Programa 1:
Puerto GPIO de la Raspberry Pi en simulador

Fundamentos de Sistemas Embebidos

Autor: José Mauricio Matamoros de Maria y Campos

1. Objetivo

El alumno se familiarizará con el puerto GPIO de la Raspberry Pi (simulado), configurándo varios pines como salidas digitales para el control de leds y circuitos de lógica TTL.

2. Material

Ninguno

3. Instrucciones

1. Descargue y pruebe la tarjeta simuladora siguiendo los pasos de la subsección 3.1
2. Realice los programas de las subsecciones 3.2 a 3.4
3. Analice los programas de las subsecciones 3.2 a 3.4, realice los experimentos propuestos en la **??** y con los resultados obtenidos responda el cuestionario de la sección 6.

3.1. Paso 1: Configuración del Simulador

Descargue el simulador de `https://github.com/kyordhel/RPiVirtualBoard` ejecutando la siguiente línea de comandos:

```
git clone https://github.com/kyordhel/RPiVirtualBoard.git
cd RPiVirtualBoard
```

A continuación instale todas las dependencias requeridas por el simulador usando *pip*:

```
sudo apt install python3-tk
pip install --user -r requirements.txt
```

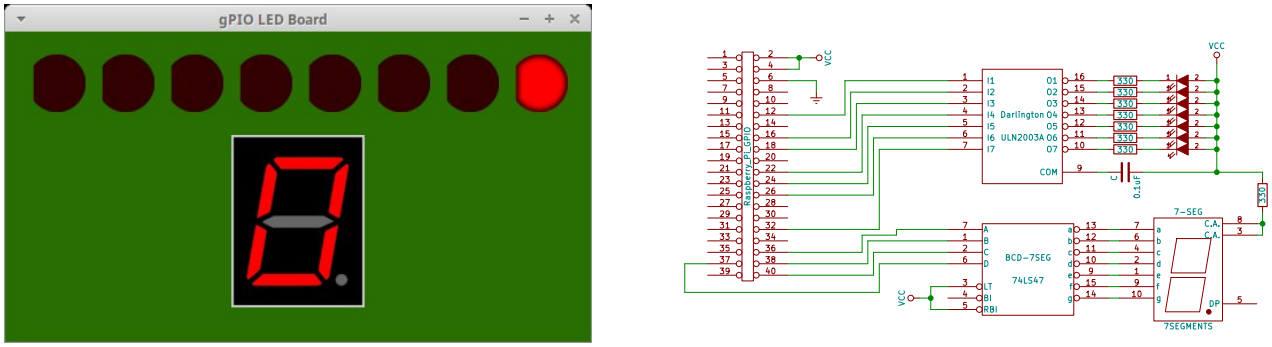
Finalmente, pruebe el simulador ejecutando la siguiente línea:

```
./blink.py
```

O bien, si desea mantener el simulador conmo un proyecto aislado y cuenta con la utilería *pipenv*[1] para tal propósito, después de clonar el proyecto basta con ejecutar:

```
pipenv run python blink.py
```

Si la configuración es correcta, verá una ventana similar a la de la Figura 1a con uno de los leds virtuales parpadeando. Este simulador implementa el circuito mostrado en la Figura 1b



(a) Simulador de tarjeta con leds para la GPIO de la Raspberry Pi

(b) Circuito implementado en el simulador

Figura 1: Simulador y circuito implementado

3.2. Paso 2: Led parpadeante

El código mostrado en Código de Ejemplo 1 muestra cómo se haría parpadear un LED mediante tiempos de espera o *sleeps* utilizando la Raspberry Pi.

Código ejemplo 1: `blink.py`

```
# Import Raspberry Pi's GPIO control library
import RPi.GPIO as GPIO
# Imports sleep functon
from time import sleep
# Initializes virtual board (comment out for hardware deploy)
import virtualboard

# Set up Rpi.GPIO library to use physical pin numbers
GPIO.setmode(GPIO.BOARD)
# Set up pin no. 32 as output and default it to low
GPIO.setup(32, GPIO.OUT, initial=GPIO.LOW)

# Blink the led
while True: # Forever
  sleep(0.5)                  # Wait 500ms
  GPIO.output(32, GPIO.HIGH) # Turn led on
  sleep(0.5)                  # Espera 500ms
  GPIO.output(32, GPIO.LOW)  # Turn led off
```

Estudie el código y véalo en funcionamiento, ejecutándolo de la siguiente manera:

```
./blink.py
```

3.3. Paso 3: 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.

El código mostrado en Código de Ejemplo 2 muestra cómo se haría parpadear un LED mediante *PWM* utilizando la Raspberry Pi.

Código ejemplo 2: `pwm.py`

```
# Import Raspberry Pi's GPIO control library
```

[1] *pipenv* es una herramienta que facilita la creación y administración de entornos virtuales en cualquier proyecto escritos en Python, llevando un control riguroso de los paquetes de los que depende dicho proyecto. Se puede instalar fácilmente con la línea `pip install -user pipenv`

```
import RPi.GPIO as GPIO
# Initializes virtual board (comment out for hardware deploy)
import virtualboard

# Set up Rpi.GPIO library to use physical pin numbers
GPIO.setmode(GPIO.BOARD)
# Set up pin no. 32 as output and default it to low
GPIO.setup(32, GPIO.OUT, initial=GPIO.LOW)
# Set up PWM on pin 32 at 1Hz
pwm = GPIO.PWM(32, 1)
print("Starting pwm")
# Set duty cycle to 50% to blink 500ms on 500ms off
pwm.start(50)
print("Pwm started")
flag = True

# Blink the led
while flag:
  try:
    dutyCycle = int(input("Set duty cycle: "))
    pwm.ChangeDutyCycle(dutyCycle)
  except:
    flag = False
    pwm.ChangeDutyCycle(0)

# Stop the PWM
pwm.stop()
# Reset all ports to its default state (inputs)
GPIO.cleanup()
```

Estudie el código y véalo en funcionamiento, ejecutándolo de la siguiente manera:

```
./pwm.py
```

3.4. Paso 4: Display de siete segmentos

El código mostrado en Código de Ejemplo 3 muestra cómo se operaría un display de siete segmentos mediante una controladora TTL 74LS47 utilizando la Raspberry Pi.

Código ejemplo 3: `bcd.py`

```
# Import Raspberry Pi's GPIO control library
import RPi.GPIO as GPIO
# Imports sleep functon
from time import sleep
# Initializes virtual board (comment for hardware deploy)
import virtualboard

# Set up Rpi.GPIO library to use physical pin numbers
GPIO.setmode(GPIO.BOARD)
# Set up pins 36, 38, 40 and 37 as output and default them to low
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)

def bcd7(num):
  """Converts num to a BCD representation"""
  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 )

# Request a number and send it to the display
flag = True
while flag:
  try:
```

```
    num = int(input("Enter int between 0 and 15: "))
    bcd7(num)
  except:
    flag = False

# Reset all ports to its default state (inputs)
GPIO.cleanup()
```

Estudie el código y véalo en funcionamiento, ejecutándolo de la siguiente manera:

```
./bcd.py
```

4. Programas

Genere un conjunto de 5 programas que resuelvan los siguientes problemas.

1. [1pt] Modifique el código de la subsección 3.2 para todos los leds de la fila parpadeen. Nombre el archivo de código fuente `blink8.py`
2. [1pt] Modifique el código de las subsecciones 3.2 y 3.3 para que los leds de la fila enciendan de manera continua en una marquesina de izquierda a derecha. Nombre el archivo de código fuente `marquee.py`
3. [2pt] Modifique el código de la subsección 3.2 para que ocho leds parpadeen en el simulador en línea `https://create.withcode.uk/python/A3`. Nombre el archivo de código fuente `osblink8.py` Proporcione un video (captura de pantalla) con nombre `osblink8.mp4`, sin audio y con *codec* h.264 a 15fps como evidencia.
4. [2pt] Modifique el código de las subsecciones 3.2 y 3.3 para que 8 leds en la misma fila del simulador en línea el simulador en línea `https://create.withcode.uk/python/A3` enciendan de manera continua en una marquesina de izquierda a derecha. Nombre el archivo de código fuente `osmarquee.py` Proporcione un video (captura de pantalla) con nombre `osmarquee.mp4`, sin audio y con *codec* h.264 a 15fps como evidencia.
5. [2pt] Modifique el código de las subsecciones 3.2 a 3.4, para que los últimos 4 leds de la derecha muestren el código BCD enviado al display de 7 segmentos. Nombre el archivo de código fuente `bcd.py`

5. Especificaciones técnicas de los programas

- No utilice paquetes adicionales.
- El código deberá ser ejecutable con Python versión 3.5 o posterior.
- Todos los programas deberán comenzar con la línea de intérprete o *she-bang* correspondiente
- Todos los programas deberán tener el nombre del autor de la forma:

```
# Author: Nombre del Alumno
```

- Incluya sólo los videos, el cuestionario, y el código fuente de los programas ***sin librerías ni paquetes***.
- Los archivos de código python deberán estar en raíz `./`.
- Los videos-evidencia deberán estar en el subdirectorio `./vid/`.
- Los videos-evidencia deberán durar no más de 60 segundos, incluir sólo la ventana del simulador y contar únicamente con *stream* de video comprimido con *codec* h.264 a $15 fps$ con una resolución máxima de 1280×720 y con un tamaño máximo de 3MB por archivo (velocidad de datos aproximada de $1500 kbps$)[2].
- Anexe el cuestionario en formato *pdf* bajo `./doc/` como `questionnaire.pdf`.
- Los nombres de todos los archivos proporcionados deberán llevar un prefijo p de 5 caracteres, tales que:
  - $p[0]$ Primera letra del apellido paterno del alumno excluyendo preposiciones.
  - $p[1]$ Segunda letra del apellido paterno del alumno excluyendo preposiciones.
  - $p[2]$ Primera letra del apellido materno del alumno excluyendo preposiciones.

[2] `ffmpeg -i input -an -vf scale=-1:720 -c:v libx264 -crf 28 -r 15 -preset veryslow hicm_osblink8.mp4`

- $p[3]$ Primera letra del primer nombre del alumno o X en caso de nombres compuestos principiantes por J, J., José, Ma., Ma o María.
- $p[4]$ Guión bajo (caracter ASCII 95).
- Toda Ñ se reemplaza con una X

Es decir, las primeras 4 letras de su RFC o CURP seguidas de un guión bajo. Así, el cuestionario de Miguel Hidalgo y Costilla quedaría almacenado como `./doc/hicm_questionnaire.pdf`.

- El conjunto de programas, videos y el questionario deberá estar empaquetado en un archivo comprimido de nombre `[prefijo]_p01`, por ejemplo `hicm_p01.zip`. Los formatos aceptables son *7z*, *rar*, *tar.bz2*, *tar.gz* y *zip*.

6. Cuestionario

1. [2pt] Investigue el efecto que se produciría al regular el tiempo de encendido de un LED mediante la modulación del ciclo de trabajo del PWM en alta frecuencia (ej. 1kHz) y explique por qué no es posible observar este efecto en los simuladores.