Morse Code Translator Using Moore Machine

Simon Daniel Dela Cruz
Computer Science Department
BSCS STEM – 3AB
Technological University of the Philippines
Ayala Blvd, Ermita, Manila, 1000 Metro Manila, Philippines
simondaniel.delacruz@tup.edu.ph

Joshuel Ernest Simbulan
Computer Science Department
BSCS STEM – 3AB
Technological University of the Philippines
Ayala Blvd, Ermita, Manila, 1000 Metro Manila, Philippines
joshuelernest.simbulan@tup.edu.ph

Andrew James Tejerero
Computer Science Department
BSCS STEM – 3AB
Technological University of the Philippines
Ayala Blvd, Ermita, Manila, 1000 Metro Manila, Philippines
andrewjames.tejerero@tup.edu.ph

Mark Zaired Ugay
Computer Science Department
BSCS STEM – 3AB
Technological University of the Philippines
Ayala Blvd, Ermita, Manila, 1000 Metro Manila, Philippines
markzaired.ugay@tup.edu.ph

***Abstract*— Morse Code is a communication system that has been around for decades. It works by representing the alphabet, numbers, and some symbols into an arrangement of dots, dashes, and spaces. Although it is less used now due to innovations of streamlined communication, it still has its uses for communication during emergencies, assistive communication tools for some PWDs, and some professions such as pilots and navigators where they send their identification letters for station names. Thus, in this study it recreates a Morse Code translator as an application of Moore Machine, enabling the user to translate English text to Morse Code and vice versa.**

Keywords—Morse Code, Moore Machine

I. INTRODUCTION

Morse Code is a telecommunication system that uses dots, dashes, and spaces to convey messages. The coded messages can be transmitted using different methods such as varied lengths of electrical pulses or visual signals as flashing lights [1]. Morse code was first developed by Samuel F. B. Morse, born on April 27, 1791, in Charlestown, Massachusetts. Morse graduated from Yale Collage and had interests both in painting and electricity. His inspiration for developing an electronic telegraph sparked back from 1832 after hearing a conversation about electromagnetism during his travel back to the United States from Europe. While Morse was devoting his time in developing his first telegraph, he was introduced to Alfred Vail who offered to provide materials and labor for help. Then, by 1838 Morse and Vail develop the first Morse Code that is used as a dictionary for decoding the message produced by their one-wire telegraph system [2] .

The first version of Morse code displayed deficiencies in transmitting messages for non-English text. European countries use diacritics often for their texts, such examples are *ä*, ö, é, and many more. Thus, the conference of European nations in 1851 developed a new version of Morse code and called it the International Morse Code [1]. The improvements showed by the International Morse Code were displayed in its simpleness and precision compared to the earlier version. It defined constant length for dashes compared to the original Morse code. It specified that the dot is the unit, a dash is equal to three dots, the space between the signals forming the same letter is equal to one dot, the space between two letterse is equal to threee dots, and the space between two words is equal to seven dots [3]. In figure 1 it details the corresponding string of dash and dots for each letter and number as per the International Morse Code standard.

Morse code is still used to this day. People such as amateur radio enthusiasts still use Morse code for their hobbies. Sectors like aeronautical fields prominently use Morse code since it is still widely used in radio navigational aids such as Very-High-Frequency Omnidirectional Radio Range (VORs) and Non-Directional Beacons (NDBs). Morse code also found an alternative use as it becomes as a mean of communication for some person with disabilities [4].

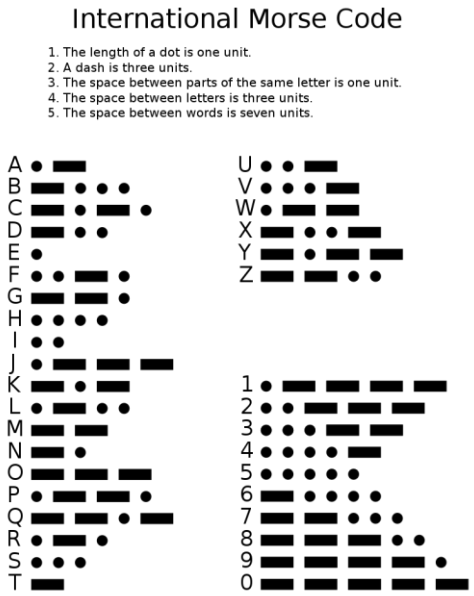

Figure 1. International Morse Code Chart

Using Python, the researchers intend to develop a Morse translator program. Python is a high-level programming language developed by Guido Van Rossum in February 1991[6]. To facilitate the conversion of strings of dots "." and dashes "-" Moore machine will be used to emulate the Morse code translator. Moore machines are finite state transducers developed by Edward Moore. For a given output Moore machine will generate an output depending on the current state of the machine [5]. This project will be limited only to the English alphabet which means it will not include any diacritics, numbers 0-9, and symbols "+", "=", and '/'.

II. Developing The Morse Translator

A. Defining the Programs Function

To develop the project, the researchers first laid out the flow of processes to visualize the project.

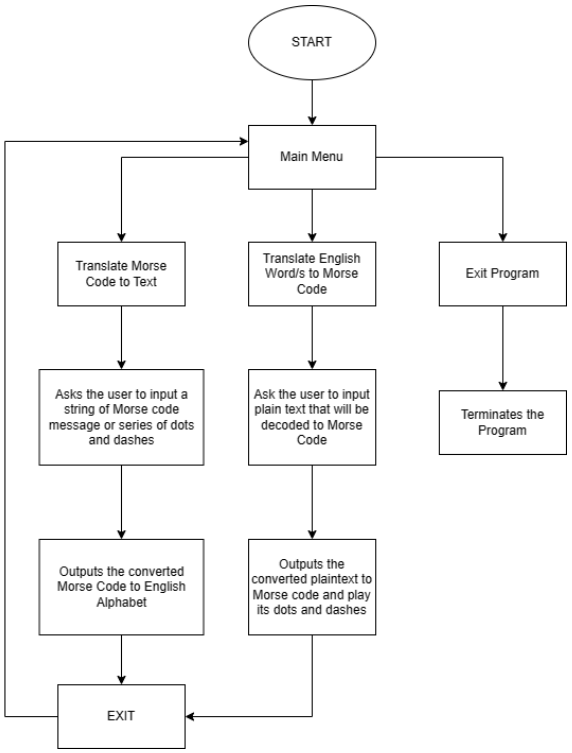

Figure 2. Flow of the Morse Translator Program

The program starts with a menu UI to help the users pick an action first. The actions can be divided into three choice which are *translate Morse code to text*, *translate English word/s to Morse code*, and *exit program.* Python will be the programming language to be used to build the program together with Tkinter library for the GUI of the project.

B. Translate Morse Code to Text

The *translate Morse code to text* details that the user will input a string of dash and dots, representing a Morse code for the program to output the corresponding English text. A new window will appear prompting the user to enter the Morse code they want to translate. Each letter will be separated by a space (e.g., "-. ---" = NO). Then to separate two words "/" will be used (e.g., ".- -. -… / -. ----" = AND NO). To facilitate the translation of Morse Code strings to English text construction of Moore machine will be applied. The Moore machine can be described using by 6-tuples (Q, q0, ∑, O, δ, λ) where:

- Q: finite set of states
- q0: initial state of machine
- ∑: finite set of input symbols
- O: output alphabet

- δ: transition function where Q × ∑ → Q
- λ: output function where Q → O

To describe the Moore machine relative to the Morse Translator project it will be described as follows:

- Q = 43 states. {A-Z, 0-9, and symbols "+", "=", and "/", 4 – ε-states}
- q0 = S
- ∑ = {".", "-"}
- O: {A-Z, 0-9, +, =, /, ε}

To visualize the Moore Machine a diagram is shown below:

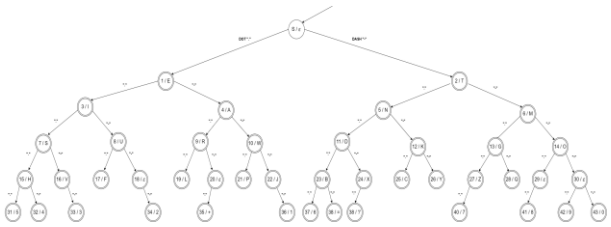

Figure 3 *Moore Machine Morse Translator*

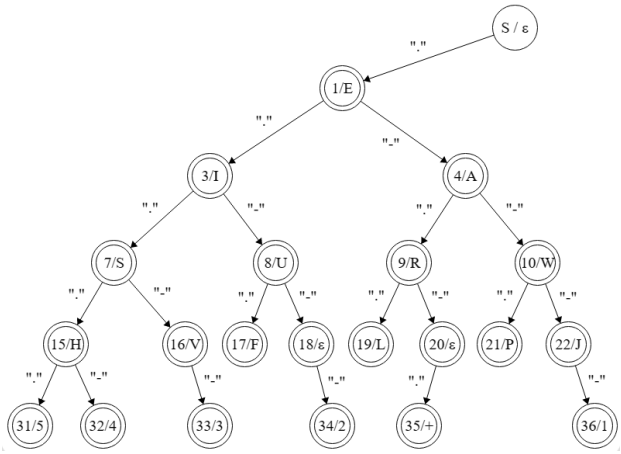

Figure 4 *Moore Machine Morse Translator Left Side*

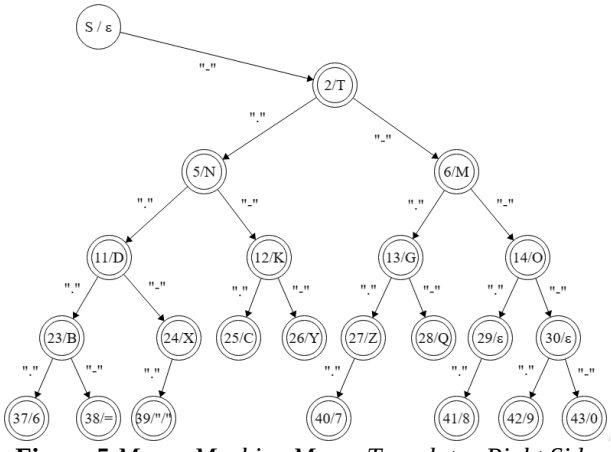

Figure 5 *Moore Machine Morse Translator Right Side*

The Python implementation of the *Translate Morse Code to Text* starts by declaring a class for the Moore machine as seen in figure 6. The class itself will serve as state. The self.left will serve as transition function if the input was a dot ".", while the self.right will serve as the transition function if the input was a dash "-". The self.char will serve as supposed output for each corresponding state.

```
5   class State:
6       def __init__(self, char='', left=None, right=None):
7           self.char= char
8           self.left= left
9           self.right= right
```

Figure 6 *Class Declaration for the Moore Machine*

```
54  # Morse FSM
55  morse_fsm = State('',
56                  State('E',
57                      State('I',
58                          State('S',
59                              State('H',
60                                  State('5'),
61                                  State('4')),
62                              State('V',
63                                  State(''),
64                                  State('3'))),
65                          State('U',
66                              State('F',
67                                  State(''),
68                                  State('')),
69                              State('',
70                                  State(''),
71                                  State('2')))),
72                      State('A',
73                          State('R',
74                              State('L',
75                                  State(''),
76                                  State('')),
77                              State('',
78                                  State('+'),
79                                  State(''))),
80                          State('W',
81                              State('P',
82                                  State(''),
83                                  State('')),
84                              State('J',
85                                  State(''),
86                                  State('1'))))),
87                  State('T',
88                      State('N',
89                          State('D',
90                              State('B',
91                                  State('6'),
92                                  State('=')),
93                              State('X',
94                                  State('/'),
95                                  State(''))),
96                          State('K',
97                              State('C',
98                                  State(''),
99                                  State('')),
100                             State('Y',
101                                 State(''),
102                                 State('')))),
103                 State('M',
104                     State('G',
105                         State('Z',
106                             State('7'),
107                             State('')),
108                         State('Q',
109                             State(''),
110                             State(''))),
111                     State('O',
112                         State('',
113                             State('8'),
114                             State('')),
115                         State('',
116                             State('9'),
117                             State('0'))))))
118
```

Figure 7 *Moore Machine Class*

The actual translation of *Morse code to text* takes place in the functions displayed in Figure 8. The input Morse code string is split using spaces. This is done to divide each Morse code string to their corresponding characters resulting to combinations of dots and dashes treated as a single element in an array. If a '/' is read a space will be added. Then, each dot "." and dash "-" read will be used as an input for the transition function of Moore class states resulting to either state.left or state.right depending on the input. This will be recursively done until the end of the split Morse code string is read. The corresponding output character of the last state it visited will be returned. The character returned will be appended to the variable that will contain the whole translation of the whole Morse code string. The process is repeated until no elements from the split Morse Code string remains.

```
# Functions for decoding Morse Code
def morse_to_char(morse_fsm, morse_string, i=0):
    if i==len(morse_string):
        return morse_fsm.char
    elif morse_string[i] == '.':
        return morse_to_char(morse_fsm.left, morse_string, i+1)
    else:
        return morse_to_char(morse_fsm.right, morse_string, i+1)

def decode_morse(morse_fsm, str):
    decoded = ''
    morse_string = str.split(' ')
    for decode_string in morse_string:
        if decode_string == '/':
            decoded += ' '
        else:
            decoded += morse_to_char(morse_fsm, decode_string)
    return decoded

```

Figure 8 *Functions to Translate Morse Code to Text*

C. Translate English word/s to Morse Code

The *translate English word/s to Morse code* details that the user will input a string of English plaintext that will be encoded in Morse Code. A new window will appear prompting the user to type the string they want to be encoded in Morse code. To encode the English plaintext in Morse code the already built Moore machine can be utilized. Since every Q-state contains an output, each character in the English plaintext can be compared if they are equal. The inputs used to traverse that state can then be recorded resulting to equivalent Morse code of the English plaintext. To explain this further a pseudocode is displayed below:

Pseudocode:

1. Read each character of the input string.
2. In each iteration check if the current character in read is equal to the current output of the state.
3. If not move to another state and record the transition function.
4. Repeat step 3 until the input character is equal to the output character of the state.
5. The $\sum$ used in each transition function will now serve as the Morse Code translation of the input string.

The Python implementation of *Translate English Word/s to Morse Code* utilizes the existing Moore Machine Class declared previously as shown in figure 9. The input English Word/s string is being read character by character using a for loop. Once an input character is being read, a temporary variable will be declared. This is where the Morse code string of the current character in read will be temporarily stored. Then it continuously finds the state where the character output is equal to itself, while also recording the path it took to reach the state and storing it in the temporary variable. Once it finds the state that has equal output character to the character being read it returns the path taken which results to the combination of dots and dashes. The returned Morse code string will be joined to the *encoded_tmp* variable added with a space to separate each translated character. Once the final character of the input string was read the for loop breaks.

```
# Function for encoding a String to Morse Code
def char_to_morse(morse_fsm, character, tmp):
    if morse_fsm == None:
        return False
    elif morse_fsm.char == character:
        return True
    else:
        if char_to_morse(morse_fsm.right,character, tmp) == True:
            tmp.insert(0, "-")
            return True
        elif char_to_morse(morse_fsm.left,character, tmp) == True:
            tmp.insert(0, ".")
            return True
```

```
    encoded = ''
    user_string = input.upper()
    for character in user_string:
        tmp = []
        mt.char_to_morse(mt.morse_fsm, character, tmp)
        encoded_tmp = "".join(tmp)
        encoded = encoded + encoded_tmp + " "
```

Figure 9 *Functions to Translate Text to Morse Code*

D. Devolping the GUI of the Project

Python has a lot of GUI frameworks, and one of those is the Tkinter. The only difference of Tkinter among other frameworks is that it is built into the Python standard library. It renders its widgets which is a controllable element within the GUI using native operating system elements. Thus, making it lightweight and easy to use compared to other frameworks.

Figure 10 *Whole Code for the Main Menu UI*

```
#Translate Morse Code to Text Button
btn1 = tk.Button(frame, text="Translate Morse Code to Text", font=('Ari
    command = morse_to_text)
btn1.pack(fill="both")
```

Figure 11 *Translate Morse Code to Text Button*

```
#Translate English Word/s to Morse Code Button
btn2 = tk.Button(frame, text="Translate English Word/s to Morse Code", font=('Ar
    command = text_to_morse)
btn2.pack(fill="both")
```

Figure 12 *Translate English Word/s to Morse Code Button*

III. RESULTS

A. Morse Translator Menu Interface

The developed Morse Translator project used Tkinter as the GUI framework. The functionalities of the program were displayed using buttons for ease of use for the user. During the event of the button clicked it calls the corresponding function in the program.

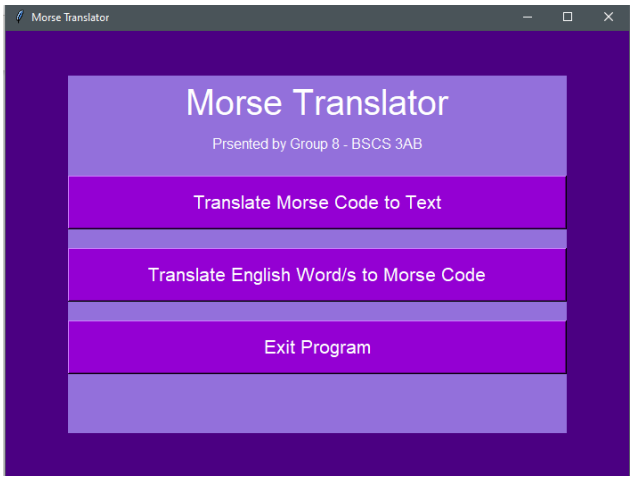

Figure 13 *Morse Translator Main Menu UI*

B. Morse Code to Text

The *Translate Morse Code to Text* button displays a new window asking the user to input a Morse Code string for the program to translate. The corresponding letters for each Morse code string is separated by a single space. To input a Morse Code string that contains more than one word, it is required to include '/' as a divider to be able to distinguish individual words given a Morse code string. Once the user input a Morse code string inside the textbox, the *Enter Morse Code* button will become active allowing the user to translate their input. After clicking the *Enter Morse Code* button the equivalent English text of the Morse Code input will be displayed together with an *Exit* button that will send back the user to the main menu.

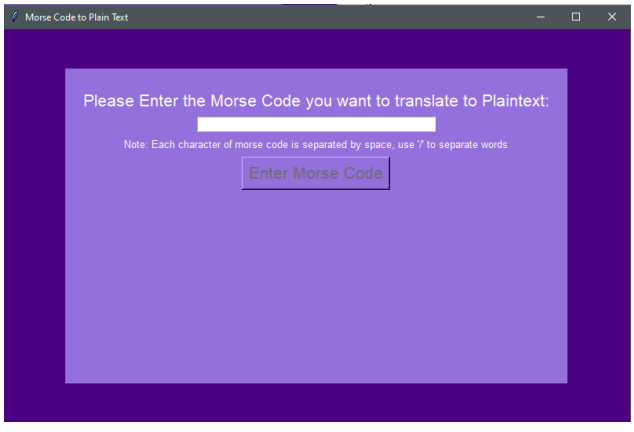

Figure 14 *New window prompt asking the user to input a Morse String*

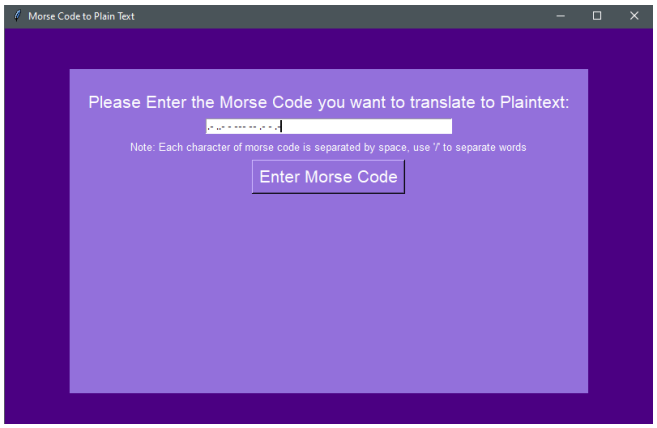

Figure 15 *Enter Morse Code button becomes active*

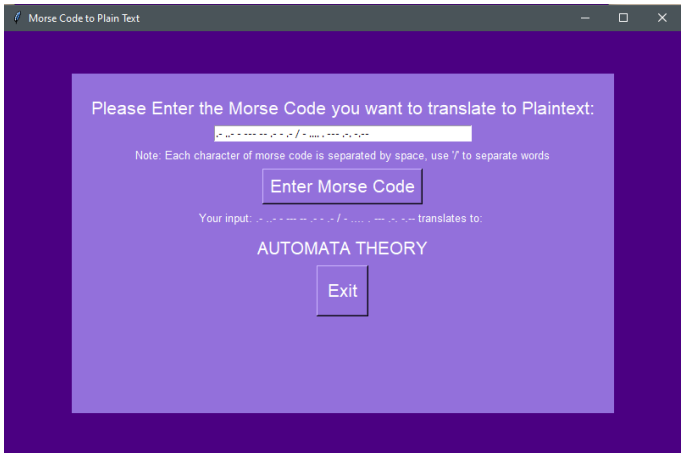

Figure 16 *Displayed Output of the Translate Morse Code to Text Function*

C. English Word/s to Morse Code

The *Translate English Word/s to Morse Code* button will also display a new window asking the user to input English word/s. In the moment that the textbox is filled the *Enter Plaintext* button will be active. After clicking the active *Enter English Word/s* button a beeping sound for the corresponding dots and dashes will be played before displaying the encoded message. After the beeping noises the Morse Code will be displayed together with an *Exit* button that will return the user back to the main menu.

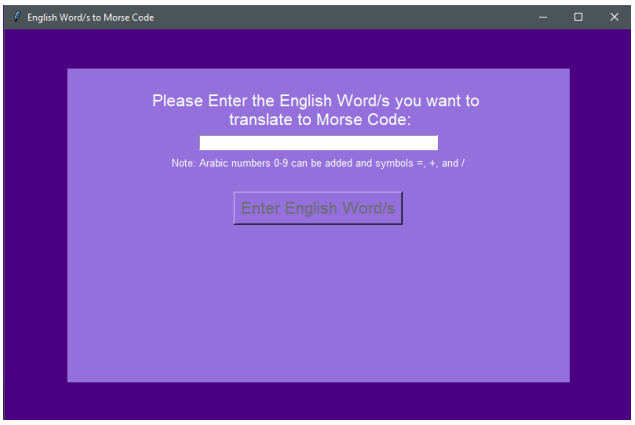

Figure 17 *New window prompt asking the user to input an English Word/s*

```
44  # Function to play Morse Beep sounds
45  def morse_beep (morse_code):
46      for morse_code in morse_code:
47          if morse_code == '.':
48              playsound("MorseBeeps\dot.wav")
49          elif morse_code == '-':
50              playsound("MorseBeeps\dash.wav")
51          else:
52              time.sleep(0.3)
53
```

Figure 18 *Function that plays the Morse beeps*

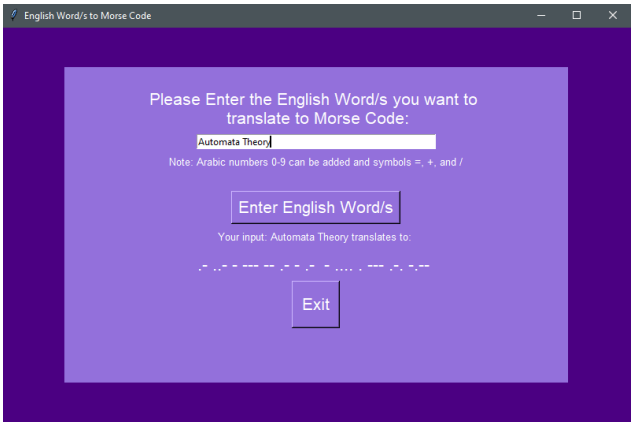

Figure 19 *Displayed Output of the Translate English Word/s to Morse Code Function*

IV. CONCLUSION AND RECOMMENDATIONS

Morse Code is one of the most elegant and simple systems of communication, as it only uses combination of dots and dashes to convey a message. Additionally, Morse Code messages can also be transmitted using various methods such as mechanically flicking a light, using sound frequencies, electrical pulses and even blinking of the eye[7]. Due to its flexibility in transmission, it withstands various innovations done in the sector of communication that it is still used functionally to this age, especially in case of emergencies, which makes

it a life-saving knowledge. It can also be built as an application of trees, or FSM. As seen in this project the researchers successfully developed a program that can translate Morse Code to text and vice versa. Using Moore machine, a finite state transducer, the machine outputs the character corresponding to input combination of dots and dashes. Displaying a degree of understanding of Finite State Machines by constructing a Moore machine and translating it to computer code using Python. To further improve this project the following can be applied in future:

- Create a mobile application that can also do Morse Code transmission using the device's native features. For example, blinking using its flashlight or screen, and/or beeping sounds.
- Include a speech to Morse Code functionality.

V. REFERENCES

[1] T. E. o. E. Britannica, "Encyclopædia Britannica," Encyclopædia Britannica, inc., 05 January 2023. [Online]. Available: https://www.britannica.com/topic/Morse-Code. [Accessed 28 January 2023].

[2] C. Mabee, "Samuel F.B. Morse," Encyclopædia Britannica, inc., 9 May 2022. [Online]. Available: https://www.britannica.com/biography/Samuel-F-B-Morse. [Accessed 28 January 2023].

[3] I.-R. Radiocommunication Sector, "Recommendation ITU-R M.1677-1 International Morse Code," International Telecommunication Union, 2009.

[4] N. Treasure, "Is Morse Code Used Today? The Brief History and Importance of Morse Code," Owlcation, 20 December 2022. [Online]. Available: https://owlcation.com/humanities/morse_code. [Accessed 28 January 2023].

[5] Z. Ilyas, "Educative: Interactive Courses for Software Developers," [Online]. Available: https://www.educative.io/answers/what-is-a-moore-machine. [Accessed 28 January 2023].

[6] B. Klein, "History and Philosophy of Python," python-course.eu, 1 Febuary 2022. [Online]. Available: https://python-course.eu/python-tutorial/history-and-philosophy-of-python.php. [Accessed 28 January 2023].

[7] P. Dancstep, "Speaking With Your Eyes | Denton Blinks Morse Code," exploratorium, 20 January 2017. [Online]. Available: https://www.exploratorium.edu/blogs/tangents/denton. [Accessed 29 January 2023].