Práctica 1: Salida Digital

LED Parpadeante y Simulación de Semáforo

Objetivos de Aprendizaje

Al finalizar esta práctica de laboratorio, serás capaz de:

1. Establecer una conexión serial segura y gestionada por software con una placa Arduino UNO en Python utilizando la biblioteca `pyFirmata`.
2. Configurar pines digitales como salidas y manipular sus estados lógicos (ALTO/BAJO).
3. Aplicar retardos de tiempo y bucles básicos en Python para controlar actuadores físicos (LEDs).
4. Implementar un sistema de control secuencial por estados (Simulación de Semáforo) con procedimientos adecuados de inicialización y limpieza del hardware.

1 Introducción

Los sistemas de ingeniería modernos a menudo utilizan un lenguaje de programación de alto nivel como Python para algoritmos de control, toma de decisiones e interfaces de usuario, delegando la interacción de bajo nivel con el hardware (sensores y actuadores) a microcontroladores.

En esta práctica, utilizamos el **protocolo Firmata**, un protocolo de comunicación estándar para microcontroladores desde software en una computadora host. Al cargar el sketch `StandardFirmata` al Arduino UNO, convertimos el microcontrolador en un dispositivo de E/S (Entrada/Salida) que responde directamente a los comandos enviados por USB/Serial desde nuestro entorno de Python utilizando la biblioteca `pyFirmata`.

2 Requerimientos de Hardware

Reúna los siguientes componentes antes de comenzar el ensamblaje del circuito:

Descripción del Componente	Especificación	Cantidad
Placa Arduino UNO	Cable USB tipo A a B	1
Protoboard	Tamaño completo o medio	1
LED Rojo	Estándar de 5mm	1
LED Amarillo	Estándar de 5mm	1
LED Verde	Estándar de 5mm	1
Resistencia	220 Ω (0.25 W)	3
Cables Jumper	Macho a Macho	4

3 Diagrama de Circuito y Conexiones

Para proteger los LEDs de una corriente excesiva, se debe conectar una resistencia de limitación de corriente de 220 Ω en serie con el ánodo de cada LED (la pata más larga). Arme el circuito en su protoboard de acuerdo con la siguiente distribución:

Detalles de Conexión del Circuito

Color de LED	Pin de Ánodo (+)	Resistencia Limitadora	Pin de Cátodo (-)
LED Rojo	Pin Digital 13	220 Ω	GND del Arduino
LED Amarillo	Pin Digital 12	220 Ω	GND del Arduino
LED Verde	Pin Digital 11	220 Ω	GND del Arduino

Instrucciones de ensamble:

- Conecte un cable jumper negro desde uno de los pines **GND** del Arduino al riel de tierra común azul/menos (-) en la protoboard.
- Para cada LED, inserte el cátodo (pata corta, lado plano del encapsulado) directamente en el riel común de GND.
- Coloque el ánodo (pata más larga) en una fila de terminales independiente de la protoboard.
- Conecte una resistencia de 220 Ω desde la fila del ánodo a otra fila vacía.
- Conecte un cable jumper desde el otro extremo de la resistencia al pin digital de Arduino correspondiente (13, 12 o 11).

Nota Importante / Resolución de Problemas

Advertencia: Nunca conecte un LED directamente entre los pines digitales/5V y Tierra sin una resistencia. Hacerlo provocará un flujo excesivo de corriente, dañando permanentemente el LED y posiblemente destruyendo el puerto digital de E/S del Arduino.

4 Configuración del Entorno de Software

En esta sección se detalla el procedimiento para configurar las herramientas en sistemas operativos Windows.

4.1 Paso 1: Instalación y uso de VS Code (Visual Studio Code)

1. Abra **VS Code**. Si aún no lo ha hecho, cree una carpeta dedicada para sus prácticas en su computadora (por ejemplo, llamada `arduPyLab`).
2. En el menú superior de VS Code, vaya a **Archivo** → **Abrir carpeta...** (File → Open Folder) y seleccione la carpeta que creó.
3. En la barra lateral izquierda (Explorador de archivos), haga clic en el botón de **Nuevo archivo** (icono con forma de hoja y un signo +).

4. Cree dos archivos en esta carpeta con los siguientes nombres exactos:
   - `firmata_base.py` (Copie en él el código de la plantilla base).
   - `lab1_blink.py` (Copie en él el código de la Tarea 1).

4.2 Paso 2: Identificación del Puerto COM del Arduino en Windows

Para que su programa de Python sepa a qué puerto enviar los comandos, debe averiguar qué puerto COM ha asignado Windows a su Arduino:

1. Conecte la placa Arduino UNO a su computadora mediante el cable USB.
2. Haga clic derecho sobre el botón de inicio de Windows y seleccione **Administrador de dispositivos** (Device Manager).
3. Despliegue la sección llamada **Puertos (COM y LPT)**.
4. Busque un dispositivo llamado *Arduino Uno* o *Dispositivo serie USB* y anote el número entre paréntesis (por ejemplo, **COM3** o **COM4**).
5. **Importante:** Si su puerto asignado es diferente a COM3 (por ejemplo, COM5), recuerde modificar la línea del código de Python donde se define el puerto para que coincida (cambie `port='COM3'` por `port='COM5'`).

Nota Importante / Resolución de Problemas

Error común de conexión: Si tiene abierto el **Monitor Serie** del IDE de Arduino, este mantendrá bloqueado el puerto COM. Asegúrese de cerrarlo antes de intentar ejecutar su programa en Python; de lo contrario, la conexión en Python fallará.

4.3 Paso 3: Apertura y uso de la Terminal en VS Code

La terminal es una interfaz de texto que nos permite interactuar con el sistema operativo y ejecutar scripts:

1. En VS Code, abra la terminal integrada presionando la combinación de teclas **Ctrl** + ~ (o desde el menú superior: **Terminal** → **Nuevo Terminal**). Se abrirá un panel de texto en la parte inferior.
2. Verifique que Python y su gestor de paquetes (`pip`) estén listos ejecutando el siguiente comando en la terminal y presionando Enter:

```
python --version

```

   Si la terminal indica que no reconoce el comando, intente con:

```
py --version

```

3. Instale la biblioteca `pyFirmata2` escribiendo en la terminal:

```
    pip install pyfirmata2

```

Nota Importante / Resolución de Problemas

¡Atención - Conflicto de bibliotecas!: Si previamente instalaste la biblioteca obsoleta `pyfirmata`, esta causará un conflicto e impedirá el correcto funcionamiento de la nueva versión. Debes desinstalarla y reemplazarla ejecutando los siguientes comandos en la terminal de VS Code:

```
pip uninstall pyfirmata -y
pip install pyfirmata2
```

Asegúrese también de que el Arduino UNO tenga cargado el sketch `StandardFirmata` usando el IDE de Arduino (menú `Archivo` → `Ejemplos` → `Firmata` → `StandardFirmata`).

5 Tareas Experimentales

5.1 Tarea 1: Probar la Conexión y Parpadeo de un Solo LED

En esta tarea, creará un script de Python que importe el módulo base de conexión `firmata_base.py` y haga parpadear el LED Rojo (Pin 13) a una frecuencia de 1 Hz (0.5 segundos encendido, 0.5 segundos apagado).

A continuación se presenta la plantilla del código. Guarde este archivo como `lab1_blink.py`.

```
import time
from firmata_base import ArduinoConnection

# Asignación de pin objetivo
RED_PIN = 13

# Establecer conexión usando el administrador de contexto (puerto COM
    asignado)
with ArduinoConnection(port='COM3') as board:
    # Configurar el pin 13 como salida digital
    # Sintaxis: board.get_pin('d:numero_de_pin:modo')
    # d = digital, 13 = pin, o = output (salida)
    red_led = board.get_pin(f'd:{RED_PIN}:o')

    print("Haciendo parpadear el LED Rojo... Presiona Ctrl+C para detener.
    ")
    try:
        while True:
            # Encender el LED (alto lógico)
            red_led.write(1)
            time.sleep(0.5)

            # Apagar el LED (bajo lógico)
            red_led.write(0)
            time.sleep(0.5)

```

```
25      except KeyboardInterrupt:
26          # La limpieza segura es manejada por el administrador de contexto
    al salir
27          print("\nParpadeo detenido por el usuario.")
```

Listing 1: Parpadeo de un LED usando pyFirmata

5.2 Tarea 2: Simulador de Semáforo

Ahora, construirá una lógica de máquina de estados secuencial que simula un semáforo estándar. Amplíe el circuito para incluir el LED Amarillo (Pin 12) y el LED Verde (Pin 11). El programa debe recorrer los estados que se muestran a continuación:

Estado	LED Activo	Pin de Arduino Activo	Duración (Segundos)
1	Verde	Pin 11	5
2	Amarillo	Pin 12	2
3	Rojo	Pin 13	5

5.2.1. Requisitos Clave:

- Al realizar la transición de un estado a otro, asegúrese de apagar explícitamente todos los demás LEDs (es decir, escribiendo un 0 lógico).
- El script debe manejar interrupciones de teclado (`KeyboardInterrupt` mediante Ctrl+C) para limpiar y garantizar que **todos los LEDs se apaguen** antes de salir, evitando que los LEDs se queden encendidos en un estado activo.

6 Preguntas y Análisis

Proporcione respuestas claras y escritas en computadora a las siguientes preguntas en su reporte de práctica:

1. Explique la función de la clase `Iterator` de `pyfirmata.util` dentro de `firmata_base.py`. ¿Qué sucede con la lectura del búfer de datos seriales si el iterador no se está ejecutando?
2. ¿Cuál es el propósito de la instrucción `board.exit()` al cerrar el programa? ¿Por qué es importante realizar una etapa de limpieza en aplicaciones prácticas de ingeniería?
3. Modifique el bucle del semáforo para agregar un estado intermedio: un estado de “Cruce Peatonal” donde el LED Rojo parpadee rápidamente (0.2 segundos encendido/apagado) durante 3 segundos antes de regresar al estado Verde. Escriba el fragmento de bucle en Python modificado a continuación.