Práctica 3:
Uso del puerto GPIO de la Raspberry Pi

Fundamentos de Sistemas Embebidos

Autor: José Mauricio Matamoros de Maria y Campos

1. Objetivo

El alumno aprenderá a utilizar el puerto GPIO de la Raspberry Pi, configurándo varios pines como salidas digitales para el control de leds y circuitos de lógica TTL.

2. Material

Se asume que el alumno cuenta con un una Raspberry Pi con sistema operativo Raspbian e interprete de Python instalado. Se aconseja encarecidamente el uso de *git* como programa de control de versiones.

Además, el alumno necesitará:

- 7 Diodos emisores de luz LEDS
- 8 resistencias de 330Ω
- 1 Condensador de 0.1μF
- 1 Array Darlington ULN2003 (o 7 transistores de potencia)
- 1 Decodificador de 7–segmentos ánodo común
- 1 Display de 7 segmentos ánodo común
- 1 Conector DIL con cable plano tipo listón para el GPIO de la Raspberry Pi (similar al de un Disco Duro PATA, véase Figura 1)
- 1 protoboard o circuito impreso equivalente
- 1 fuente de alimentación regulada a 5V y al menos 2 amperios de salida
- Cables y conectores varios

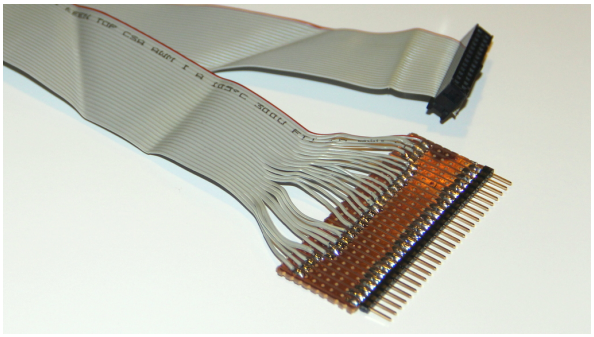

Figura 1: Cable plano con conector DIL

3. Instrucciones

1. Alambre el circuito tal y como se detalla en la subsección 3.1
2. Antes de conectar la Raspberry Pi, pruebe el circuito como se explica en la subsección 3.2
3. Realice los programas de las subsecciones 3.3 a 3.5
4. Analice los programas de las subsecciones 3.3 a 3.5, realice los experimentos propuestos en la sección 4 y con los resultados obtenidos responda el cuestionario de la sección 5.

3.1. Paso 1: Alambrado

El proceso de alambrado de esta práctica considera dos circuitos.

El primer circuito permitirá controlar con las salidas digitales del GPIO de la Raspberry Pi el encendido y apagado de siete leds mediante el uso de un encapsulado de varios controladores de potencia tipo Darlington

(*Darlington Array*). De manera simimlar, el segundo circuito se auxiliará de un integrado TTL para desplegar números del 0 al 9 en un display de 7 segmentos (véase Figura 2).

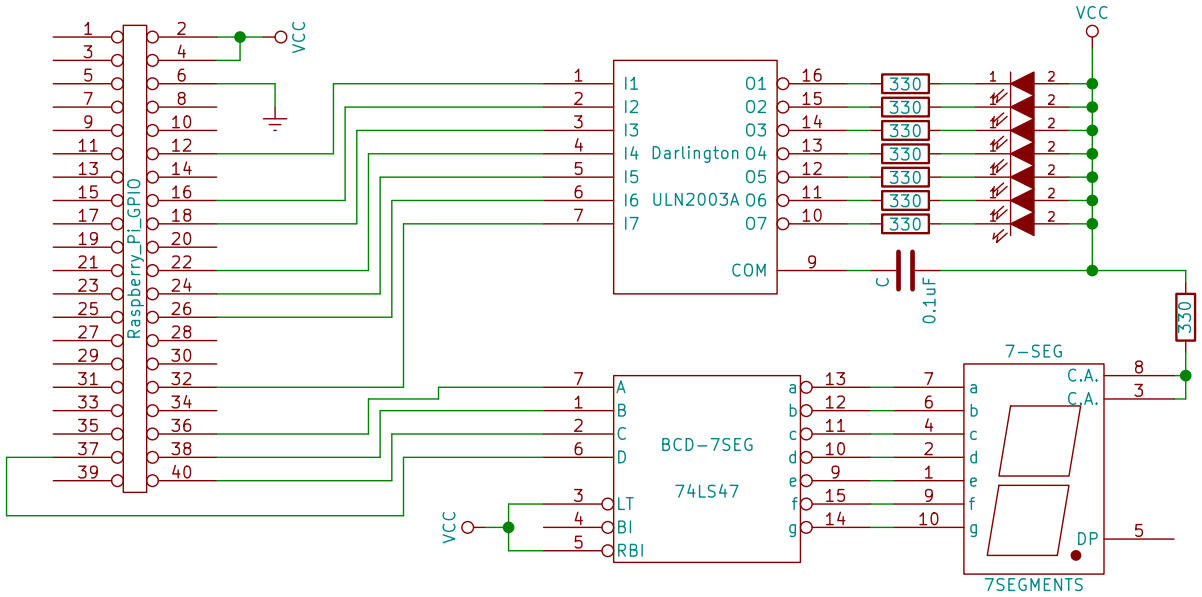

Figura 2: Diagrama de conexiones del circuito a alambrar

3.1.1. Subcircuito 1: leds en línea

. Forme los siete leds en línea, cuidando de que todos tengan la misma orientación. A continuación, conecte el cátodo de cada LED a una resistencia de 300Ω, y ésta a su vez a una salida libre del controlador ULN2003, de tal manera que el primer led de la fila esté conectado a la salida 1, el segundo a la salida 2, y así sucesivamente.

De manera similar, conecte las entradas 1 a 7 del ULN2003 a las salidas GPIO 18, 23, 24, 25, 8, 7, y 12 de la Raspberry Pi mediante el cable tipo listón (pines 12, 16, 18, 22, 24, 26 y 32). La conexión de uno de los leds se conecte al GPIO12/PWM es importante, pues éste pin se utilizará para variar la intensidad del led más adelante.

Complete el alambrado del ULN2003 conectándolo a tierra, conectando el común de éste a VCC mediante el capacitor de 0.1μF. Finalmente, conecte el ánodo de todos los leds a VCC.

3.1.2. Subcircuito 2: Display de 7 segmentos

. Comience conectando las salidas a a g del display del circuito controlador 74LS47 con los pines homónimos del display de siete segmentos. A continuación, conecte el integrado a tierra y las terminales LT y RBI (pines 3 y 5) a VCC, dejando BI (pin 4) sin conectar tal como se muestra en la Figura 2. Como siguiente paso, conecte el ánodo común del display a VCC mediante una resistencia de 330Ω.

A continuación conecte las entradas A, B, C, D a las salidas GPIO 16, 20, 21 y 26 respectivamente (pines 36, 38, 40 y 37 de la Raspberry Pi) mediante el cable tipo listón.

Finalmente, para terminar el alambrado, conecte el 74LS47 a VCC y tierra.

3.2. Paso 2: Configuración y prueba del circuito

Antes de proceder, verifique conexiones con un multímetro en busca de corto circuitos. En particular verifique que exista una impedancia muy alta entre los pines 4 y 6 (VCC y tierra) del cable listón que conectará al GPIO de la Raspberry Pi.

Tras lo anterior, conecte VCC y tierra del circuito alambrado a un eliminador de corriente de 5V y cierre el circuito entre los pines 4 y 32 del cable listón. Si todo está alambrado correctamente, el último del de la fila deberá encender.

Ahora cierre el circuito entre los pines 4 y 37 del cable listón. Si todo está alambrado correctamente, el display de siete segmentos deberá mostrar un 1.

Importante: Ninguno de los circuitos o resistencias debe calentarse. Si alguno de los componentes emitiera calor, verifique las conexiones en busca de corto circuitos o reemplace los componentes dañados.

Verificadas las conexiones, instale los complementos de desarrollo del puerto de propósito general de la Raspberry Pi (deberían venir instalados por defecto).

```
$ sudo apt-get install python-rpi.gpio python3-rpi.gpio
```

3.3. Paso 3: Led parpadeante

Con todas las conexiones probadas y verificadas, es seguro proceder al control de señales digitales utilizando el GPIO de la Raspberry Pi.

Inicie su Raspberry Pi y, ya sea mediante una terminal remota o directamente en ella, ejecute el siguiente código Python para hacer parpadear uno de los leds del circuito alambrado.

```
# Importa la librería de control del GPIO de la Raspberry Pi
import RPi.GPIO as GPIO
# Importa la función sleep del módulo time
from time import sleep

# Desactivar advertencias (warnings)
GPIO.setwarnings(False)
# Configurar la librería para usar el número de pin.
# Llame GPIO.setmode(GPIO.BCM) para usar el canal SOC definido por Broadcom
GPIO.setmode(GPIO.BOARD)
# Configurar el pin 32 como salida y habilitar en bajo
GPIO.setup(32, GPIO.OUT, initial=GPIO.LOW)

# El siguiente código hace parpadear el led
while True: # Bucle infinito
  sleep(0.5)                # Espera 500ms
  GPIO.output(32, GPIO.HIGH) # Enciende el led
  sleep(0.5)                # Espera 500ms
  GPIO.output(32, GPIO.LOW)  # Apaga el led
```

3.4. Paso 4: Led parpadeante con PWM

En lugar de utilizar tiempos de espera (mismos que consumen tiempo de procesamiento y energía), es posible hacer parpadear el led de manera mucho más precisa y rápida utilizando uno de los moduladores de ancho de pulso (en inglés *Pulse Width Modulation* o *PWM*) por hardware que incorpora la Raspberry Pi.

Ejecute el siguiente código Python para hacer parpadear uno de los leds del circuito alambrado.

```
# Importa la librería de control del GPIO de la Raspberry Pi
import RPi.GPIO as GPIO
# Importa la función sleep del módulo time
from time import sleep

# Desactivar advertencias (warnings)
GPIO.setwarnings(False)
# Configurar la librería para usar el número de pin.
GPIO.setmode(GPIO.BOARD)
# Configurar el pin 32 como salida y habilitar en bajo
GPIO.setup(32, GPIO.OUT, initial=GPIO.LOW)
# Inicializar el pin 32 como PWM a una frecuencia de 2Hz
pwm = GPIO.PWM(32, 1)

# El siguiente código hace parpadear el led
pwm.start(50)

```

```
flag = True
while flag:
  try:
    dutyCycle = int(input("Ingrese ciclo de trabajo: "))
    pwm.ChangeDutyCycle(dutyCycle)
  except:
    flag = False
    pwm.ChangeDutyCycle(0)
# Detiene el PWM
pwm.stop()
# Reinicia los puertos GPIO (cambian de salida a entrada)
GPIO.cleanup()
```

3.5. Paso 5: Display de siete segmentos

El último ejemplo consiste en mostrar números en el display de siete segmentos.
Analice y ejecute el código mostrado a continuación.

```
# Importa la librería de control del GPIO de la Raspberry Pi
import RPi.GPIO as GPIO
# Importa la función sleep del módulo time
from time import sleep

# Desactivar advertencias (warnings)
GPIO.setwarnings(False)
# Configurar la librería para usar el número de pin.
GPIO.setmode(GPIO.BOARD)
# Configurar pines 36, 38, 40 y 37 como salida y habilitar en bajo
GPIO.setup(32, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(38, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(40, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(37, GPIO.OUT, initial=GPIO.LOW)

# Mapea bits a los pines de la GPIO
def bcd7(num):
  GPIO.output(32, GPIO.HIGH if num & 0x00000008 else GPIO.LOW )
  GPIO.output(38, GPIO.HIGH if num & 0x00000004 else GPIO.LOW )
  GPIO.output(40, GPIO.HIGH if num & 0x00000002 else GPIO.LOW )
  GPIO.output(37, GPIO.HIGH if num & 0x00000001 else GPIO.LOW )

flag = True
while flag:
  try:
    num = int(input("Ingrese número entero: "))
    bcd(num)
  except:
    flag = False
# Reinicia los puertos GPIO (cambian de salida a entrada)
GPIO.cleanup()
```

4. Experimentos

1. [1pt] Modifique el código de la subsección 3.3 para todos los leds de la fila parpadeen.
2. [1pt] Modifique el código de las subsecciones 3.3 y 3.5 para que los leds de la fila enciendan de manera continua en una marquesina de izquierda a derecha.
3. [2pt] Modifique el código de las subsecciones 3.3 y 3.5 para que los leds de la fila enciendan de manera continua en una marquesina de derecha a izquierda con velocidad variable definida por el usuario.

4. [1pt] Con base en las modificaciones anteriores, genere una marquesina que haga parecer que «la luz rebota» al llegar a las orillas (efecto ping-pong) con velocidad variable definida por el usuario.

5. [3pt] Tomando como base el código de la subsección 3.4, haga que uno de los leds encienda gradualmente a lo largo de un segundo hasta adquirir máxima potencia, permanezca encendido medio segundo, y después se apague gradualmente a lo largo de otro segundo.

5. Cuestionario

1. [0.5pt] Explique por qué usar corrimientos es la manera más eficiente de generar una marquesina.

2. [0.5pt] Explique las ventajas o desventajas que tiene utilizar un modulador de ancho de pulso sobre tiempos de espera programados (*delays*).

3. [0.5pt] ¿Sería posible generar una marquesina circular utilizando el 74LS47 y el display de siete segmentos? Justifique su respuesta.

4. [0.5pt] ¿Es posible configurar cualquier pin de la GPIO como PWM? Justifique su respuesta.

A. Programa Ejemplo: `blink.py`

src/blink.py

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

# Importa la librería de control del GPIO de la Raspberry Pi
import RPi.GPIO as GPIO
# Importa la función sleep del módulo time
from time import sleep

# Desactivar advertencias (warnings)
# GPIO.setwarnings(False)
# Configurar la librería para usar el número de pin.
# Llame GPIO.setmode(GPIO.BCM) para usar el canal SOC definido por Broadcom
GPIO.setmode(GPIO.BOARD)

# Configurar el pin 32 como salida y habilitar en bajo
GPIO.setup(32, GPIO.OUT, initial=GPIO.LOW)

# El siguiente código hace parpadear el led
while True: # Bucle infinito
  sleep(0.5)                 # Espera 500ms
  GPIO.output(32, GPIO.HIGH) # Enciende el led
  sleep(0.5)                 # Espera 500ms
  GPIO.output(32, GPIO.LOW)  # Apaga el led
```

B. Programa Ejemplo: `bcd.py`

src/bcd.py

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

# Importa la librería de control del GPIO de la Raspberry Pi
import RPi.GPIO as GPIO
# Importa la función sleep del módulo time
from time import sleep

# Desactivar advertencias (warnings)
# GPIO.setwarnings(False)
# Configurar la librería para usar el número de pin.
GPIO.setmode(GPIO.BOARD)
# Configurar pines 36, 38, 40 y 37 como salida y habilitar en bajo
GPIO.setup(36, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(38, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(40, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(37, GPIO.OUT, initial=GPIO.LOW)

# Mapea bits a los pines de la GPIO
def bcd7(num):
  GPIO.output(36, GPIO.HIGH if (num & 0x00000001) > 0 else GPIO.LOW )
  GPIO.output(38, GPIO.HIGH if (num & 0x00000002) > 0 else GPIO.LOW )
  GPIO.output(40, GPIO.HIGH if (num & 0x00000004) > 0 else GPIO.LOW )
  GPIO.output(37, GPIO.HIGH if (num & 0x00000008) > 0 else GPIO.LOW )

flag = True
while flag:
  try:
    num = int(input("Ingrese número entero: "))
    bcd7(num)
  except:
    flag = False
  #end try
#end while

# Reinicia los puertos GPIO (cambian de salida a entrada)
GPIO.cleanup()
```

C. Programa Ejemplo: `pwm.py`

src/pwm.py

```
1
2  # Future imports (Python 2.7 compatibility)
3  from __future__ import absolute_import
4  from __future__ import division
5  from __future__ import print_function
6
7  # Importa la librería de control del GPIO de la Raspberry Pi
8  import RPi.GPIO as GPIO
9  # Importa la función sleep del módulo time
10 from time import sleep
11
12 # Desactivar advertencias (warnings)
13 GPIO.setwarnings(False)
14 # Configurar la librería para usar el número de pin.
15 GPIO.setmode(GPIO.BOARD)
16 # Configurar el pin 32 como salida y habilitar en bajo
17 GPIO.setup(32, GPIO.OUT, initial=GPIO.LOW)
18 # Inicializar el pin 32 como PWM a una frecuencia de 1Hz
19 pwm = GPIO.PWM(32, 1)
20
21 # El siguiente código hace parpadear el led
22 pwm.start(50)
23 flag = True
24 while flag:
25   try:
26     dutyCycle = int(input("Ingrese ciclo de trabajo: "))
27     pwm.ChangeDutyCycle(dutyCycle)
28   except:
29     flag = False
30     pwm.ChangeDutyCycle(0)
31   #end try
32 #end while
33 # Detiene el PWM
34 pwm.stop()
35 # Reinicia los puertos GPIO (cambian de salida a entrada)
36 GPIO.cleanup()
```

D. Programa Ejemplo: `pwm_fast.py`

src/pwm_fast.py

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(32, GPIO.OUT, initial=GPIO.LOW)
pwm = GPIO.PWM(32, 1000)

pwm.start(50)
flag = True
while flag:
  try:
    dutyCycle = int(input("Duty cycle: "))
    pwm.ChangeDutyCycle(dutyCycle)
  except:
    flag = False
    pwm.ChangeDutyCycle(0)
  #end try
#end while
pwm.stop()
GPIO.cleanup()
```