UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Universidad Distrital Francisco José de Caldas

Computer Science III

Computer Science III Report

Torres Mendieta David Santiago

Supervisor: Carlos Andrés Sierra Virguez

A report submitted in partial fulfilment of the requirements of
the University of Francisco José de Caldas for the degree of
Computer Engineering in *Data Science and Advanced Computing*

May 13, 2025

Declaration

I, David Santiago Torres Mendieta, of the Department of Computer Science, University of Francisco José de Caldas, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UoR and public with interest in teaching, learning and research.

David Torres
May 13, 2025

Abstract

In the current context, understanding and use of Morse code have significantly declined, despite its historical value and potential in alternative or emergency communication situations. As a solution, Python software was developed capable of bidirectionally translating natural language text and Morse code, incorporating features such as a graphical interface, file reading and writing, and input validation. The results show that the tool enables accurate, rapid, and accessible translation for users with basic knowledge, strengthening both the learning and practical use of Morse code.

Keywords: Morse code, machine translation, natural language, alternative communication, educational software, bidirectionality

Contents

List of Figures

List of Tables

List of Abbreviations

GUI	Graphical User Interface
DSL	Domain-Specific Language
OS	Operating System
BPM	Beats Per Minute
I/O	Input/Output
API	Application Programming Interface
MVP	Minimum Viable Product
ASCII	American Standard Code for Information Interchange
TTS	Text-to-Speech
IDE	Integrated Development Environment
UX	User Experience
PyPI	Python Package Index
PDF	Portable Document Format

Chapter 1

Introduction

Morse code, developed in the 19th century, was a key tool in the history of communications, allowing the transmission of messages through signals encoded in dots and dashes. Although its use has declined with the advancement of digital technologies, it continues to have relevant applications in emergency, educational, and accessibility contexts. In this context, modern tools are needed to facilitate its understanding and use, especially in educational or resource-limited settings.

Currently, there are multiple solutions for translation between Morse code and natural text, most available as mobile applications or web platforms. However, many of them have limited connectivity, portability, or functionality, and often focus only on one-way conversion. Therefore, we propose the development of educational software in Python that performs bidirectional translation between text and Morse code, integrating functions such as a graphical interface (Tkinter), reading and writing .txt files, input validation, and exporting results. This solution, developed with basic programming and text processing techniques, seeks to promote practical learning of coding systems and offer a useful, accessible, and scalable tool.

1.1 Background

Morse code, developed in the 19th century, was a fundamental communication system that allowed the transmission of messages using sequences of dots and dashes. Despite the rise of modern digital technologies, Morse code remains relevant in specific contexts such as emergency signaling, accessibility support, and educational environments. However, few interactive tools exist today that make learning and using Morse code intuitive and accessible. This project arises from the need to create a modern, educational, and functional solution that promotes the understanding and practical application of Morse code in a user-friendly environment.

1.2 Problem statement

Although Morse code continues to be relevant in education and alternative communication, existing tools are often limited in functionality, difficult to use, or lack interactive interfaces. Most solutions available online are unidirectional (text-to-Morse or Morse-to-text), require internet access, or are not designed with an educational focus. There is a lack of a comprehensive, offline, and open-source application that enables users to both learn and utilize Morse code in a practical and bidirectional manner.

1.3 Aims and objectives

Aim: The main aim of this project is to develop a Python-based software named Morse-Mind, capable of bidirectional translation between Morse code and natural language. The key objectives include:

Objectives:

- Designing an intuitive graphical interface for text input and output.
- Implementing core translation functionalities for Morse-to-text and text-to-Morse.
- Supporting file input/output for saving and loading messages.
- Providing real-time validation and error handling.
- Ensuring modular design for future enhancements (e.g., audio input).

1.4 Solution approach

The language uses simplified keywords such as `tempo`, `pattern`, `melody`, and `mix` to abstract the creation and combination of musical components. The interpreter, written in Python, parses user code through regular expressions and maps it to audio segments using PyDub. Pre-recorded samples in WAV or MP3 format are sequenced or mixed based on the user-defined script. The focus remains on loop-based composition without involving complex synthesis or MIDI integration.

1.4.1 Language Design

The solution will be structured into modular components: user input, validation, translation engine, and output (screen and file). Python was selected for its clarity and educational suitability. The GUI will be implemented using the Tkinter library, enabling interactive user interaction without requiring advanced system dependencies. A dictionary-based translation mechanism will map characters to Morse symbols and vice versa. Real-time validation will ensure user inputs conform to expected formats. The solution is designed to be extendable, lightweight, and usable offline.

1.4.2 Interpreter Implementation

Although MorseMind does not introduce a new programming language, it effectively treats Morse code as a minimal symbolic language. Its internal "language design" involves:

- A predefined character set (A-Z, 0-9, and basic punctuation).
- A one-to-one mapping between Morse sequences and characters.
- Rules for interpreting spaces as letter or word separators. This internal design enables efficient and accurate translation while preserving Morse's structural integrity. The system handles both encoding and decoding rules consistently.

Chapter 2

Literature Review

This chapter provides an overview of current tools and frameworks related to Morse code translation and educational software. It presents existing literature and applications that address the problem of converting between Morse code and natural language, and highlights the gap that this project, MorseMind, seeks to fill.

2.1 State of the Art

Several tools have been developed to support Morse code learning and translation. Among the most relevant are:

- **Morse Translator (Browser-based)**: Simple online tools that allow users to convert between text and Morse code using input boxes and output fields.
- **Morse Code Apps**: Mobile apps for Android and iOS that offer real-time conversion, some with audio playback or flashlight signaling.
- **Arduino-based Morse Systems**: Hardware projects that use Arduino boards to send or interpret Morse signals, often for educational or hobbyist use.

These tools often offer basic functionality or hardware interactivity, but rarely combine accessibility, educational value, and advanced software structure in one environment.

2.2 Context of the Project in the Literature

While existing tools are functional, they are typically fragmented or limited in scope. Web-based converters are useful for quick lookups but lack validation features, file I/O support, and educational design. Mobile applications tend to emphasize translation speed or entertainment, with little focus on learning mechanisms. Arduino projects, on the other hand, require technical and hardware knowledge.

MorseMind situates itself in this context by proposing an interactive, Python-based application that is educationally oriented. It supports bidirectional translation, graphical user interaction, real-time validation, and file export. Unlike most tools that focus solely on translation, MorseMind incorporates programming concepts and aims to strengthen computational thinking through a practical coding activity.

2.3 Relevance to the Intended Problem

The persistence of Morse code in emergency communication, accessibility technology, and historical education creates a need for an intuitive, modern tool for exploration and learning. MorseMind responds by offering a minimal, user-friendly interface that hides technical complexity and focuses on user understanding. By combining translation accuracy with real-time feedback and modular software design, the project addresses both practical and pedagogical challenges.

2.4 Critique of Existing Work

Although current tools serve specific needs, they often fall short in usability, educational value, or technical scalability. For example:

- **Online translators** provide little or no input validation, and typically lack interactivity beyond text boxes.
- **Mobile apps** are often closed systems with limited functionality and no educational guidance or export options.
- **Hardware projects** require programming skills and hardware setup, making them less accessible for general users.

MorseMind addresses these limitations by providing a desktop-based, modular, Python-implemented system with a focus on both function and form. It balances simplicity with functionality and supports users from novice to intermediate levels.

2.5 Summary

This literature review has identified a lack of educational, bidirectional, and extensible tools for Morse code translation. MorseMind addresses this gap by leveraging Python's readability and GUI capabilities to create a comprehensive application. It emphasizes accessibility, modularity, and learning by doing, contributing both as a translation tool and as a means to teach foundational programming and encoding concepts.

Chapter 3

Methodology

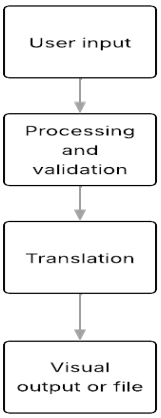
This chapter outlines the methodology used for designing and implementing MorseMind, an interactive Python-based software for bidirectional Morse code translation. As discussed in Chapter **??**, the project focuses on providing an educational tool that allows users to convert between Morse code and natural language, while also supporting accessibility features and practical programming concepts. The implementation leverages Python's simplicity and modularity to promote learning and usability.

3.1 System Design

The core design principle of MorseMind is simplicity and modularity. The system architecture is composed of the following functional layers:

- **Input Layer** – accepts user input in text or Morse code.
- **Processing Layer** – validates and cleans the input for conversion.
- **Translation Engine** – uses dictionaries to perform encoding/decoding.
- **Output Layer** – displays results in the GUI or writes them to files.

This structure supports extensibility and helps isolate responsibilities across the system, facilitating easier testing and future upgrades.



(Figure 1. General diagram of the Morse text translation system)

3.2 Interpreter Implementation

The interpreter, responsible for translation, was implemented using native Python data structures. A dictionary maps Morse symbols to alphanumeric characters and is reversed dynamically for text-to-Morse conversion. The interpreter performs the following steps:

1. Cleans and validates the input.
2. Splits input by word and character based on Morse syntax.
3. Maps symbols to corresponding characters or vice versa.
4. Returns the translated string for output or storage.

Error handling ensures that any invalid input is caught and flagged with informative feedback for the user.

3.3 Graphical User Interface

The GUI was developed using `Tkinter`, Python's standard library for graphical applications. The interface includes text boxes for input and output, buttons to trigger translation or clear data, and options to load or save files. Real-time input validation is incorporated to improve usability. The design emphasizes minimalism to maintain clarity and accessibility.

3.4 Workflow of the System

Figure **??** illustrates the high-level workflow of MorseMind, divided into three main stages:

1. **User Input and Interface Interaction:** Users input either Morse code or natural language through a graphical interface.
2. **Processing and Translation:** Input is validated and translated using internal dictionaries.
3. **Result Display and Export:** The translated message is displayed and optionally saved to a file.

3.5 File Handling

The system supports reading and writing to text files. Users can load messages to be translated or export results for later use. File operations are handled with Python's `os` and `tkinter.filedialog` modules to ensure cross-platform compatibility and usability.

3.6 System Architecture

MorseMind is composed of four main modules:

- **Main Controller**: Manages execution and links all modules.
- **Translator**: Encodes and decodes between Morse and text.

- **Validator**: Verifies the correctness of the input format.
- **UI Handler**: Controls all interface-related functions and user interactions.

Each module is contained in a separate Python file to ensure modularity and ease of maintenance.

3.7 Development Environment

- **Programming Language**: Python 3.11
- **Libraries Used**: Tkinter, OS, Regex
- **Operating System**: Windows

These technologies were chosen for their accessibility, stability, and broad compatibility across educational settings.

3.8 Testing Strategy

The software was tested through:

- **Unit Tests**: Validating each module (e.g., translation, validation, file handling).
- **Functional Tests**: Verifying overall system behavior from input to output.
- **Usability Tests**: Ensuring the GUI is intuitive and responsive to typical user workflows.

Sample test cases included correct inputs, invalid Morse code, empty text, and edge cases with special characters.

3.9 Ethical Considerations

Although the project does not involve personal data or human subjects, ethical practices were upheld:

- **Originality**: All implementation is original or based on open-source references.
- **Licensing**: External libraries are used in accordance with their respective licenses.
- **Purpose**: The tool is intended solely for educational, assistive, and non-commercial purposes.

3.10 Summary

This chapter presented the design and implementation methodology for MorseMind. The project followed a modular, user-centered approach using Python and Tkinter to ensure ease of use and future expandability. Testing and validation strategies were applied to confirm system reliability, and ethical software development standards were maintained throughout.

Chapter 4

Conclusions and Future Work

4.1 Conclusions

This project will address the need for an accessible and educational tool capable of translating between Morse code and natural language. The motivation behind MorseMind arises from the lack of comprehensive, beginner-friendly applications that support bidirectional translation and reinforce programming and encoding principles in an interactive way. The software will be designed to fill this gap by offering a modular, Python-based system with a graphical interface, file handling capabilities, and real-time input validation.

The implementation of MorseMind will follow a user-centered methodology, using Python's simplicity and modular design to promote both usability and extensibility. The program will feature a clean separation of concerns, enabling each module (translation, validation, GUI, and file I/O) to operate independently while contributing to the overall functionality of the system. The use of a GUI with Tkinter will enhance user interaction and accessibility, making it suitable for both educational environments and individual learners.

Although the project will not yet have yielded concrete results at the time of writing, it is expected that the development of MorseMind will demonstrate the feasibility and value of combining translation tools with educational design principles. The anticipated contributions will include a functioning prototype of a bidirectional Morse code translator with a user-friendly interface and file support. This tool will aim to lower the barriers to understanding Morse code and promote active learning through the integration of programming logic and symbolic communication.

In summary, MorseMind is projected to offer a novel contribution to the fields of educational software and assistive technology by providing a lightweight, scalable, and intuitive platform. While future validation will be necessary to measure the effectiveness and usability of the tool, the methodological approach and design choices provide a solid foundation for further development. The project's anticipated impact lies in empowering students, educators, and curious users to explore Morse code in a modern, engaging context.

4.2 Future Work

While the initial version of MorseMind will focus on essential features such as bidirectional translation, file handling, and user interface design, several areas of future work will remain open for exploration and enhancement. These improvements will stem from limitations identified during implementation and testing, as well as opportunities for broadening the scope and impact of the tool.

One of the most promising areas for future development will be the integration of audio functionality. The software could be extended to detect Morse code through microphone input or convert text to audible Morse signals using beeps or tones. This would allow users to both receive and transmit Morse in real-world communication scenarios, thereby increasing the practical relevance of the tool. To achieve this, Python libraries such as `pyaudio` or `pygame` could be incorporated to capture and generate sound-based Morse sequences.

Another key enhancement will be the development of a web-based version of MorseMind using frameworks such as Flask or Streamlit. This would enable wider access across platforms, including mobile and tablets, and remove the dependency on local installations. The web version could also incorporate cloud storage or collaborative features for educational use in classrooms or remote learning environments.

Additionally, accessibility improvements could be made by integrating support for screen readers, alternative text input methods (e.g., switches or eye tracking), or text-to-speech feedback. These features would expand MorseMind's usefulness to users with disabilities, aligning with inclusive design principles.

From a pedagogical perspective, a future version of MorseMind might include a "learning mode" or interactive tutorials. This feature would guide users through the basics of Morse code, providing exercises and feedback to enhance engagement and knowledge retention. Gamification elements such as achievement tracking or translation challenges could also be considered.

In conclusion, while the initial scope of MorseMind will prioritize foundational functionality and user experience, its architecture and design will leave the door open for significant future enhancements. These improvements will aim to extend the tool's accessibility, pedagogical value, and real-world applicability, ensuring that MorseMind evolves into a comprehensive and impactful educational resource.

References

[1] López, J., Pérez, M., Rodríguez, F., "Uso del código Morse en sistemas de comunicación de emergencia", Revista de Tecnología y Seguridad, vol. 45, no. 2, pp. 78-85, 2020.

[2] Van Rossum, G., Drake, F. L., Python 3: The Programming Language, 3rd ed., New York, NY, USA: O'Reilly Media, 2009.

[3] Smith, A. Johnson, P., "Aplicaciones de código Morse en la era digital", Journal of Communication Engineering, vol. 12, no. 3, pp. 234-245, 2018.

[4] Jackson, R., "Implementación de interfaces gráficas con Tkinter en Python", Programación y Desarrollo, vol. 30, no. 1, pp. 112-120, 2019.