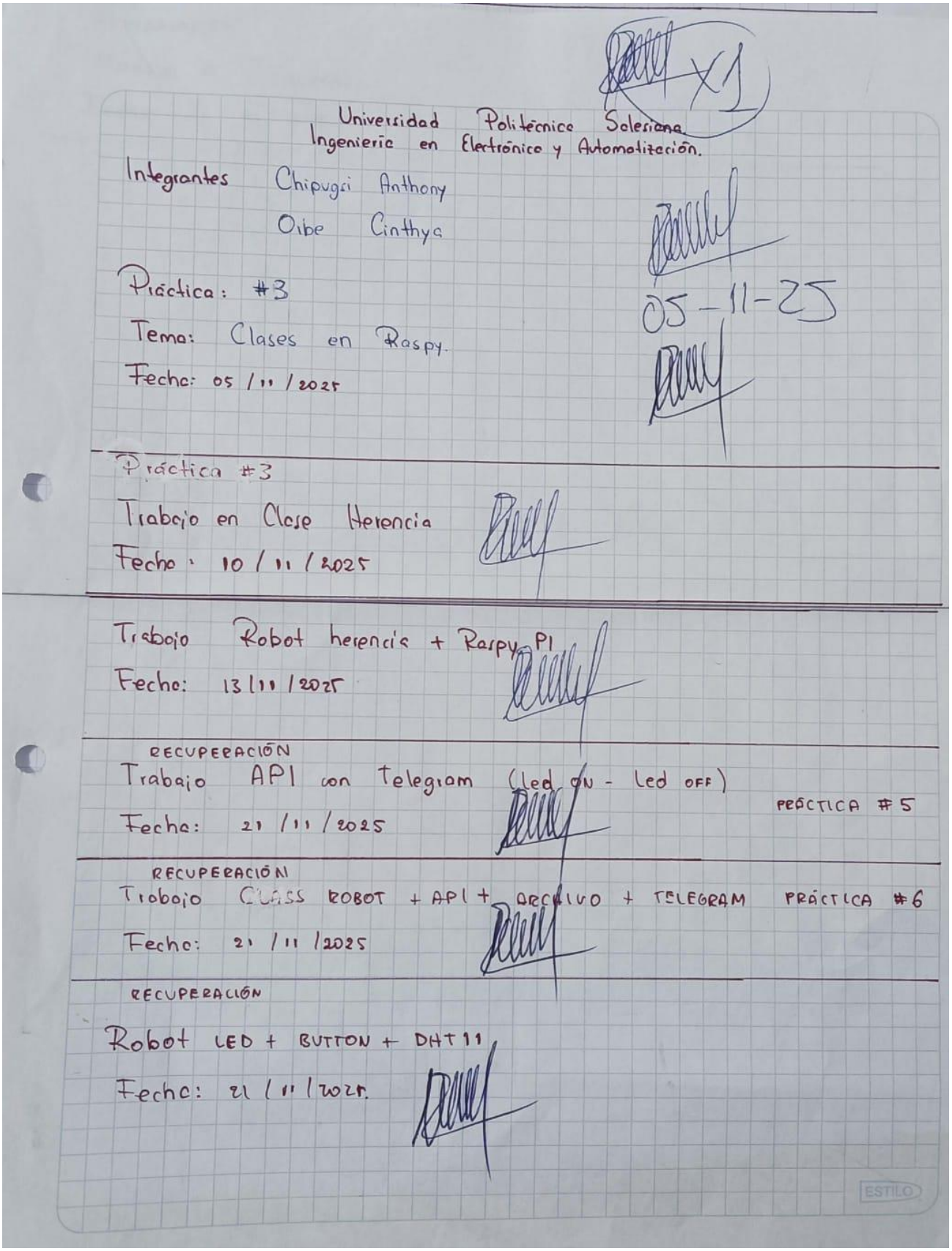
Fecha: 19/10/2025



x1

Universidad Politécnica Salesiana.
Ingeniería en Electrónico y Automatización.

Integrantes Chipugsi Anthony
Oibe Cinthya

Práctica: #3
Tema: Clases en Raspy.
Fecha: 05/11/2025

05-11-25

Práctica #3
Trabajo en Clase Herencia
Fecha: 10/11/2025

Trabajo Robot herencia + Raspy PI
Fecha: 13/11/2025

RECUPERACIÓN
Trabajo API con Telegram (Led ON - Led OFF)
Fecha: 21/11/2025
PRÁCTICA #5

RECUPERACIÓN
Trabajo CLASS ROBOT + API + ARCHIVO + TELEGRAM PRÁCTICA #6
Fecha: 21/11/2025

RECUPERACIÓN
Robot LED + BUTTON + DHT11
Fecha: 21/11/2025

Fecha: 19/10/2025

Manejo de Archivos en POO

Práctica 6

Anthony Fabricio Chipugsi Pilicita
achipugsip@est.ups.edu.ec
Cinthya Aracelly Orbe Muñoz
corbem@est.ups.edu.ec

I. Introducción

En sistemas IoT, el registro de eventos y datos es esencial para auditoría, análisis y trazabilidad. La persistencia de información permite reconstruir el comportamiento del sistema, detectar fallos y generar reportes. Esta práctica se enfoca en implementar un sistema de registro de acciones mediante archivos .txt, utilizando Programación Orientada a Objetos (POO) en Python. Se integró el control de un LED, un botón físico y un sensor DHT11, registrando cada evento en tiempo real dentro de un archivo de texto.

II. Metodología

- Configuración **del entorno**
  - Se utilizó una Raspberry Pi con Python 3 instalado.
  - Se conectó un LED al pin **GPIO 18**, un botón al pin **GPIO 23**, y el sensor DHT11 al pin **GPIO 4** [1].
  - Se instalaron las librerías necesarias:

    pip install adafruit-circuitpython-dht
    pip install board

- Clase **Logger**
  - Se creó una clase Logger para encapsular el manejo de archivos.
  - El método escribir() permite registrar mensajes en un archivo .txt.
  - Ejemplo de uso:

    logger = Logger("log.txt")
    logger.escribir("Sistema iniciado")

- Script **principal con lectura del sensor**
  - Se utilizó la librería adafruit_dht para interactuar con el sensor.
  - El usuario selecciona si desea observar temperatura o humedad.
  - Cada lectura se registra en el archivo log.txt con marca de tiempo.

```
import time
import adafruit_dht
import board

class Logger:
    def __init__(self, archivo):
        self.archivo = archivo

    def escribir(self, mensaje):
        with open(self.archivo, 'a') as f:
            f.write(mensaje + '\n')

logger = Logger("log.txt")
logger.escribir("Sistema iniciado")

dht_device = adafruit_dht.DHT11(board.D4)

print("Seleccione una opción que desea observar:")
print("1-Temperatura")
print("2-Humedad")

res = int(input("Ingresar la opción (1, 2): "))

if res == 1:
    for i in range(10):
        try:
            temperature_c = dht_device.temperature
            temperature_f = temperature_c * (9 / 5) + 32
            mensaje = f"Temperatura: {temperature_c:.1f}°C / {temperature_f:.1f}°F a las {time.strftime('%H:%M:%S')}"
            print(mensaje)
            logger.escribir(mensaje)
        except RuntimeError as err:
            print("Error al leer el sensor:", err.args[0])
        time.sleep(1)

if res == 2:
    for i in range(10):
        try:
            humidity = dht_device.humidity
            mensaje = f"Humedad: {humidity:.1f}% a las {time.strftime('%H:%M:%S')}"
            print(mensaje)
            logger.escribir(mensaje)
        except RuntimeError as err:
            print("Error al leer el sensor:", err.args[0])
        time.sleep(1)
```

Fig. 1. Código implementado (Spyder)

III. Resultados

- Se generó el archivo log.txt con registros de temperatura y humedad.
- Cada línea contiene una marca de tiempo y una descripción clara del evento.
- El sistema respondió correctamente a la selección del usuario, mostrando y registrando los datos.

```
Sistema iniciado
Temperatura: 24.0°C / 75.2°F a las 00:12:03
Temperatura: 24.1°C / 75.4°F a las 00:12:04
Humedad: 58.0% a las 00:12:05
```

Fig. 2. Salida de archivo .txt

IV. Discusión

La implementación de la clase Logger permitió separar la lógica de registro del resto del código, cumpliendo con los principios de la POO [3]. Esta modularidad facilita la reutilización y mantenimiento del sistema. El uso de archivos .txt como medio de persistencia es adecuado para sistemas simples, aunque en aplicaciones más robustas se

recomienda el uso de bases de datos. La integración con el sensor DHT11 demuestra cómo los sistemas IoT pueden registrar eventos en tiempo real para análisis posterior.

V. Conclusiones

- Se logró implementar un sistema de registro modular utilizando POO en Python.
- El archivo log.txt contiene un historial completo de lecturas del sensor.
- La práctica refuerza la importancia de la persistencia de datos en sistemas IoT.
- La clase Logger puede extenderse fácilmente para registrar en otros formatos como .csv o enviar datos a la nube.

VI. Referencias

[1] Raspberry Pi Foundation, “GPIO Pinout,” Raspberry Pi Documentation, 2025. [Online]. Available: https://www.raspberrypi.com/documentation/computers/os.html#gpio

[2] Adafruit, “DHT Humidity Sensing on Raspberry Pi,” Adafruit Learning System, 2025. [Online]. Available: https://learn.adafruit.com/dht

[3] Python Software Foundation, “Classes and Objects,” Python Docs, 2025. [Online]. Available: https://docs.python.org/3/tutorial/classes.html