Technoscience Journal for Community Development in Africa 1:1 (2020) 71–80
ISSN 2659-0573

Research Article

Implementation of Yorùbá Unicode generation for an indigenous keyboard

Jumoke Falilat Ajao*, Ronke Seyi Babatunde, Suliyat Oyindamola Asapetu, Shakirat Ronke Yusuff

Department of Computer Science, Kwara State University, Malete, Nigeria

Abstract: Characters are generally represented on standard keyboards by the use of coding schemes such as ASCII and Unicode but some characters of the Yorùbá language were not captured because of the accent and diacritic signs inherent in Yorùbá language. Coding representation for Yorùbá characters has become a key problem in circuit implementations of keyboards for typing Yorùbá documents correctly. The standard keyboard layouts do not have a simple key combination for all the characters. To represent Yorùbá characters in human-computer interaction, this paper proposes the use of Unicode for the deployment of Yorùbá keyboard for effective and efficient typing of documents in Yorùbá language. The approach used for the development of the Yorùbá keyboard uses both binary and hexadecimal representation of the coding standards to codify Yorùbá characters with their diacritic signs and the under dot. This dimension introduces the complex Yorùbá character coding representations using a single code point standard. The generated Unicode point was transformed into HTML entities. The HTML entities generated were converted to its character equivalent for the Yorùbá keyboard. The new Unicode was compared with the results of the two encoding standard scheme using the bit relationships of upper and lower case characters to ascertain conformity of the new Unicode with the standard encoding scheme. This provided some insights about the efficient representation of Yorùbá character scheme. The representation is expected to quickly identify solutions to the design of Yorùbá keyboards and signage.

Keywords: Unicode; Yorùbá character; Yorùbá keyboard; HTML encoding; code point

1. Introduction

Unicode is a standard encoding system used in representing and manipulating text expressed in most of the world's writing systems (McGowan, 2018). The Unicode standard enables the implementation of useful process operation on textual data. Unicode consists of more than 100,000 characters which can be implemented by different character encodings. It defines two mapping methods which are the Unicode transformation format (UTF) and the Universal character set (UCS) (Mastronarde, 2008; Unicode, 2010). The most commonly used encoding schemes are UTF-8, UTF-16, and UTF-32. The Unicode standard code point is an integer value which is used to describe how characters are being represented; usually denoted in base 16 (Mastronarde, 2008; Unicode, 2010).

One of the important input devices for a computer is the keyboard. The keyboard has been used in various aspects of our day-to-day activities for typing different documents such as journals, novels, poem and books in different languages (Sabni et al., 2017; Habib et al., 2012). There is justification in the need to create a keyboard which has keys for Yorùbá language characters complete with diacritical marks. This is to enhance the typing of Yorùbá documents and facilitate efficient use of time spent in creating such documents. Characters are represented on a keyboard by the use of their Unicode. Canonical Equivalence in Unicode is used to resolve the ambiguity introduced by combining characters with acute accent signs and grave accent signs e.g. Latin Small Letter E with Acute (U+00E9) and Latin Small Letter E (U+0065) with combining acute accent (U+0301) are considered to be equivalent (McGowan,

* Corresponding author:
Email: jumoke.ajao@kwasu.edu.ng

KWASU PRESS AND PUBLISHING

2018; Unicode, 2010). The keyboard layout cannot have a simple key combination for all characters. Several operating systems provide an alternative input method that allows access to the entire repertoire. Other means of inputting characters are utilizing handheld devices and by typing a certain sequence of keys on the physical keyboard (Mastronarde, 2008).

Extensive research has been carried out in generating Unicode characters for all the alphabets in different languages such as Latin, Chinese, Arabic, and Yorùbá (McGowan, 2018; Mastronarde, 2008; Unicode, 2010). However, not all the alphabets in the Yorùbá language were captured in the Unicode encoding schemes. This hinders the typing of documents. The present research is based on the concept of using encoding scheme of ASCII and Unicode to generate Unicode for Yorùbá characters with diacritic signs and to design and implement Yorùbá character keyboard for effective and efficient typing of Yorùbá documents. This work aims to generate a single code point for Yorùbá characters with diacritic signs without the need to combine two or more keys on the keyboard to generate a single character with under dot.

2. Yorùbá orthography and character keyboard

The Yorùbá alphabet has a Latin-based character set just like English. Yorùbá is a tonal language which means that the pitch with which words are pronounced dictates the meaning of such words (Ajao et al., 2016; Ajao et al., 2015). Words can have the same basic spelling but the pronunciation may differ based on the pitch with which they are pronounced. Accent and diacritics are rare in English. A character with diacritical marks can generally be represented either as a single pre-composed character or as a decomposed sequence of a base letter plus one or more non-spacing marks. The unavailability of a single code point for a character with a dot under and their combination with an acute accent or grave accent, which represent the high mark tone and low mark tone respectively on the current keyboard, makes it difficult to type Yorùbá documents correctly on the virtual and physical keyboard (Robert, 2002; Unicode, 2010; Zhang, 2014).

The keyboard is emerging as a convenient and effective technology for document creation, editing and digitization in Human-Computer Interaction (Khan et al., 2009). The falling prices of tablet PCs, combined with high-quality handwritten recognition software is evolving a computing paradigm that can be exploited in the generation and documentation of African language resources (Khan et al., 2009; Sabni et al., 2017). The paradigm also holds great potential for language education applications. Considering some of the available Unicode, it was observed that there is no single code point for the following characters *Ẹ́ẹ́ Ẹ̀ẹ̀ Ọ́ọ́ Ọ̀ọ̀* and no single code point for GB gb. It is necessary to implement Yorùbá Character Keyboard which will contain all the Yorùbá language characters including diacritic marks. In this research, we present a system for diacritically marked uppercase and lowercase Yorùbá letters in offline mode. The need to incorporate the Unicode of some of the alphabets that were not captured in the Unicode standard necessitates the formation of Unicode characters for no available character to enhance the typing of Yorùbá documents.

The Yorùbá orthography is a variant of the English alphabet system. This is shown in Figure 1 and Figure 2.

A	À	Á	B	E	È	É	Ẹ	Ẹ̀	Ẹ́
F	G	GB	H	I	Ì	Í	J	K	L
M	N	O	Ò	Ó	Ọ	Ọ̀	Ọ́	P	R
S	Ṣ	T	U	Ù	Ú	W	Y		

Figure 1: Uppercase Yorùbá letters

a	à	á	b	e	è	é	ẹ	ẹ̀	ẹ́
f	g	gb	h	i	ì	í	j	k	l
m	n	o	ò	ó	ọ	ọ̀	ọ́	p	r
s	ṣ	t	u	ù	ú	w	y		

Figure 2: Lowercase Yorùbá letters

3. Materials and methods

The method used for the development of Yorùbá character keyboard included reviewing the existing Unicode standard. The reviewed standard encoding scheme revealed some standard format that must be followed in generating the Unicode that was captured for Yorùbá characters. Table 1 consists of a set of existing Unicode characters and each character has an entity name which starts with an ampersand (&) followed by the letter and the name of the diacritic mark on it. The character also has an entity number which is in the form of a code point that can be either single or double. The double code points are separated by a semicolon. The entity number starts with uppercase u (U) which tells

Table 1: Set of existing Unicode characters (Source: Unicode, 2010)

Character	Entity Name	Entity Number	Character	Entity Name	Entity Number
A	&A	U+0041	a	&a	U+0061
B	&B	U+0042	b	&b	U+0062
D	&D	U+0044	d	&d	U+0064
E	&E	U+0045	e	&e	U+0065
F	&F	U+0046	f	&f	U+0066
G	&G	U+0047	g	&g	U+0067
H	&H	U+0048	h	&h	U+0068
I	&I	U+0049	i	&i	U+0069
J	&J	U+004A	j	&j	U+006A
K	&K	U+004B	k	&k	U+006B
L	&L	U+004C	l	&l	U+006C
M	&M	U+004D	m	&m	U+006D
N	&N	U+004E	n	&n	U+006E
O	&O	U+004F	o	&o	U+006F
P	&P	U+0050	p	&p	U+0070
R	&R	U+0052	r	&r	U+0072
S	&S	U+0053	s	&s	U+0073
T	&T	U+0054	t	&t	U+0074
U	&U	U+0055	u	&u	U+0075
W	&W	U+0057	w	&w	U+0077
Y	&Y	U+0059	y	&y	U+0079

Table 2: Comparison between Yorùbá lower case and uppercase letters Unicode formation

Character	Entity Name	Entity Number	Character	Entity Name	Entity Number
À	&Agrave	U+00C0	à	&agrave	U+00E0
Á	&Aacute	U+00C1	á	&aacute	U+00E1
È	&Egrave	U+00C8	è	&egrave	U+00E8
É	&Eacute	U+00C9	é	&eacute	U+00E9
Ì	&Igrave	U+00CC	ì	&igrave	U+00EC
Í	&Iacute	U+00CD	í	&iacute	U+00ED
Ò	&Ograve	U+00D2	ò	&ograve	U+00F2
Ó	&Oacute	U+00D3	ó	&oacute	U+00F3
Ù	&Ugrave	U+00D9	ù	&ugrave	U+00F9
Ú	&Uacute	U+00D8	ú	&uacute	U+00FA

the system that the entity number is a Unicode followed by a plus sign (+) and the hexadecimal value of the code point which ranges from 0 to F.

Table 2 compares some of the available Unicode for Yorùbá lower case letters and uppercase characters. Table 3 compares the existing Unicode formation and the disparity between lower case letters and the uppercase letters to establish the standard format of generating code points for upper case and lower case letters. This is used as a guide for generating Yorùbá characters that were not captured in the existing Unicode.

The generation of a single code point for characters with double code point and the point of examining the standards coding scheme using the bit relationships of upper and lower case characters. CD = BL + DS

Where CD is the character with a diacritic sign and can be obtained by adding a base letter (BL) to the combining diacritic sign (DS).

Table 3: Comparison of Yorùbá uppercase letters and lowercase letters

Char	Hex Val	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Ẹ	1EB8	0	0	0	1	1	1	1	0	1	0	1	1	1	0	0	0
ẹ	1EB9	0	0	0	1	1	1	1	0	1	0	1	1	1	0	0	1
Ọ	1ECC	0	0	0	1	1	1	1	0	1	1	0	0	1	1	0	0
ọ	1ECD	0	0	0	1	1	1	1	0	1	1	0	0	1	1	0	1
Ṣ	1E62	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	0
ṣ	1E63	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	1
A	0041	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
a	0061	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
B	0042	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
b	0062	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0
D	0044	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
d	0064	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

Table 4: The double code point for E with under dot and acute accent

Char	Hex Val	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Ẹ	1EB8	0	0	0	1	1	1	1	0	1	0	1	1	1	0	0	0
̀	0300	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
Ẹ̀	21B8	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0	0

Table 5: The double code point for e with under dot and acute accent

Char	Hex Val	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
ẹ	1EB9	0	0	0	1	1	1	1	0	1	0	1	1	1	0	0	1
̀	0300	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
ẹ̀	21B9	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0	1

Table 6: Single code point for Yorùbá letter E with an acute diacritic sign and under dot

Char	Hex Val	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Ẹ̀	21B8	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0	0
ẹ̀	21B9	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0	1

Table 7: The single code point for uppercase character ẹ with a grave accent

Char	Hex Val	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Ẹ	1EB8	0	0	0	1	1	1	1	0	1	0	1	1	1	0	0	0
́	0301	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
Ẹ́	21B9	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0	1

Table 8: The single code point for character ẹ̀ with a grave accent

Char	Hex Val	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
ẹ	1EB9	0	0	0	1	1	1	1	0	1	0	1	1	1	0	0	1
̀	0301	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
ẹ̀	21BA	0	0	1	0	0	0	0	1	1	0	1	1	1	0	1	0

Table 9: Comparison between the generated single code point of character Ẹ̀ and ẹ̀

Char	Hex Val	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Ẹ̀	21B9	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0	0
ẹ̀	21BA	0	0	1	0	0	0	0	1	1	0	1	1	1	0	1	1

The comparison established that the uppercase and lowercase letters have their differences in its first bit (b0), i.e., uppercase characters with a dot are followed immediately by its lowercase characters. Comparing characters without dot shows that their difference lies in the six-bit (b5). I.e. uppercase character (0041 – 005A) are followed by six symbols before the lowercase character (0061 – 007A). The detailed double code point for character E and e with a diacritic sign is shown in Table 4 and Table 5. Single code point for the character with under dot and diacritic sign, the value of the character with under dot was added to the code point of the diacritic symbol which resulted into Table 6 and Table 7. The single code point was formulated from each of the identified Yorùbá characters which are represented in Table 6 and Table 7.

Table 6 shows the single code point for character Ẹ̀, which is generated by adding the Unicode of the base letter Ẹ (1EB8) to the Unicode of the diacritic mark ̀ (0300) on it. Table 7 shows the single code point for character ẹ̀ is been generated by adding the Unicode of the base letter ẹ (1EB9) to the Unicode of the diacritic mark ̀ (0300) on it. Table 8: the single code point for character ẹ̀ with grave accent Table 9 shows the comparison between the generated single code point of character Ẹ̀ and ẹ̀. The difference of the Unicode lies in the first bit (b_0).

3.1. The keyboard encoding scheme

The deployment of the Yorùbá character keyboard was developed by decoding and translating the Unicode generated, loading of decoded Unicode entities, creating an HTML entity database, and writing of decoded entities into Yorùbá characters. Detail of this process is explained in this section.

3.2. HTML entities decoding

To process the HTML entity, an HTML parser must ascertain what character encoding scheme is used to encode the content. The HTML parser first detects the character encoding in the HTML entities. If there is no valid character encoding information detected, a predefined character-encoding scheme will be invoked. The HTML parser processes the HTML entities by using byte sequences.

3.3. HTML character entity converter

The HTML character entity converter is designed to translate HTML entities Unicode to corresponding byte sequences of Unicode's code point. The converter can handle both character entity references and numeric character references. There are 107 character entity references defined for the HTML entity, which act as mnemonic aliases for certain characters. The converter maintains a mapping table between the 107 character entity references and their represented byte sequences in a hexadecimal number. When a character entity reference is detected by the converter, it replaces the entity with its associated byte sequences. For numeric character references, the converter performs a real-time decoding process on it. The converter will convert the character reference (from decimal base 10) number to byte sequences.

4. Result and discussion

Since one code point can only be assigned to one character, the character ẹ̀ and Ẹ̀ generate the same code

Table 10: Code point generated for ẹ plus one bit added to it

Char	Hex Val	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
ẹ	1EB9	0	0	0	1	1	1	1	0	1	0	1	1	1	0	0	1
̀	0300	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1 bit	0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
ẹ̀	21BA	0	0	1	0	0	0	0	1	1	0	1	1	1	0	1	0

Table 11: Code point generated for g and b plus one bit added to it

Char	Hex Val	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
g	0062	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0
1 bit	0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
b	0067	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1
gb	006A	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0

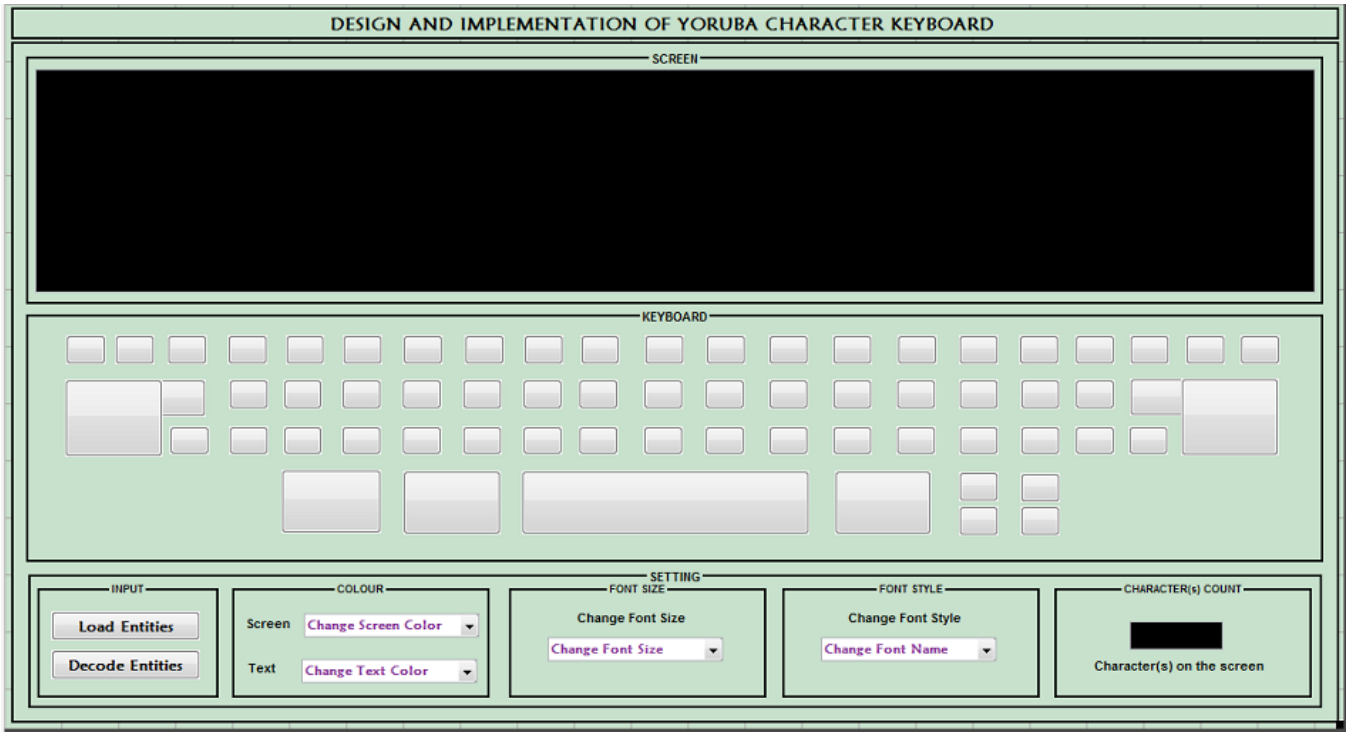

Figure 3: The GUIDE design and development environment

point by using the general formula CD = BL + DS. CD is the character with a diacritic sign and can be obtained by adding a base letter (BL) to the combining diacritic sign (DS). The result generated is more accurate by adding one bit to the code point generated which is shown in Table 10 and Table 11.

The input to the developed Yorùbá character keyboard system is the plain code generated for Yorùbá character with under dot and diacritic signs. Each character with its entity name and number is to be translated and decoded by the system. The output of the developed Yorùbá character keyboard application software correctly translates and decodes into HTML entity. The first stage in the system evaluation is to load in HTML entities by clicking the 'Load Entities' button. The system then creates a database to store the HTML entities Unicode. Once the database has been created, the system writes the character HTML entities to the database for further processing. The detail of the application developed for the interface for the Keyboard is shown in Figure 3. Figure 4 displays the transformation of the Unicode generated to HTML entities. Figure 5 displays the

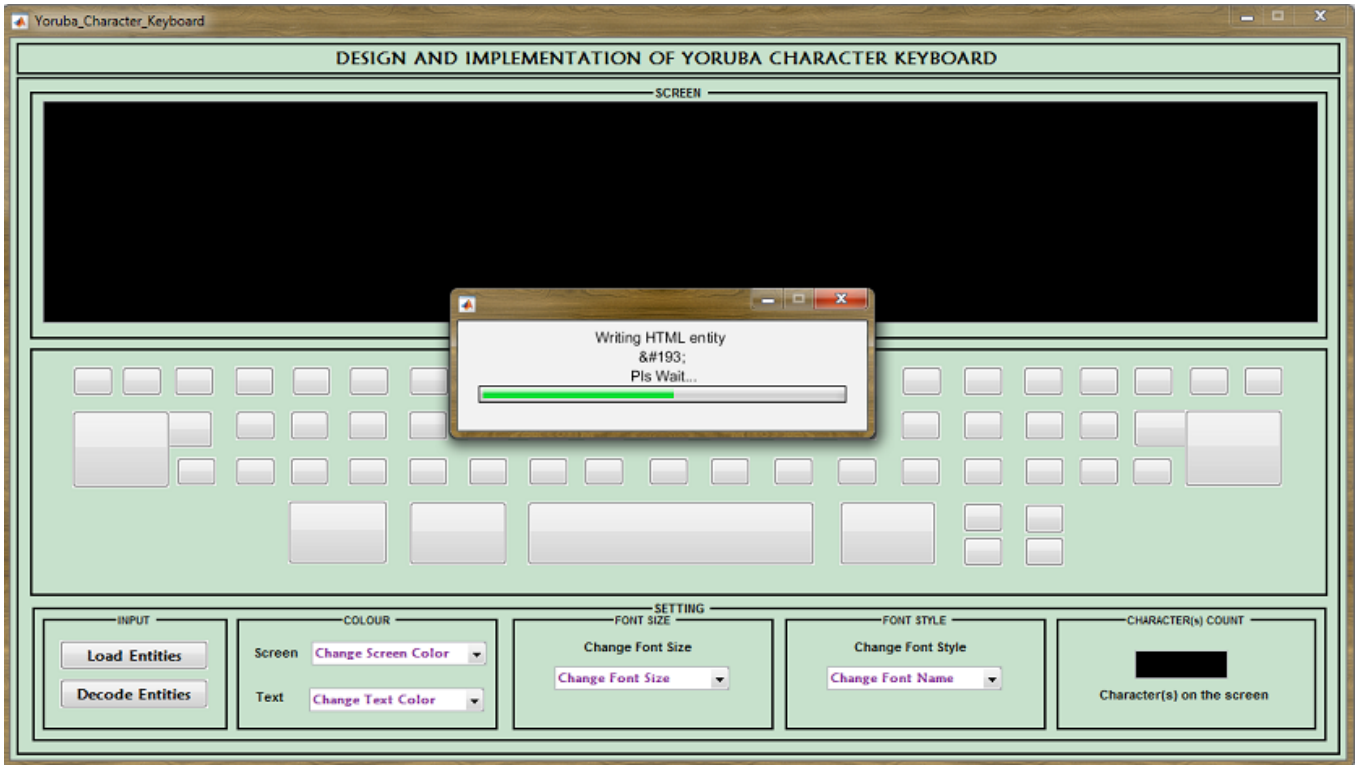

Figure 4: Writing HTML entities Unicode to database

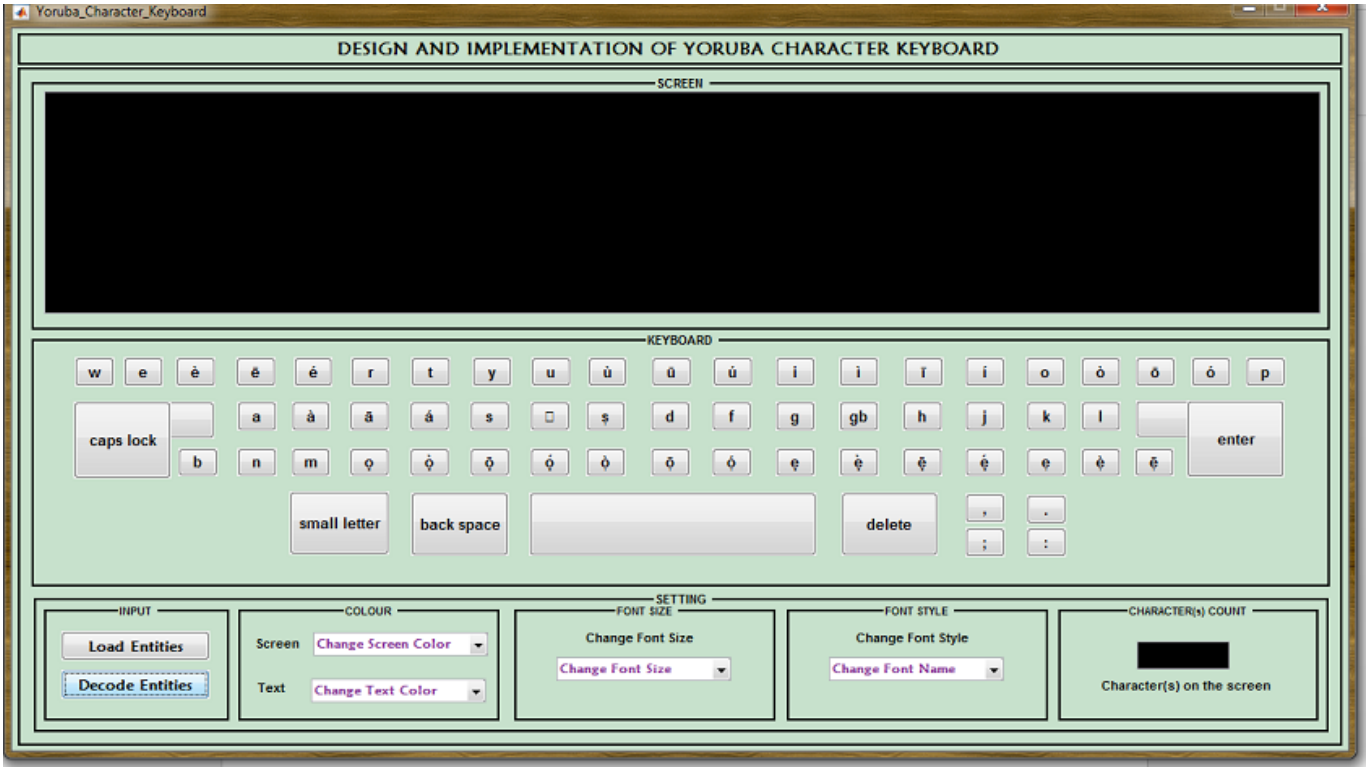

Figure 5a: HTML character entity Unicode converter

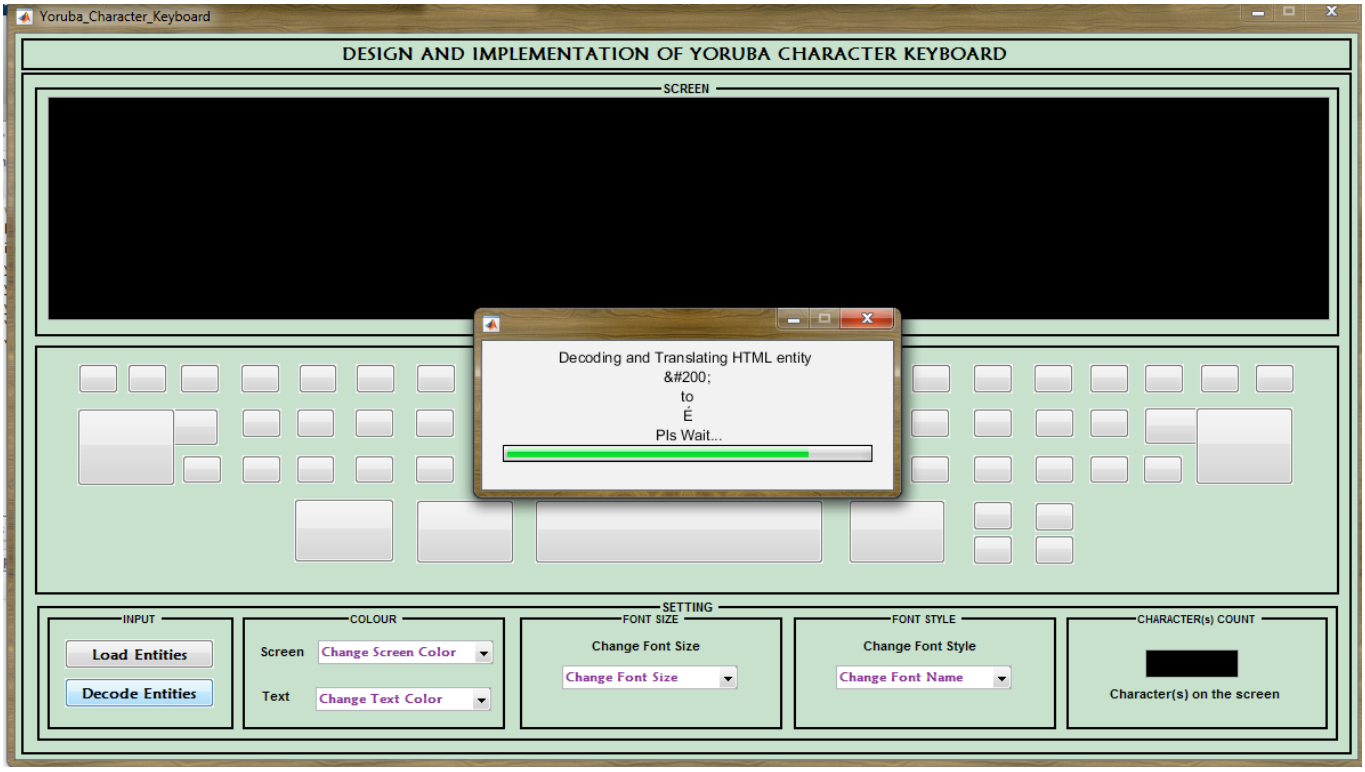

Figure 5b: HTML character entity Unicode converter

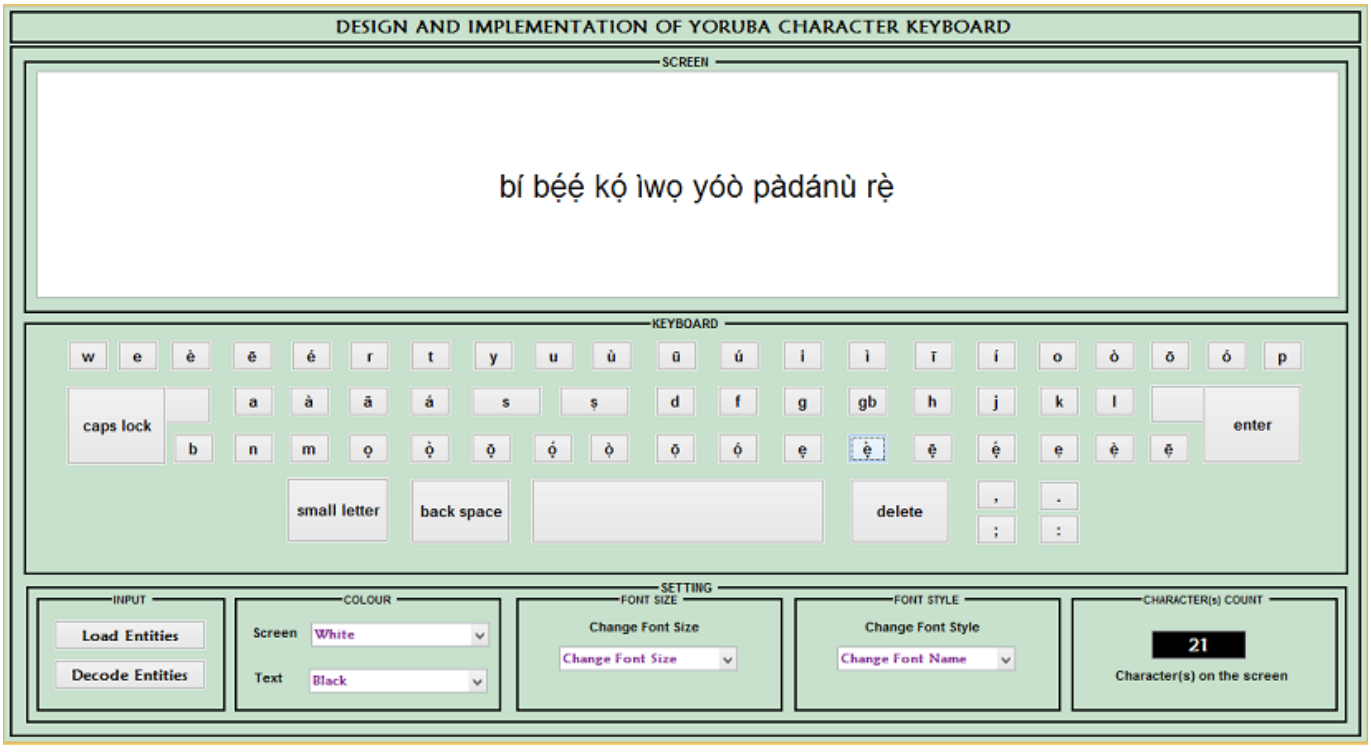

Figure 6: Lowercase Yorùbá character showing on keyboard

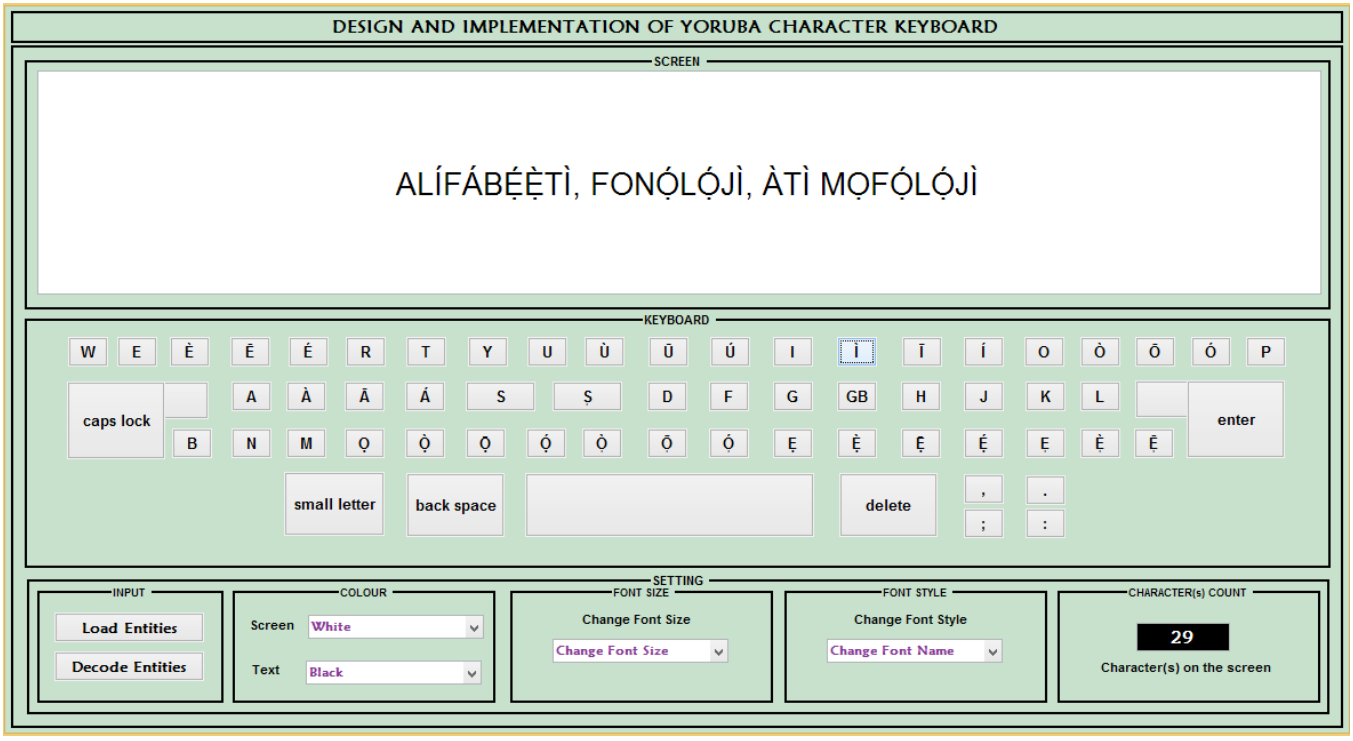

Figure 7: Uppercase Yorùbá character showing on keyboard

converted HTML entities into Yorùbá characters on the Keyboard to validate the designed Keyboard. Samples of Yorùbá characters and text were typed on the keyboard and are shown in Figure 7.

Once the HTML entries have been translated to its corresponding character, each of the character will be loaded in an appointed push button for it to be accessed when clicked and displayed on the screen.

5. Conclusion and recommendation

The design of the Yorùbá keyboard presented is based on generating a single code point for Yorùbá characters that are not captured in the conventional keyboard. This was achieved by making the diagraph character as a single character instead of two separate characters. It was observed that character Ẹ́ and ẹ̀ have the same code point (21B9), by adding bit one to the lowercase character ẹ̀ (21B9) and ẹ́ (21BA); it changes the generated code point for the characters to (21BA) and (21BB) respectively. The combination of character g and b and the additional 1-bit added makes it to be treated as a single character instead of separate keys on the existing keyboards. The generated single code point for the Yorùbá characters will be communicated to the Universal encoding standard for further processing. It is therefore recommended that the keyboard be extended to include characters from other languages that are not present on the standard keyboard.

References

Ajao, J. F., Olabiyisi, S. O., Omidiora, O. O., & Odejobi, O. O. (2015). Yorùbá handwriting word recognition quality evaluation of preprocessing attributes using information theory approach. *International Journal of Applied Information System IJAIS, 9*(1), 18-23.

Ajao, J. F., Olabiyisi, S. O., Omidiora, O. O., & Okediran, O. O. (2016). Hidden Markov model approach for offline Yorùbá handwritten word recognition. *British Journal of Mathematics and Computer*, *18*(6), 1-20.

Habib, A., Iwatate, M., Asahara, M., & Matsumoto, Y. (2012). Keypad for large letter-set languages and small touch-screen devices (case study: Urdu). *International Journal of Computer Science Issues*, *9*(3), 47-58.

Khan, M., Khan, M., & Ali, M. (2009). Design of Urdu virtual keyboard. *Proceedings of the Conference on Language and Technology.* http://www.cle.org.pk/clt09/download/Papers/Paper19.pdf

Mastronarde, D. (2008). Before and after Unicode: working with Polytonic Greek. *Montreal APA Unicode Presentation*. https://ucbclassics.dreamhosters.com/djm/unicodeTalk/BeforeAndAfterUnicode.pdf

McGowan, R. (2018). A summary of Unicode consortium procedures, policies, stability, and public access tools. https://doi.org/10.17487/RFC3718

Robert G. J. (2002). Emerging technologies for multilingual computing. *Language, Learning and Technology, 6*(2), 6-11.

Sabni, T., Yavatkare, J., Divyanand, S., & Kakde V. (2017). Laser projection virtual keyboard: a laser and image processing based human-computer interaction device. *International Journal of Advanced Research in Computer and Communication Engineering*, *6*(3), 3297-2007.

Unicode. (2010). *Unicode Consortium*. http://www.unicode.org

Zhang, X. (2014). *Optimization of switch virtual keyboard by using computational modelling* [Master's thesis). Purdue University West Lafayette, Indiana.