Práctica 8:
Desplegado de temperatura en un display digital usando la Raspberry Pi

Fundamentos de Sistemas Embebidos

Autor: José Mauricio Matamoros de Maria y Campos

1. Objetivo

El alumno aprenderá a desplegar información proveniente de un sensor 1-Wire en una pantalla de cristal líquido de 32 caracteres organizados en dos filas.

2. Introducción

La presente práctica resume los pasos a seguir para desplegar texto en una pantalla de cristal líquido de 16 caracteres en dos líneas, mejor conocido como display LCD 16×2. En particular, se interesa en el desplegado de una cadena de más de 16 caracteres en la primera línea del display y en la segunda la temperatura registrada por un sensor DS18B20 mediante el bus 1-Wire.

2.1. Bus 1-Wire

1-Wire es un protocolo serial inventado por Dallas Semiconductor y diseñado para conectar dispositivos de muy baja velocidad mediante una interfaz de un sólo hilo (Figura 2) para transmisión datos. El bus 1-Wire es popular en meteorología (mediciones de temperatura, humedad y presión) debido a su facilidad de uso, fácil configuración y largo alcance (hasta 500 mestros) [1, 2].

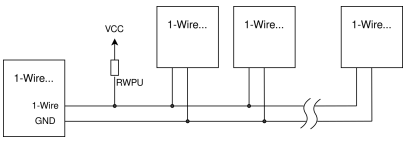

Figura 1: Bus 1-Wire con esclavos parásitos

La transferencia de datos es serial asíncrona y transmite paquetes de 8 bits con velocidades de hasta 16.3 kbit/s y un voltaje variable entre 2.8V y 5.25V. El dispositivo maestro, llamado MicroLAN, negocia la velocidad con los dispositivos esclavos y coordina la transmisión de datos. Cada dispositivo esclavo es identificado mediante un paquete de 64 bits único y definido por el fabricante, donde los 8 bits más significativos especifican la familia del producto, es decir su tipo y función. Además, la baja velocidad de operación del bus permite que el bus opere en modo *parásito* con tan sólo dos hilos: datos y tierra. Esto se logra mediante la inclusión de un capacitor de 800pF que almacena energía cuando el bus de datos está activo [1, 2].

Al ser un sensor completamente digital con un protocolo de transmisión de datos predefinido la lectura de datos del bus 1-Wire requiere de un controlador, mismo que viene integrado en la Raspberry Pi. Una vez habilitado dicho controlador, éste enumerará todos los dispositivos conectados al bus 1-Wire, mismos que serán visibles bajo `/sys/bus/w1/devices`.

2.2. Bus I^{2}C

I^{2}C es un protocolo serial inventado por Phillips y diseñado para conectar dispositivos de baja velocidad mediante interfaces de dos hilos (Figura 2). El protocolo permite un número virtualmente ilimitado de dispositivos interconectados donde más de uno puede ser un dispositivo maestro. El bus I2C es popular debido a su facilidad de uso y fácil configuración. Sólo es necesario

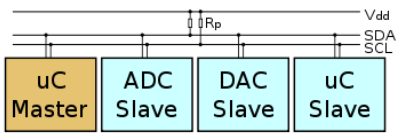

Figura 2: Bus I^{2}C

definir la velocidad máxima del bus, que está conformado por dos cables con resistencias pull-up [3].

I^2C utiliza solamente dos cables: SCL (reloj) y SDA (datos). La transferencia de datos es serial y transmite paquetes de 8 bits con velocidades de hasta 5MHz. Además, es requisito que cada dispositivo esclavo tenga una dirección de 7 bits que (el bit más significativo se utiliza para indicar si el paquete es una lectura o una escritura) debe ser única en el bus. Los dispositivos maestros no necesitan dirección ya que estos generan la señal de reloj y coordinan a los dispositivos esclavos [3].

2.3. LCD 1602 I^2C

El LCD 1602 es un display de cristal líquido de 32 caracteres organizados en 2 líneas de 16 caracteres con iluminación trasera que cuenta con 8 pines de datos que pueden ser usados en modo de byte completo o por *nibbles* (dos segmentos de 4 bits).

La inicialización del display usando los pines de datos no es tarea trivial. Por facilidad, es común acoplar el display a un adaptador de puerto paralelo cuasi bidireccional con interfaz I^2C: el integrado PCF8574. Este integrado se encarga de la escritura de los datos al display LCD 16×02, permitiendo su uso mediante I^2C.

IMPORTANTE: Normalmente el PCF8574 está preconfigurado con las direcciones I^2C $0 \times 3F$ o 0×27. Es crucial identificar correctamente la dirección I^2C del adaptador o el display no funcionará correctamente.

2.3.1. Comunicación

La comunicación con el LCD 1602 se realiza mediante la lectura y escritura de la memoria del display o el envío de comandos. Como el display está conectado al PCF8574 que sólo tiene 8 pines, el display no puede ser operado en modo de byte completo pues no habría manera de enviar las 3 señales de control que el display requiere para operar: $\overline{\texttt{ENA}}$, $\texttt{R}\overline{\texttt{W}}$ y $\texttt{R}\overline{\texttt{S}}$. Las señales de control operan como sigue:

Tabla 1: Señales de control del LCD 1601

Señal	Descripción
$\overline{\texttt{ENA}}$	Habilita el display. Poner esta línea en alto pone al LCD 1602 en modo de bajo consumo.
$\texttt{R}\overline{\texttt{W}}$	Establece que la operación es una lectura (alto) o una escritura (bajo) de la memoria del display.
$\texttt{R}\overline{\texttt{S}}$	Establece si la información enviada debe ser interpretada como datos (alto) o como un comando para el display (bajo).

Al haber sólo 4 bits disponibles, el envío de datos y de comandos requerirá de dos operaciones de escritura en el bus I^2C: el primero para la parte alta o nibble superior y el segundo para la parte baja o nibble inferior.

Los Algoritmos 1 y 2 resumen el envío de datos y comandos. Ambos algoritmos son, escencialmente, idénticos, con la única diferencia de un cambio de valor en la bandera $\texttt{R}\overline{\texttt{S}}$.

Algorithm 1 Envío de commandos al LCD 1601

procedure SENDCOMMAND(cmd)
- buffer ← cmd & $0 \times f0$ ▷ Elimina el nibble inferior
- buffer ← buffer | 0×04 ▷ Banderas $\overline{\text{ENA}}$= 1, R$\overline{\text{W}}$= 0, R$\overline{\text{S}}$= 0
- SENDWORD(LCD_ADDR, buffer) ▷ Envía bits 7–4
- SLEEP(2ms) ▷ Tiempo de espera del display
- buffer ← buffer & $0 \times FB$ ▷ Cambia bandera $\overline{\text{ENA}}$= 0
- SENDWORD(LCD_ADDR, buffer) ▷ Envía confirmación de fin de nibble

- buffer ← (cmd & $0 \times f0$) $\ll 4$ ▷ Elimina el nibble superior y recorre el inferior a la parte alta
- buffer ← buffer | 0×04 ▷ Banderas $\overline{\text{ENA}}$= 1, R$\overline{\text{W}}$= 0, R$\overline{\text{S}}$= 0
- SENDWORD(LCD_ADDR, buffer) ▷ Envía bits 3–0
- SLEEP(2ms) ▷ Tiempo de espera del display
- buffer ← buffer & $0 \times FB$ ▷ Cambia bandera $\overline{\text{ENA}}$= 0
- SENDWORD(LCD_ADDR, buffer) ▷ Envía confirmación de fin de nibble

end procedure

Algorithm 2 Envío de datos al LCD 1601

procedure SENDDATA(data)
- buffer ← data & $0 \times f0$ ▷ Elimina el nibble inferior
- buffer ← buffer | 0×05 ▷ Banderas $\overline{\text{ENA}}$= 1, R$\overline{\text{W}}$= 0, R$\overline{\text{S}}$= 0
- SENDWORD(LCD_ADDR, buffer) ▷ Envía bits 7–4
- SLEEP(2ms) ▷ Tiempo de espera del display
- buffer ← buffer & $0 \times FB$ ▷ Cambia bandera $\overline{\text{ENA}}$= 0
- SENDWORD(LCD_ADDR, buffer) ▷ Envía confirmación de fin de nibble

- buffer ← (data & $0 \times f0$) $\ll 4$ ▷ Elimina el nibble superior y recorre el inferior a la parte alta
- buffer ← buffer | 0×05 ▷ Banderas $\overline{\text{ENA}}$= 1, R$\overline{\text{W}}$= 0, R$\overline{\text{S}}$= 0
- SENDWORD(LCD_ADDR, buffer) ▷ Envía bits 3–0
- SLEEP(2ms) ▷ Tiempo de espera del display
- buffer ← buffer & $0 \times FB$ ▷ Cambia bandera $\overline{\text{ENA}}$= 0
- SENDWORD(LCD_ADDR, buffer) ▷ Envía confirmación de fin de nibble

end procedure

La función SENDWORD se presenta a continuación en lenguage Python. Esta dependerá de la implementación subyaciente de `smbus2`.

```python
import smbus2
BUS = smbus2.SMBus(1)
def send_word(addr, data):
  global BLEN
  temp = data
  if BLEN == 1:
    temp |= 0x08
  else:
    temp &= 0xF7
  BUS.write_byte(addr ,temp)
```

2.3.2. Inicialización

La rutina de inicialización del LCD 1602 es la siguiente:

Algorithm 3 Inicialización del LCD 1601

procedure InitLCD
  SendCommand(0×33) ▷ Inicializar display
  Sleep(5ms)
  SendCommand(0×32) ▷ Cambiar a modo de 4 líneas
  Sleep(5ms)
  SendCommand(0×28) ▷ Configurar modo: 2 líneas y caracteres de 35 puntos
  Sleep(5ms)
  SendCommand($0 \times 0C$) ▷ Habilitar display y ocultar cursor
  Sleep(5ms)
  SendCommand(0×01) ▷ Borrar pantalla
  Sleep(5ms)
end procedure

Nótese la espera activa de 5ms después de cada comando. Esta espera es para dar tiempo al display de cambiar de modo, ya que la inicialización es un proceso lento.

2.3.3. Comandos

Los comandos del LCD 1602 son los siguientes.

Comando	Descripción
0×01	Borrar pantalla
0×08	Activar luz posterior
$0 \times 0C$	Habilitar display sin cursor
0×33	Inicializar display (modo de 8 pines)
0×32	Cambio a modo de 4 pines
0×28	Configurar display con 2 líneas y caracteres de 5×7 puntos
$0 \times 80 + 0 \times 40 \times y + x$	Mover cursor a posición (x, y)

3. 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.

- 1 Display de cristal líquido $LCD - 16 \times 2$
- 1 sensor de temperatura DS18B20 en encapsulado TO-92[1]
- 1 adaptador I^2C a $LCD - 16 \times 2$ con módulo expansor PCF8574
- 2 resistencias de 1kΩ
- 1 resistencia de 10kΩ
- 1 diodo emisor de luz Led rojo
- 1 diodo emisor de luz Led verde
- 1 protoboard o circuito impreso equivalente
- 1 fuente de alimentación regulada a 5V y al menos 2 amperios de salida
- Cables y conectores varios

4. Instrucciones

1. Alambre el circuito descrito en la Subsección 4.1.
2. Siga los procedimientos de las Subsecciones 4.2 y 4.3
3. Realice los experimentos propuestos en la sección 5.

[1] En lugar del sensor DS18B20 es posible usar el alambrado de arduino con LM35 de la práctica 6.

Tabla 2: Conexiones I^{2}C entre la Raspberry Pi y el adaptador I^{2}C a $LCD-16\times2$

Pin Raspberry	Conexión
3 (GPIO2)	Raspberry Pi SDA → Adaptador SDA
5 (GPIO3)	Raspberry Pi SCL → Adaptador SCL
6 (GND)	Raspberry Pi GND → Adaptador GND
6 (GND)	Raspberry Pi VCC → Adaptador VCC

4.1. Paso 1: Alambrado

El proceso de alambrado de esta práctica es relativamente simple y considera dos circuitos: el sensor DS18B20 y el display $LCD-16\times2$. El primer circuito, mostrado en la Figura 3, permite leer directamente del sensor de temperatura DS18B20 mediante el bus 1-Wire. El segundo circuito (Figura 4) consiste en la interfaz de conexión entre el $LCD-16\times2$ y el bus I^{2}C vía el módulo expansor de 8 bits a I^{2}C el PCF8574.

Alambre primero el circuito del sensor de temperatura DS18B20 mostrado en la Figura 3 consistente en el integrado DS18B20, las dos resistencias de 1kΩ, los diodos emisores de luz y la resistencia de pull-up de 10kΩ. Verifique que el diodo emisor de luz entre Vcc y tierra sea el Led **ROJO**, y que el diodo emisor de luz entre Vcc y Sig sea el Led **VERDE**. Finalmente, conecte los cables de Vcc y Gnd a la Raspberry Pi, así como el cable de datos Sig al pin número 7 (Gpio4).

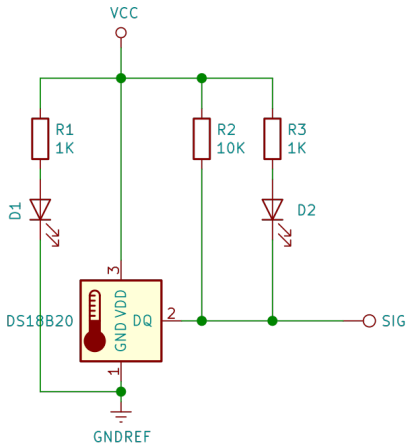

Figura 3: Alambrado del sensor de temperatura 1-Wire DS18B20

Tras alambrar el primer circuito realice el experimento prueba indicado en la Subsección 4.2.

A continuación ensamble el display al adaptador I^{2}C a $LCD-16\times2$ y conecte este último al bus I^{2}C de la Raspberry Pi como ilustran la Tabla 2 y la Figura 4.

Hay tutoriales que sugieren utilizar un convertidor de niveles de voltaje cuando se conecta una Raspberry Pi a un arduino mediante I^{2}C, especialmente cuando la Raspberry Pi opera a 3.3V. Esto **NO** es necesario cuando la Raspberry Pi está configurada como dispositivo maestro o *master*.

Esto es posible debido a que el adaptador no tiene resistencias de acoplamiento a positivo o *pull-up* integradas, mientras que los pines I^{2}C de la Raspberry Pi están conectados internamente a la línea de 3.3V mediante resistencias de 1.8kΩ. Por este motivo, tendrán que quitarse las resistencias de *pull-up* a cualquier otro dispositivo esclavo que se conecte al bus I^{2}C de la Raspberry Pi.[2]

4.2. Paso 2: Lectura del sensor DS18B20

Para poder leer el sensor de temperatura DS18B20 es necesario habilitar la carga del controlador 1-Wire en el kernel. Para esto edite el archivo `/boot/config.txt` como superusuario y agregue al final la línea

```
dtoverlay=w1-gpio
```

A continuación reinicie la Raspberry Pi.

Cuando la Pi termine de iniciar, verifique que el controlador está habilitado con los siguientes comandos

```
# modprobe w1-gpio
# modprobe w1-therm
# ls sys/bus/w1/devices
```

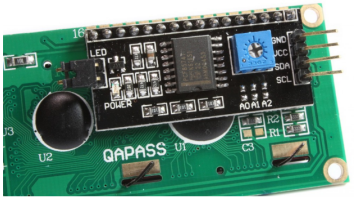

Figura 4: Adaptador I^{2}C acoplado a display[3]

[2]Para más información sobre el papel de las resistencias de acoplamiento a positivo o *pull-up* en un bus I^{2}C se puede consultar http://dsscircuits.com/articles/effects-of-varying-i2c-pull-up-resistors

[3]Imagen obtenida de https://alselectro.wordpress.com/2016/05/12/serial-lcd-i2c-module-pcf8574/

El resultado debería ser muy parecido al siguiente

```
28-00000495db35 w1_bus_master1
```

donde `28-00000495db35` es el número de serie del sensor de temperatura.

Si se montó correctamente, las funciones del sensor DS18B20 serán accessibles como archivos en el directorio mostrado por el comando `ls`, a saber:

```
root@raspberrypi:/sys/bus/w1/devices/28-00000495db35# ls
driver         id              name            power
subsystem      uevent          w1_slave
```

La lectura del sensor (temperatura) se realiza leyendo el archivo `w1_slave` en modo texto, por ejemplo con `cat`:

```
root@raspberrypi:/sys/bus/w1/devices/28-00000495db35# cat w1_slave
a3 01 4b 46 7f ff 0d 10 ce : crc=ce YES
a3 01 4b 46 7f ff 0d 10 ce t=26187
```

donde `t=26187` es la temperatura en m°C. Es decir es necesario dividir el valor `t` entre 1000 para obtener la temperatura en °C:

$$\frac{26187 m^oC}{1000} = 26.187^oC$$

4.3. Paso 3: Configuración de comunicaciones I²C

Primero ha de configurarse la Raspberry Pi para funcionar como dispositivo maestro o *master* en el bus I²C. Para esto, inicie la utilidad de configuración de la Raspberry Pi con el comando

```
# raspi-config
```

y seleccione la opción 5: Opciones de Interfaz (*Interfacing Options*) y active la opción `P5` para habilitar el I²C.

A continuación, verifique que el puerto I²C no se encuentre en la lista negra. Edite el archivo `/etc/modprobe.d/raspi-blacklist.conf` y revise que la línea `blacklist spi-bcm2708` esté comentada con #.

Código ejemplo 1: /etc/modprobe.d/raspi-blacklist.conf

```
# blacklist spi and i2c by default (many users don't need them)
# blacklist i2c-bcm2708
```

Como paso siguiente, se habilita la carga del driver I²C. Esto se logra agregando la línea `i2c-dev` al final del archivo `/etc/modules` si esta no se encuentra ya allí.

Por último, se instalan los paquetes que permiten la comunicación mediante el bus I²C y se habilita al usuario predeterminado *pi* (o cualquier otro que se esté usando) para acceder al recurso.

```
# apt-get install i2c-tools python3-smbus
# adduser pi i2c
$ pip install smbus2
```

Reinicie la Raspberry Pi y pruebe la configuración ejecutando `i2cdetect -y 1` para buscar dispositivos conectados al bus I²C. Debería ver una salida como la siguiente:

```
$ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

5. Experimentos

1. [2pt] Realice un programa en Python que despliegue en consola la temperatura sensada por el DS18B20 cada segundo.
2. [6pt] Con base en las especificaciones de la subsección 2.3 realice un programa en Python que imprima en la primera línea del display el apellido paterno de uno de los integrantes del equipo de trabajo.
3. [2pt] Modifique el programa anterior para que el display muestre en la segunda línea del display la temperatura en grados centígrados registrada por el DS18B20, actualizada cada segundo.
4. [+2pt] Modifique el programa del punto 3 para que el display muestre la temperatura tanto en grados Celcius como en Farenheit.
5. [+3pt] Modifique el programa del punto 3 para que el display muestre en la primera línea del display el apellido paterno de cada integrante del equipo de trabajo, separados por espacio, como un corrimiento infinito de marquesina izquierda además de la temperatura en la segunda línea.
6. [+5pt] Con base en lo aprendido, modifique el programa de los puntos 3 a 5 para que la Raspberry Pi sirva una página web donde se pueda variar la velocidad y dirección de la marquesina, además de seleccionar si la temperatura se muestra en escala centígrada, Farenheit o ambas.
7. [+5pt] Con base en lo aprendido, modifique el programa del punto 3 para que la Raspberry Pi sirva una página web donde se pueda observar la gráfica de temperatura (histórico) desde la bitácora con resolución de hasta 1 minuto.

6. Referencias

Referencias

[1] Linke, Bernhard. Overview of 1-wire technology and its use, 2008. https://www.analog.com/media/en/technical-documentation/tech-articles/guide-to-1wire-communication--maxim-integrated.pdf, Last accessed on 2023-12-01.

[2] Ludmila Maceková. 1-wire-the technology for sensor networks. *Acta Electrotechnica et Informatica*, 12(4):52, 2012.

[3] I2C Info: A Two-wire Serial Protocol. I2c info – i2c bus, interface and protocol, 2020. https://i2c.info/, Last accessed on 2020-03-01.

[4] *LM35 Precision Centigrade Temperature Sensors*. Texas Instruments, August 1999. Revised: December, 2017.

[5] The Arduino Project. Introduction to the arduino board, 2020. https://www.arduino.cc/en/reference/board, Last accessed on 2020-03-01.