FICHA [05]: Electrónica Digital & Raspberry Pi

Práctica : **Prácticas con LED de 7 segmentos (manejo vía chip 4511)**
Nivel de dificultad teoria: **básico**
Nivel de dificultad hardware: **básico**
Nivel de dificultad software: **básico**

Nota : precisa el KIT básico de prácticas de Webtronika ***(Ref.*** KIT001***) o material equivalente.***

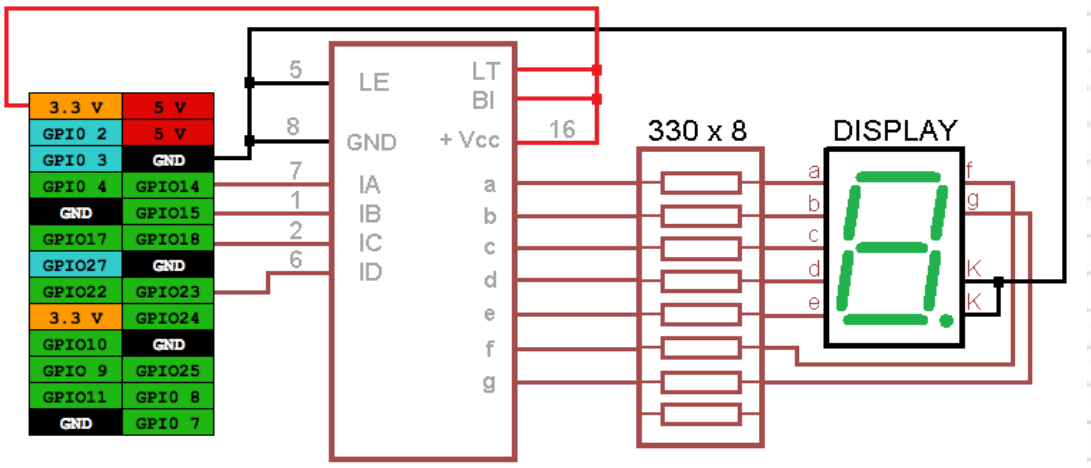
DESCRIPCIÓN DE LA PRÁCTICA

Conexión de DISPLAY (LED de 7 segmentos) a la tarjeta Raspberry Pi y manejo del mismo con explicación teórico práctica. Utiliza solo 4 patillas (OUT) para su manejo debido a que la decodificación de los 7 segmentos se realiza en el chip **4511**. Se proporciona código para su operativa.

MATERIALES UTILIZADOS

1 Tarjeta Raspberry Pi (con s.o. Linux Wheezy instalado).
1 Placa prototipo pequeña.
1 array de resistencias (DIL) de 330 ohmios.
1 chip decodificador BCD a 7 segmentos 4511
1 DISPLAY LED de 7 segmentos (cátodo común).
Varios cables para prácticas.

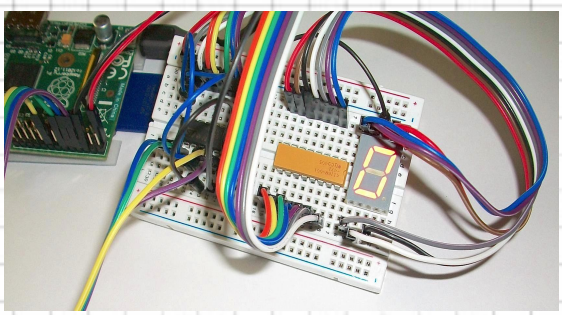
ESQUEMA ELECTRONICO



Esquema electrónico del montaje propuesto.

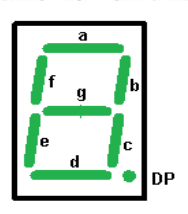
OPERATIVA DE MONTAJE

- Apagamos la tarjeta Raspberry Pi (sudo halt)
- Conectamos en la placa de prototipos los componentes del montaje de acuerdo al esquema mostrado.
- Conectamos los hilos correspondientes desde la placa de prototipos al puerto GPIO de la Raspberry Pi.
- Nos aseguramos de que el cableado es el correcto
- Arrancamos la Raspberry Pi.
- Copiamos el sowtware suministrado.
- Ejecutamos el software de control.

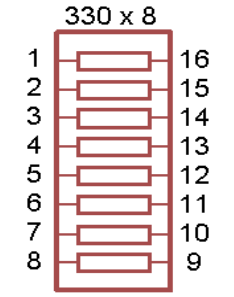


INFORMACIÓN ADICIONAL PARA EL MONTAJE

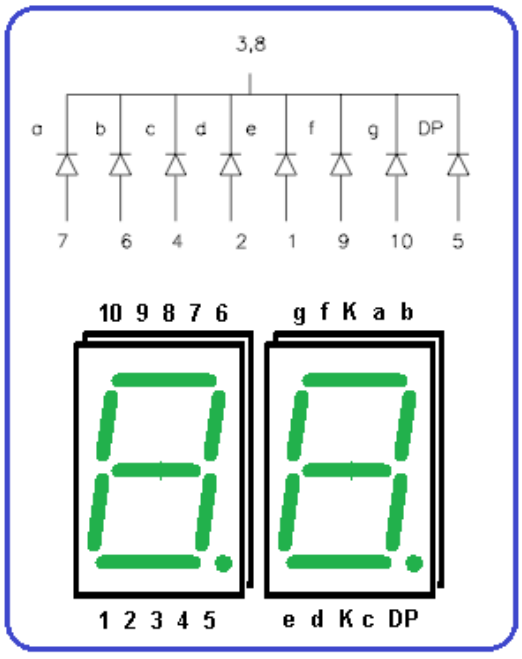
La distribución de los segmentos en un DISPLAY-7 suele responder a un esquema como este:



Por cada diodo LED conectado a cada patilla del puerto GPIO hemos de conectar una resistencia limitadora. En vez de utilizar 8 resistencias hemos optado por utilizar un "array". El array de resistencias utilizado nos evita la engorrosa tarea de conectar las mismas de forma independiente. El encapsulado DIL que las contiene (similar a un chip de 16 patillas) responde al esquema interno siguiente:

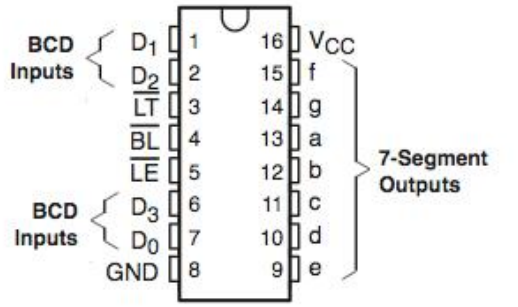


El punto decimal (DP) representa el octavo LED existente en un DISPLAY-7. De acuerdo al conexionado utilizado, existen DISPLAY-7 de tipo cátodo común y ánodo común. El utilizado en nuestro montaje es de tipo cátodo común. El conexionado del DISPLAY-7 que hemos utilizado responde a 10 patillas, cuya numeración, conexionado interno y relación con los segmentos (LEDs) internos puede verse en este esquema.



Nota : las patillas 3 y 8 del DISPLAY de 7 segmentos – unidas internamente - constituyen el cátodo común a los 8 LEDs internos.

El detalle del patillaje del chip decodificador 4511, así como la tabla de asignación de salidas a segmentos (a...g) en función de la entrada BCD (D0 .. D3) puede verse aquí.



FUNCTION TABLE

INPUTS							OUTPUTS							
$\overline{LE}$	$\overline{BL}$	$\overline{LT}$	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g	DISPLAY
X	X	L	X	X	X	X	H	H	H	H	H	H	H	8
X	L	H	X	X	X	X	L	L	L	L	L	L	L	Blank
L	H	H	L	L	L	L	H	H	H	H	H	H	L	0
L	H	H	L	L	L	H	L	H	H	L	L	L	L	1
L	H	H	L	L	H	L	H	H	L	H	H	L	H	2
L	H	H	L	L	H	H	H	H	H	H	L	L	H	3
L	H	H	L	H	L	L	L	H	H	L	L	H	H	4
L	H	H	L	H	L	H	H	L	H	H	L	H	H	5
L	H	H	L	H	H	L	L	L	H	H	H	H	H	6
L	H	H	L	H	H	H	H	H	H	L	L	L	L	7
L	H	H	H	L	L	L	H	H	H	H	H	H	H	8
L	H	H	H	L	L	H	H	H	H	L	L	H	H	9
L	H	H	H	L	H	L	L	L	L	L	L	L	L	Blank
L	H	H	H	L	H	H	L	L	L	L	L	L	L	Blank
L	H	H	H	H	L	L	L	L	L	L	L	L	L	Blank
L	H	H	H	H	L	H	L	L	L	L	L	L	L	Blank
L	H	H	H	H	H	L	L	L	L	L	L	L	L	Blank
L	H	H	H	H	H	H	L	L	L	L	L	L	L	Blank
H	H	H	X	X	X	X	†	†	†	†	†	†	†	†

Nota : en este montaje se ha obviado el punto decimal (DP) del DISPLAY

PROGRAMA EJEMPLO (lenguaje Python)

Programa ejemplo **pi4511.py** desarrollado en lenguaje python. Véanse notas más abajo.

```
# ================================================
# -  Programa para control de DISPLAY-LED via chip  -
# -  (control via chip 4511)
# -  utiliza 4 salidas de la Raspberry Pi
# ------------------------------------------------
# -  J.C.G.P. - (c) Webtronika 2013 -
# ================================================
#!/usr/bin/python
import sys
import curses
import time
import RPi.GPIO as GPIO
# ------------------------------------------------------------
# - Rutina de configuracion de puertos en GPIO ------------------
# ------------------------------------------------------------
def confiGPIO(param1):
          GPIO.setmode(GPIO.BCM)
          GPIO.setwarnings(False)
          GPIO.setup(14, GPIO.OUT)
          GPIO.setup(15, GPIO.OUT)
          GPIO.setup(18, GPIO.OUT)
          GPIO.setup(23, GPIO.OUT)
          return 0
# ------------------------------------------------------------
# - Rutina RESET-DISPLAY ----------------------------------------
# - valores de param1 : 0 => 0 // 8 => 8 // otro valor => Blank
# ------------------------------------------------------------
def RESET_DISPLAY(param1):
    if param1 == 8:
        GPIO.output(14, False)
        GPIO.output(15, False)
        GPIO.output(18, False)
        GPIO.output(23, True)
    elif param1 == 0:
        GPIO.output(14, False)
        GPIO.output(15, False)
        GPIO.output(18, False)
        GPIO.output(23, False)
    else:
        GPIO.output(14, True)
        GPIO.output(15, True)
        GPIO.output(18, True)
        GPIO.output(23, True)
          return 0
# ------------------------------------------------------------
# - Rutina TEST-DISPLAY -----------------------------------------
# - param1= tiempo de retardo en ms.
# ------------------------------------------------------------
def TEST_DISPLAY(param1):
          RESET_DISPLAY(1)
          time.sleep(param1)
          for i in range(10):
                    SHOW_DISPLAY(48+i)
                    time.sleep(param1)
          RESET_DISPLAY(1)
          return 0
# ------------------------------------------------------------
# - Rutina Muestra cifra en DISPLAY -----------------------------
# - param1= cifra a mostrar (0 ... 9)
# ------------------------------------------------------------
def SHOW_DISPLAY(param1):
          RESET_DISPLAY(0)
          if param1 == 48: # numero 0
                    GPIO.output(14, False)
                    GPIO.output(15, False)
                    GPIO.output(18, False)
                    GPIO.output(23, False)
          elif param1 == 49: # numero 1
                    GPIO.output(14, True)
                    GPIO.output(15, False)
                    GPIO.output(18, False)
                    GPIO.output(23, False)
          elif param1 == 50: # numero 2
                    GPIO.output(14, False)
                    GPIO.output(15, True)
                    GPIO.output(18, False)
                    GPIO.output(23, False)
          elif param1 == 51: # numero 3
                    GPIO.output(14, True)
                    GPIO.output(15, True)
                    GPIO.output(18, False)
                    GPIO.output(23, False)
          elif param1 == 52: # numero 4
                    GPIO.output(14, False)
                    GPIO.output(15, False)
                    GPIO.output(18, True)
                    GPIO.output(23, False)
          elif param1 == 53: # numero 5
                    GPIO.output(14, True)
                    GPIO.output(15, False)
                    GPIO.output(18, True)
                    GPIO.output(23, False)
          elif param1 == 54: # numero 6
                    GPIO.output(14, False)
                    GPIO.output(15, True)
                    GPIO.output(18, True)
                    GPIO.output(23, False)
```

```
        elif param1 == 55: # numero 7
                GPIO.output(14, True)
                GPIO.output(15, True)
                GPIO.output(18, True)
                GPIO.output(23, False)
        elif param1 == 56: # numero 8
                GPIO.output(14, False)
                GPIO.output(15, False)
                GPIO.output(18, False)
                GPIO.output(23, True)
        elif param1 == 57: # numero 9
                GPIO.output(14, True)
                GPIO.output(15, False)
                GPIO.output(18, False)
                GPIO.output(23, True)
        return 0
# ---------------------------------------------------------------
# ###############################################################
# - PROGRAMA PRINCIPAL ----- Usa curses para salida a pantalla --
# ###############################################################
# ---------------------------------------------------------------
stdscr = curses.initscr()
curses.cbreak()
curses.start_color()
# definicion de paletas de color
curses.init_pair(1, curses.COLOR_WHITE, curses.COLOR_BLUE)
curses.init_pair(2, curses.COLOR_YELLOW, curses.COLOR_BLACK)
hsize = curses.COLS
vsize = curses.LINES
curses.curs_set(0)
curses.noecho
stdscr.border(0)
stdscr.keypad(1)
stdscr.nodelay(1)
configPIO(0)
RESET_DISPLAY(8) # (0) = 0 // (8) = 8 // (otro valor = Blank)
try:
        stdscr.bkgd(curses.color_pair(1)) # seleccion de paleta de color (1/2)
        while True:  # Rutina principal del programa
                char = stdscr.getch()
                if (char == 81 or char == 113):   # Teclas Q/q
                        break
                elif (char == 84 or char == 116): # Teclas T/t
                        TEST_DISPLAY(0.2)
                elif (char >= 48 and char <= 57): # Teclas NUMERICAS
                        SHOW_DISPLAY(char)
                else:
                        stdscr.addstr(vsize/6, (hsize/2)-24,"  ---- Test de DISPLAY LED via chip 4511 ---- ")
                        stdscr.addstr((vsize/5)+5, (hsize/2)- 22,"[T] - efectua un TEST del DISPLAY")
                        stdscr.addstr((vsize/5)+7, (hsize/2)- 22,"[1..0] - pulsa numero para mostrar en DISPLAY")
                        stdscr.addstr(vsize-4, (hsize/2)- 22," Pulsa [Q] para salir   ")
                        stdscr.refresh()
finally:
    curses.nocbreak()
    stdscr.keypad(0)
    curses.echo()
    curses.endwin()
```

OPERATIVA SOFTWARE

Las librerías (python) para el control del puerto GPIO vienen ya pre-instaladas en el sistema operativo Raspbian. Conviene, no obstante, estar actualizado a la última versión (Wheezy).

PROGRAMA EJEMPLO (**pi4511.py** desarrollado en lenguaje python.)

Este programa puede copiarse tal como está y guardarse en un fichero llamado **pi4511.py** en nuestra Raspberry Pi. Para ponerlo en marcha hemos de seguir cuidadosamente estos pasos:

(1) Tener el sistema operativo Raspbian Wheezy instalado en nuestra Raspberry Pi.
(2) Efectuar correctamente el conexionado del circuito a nuestra Raspberry Pi.
(3) Vamos al directorio del programa y damos al fichero **pi4511.py** permisos de ejecución con

sudo chmod 777 pi4511.py

(5) En un terminal, desde el directorio del programa, ejecutamos (**IMPORTANTE** : como root)

sudo python pi4511.py

ENLACES DE INTERÉS

RASPBERRY PI: página oficial del proyecto Raspberry Pi.

DIVERTEKA: notas técnicas de utilidad que pueden usarse con el Kit de Webtronika.

Nota : Webtronika no se responsabiliza de cualquier daño en los componentes empleados en la práctica. Es responsibilidad del usuario final la cuidadosa verificación y la toma de precauciones adecuadas para evitar posibles daños a los mismos.