Microcontroladores II

Juan Esteban Giraldo Hoyos
Ingeniero Electrónico
Magíster en Gestión de Ciencia, Tecnología e Innovación

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Timers

Timer	Resolución	Modos de Operación	Modos de Interrupción	Rango de Temporización	Uso Común
Timer0	8 bits	Normal, CTC	Overflow	0 a 256 ciclos	Control de PWM, temporización básica
Timer1	16 bits	Normal, CTC, PWM	Overflow, Compare A/B	0 a 65536 ciclos	Control de PWM, temporización precisa
Timer2	8 bits	Normal, CTC, PWM	Overflow, Compare A/B	0 a 256 ciclos	Control de PWM, temporización básica

Microcontroladores II - Timers

Cuando se trata de interrupciones por temporizador para tiempos prolongados, Timer1 es generalmente la mejor opción debido a su resolución de 16 bits, lo que le permite temporizar durante un rango más amplio sin perder precisión. Sin embargo, si la aplicación no requiere una precisión extrema y se puede ajustar dentro del rango de Timer0 o Timer2, también pueden ser opciones viables.

Es importante tener en cuenta los límites de tiempo al usar temporizadores en Arduino Uno. Los límites están determinados por la frecuencia del reloj del microcontrolador y la resolución del temporizador. Para calcular el tiempo máximo, puedes usar la fórmula:

$$\text{Tiempo Máximo} = \frac{2^n}{\text{Frecuencia del Reloj}}$$

Donde *n* es el número de bits del temporizador. Por ejemplo, para Timer1 con una frecuencia de reloj de 16 MHz:

$$\text{Tiempo Máximo} = \frac{2^{16}}{16 \times 10^6} \approx 4.1 \text{ segundos}$$

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Modos de bajo consumo

Wake-up Sources	Idle	ADC Noise Reduction	Power-save	Standby	Power-down
INT1, INT0 and Pin Change	X	X	X	X	X
TWI Address Match	X	X	X	X	X
Timer2	X	X	X		
SPM/EEPROM Ready	X	X			
A/D Converter	X	X			
Watchdog Timer	X	X	X	X	X
Other I/O	X				

Dependiendo del modo Sleep se tiene la posibilidad de despertar el microcontrolador de diferentes fuentes

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Hoy trabajaremos

com x

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

Requerimientos:

1) Tener un editor de código instalado: Visual Code (Recomendado)

   https://code.visualstudio.com/Download

2) Instalar python en sus computadoras: Verificar usando comando:

   ```
   python --version
   ```

   https://www.python.org/downloads/

3) Instalar las siguientes librerías de python: `pip install tk pyserial`

   - pyserial (Manejar la comunicación serial)
   - tkinter (Manejar la interfaz gráfica)

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

Pasos:

1) Abrir Visual Code y empezar a editar un nuevo archivo
2) Agregar las librerías de python

```
import tkinter as tk
from tkinter import ttk
import serial
from serial.tools import list_ports
```

3) Se crea la ventana principal de la interfaz gráfica, se especifica el nombre de la ventana, en tamaño y el color de fondo.

```
# Crea la ventana de la interfaz grafica
ventana = tk.Tk(className=" LEDs")
ventana.geometry("305x350")
ventana.config(bg="#405158")
```

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

Pasos:

4) Colocamos una etiqueta como título de la ventana.

```
# Etiqueta para mostrar el titulo del proyecto
etiqueta = tk.Label(ventana, text="Encender LEDs", font=("Arial", 14), bg="#5C6B72", fg="white")
etiqueta.pack(side=tk.TOP, fill="both")
```

5) Se coloca un «ComboBox» para mostrar los puertos COM conectados a la computadora.

```
# Crear ComboBox para mostrar los puertos COM disponibles
combo_puertos = ttk.Combobox(ventana, state='readonly')
combo_puertos.place(x=5, y=35)
```

6) Se crea un botón para actualizar los puertos COM conectados, ejecutando la función «mostrar_puertos_com».

```
# Botón para actualizar la lista de puertos COM disponibles
btn_actualizar = tk.Button(ventana, text="Actualizar", command=mostrar_puertos_com)
btn_actualizar.place(x=235, y=30)
```

Microcontroladores II - Interfaz gráfica usando python

Pasos:

7) Se crea un botón para conectarse o desconectarse del puerto COM seleccionado en el «ComboBox», ejecutando la función «conectar_o_desconectar».

```
# Botón para conectarse al puerto COM seleccionado
btn_conectar = tk.Button(ventana, text="Conectar", command=conectar_o_desconectar)
btn_conectar.place(x=155, y=30)
```

8) La función «obtener_puerto_com» obtiene la lista de los puertos COM conectados a la computadora.

```
def obtener_puertos_com():
    puertos_com = [port.device for port in list_ports.comports()]
    return puertos_com
```

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

Pasos:

9) La función «mostrar_puertos_com», obtienen los puertos COM de la función «obtener_puerto_com» y los muestra en el «ComboBox» de la ventana.

```
def mostrar_puertos_com():
    combo_puertos['values'] = obtener_puertos_com()
    combo_puertos.set("")
```

10) La función «conectar_puerto» se conecta al puerto COM seleccionado en el «ComboBox», y retorna la comunicación serial, si no se establece la conexión, retorna «None».

```
def conectar_puerto(puerto):
    try:
        conectar = serial.Serial(puerto, baudrate=9600, stopbits=1, parity='N', bytesize=8)
        return conectar
    except serial.SerialException as e:
        return None
```

Microcontroladores II - Interfaz gráfica usando python

Pasos:

11) La función «desconectar_puerto» se desconecta al puerto COM seleccionado en el «ComboBox».

```
def desconectar_puerto(conectar):
    if conectar:
        conectar.close()
```

12) La función «conectar_o_desconectar», simplemente se conecta o desconecta del puerto COM seleccionado ejecutando las diferentes funciones. Si no existe comunicación serial con un puerto «COM», la computadora se conectará al puerto COM seleccionado y el texto del botón «btn_conectar» se cambiará a «desconectar», si existe una comunicación serial con un puerto COM, la computadora se desconectara del puerto COM y el texto del botón «btn_conectar» se cambiará a «conectar».

```
def conectar_o_desconectar():
    global conectar
    if conectar is None:
        puerto_seleccionado = combo_puertos.get()
        conectar = conectar_puerto(puerto_seleccionado)
        if conectar:
            btn_conectar.config(text="Desconectar")
    else:
        desconectar_puerto(conectar)
        conectar = None
        btn_conectar.config(text="Conectar")
```

Microcontroladores II - Interfaz gráfica usando python

Pasos: Una vez colocado el código correspondiente a la comunicación serial entre la computadora y el microcontrolador, se realiza el código, en este caso para enviar diferentes comandos al microcontrolador para encender o a pagar diferentes LEDs mediante botones de encendido y apagado.

13) Inicialmente se coloca una etiqueta para identificar el LED.

```
# Etiqueta para mostrar led1
etiqueta_led1 = tk.Label(ventana, text="LED1", font=("Arial", 20), padx=1, pady=10, bg="#49646F", fg="white")
etiqueta_led1.place(x=10, y=80)
```

14) Se crea un botón de encendido y un botón de apagado para enviar el comando correspondiente para encender o apagar el LED, cada botón ejecuta una función que enviará el comando por el puerto COM seleccionado «command=led1_on».

```
# Botón para encender el led1
boton1_ON = tk.Button(ventana, text="ON", font=("Arial", 24), padx=1, pady=1, bg="green4", fg="white",
                      activebackground="green2", activeforeground="black", command=led1_on)
boton1_ON.place(x=110, y=80)

# Botón para apagar el led1
boton1_OFF = tk.Button(ventana, text="OFF", font=("Arial", 24), padx=1, pady=1, bg="red4", fg="white",
                       activebackground="red2", activeforeground="black", command=led1_off)
boton1_OFF.place(x=200, y=80)
```

IUSH Institución Universitaria SALAZAR Y HERRERA
ABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

Pasos:

15) Se crean las funciones para enviar el comando por el puerto COM seleccionado al microcontrolador cuando se pulsa el botón correspondiente, «conectar.write()» indica que enviará los datos por la comunicación serial establecida, se coloca «b» que convierte los datos en Bytes y entre comillas «datos» se colocan los datos que se desean enviar, en este caso se envía el comando seguido del retorno de carro «\r» y el salto de línea «\n».

```
# Envia el comando por el puerto seleccionado
def led1_on():
    conectar.write(b"LED1_ON\n")

def led1_off():
    conectar.write(b"LED1_OFF\n")
```

Microcontroladores II - Interfaz gráfica usando python

Pasos:

16) Ir al Campus y completar el código del proyecto usando el código de ejemplo entregado en el material de la semana 2

17) Guardamos el programa con un nombre como: “control_led.py”

18) Vamos a la ubicación del archivo y ejecutamos python control_led.py

Microcontroladores II - Firmware Arduino

control_leds_serial

```
void setup() {

  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
}

void loop() {
    if (Serial.available() > 0) {
        String command = Serial.readStringUntil('\n');

        if(command == "LED1_ON") {
            digitalWrite(LED_BUILTIN,HIGH);
        }else if(command == "LED1_OFF") {
            digitalWrite(LED_BUILTIN,LOW);
        }
    }
}
```

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

PRÁCTICA #2 10%

Implementar un firmware que permita controlar (BOTÓN OFF y ON) 3 leds de diferentes colores al recibir comandos por la comunicación serial con la interfaz creada en python corriendo en el PC. Tomar el firmware ejemplo y modificarlo para adaptarlo al requerimiento. Para el led 3 se debe agregar en la interfaz de usuario un método adicional que al presionar el botón envíe la orden pasados 5s al microcontrolador ya sea al dar la orden de encender o al apagar. (Ver https://www.youtube.com/watch?v=fRncEqTy_Lw)

Modificar la interfaz creada de usuario para cambiar los colores y personalizar aspectos como el tamaño de la ventana, el nombre del programa, etc

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Bibliografía

- https://pythonbasics.org/tkinter-button/
- https://microchipotle.com/interfaz-grafica-en-python-com-usb-y-rs232-pic-c-compiler/