Simulación De Una Red De Sensores. (26 de Noviembre, 2024)

Participantes: Johan Yesid Ramírez Torres, Néstor Iván Castellanos Merchán

Resumen: En el presente documento se realiza la respectiva documentación en Formato IEEE del punto C de laboratorio 2, el fin de este documento es presentar la metodología de dicho ejercicio.

I. INTRODUCCIÓN

En el presente formato encontraremos la solución del Punto C de laboratorio 2. El presente punto es referente a la simulación de una red de sensores, en este caso la simulación de sensores de temperatura e indicar en el programa temperaturas altas y/o graves.

II. ANÁLISIS

Se tomarán en cuenta una serie de pasos y organización para la solución del ejercicio, es por ende de que como en la solución de este es poder detectar niveles críticos de temperaturas.

- Contexto: Se realiza el punto C de laboratorio que consiste en la Simulación de una red de sensores e indicar niveles críticos de temperatura.
- Población: Para usuarios que busquen dar un análisis para sus empresas sobre cuartos que manejen altas temperaturas.
- Limitación y Alcance: Usuarios interesados en implementar la red de sensores.

PROGRAMACION MODULAR

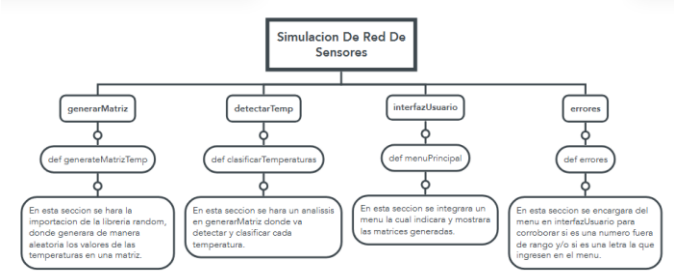

Figura 1: ProgramacionModular (Imagen fuente)

En la figura 1 se muestra la el mapa de programacion modular, la cual sera la estructura incial para trabajar sobre el ejercico C de laboratorio 2.

III. OBJETIVOS

- General:
  En el presente formato es dar información compleja sobre la realización de dicho ejercicio, e implementarlo y documentar respectivas funciones del código.

- Específicos:
  - Presentar un código preciso en la codificación totalmente funcional.
  - Enfatizar el uso respectivo de matrices y solución de errores en la codificación.
  - Dar en la metodología de problemas las fases completas.

IV. CODIFICACIÓN

A. *Tipos de aplicación*

En el presente software a realizar se tendrá en cuenta en la fase de Análisis de la Programación Modular y los conceptos completos respectivos a las Matrices y bucles para la verificación de valores respectivos.

B. *Instrucciones de uso*

- Para lograr realizar la toma de temperatura, como tal es una simulación, se realizará la toma de valores de manera aleatoria en los valores según corresponda.

- En la codificación se utilizará matrices la cual ira los valores de las temperaturas y para recorrer e indicar los niveles de temperatura se implementarán bucles.

Python:
En el presente lenguaje de programación “Python” se dará la realización del Punto C de laboratorio 2, se utilizarán funciones, matrices y bucles para su respectiva solución.

```python
import random


def generateMatrizTemp(n, m):
    '''
    Generar matrices con valores aleatorios
    '''
    matrizTemp = []
    for i in range(n):
        fila = []
        for j in range(m):
            # Generar una temperatura aleatoria entre 0 y 100 grados Celsius
            temperatura = round(random.uniform(0, 100), 2)
            fila.append(temperatura)
        matrizTemp.append(fila)
    return matrizTemp
```

Figura 2: Codificacion_Python1 (Imagen fuente)

- En la figura 2 encontraremos la función “def gemerateMatrizTemp”, esta va lazada a la importación random, la cual con la ayuda de los bucles esta misma es generada, para la toma de valores se le asigno a una variable llamada “temperatura”, la cual con ayuda de round ingresa los valores de manera aleatoria de o a 100.

```python
def clasificarTemperaturas(matrizTemp):
    '''
    Clasificacion de las temperaturas
        *Criticas
        *Altisimas
        *Altas
        *Normales
        *Frias
    '''
    categorias = {
        "🔥 Temperaturas Criticas (superior a 80°C)": [],
        "🌡️ Temperaturas Altisimas (superior a 60°C)": [],
        "☀️ Temperaturas Altas (superior a 40°C)": [],
        "🌤️ Temperaturas Normales (15 a 40°C)": [],
        "❄️ Temperaturas Frias (inferior a 15°C)": []
        }

    for i in range(len(matrizTemp)):
        for j in range(len(matrizTemp[i])):
            temp = matrizTemp[i][j]
            if temp > 80:
                categorias["🔥 Temperaturas Criticas (superior a 80°C)"].append((i, j, temp))
            elif temp > 60:
                categorias["🌡️ Temperaturas Altisimas (superior a 60°C)"].append((i, j, temp))
            elif temp > 40:
                categorias["☀️ Temperaturas Altas (superior a 40°C)"].append((i, j, temp))
            elif temp >= 15:
                categorias["🌤️ Temperaturas Normales (15 a 40°C)"].append((i, j, temp))
            else:
                categorias["❄️ Temperaturas Frias (inferior a 15°C)"].append((i, j, temp))
    return categorias
```

Figura 3: Codificacion_Python2 (Imagen fuente)

- En la figura 3 encontramos la codificación correspondiente a la clasificación de temperaturas, la cual se le decidió agregar, con fin de que el software a realizar sea uno más completo.
  Se ha decidido clasificarlas de 5 maneras:

  1. Critica: Superior a 80 grados Celsius.
  2. Altísima: Superior a 60 grados Celsius.
  3. Alta: Superior a 40 grado Celsius
  4. Normal: Superior a 15 grados Celsius.
  5. Fría: Inferior a 15 grados Celsius.

  Estas mismas están escrituras en la variable categorías de manera ordenada según corresponde para después ser utilizada en los if, elif y else, siendo esta necesaria para mostrar los datos de estos mismos, y con juntos a los bucles for, se les añadirá a las matrices con .append.

```python
from generarMatriz import generateMatrizTemp
from detectarTemp import clasificarTemperaturas
from errores import errores

import os
os.system('cls')

print("Bienvenido a la Simulacion de red de sensores de Temperatura\n")


def menuPrincipal():
    '''
    Menu principal
    '''
    print("\n--- Menú de Opciones ---")
    print("1. Generar matriz de temperaturas")
    print("2. Detectar y clasificar temperaturas")
    print("3. Mostrar matriz de temperaturas")
    print("4. Salir")
    opcion = input("Por favor, seleccione una de las opciones: ")
    return opcion

# Dimensiones de la matriz
n = 5  # número de filas
m = 5  # número de columnas

# Inicializar la matriz de temperaturas
matriz_temperaturas = []

rango_valido = range(1, 5)

while True:
    opcion = menuPrincipal()
    opcion_valida = errores(opcion, rango_valido)

    if opcion_valida == 1:
        # Generar la matriz de temperaturas
        matriz_temperaturas = generateMatrizTemp(n, m)
        print("\nMatriz de temperaturas generada.\n\n")

    elif opcion_valida == 2:
        if matriz_temperaturas:
            # Detectar y clasificar temperaturas
            categorias_temperaturas = clasificarTemperaturas(matriz_temperaturas)
            print("\nClasificación de temperaturas:")
            for categoria, temperaturas in categorias_temperaturas.items():
                print(f"\n{categoria}:")
                for temp in temperaturas:
                    print(f"  Posición: ({temp[0]}, {temp[1]}), Temperatura: {temp[2]}°C")
        else:
            print("Primero debes generar la matriz de temperaturas (opción 1).")

    elif opcion_valida == 3:
        if matriz_temperaturas:
            # Mostrar la matriz de temperaturas
            print("\nMatriz de temperaturas:")
            for fila in matriz_temperaturas:
                print(fila)
        else:
            print("Primero debes generar la matriz de temperaturas (opción 1).")

    elif opcion_valida == 4:
        # Salir del programa
        print("Saliendo de la simulación del programa...")
        break
```

Figura 4: Codificacion_Python3 (Imagen fuente)

- En la figura 4 encontraremos el interfaz para el usuario, siendo este que toda la codificación se lleve a cabo, donde se importan los 3 archivos que ayudan a su funcionalidad utilizando los from e import.

Adicional se le importa el sistema operativo con fin de que únicamente a la hora de ejecutar el respectivo programa ingrese lo que se ha codificado, con el propósito de verse mas limpio. Para la utilización de este mimo se ingresa os.system(‘cls’’).

```python
def errores(opcion, rango_valido):
    try:
        opcion = int(opcion)
        if opcion in rango_valido:
            return opcion
        else:
            print(f"Opción {opcion} no es válida. Debe estar dentro del rango {rango_valido}.")
            return None
    except ValueError:
        print("Entrada no válida. Por favor, ingrese un número.")
        return None

```

Figura5: Codificacion_Python4 (Imagen fuente)

- En la figura 5 econtraremos los errores, esta funcion “def errores” va enlazada con el menú del interfaz del usuario, su función da para reconocer si el usuario da un digito fuera del rango del menú de opciones o si ingresa una letra. El fin de esta función es evadir estos errores y para que el software no sufra errores al ser ejecutado.

  Para el reconocimiento de estos se ingresan el try y el except que con los encargados de esta tarea según corresponde, siendo try funcional para los valores numéricos junto con if y else para indicar si esta en el rango determinado, de lo contrario el except ValueError es por si el usuario ingresa una letra o un valor irreconocible por el software.

V. Requerimientos del sistema

- Para teléfono:
  Android: Versión 10
  RAM: 2GB
  Almacenamiento: 500Mb

- Para Ordenador:
  Windows 7 o superior
  RAM: 4GB
  Almacenamiento: 500Mb

VI. Diseño del algoritmo

En el presente punto se da a conocer el diseño de algoritmo teniendo presente su proceso y explicación, en concreto con la programación modular donde esta misma es sutil en cuanto la estructura en la codificación.

VII. Ejecución del programa

En la correspondiente ejecución del programa se establece sin novedad su funcionamiento como tal, siendo este logrando conseguir que se logre de antemano su finalidad bajo los parámetros y conclusión de errores.

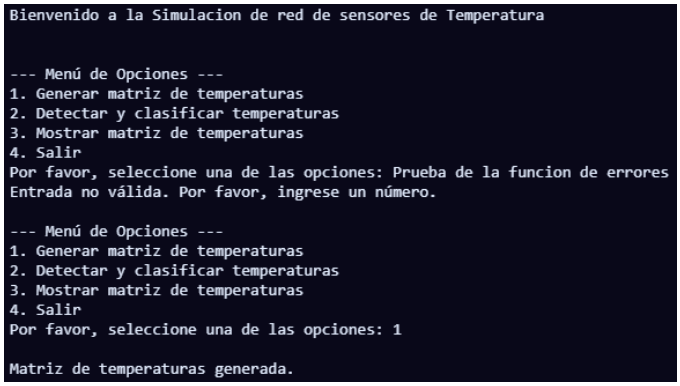

Figura 6: EjecucionPython1 (Imagen fuente)

- En la figura 6 encontramos la ejecución del programa.
  Se muestra el menú del usuario con 4 opciones_
  1. Generar matriz de temperaturas.
  2. Detectar y clasificar temperaturas
  3. Mostrar matriz de temperaturas
  4. Salir

  En la figura 6 se presenta la función realizada por def errores, la cual nos repite el menú.
  Adicionalmente se presenta la opción de generar temperaturas.

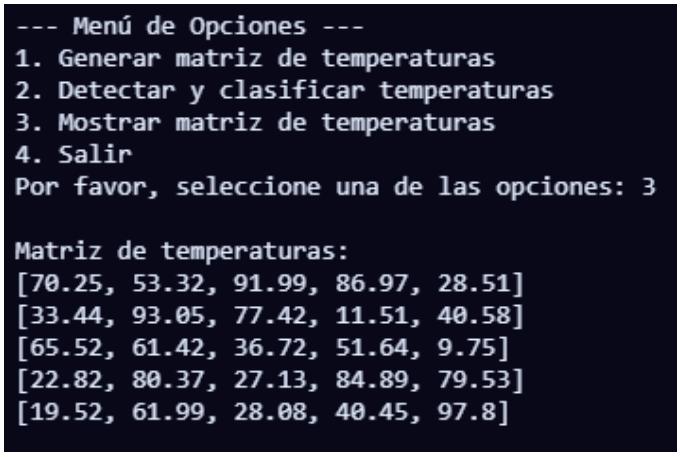

Figura 7: EjecucionPython2 (Imagen fuente)

- En la figura 7 encontramos la ejecución del menú numero 3 la cual su función es mostrar la matriz con los valores generados aleatoriamente.

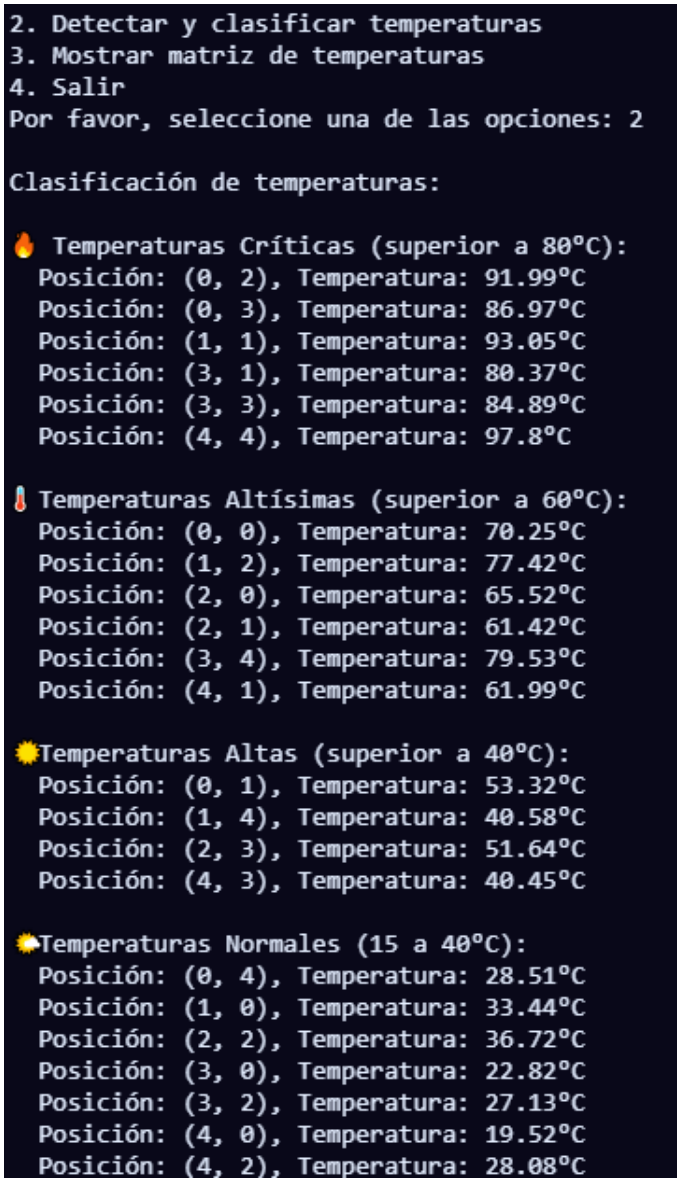

Figura 8: EjecucionPython3 (Imagen fuente)

- En la figura 8 encontramos en funcionalidad la opción del menú numero 2 la cual muestra las temperaturas según clasificadas de críticas a normales.

```
❄Temperaturas Frías (inferior a 15°C):
  Posición: (1, 3), Temperatura: 11.51°C
  Posición: (2, 4), Temperatura: 9.75°C

--- Menú de Opciones ---
1. Generar matriz de temperaturas
2. Detectar y clasificar temperaturas
3. Mostrar matriz de temperaturas
4. Salir
Por favor, seleccione una de las opciones: 4
Saliendo de la simulación del programa...
```

Figura 9: EjecucionPython4 (Imagen fuente)

- En la figura 9 encontramos la parte final de la opción 2 de generar temperaturas con las temperaturas frías e inclusive la opción numero 4 de Salir la cual finaliza el programa realizado.

VIII. DEPURACION – DOCUMENTACION Y MANTENIMIENTO

En los presentes modelos de la codificación se han realizado 3 modificaciones que la cual realizan serie de cambios que influyen en la codificación final según esta corresponda.

1. VERSION 1.0:
   En esta versión se realiza un borrador de código para sustentar la estructura del código a realizar, donde únicamente generaba la matriz aleatoria y la imprimía.

```python
import random

# Dimensiones de la matriz
n = 5  # número de filas
m = 5  # número de columnas

def generar_matriz_temperaturas(n, m):
    '''
    Genera la matriz de manera aleatoria de 0 a 100
    '''
    matriz = []
    for i in range(n):
        fila = []
        for j in range(m):
            # Generar una temperatura aleatoria entre 0 y 100 grados Celsius
            temperatura = round(random.uniform(0, 100), 2)
            fila.append(temperatura)
        matriz.append(fila)
    return matriz


def detectar_temperaturas_criticas(matriz, umbral=80):
    '''
    Detecta temperaturas criticas superior a 80
    '''
    temperaturas_criticas = []
    for i in range(len(matriz)):
        for j in range(len(matriz[i])):
            if matriz[i][j] > umbral:
                temperaturas_criticas.append((i, j, matriz[i][j]))
    return temperaturas_criticas


# Generar la matriz de temperaturas
matriz_temperaturas = generar_matriz_temperaturas(n, m)

# Detectar temperaturas críticas
temperaturas_criticas = detectar_temperaturas_criticas(matriz_temperaturas)

# Imprimir la matriz de temperaturas
print("Matriz de temperaturas:")
for fila in matriz_temperaturas:
    print(fila)

# Imprimir las temperaturas críticas
print("\nTemperaturas críticas (mayores a 80°C):")
for critico in temperaturas_criticas:
    print(f"Posición: ({critico[0]}, {critico[1]}), Temperatura: {critico[2]}°C")
```

Figura 10: Version1.0 Codificacion (Imagen fuente)

TEST VERSION 1.0:
T1:
En el test de la versión la codificación del software se implementó los requerimientos exigidos, donde su función en ejecución cumplía los requerimientos del ejercicio.

```
Matriz de temperaturas:
[64.48, 23.7, 45.0, 63.12, 7.15]
[98.75, 58.5, 57.09, 13.99, 46.22]
[18.83, 21.09, 70.85, 44.86, 68.77]
[44.06, 92.02, 38.22, 83.29, 20.78]
[61.11, 48.01, 98.84, 75.72, 46.71]

Temperaturas críticas (mayores a 80°C):
Posición: (1, 0), Temperatura: 98.75°C
Posición: (3, 1), Temperatura: 92.02°C
Posición: (3, 3), Temperatura: 83.29°C
Posición: (4, 2), Temperatura: 98.84°C
```

Figura 11: Version1.0 Test (Imagen fuente)

2. VERSION 1.1:
   En la versión 1.1 se busca realizar un código más completo para poder no solo detectar críticos de 80 grados, sino de otros más, en este caso:

   - Críticos
   - Altísima
   - Alta
   - Normal
   - Fría

```python
def clasificarTemperaturas(matrizTemp):
    ''''
    Clasificacion de las temperaturas
        *Criticas
        *Altisimas
        *Altas
        *Normales
        *Frias
    '''
    categorias = {
        "Temperaturas Criticas (superior a 80°C)": [],
        "Temperaturas Altisimas (superior a 60°C)": [],
        "Temperaturas Altas (superior a 40°C)": [],
        "Temperaturas Normales (15 a 40°C)": [],
        "Temperaturas Frias (inferior a 15°C)": []
        }

    for i in range(len(matrizTemp)):
        for j in range(len(matrizTemp[i])):
            temp = matrizTemp[i][j]
            if temp > 80:
                categorias["Temperaturas Criticas (superior a 80°C)"].append((i, j, temp))
            elif temp > 60:
                categorias["Temperaturas Altisimas (superior a 60°C)"].append((i, j, temp))
            elif temp > 40:
                categorias["Temperaturas Altas (superior a 40°C)"].append((i, j, temp))
            elif temp >= 15:
                categorias["Temperaturas Normales (15 a 40°C)"].append((i, j, temp))
            else:
                categorias["Temperaturas Frias (inferior a 15°C)"].append((i, j, temp))
    return categorias
```

Figura 12: Version1.1 Codificacion (Imagen fuente)

TEST VERSION 1.1:
T1:
Una vez integrada esta función con nuevos requerimientos a realizar en ejecución se imprimía mas matrices indicando sus posiciones de cada temperatura referente a lo indicado.

```
Clasificación de temperaturas:

Temperaturas Críticas (superior a 80°C):
  Posición: (1, 0), Temperatura: 91.5°C
  Posición: (3, 1), Temperatura: 84.08°C
  Posición: (3, 3), Temperatura: 94.22°C
  Posición: (3, 4), Temperatura: 95.25°C
  Posición: (4, 2), Temperatura: 93.6°C

Temperaturas Altísimas (superior a 60°C):
  Posición: (1, 4), Temperatura: 66.97°C
  Posición: (2, 2), Temperatura: 69.3°C
  Posición: (3, 0), Temperatura: 64.74°C
  Posición: (4, 0), Temperatura: 70.24°C
  Posición: (4, 1), Temperatura: 76.32°C
  Posición: (4, 4), Temperatura: 77.67°C

Temperaturas Altas (superior a 40°C):
  Posición: (0, 1), Temperatura: 55.38°C
  Posición: (0, 2), Temperatura: 45.71°C
  Posición: (1, 3), Temperatura: 46.84°C
  Posición: (2, 0), Temperatura: 48.92°C
  Posición: (2, 1), Temperatura: 45.32°C
  Posición: (2, 3), Temperatura: 56.05°C
  Posición: (2, 4), Temperatura: 51.87°C
  Posición: (4, 3), Temperatura: 52.62°C

Temperaturas Normales (15 a 40°C):
  Posición: (1, 1), Temperatura: 25.35°C

Temperaturas Frías (inferior a 15°C):
  Posición: (0, 0), Temperatura: 5.4°C
  Posición: (0, 3), Temperatura: 5.45°C
  Posición: (0, 4), Temperatura: 14.59°C
  Posición: (1, 2), Temperatura: 11.58°C
  Posición: (3, 2), Temperatura: 10.84°C
```

Figura 13: Versión 1.1 Test (Imagen fuente)

3. VERSION 1.2:
   En la presente versión se realiza una serie de modificaciones en el interfaz de usuario agregando un menú, la cual para que se vea una mejoría en el código y sea más completo y el usuario pueda tener interacción con el software.

```
from generarMatriz import generar_matriz_temperaturas
from detectarTemp import clasificar_temperaturas

def mostrar_menu():
    print("\n--- Menú de Opciones ---")
    print("1. Generar matriz de temperaturas")
    print("2. Detectar y clasificar temperaturas")
    print("3. Mostrar matriz de temperaturas")
    print("4. Salir")
    opcion = int(input("Seleccione una opción: "))
    return opcion

# Dimensiones de la matriz
n = 5  # número de filas
m = 5  # número de columnas

# Inicializar la matriz de temperaturas
matriz_temperaturas = []

while True:
    opcion = mostrar_menu()

    if opcion == 1:
        # Generar la matriz de temperaturas
        matriz_temperaturas = generar_matriz_temperaturas(n, m)
        print("Matriz de temperaturas generada.")

    elif opcion == 2:
        if matriz_temperaturas:
            # Detectar y clasificar temperaturas
            categorias_temperaturas = clasificar_temperaturas(matriz_temperaturas)
            print("\nClasificación de temperaturas:")
            for categoria, temperaturas in categorias_temperaturas.items():
                print(f"\n{categoria}:")
                for temp in temperaturas:
                    print(f"  Posición: ({temp[0]}, {temp[1]}), Temperatura: {temp[2]}°C")
        else:
            print("Primero debes generar la matriz de temperaturas (opción 1).")

    elif opcion == 3:
        if matriz_temperaturas:
            # Mostrar la matriz de temperaturas
            print("Matriz de temperaturas:")
            for fila in matriz_temperaturas:
                print(fila)
        else:
            print("Primero debes generar la matriz de temperaturas (opción 1).")

    elif opcion == 4:
        # Salir del programa
        print("Saliendo del programa...")
        break

    else:
        print("Opción no válida. Inténtalo de nuevo.")
```

Figura 14: Versión 1.2 Codificación (Imagen fuente)

TEST VERSION 1.2
Se realiza la implementación del menú la cual se imprime funciona para el usuario.

```
--- Menú de Opciones ---
1. Generar matriz de temperaturas
2. Detectar y clasificar temperaturas
3. Mostrar matriz de temperaturas
4. Salir
Por favor, seleccione una de las opciones:
```

Figura 15: Versión 1.2 Test (Imagen fuente)

4. VERSION 1.3:
   En la última versión se había concluido el ejercicio del Punto C de laboratorio, sin embargo, se necesitaba una función de errores para poder manejar el menú para que el software no vote error al momento de ingresar un símbolo no registrado. Es por ende que se decide realizar esta nueva función.

```
def errores(opcion, rango_valido):    Missing module docstring
    try:
        opcion = int(opcion)
        if opcion in rango_valido:
            return opcion
        else:
            print(f"Opción {opcion} no es válida. Debe estar dentro del rango {rango_valido}.")
            return None
    except ValueError:
        print("Entrada no válida. Por favor, ingrese un número.")
        return None
```

Figura 16: Solución de errores (Imagen fuente)

TESTE 1.3:
En la implementación de la nueva función se le ha realizado de incluir letras o números fuera de rango del menú de opciones y funciona al 100% sin errores volviendo a pedir al usuario que digite de nuevo una opción.

```
--- Menú de Opciones ---
1. Generar matriz de temperaturas
2. Detectar y clasificar temperaturas
3. Mostrar matriz de temperaturas
4. Salir
Por favor, seleccione una de las opciones: Teste funcion errores
Entrada no válida. Por favor, ingrese un número.

--- Menú de Opciones ---
1. Generar matriz de temperaturas
2. Detectar y clasificar temperaturas
3. Mostrar matriz de temperaturas
4. Salir
Por favor, seleccione una de las opciones:
```

Figura 17: Test función de errores (Imagen fuente)

IX. CONCLUSIÓN

- En la respectiva documentación se da por finalizado el formato IEEE referente al punto c de laboratorio 2, respectiva documentación se encuentra la programación modular y codificación completa dada en el lenguaje de programación de Python y por lo tanto dar con el resultado de la red de sensores.