 y 12 están conectados directamente entre sí. Esto guarda sus otros pines para otros sensores, luces, pantallas o motores, algunos de los cuales hemos jugado aquí y otros que veremos en el próximo capítulo.