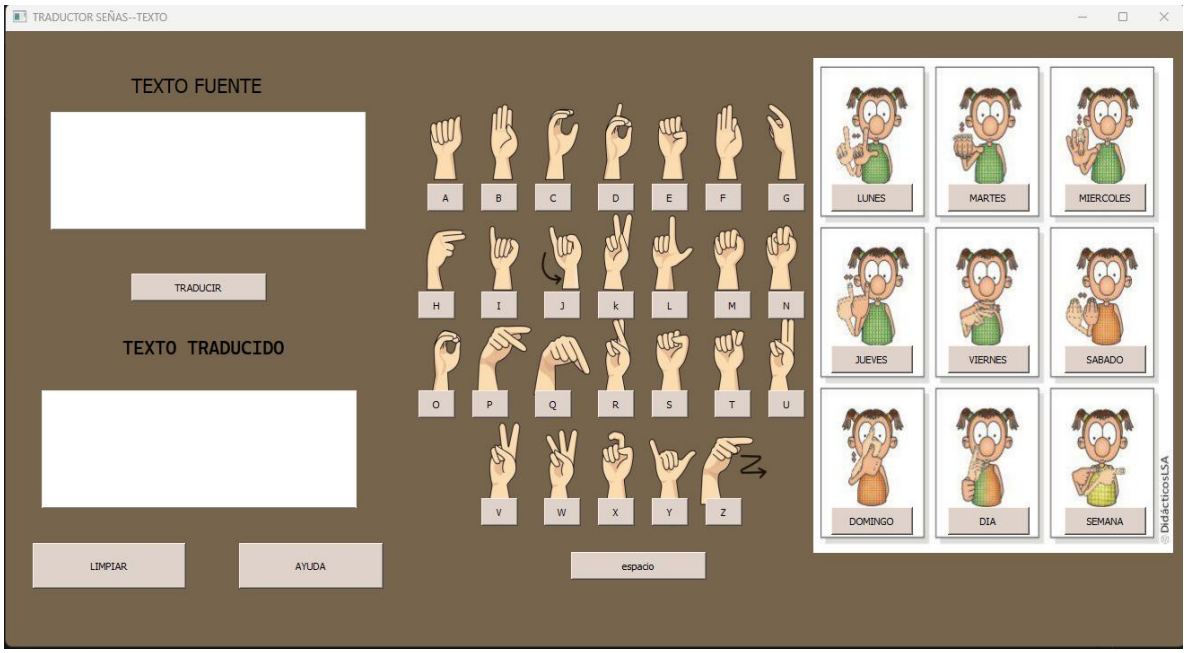
MANUAL DE USUARIO Y TÉCNICO

PROYECTO: TRADUCTOR DE LENGUA DE SEÑAS A TEXTO

- Universidad: Universidad Autónoma de Tamaulipas
- Facultad: Facultad de Ingeniería Tampico
- Materia: programación de sistemas base 2
- Semestre: 2025-3
- Profesor: Muñoz Quintero Dante Adolfo
- Integrantes del Equipo:
  - Felipe García Juan Diego
  - Jiménez Ramírez Leonel
  - Saldierna Segura Alberto Ángel



Contenido

2. Introducción al compilador y su propósito

Este proyecto consiste en una aplicación de software diseñada para traducir secuencias de gestos de la Lengua de Señas (representados por códigos numéricos) a texto legible en español.

El sistema funciona como un compilador completo, no solo traduciendo, sino analizando la estructura lógica y semántica de la entrada. Evolucionando del trabajo previo en "Programación de Sistemas de Base 1", esta versión 2.0 integra:

- Análisis Semántico: Validación de tipos de datos (Letras vs Palabras).
- Código Intermedio: Generación de instrucciones abstractas (DECODE).
- Optimización: Algoritmo de eliminación de subexpresiones comunes para reducir redundancia.
- Interfaz Gráfica Mejorada: Con soporte visual para el usuario y depuración técnica en consola.

3. Instalación y configuración

Requisitos del Sistema

- SO: Windows 10/11, macOS o Linux.
- Python: Versión 3.8 o superior.
- Dependencias: Librería PyQt5.

Pasos de Instalación

1. Descargue o clone la carpeta del proyecto.
2. Abra su terminal (Símbolo del sistema) y ejecute el siguiente comando para instalar la interfaz gráfica:

   Bash

   pip install PyQt5

3. Verifique que la estructura de carpetas sea correcta:
4. /src
5. |-- traductor.py
6. |-- Traductor.ui

4. Guía de uso paso a paso

1. Ejecución:

Abra la terminal en la carpeta principal y ejecute:

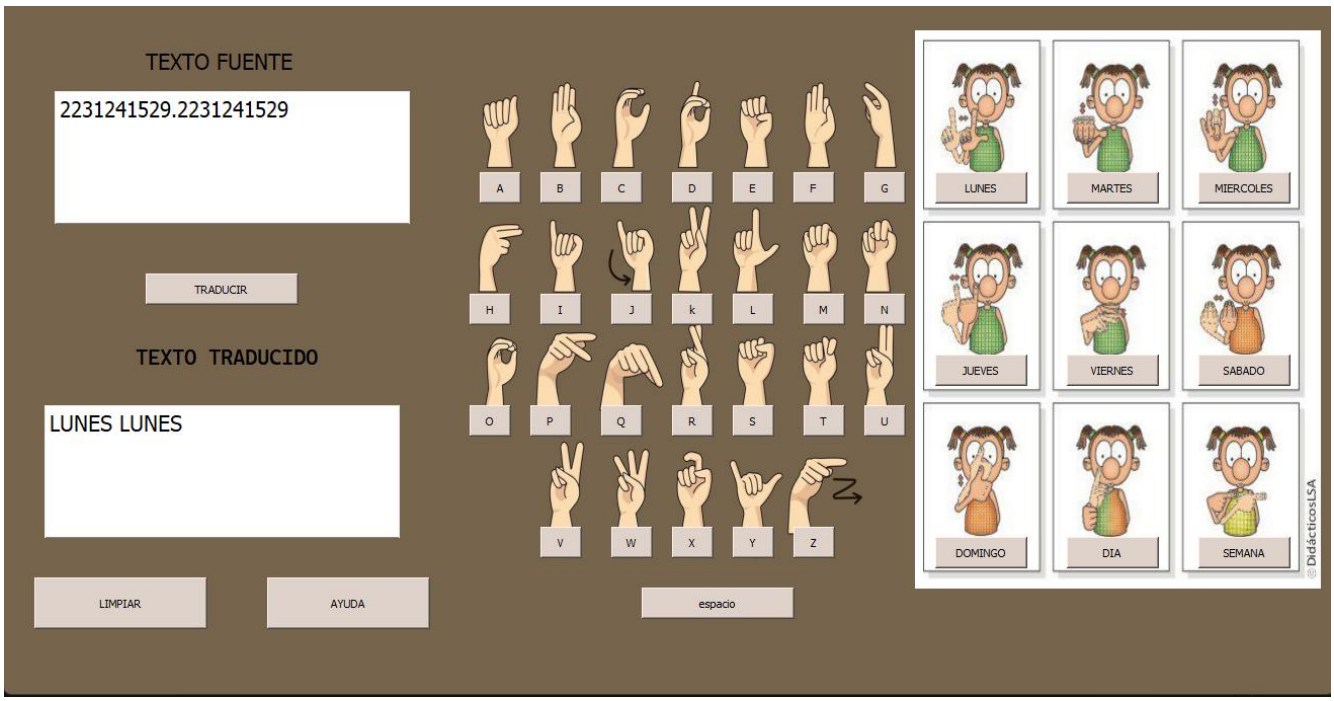
python src/traductor.py

Nota: Se abrirán dos ventanas: La Interfaz Gráfica y la Consola de Depuración (Pantalla Negra).

2. Ingreso de Señas:
   - Haga clic en los botones con imágenes de manos para ingresar letras o palabras (ej. LUNES).
   - Utilice el botón grande "espacio" para separar cada seña (inserta un punto .).
3. Proceso de Traducción:
   - Presione el botón "TRADUCIR".
   - Verifique el resultado en el cuadro de texto inferior.
4. Ayuda Interactiva:
   - Presione el botón "AYUDA"

5. Capturas de pantalla

A. Interfaz Principal y Traducción Exitosa

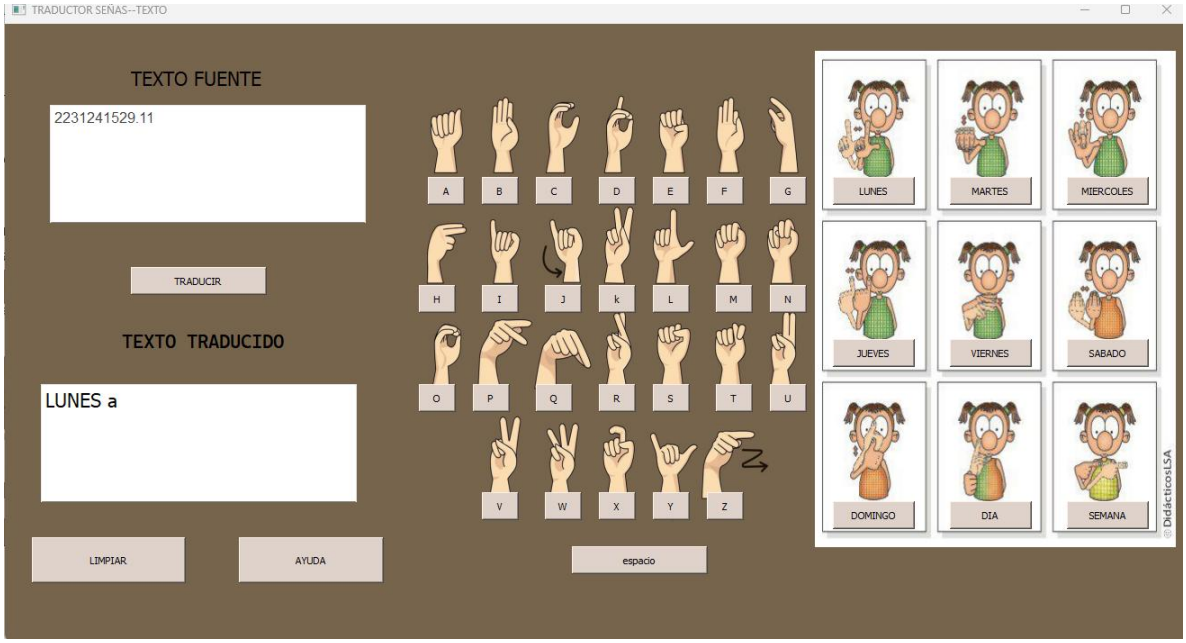


```
FASE 1: ANÁLISIS SEMÁNTICO (Tabla de Símbolos)

| POS | TOKEN (LEXEMA)       | TIPO       | VALOR           |
|-----|----------------------|------------|-----------------|
| 1   | 2231241529           | STRING     | LUNES           |
| 2   | 2231241529           | STRING     | LUNES           |
------------------------------------------------------------

FASE 2: GENERACIÓN CÓDIGO INTERMEDIO

LINEA    INSTRUCCIÓN
------------------------------------------------------------
1        T1 = DECODE(2231241529)
2        T2 = DECODE(2231241529)
------------------------------------------------------------

FASE 3: OPTIMIZACIÓN (Código Redundante)

LINEA    ACCION          DETALLE
------------------------------------------------------------
1        [NUEVO]         T1 = DECODE(2231241529)
2        [OPTIMIZADO]    REUSE T1 (Ya calculado)
------------------------------------------------------------

RESULTADO FINAL

 >> LUNES LUNES
```

Aquí se muestra el código optimizado donde hay redundancia es decir que se uso la misma letra

Entrada: 2231241529.11 (Seña de LUNES + Seña A)

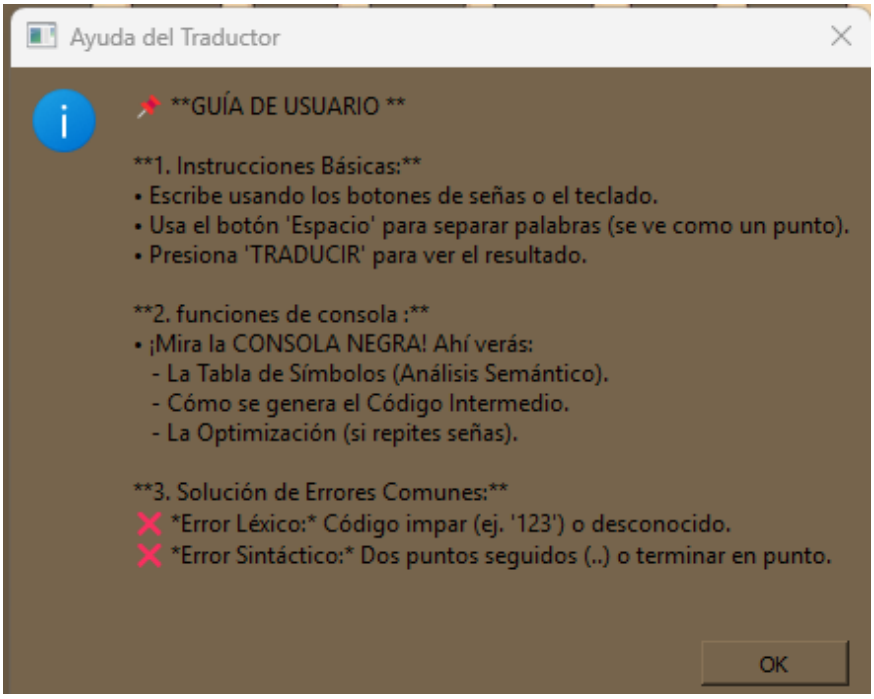
- Salida: lunes a



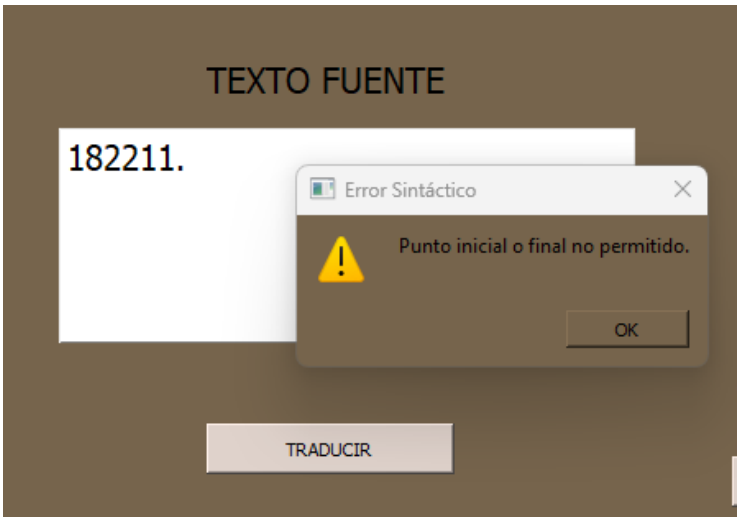
B. Análisis Semántico (Tabla de Símbolos)

```
        FASE 1: ANÁLISIS SEMÁNTICO (Tabla de Símbolos)

| POS | TOKEN (LEXEMA)       | TIPO       | VALOR           |
|-----|----------------------|------------|-----------------|
| 1   | 2231241529           | STRING     | LUNES           |
| 2   | 11                   | CHAR       | a               |
------------------------------------------------------------

        FASE 2: GENERACIÓN CÓDIGO INTERMEDIO

LINEA    INSTRUCCIÓN
------------------------------------------------------------
1        T1 = DECODE(2231241529)
2        T2 = DECODE(11)
------------------------------------------------------------

        FASE 3: OPTIMIZACIÓN (Código Redundante)

LINEA    ACCION          DETALLE
------------------------------------------------------------
1        [NUEVO]         T1 = DECODE(2231241529)
2        [NUEVO]         T2 = DECODE(11)
------------------------------------------------------------

                RESULTADO FINAL

 >> LUNES a
```

D. Manejo de Errores y Ayuda



No reconoce el numero



No puede finalizar con un punto

6. Referencia técnica del lenguaje

El compilador procesa un lenguaje fuente numérico posicional bajo las siguientes reglas:

1. Léxico:
   - Tokens válidos: Números pares.
   - Delimitador: Punto (.).
2. Semántica (Sistema de Tipos):
   - Tipo CHAR: Tokens de longitud 2. (Ej. 11 = 'a').
   - Tipo STRING: Tokens de longitud > 2. (Ej. 2231241529 = 'LUNES').
3. Código Intermedio:
   - Genera Triplos de la forma: Temporal = Operación(Operando).
   - Ejemplo: T1 = DECODE(11)

7. Código fuente (Módulos Principales)

A continuación se presentan los fragmentos clave de la lógica implementada en traductor.py:

Módulo de Análisis Semántico:
Python

```python
def analizar_semantico(self, tokens):
    self.tabla_simbolos = []
    # ...
    for i, token in enumerate(tokens):
        if len(token) == 2:
            tipo = "CHAR"  # Tipo Simple
        else:
            tipo = "STRING" # Tipo Compuesto (Palabra Reservada)
        self.tabla_simbolos.append({"token": token, "tipo": tipo, ...})
```

Módulo de Optimización (Eliminación de Redundancia):
Python

```python
def optimizar_codigo(self):
    temporales_existentes = {}
    for item in self.tabla_simbolos:
        token = item['token']
        if token in temporales_existentes:
            # OPTIMIZACIÓN: Se reutiliza el cálculo previo
            temporal_viejo = temporales_existentes[token]
            print(f"[OPTIMIZADO] REUSE {temporal_viejo}")
        else:
            # NUEVO CÁLCULO
            temporal = f"T{temp_counter}"
            temporales_existentes[token] = temporal
```

(El código fuente completo archivo traductor.py del entregable digital)

¿Cómo veo la optimización?

R: La optimización solo ocurre si hay redundancia. Ingrese la misma letra dos veces (ejemplo: 11.11) y presione Traducir. Revise la consola negra para ver el mensaje verde [OPTIMIZADO].