Blinks to Speech: Using EEGs to Detect Eye Blinks for Morse Code Communication

Esha Karlekar, Atin Pothiraj

Thomas Jefferson High School for Science and Technology

Abstract

Eye blinks, specifically deliberate ones, and their duration have been isolated and detected from EEG signals. Morse code consists of a series of dots and dashes and has been communicated through blinks in the past, where short and long blinks correspond to dots and dashes, respectively. Our project aims to build a system that uses an EEG to detect eye blinks and translates them into morse code, then to text to speech. We decided to use left and right blinks as dots and dashes, respectively. This can serve as means of augmentative and alternative communication (AAC) for individuals with afflictions like cerebral palsy, motor neuron disease, quadriplegia, and locked-in syndrome who are unable to speak verbally but have control over eyelid movement.

Engineering Goal

The goal of this project is to create a system that is able to read in the output of an EEG and isolate deliberate eye blinks. This data will then be streamed through an OSC server to a Python program that is able convert these eye blinks into morse code. We plan to utilize the international morse code system. Our goal is to use pre-specified lengths of time to correspond to short blinks vs. long blinks and different periods of rest to correspond to spaces between letters and words. Once we have received the blinks from the OSC server, we plan to translate this into morse code, which we will then translate to English and then read aloud on the click of a button using a text-to-speech program. This will allow a person with the EEG, with a slight delay to ensure full processing of the word/phrase, to communicate aloud in real-time through just a series of blinks.

Background

1.1 - EEGs (Electroencephalogram)

An EEG (electroencephalogram) is a device that records electrical patterns in your brain. The billions of neurons in the brain produce small electrical signals by producing ionic currents between each other, forming patterns. An EEG is able to detect these patterns, called brain waves, through electrodes placed on your head. This is a non-invasive device. The EEG will then amplify that signal and record those patterns in a wave pattern (Mandybur, 2018).

1.1a - EEGs to detect eye blinks

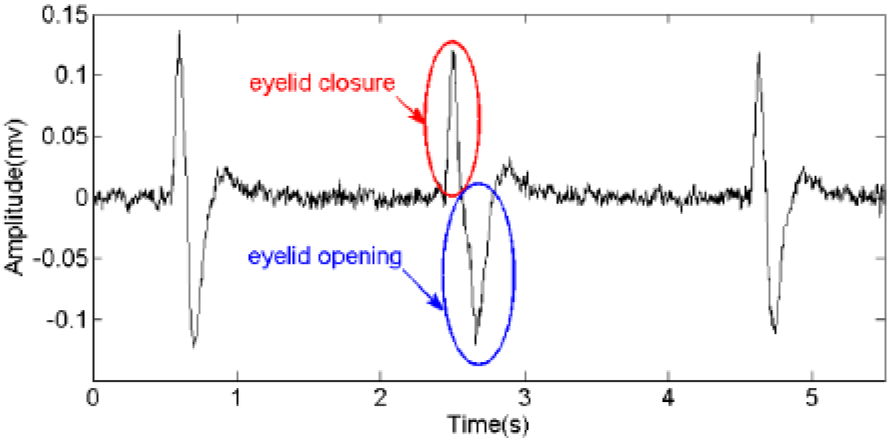


Figure 1: How eye blinks show up on an EEG output (Abo-Zahhad, 2015).

Eye blinks are very apparent in an EEG reading (Figure 1). Traditionally, natural eye blinks are seen as one of the largest artifacts in EEG data and are typically removed after the data is collected (Chang, 2015). However, deliberate eye blinks produce much clearer signals in EEG data and significant differences appear in all channels from an EEG between deliberate and involuntary eye blinks (Kanoga, 2016). Therefore, deliberate eye blinks can be used as a clear marker in an EEG output, even having been used in the past to activate home lighting systems (Rani, 2009). The duration of an eye blink can additionally be detected by measuring the difference between the peak for an eyelid closure and an eyelid opening (Abo-Zahhad, 2015).

1.2 - Morse Code Communication

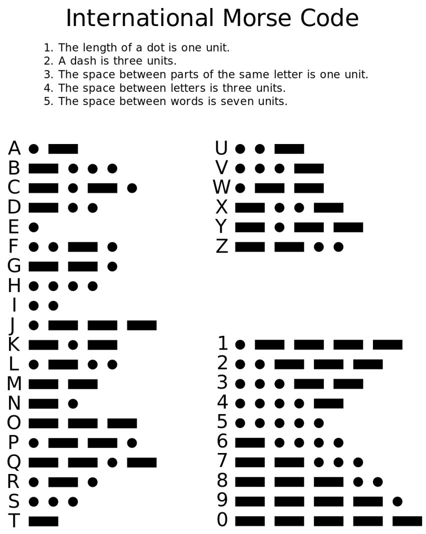


Figure 2: International Morse Code Translations (Snodgrass, 1922)

International Morse Code encodes all 26 letters of the alphabet, in addition to the 10 Arabic numerals (Figure 2). Each symbol is formed by a series of dots (*dit*) and dashes (*dah*), where the length of a *dit* is a basic time measure in Morse code transmission and a *dah* is three times the length (International Telecommunications Union, 2009). It is traditionally communicated through an on-off keying of radio waves, electrical current, visible light, or sound waves (War Department, 1939).

1.2a - Morse Code Communication using eye blinks

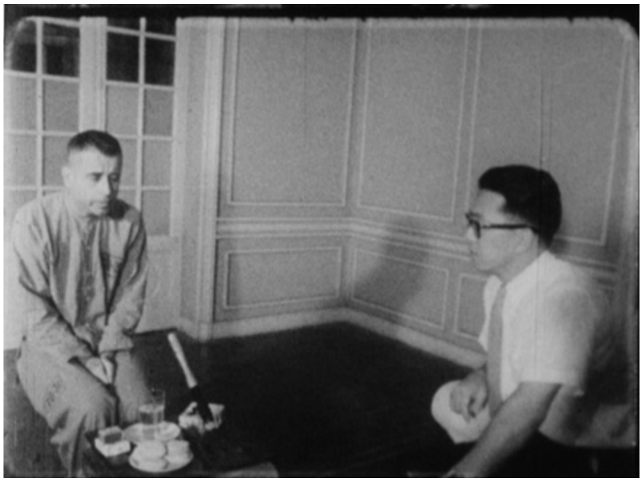


Figure 3: Comdr. Denton of the U.S. Navy, captured by the North Vietnamese, used morse code to blink out "T-O-R-T-U-R-E" while being filmed for a propaganda campaign (The National Archives, 1987).

A combination of long and short eye blinks have been used as a means of communicating through morse code in the past, specifically by prisoners of war (Figure 3) (The National Archives, 1987).

1.3 - AAC (augmentative and alternative communication)

AAC (augmentative and alternative communication) describes all the means of communicating besides talking that can be used for those unable to communicate through speech. Low-tech options include writing, drawing, pointing to photos/pictures/words on a communication board, and gesturing. Higher-tech options include an electronic means of communication or speech-generating devices. However, one major limitation in current AAC is that those who are unable to move their arms or limbs are unable to use many of the cheaper options available. This means they either can only maintain very low levels of communication or must pay for very expensive speech-generating devices (American Speech-Language-Hearing Association, 2021). A more cost-effective alternative needs to be explored.

1.3a - Specificity of AACs in the status quo

Currently, speech-generating devices are incredibly expensive because they must be very tailored to each individual. One famous example is that of Stephen Hawking. Upon losing his ability to speak, he was given a communication device by Intel which uses movements in his cheek muscles to navigate a keyboard. However, according to one of the developers, much of the system's design was "[hinged] on Stephen. [They] had to point a laser to study one individual." This caused the device to take years to make and unable to be used on other individuals easily (Medeiros, 2015). Therefore, it is necessary to develop a cheaper, less individualized system of communication through speech-generating devices.

1.4 - Afflictions where our technology could be applicable

We believe our technology could be applicable to a wide variety of people, specifically those who may have an affliction that may prevent them from communicating verbally but still have full brain activity and control over their orbicularis oculi (OO) and levator palpebrae superioris (LPS) muscles, the muscles that control eye blinks (Fitzakerley, 2015). These cases are listed below. For people with severe motor disability, it becomes difficult to communicate through large muscle movements so electroencephalography becomes necessary (Pinheiro, 2011).

1.4a - Cerebral palsy

Cerebral palsy is a disorder that affects a person’s ability to move and communicate. It can severely affect someone’s ability to communicate because it is difficult to control the muscles surrounding the mouth and tongue. Because of this, 25% of people with cerebral palsy are unable to talk. However, many are able to move their eyes and blink, which is why many who are able to afford it use eye-gaze technology, which allows them to communicate by holding their eye gaze on a certain option for a period of time. These devices can cost between $4000 to $19000 (Cerebral Palsy Alliance, 2016). Therefore, a more cost-effective device is necessary.

1.4b - Motor Neuron Disease

Motor neuron diseases are a set of conditions that cause the nerves in the brain and spine to lose function over time. Since motor neurons control the ability of the muscles to function, people who have this affliction are usually unable to move for the most part (Brazier, 2019). However, even in the last stages of one such disease (ALS), patients are able to move their eyes even having lost control over most of their motor neurons (Yee, 2014).

1.4c - Quadriplegia

Quadriplegia is a form of paralysis that affects all four limbs so patients are unable to move below the neck. If the injury or affliction affects the vertebrae nearest to the skull (C1-C3 vertebrae), this can affect the ability of the patient to speak but many can move their eyes (Kuriakose, 2020).

1.4d - Locked-In Syndrome

Locked-in syndrome is typically caused by a stroke and people with this neurological disorder have a complete paralysis of their lower face or body. This impedes their ability to eat, speak, and communicate. However, they can still “see, hear, move their eyes up and down, and blink”. Thus, many must try to communicate with their eyes (Maiese, 2020). In fact, a French author with locked-in syndrome was able to dictate an entire book by blinking his left eye. However, this took an incredible amount of time and effort on the part of the translator (Dalhburg, 1997).

1.5 - Technological Components/Unique Methods

1.5a - EMOTIV Headset

Figure 4: EMOTIV EPOC X Headset

The EMOTIV EPOC X Headset (Figure 4) is an EEG (described in section 4.1). It is developed by EMOTIV, a company dedicated to making brain wear for personal use and experimentation. It utilizes saline-soaked felt pads as its sensor material and has 14 channels or nodes that examine different areas of activity within the brain. The sampling rate can be configured to be 128 SPS or 256 SPS and uses a single ADC (analog-to-digital) sampling method. It can detect blinks, winks (left and right), surprise, frowns, smiles, clenches, laughs, and smirks (left and right) with high accuracy (EMOTIV EPOC X Headset).

1.5b - OSC Stream

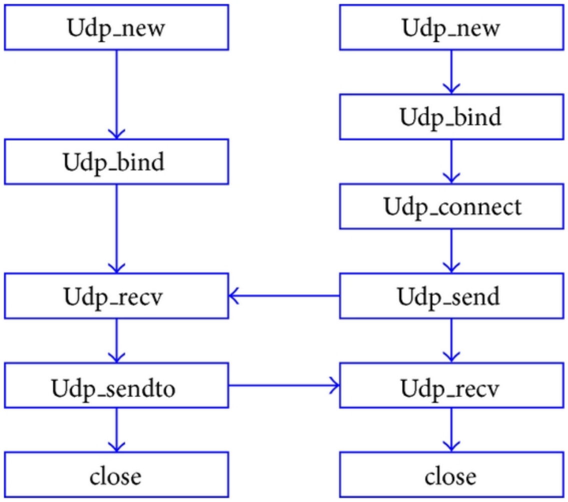


Figure 5: UDP Packet Protocol

OSC (Open Sound Control) streams are used to send audio between programs across different languages/networks. OSC messages consist of 1. A type tag string (compact string representation of argument types), 2. An address pattern (like a file system path), 3. Arguments (in binary form w/ four-byte alignment), and 4. An (optional) time tag. Data is sent in packets (of which there are multiple types). The EMOTIV headset (1.5a) uses a UDP (User Datagram Protocol) packet in order to transfer information between different devices. In Figure 5, the left side of the UDP protocol describes the computer process and the right side would describe the EMOTIV headset.

Essentially, the EEG pushes UDP send packets while the OSC server outputs UDP receive packets (shown by UDP_new, UPD_bind, and UPD_connect in Figure 5). When the data is transmitted between the packets (UDP_recv, UDP_send, and UDP_sendto), each of the packets can be closed. The UDP data can then be received by the OSC server and processed further with the translation scripts.

1.5c - Google Cloud Text to Speech API

The Google Cloud Text-to-Speech API allows users to convert text into natural-sounding speech using an API powered by Google AI. It includes features such as a custom voice, voice tuning, and tags that allow users to add pauses, numbers, formatting, etc. It also allows users to choose the speed at which the text is spoken and offers many different language options. After inputting the to-be-translated text string, a user can then save the output as a .mp3 file and play it using their own OS or an external speaker (1.5d).

1.5d - Turbo-Flask

In order to change sections of a website without refreshing the whole browser, the Turbo-Flask library can be used for Flask-based websites. It provides the advantage of not having to write javascript code while still keeping extremely low refresh rates. Turbo-Flask does this by threading the process, meaning a computer can handle the website updates while still performing other functions (like hosting the website). The three primary functions are starting the threading, updating the load, and injecting the load. Starting the thread is simple, it just requires the update load function to be specifically called. Updating the load is what dynamically updates part of the page. By splitting up the webpage into blocks, the update load function can render the new version of the block whenever needed. Injecting the load is where the translation data is received. From here, the inject load can send the data to the update load function, where it is inserted into what is called a block template, which is how it can be displayed using an HTML file.

1.5e - Farnsworth Spacing

Farnsworth spacing is a method of teaching Morse code in which character elements and inter-elements have a high wpm but characters and word spaces are longer than usual. This promotes an easier use of morse code for novices because it promotes learning characters as patterns instead of analytically identifying a character. This leads to a faster translation of morse code and a better understanding of Morse code overall (Walker, 2016).

Materials

EMOTIV EPOC X Headset ($849.00)
EMOTIVBCI Software - Free
BCI-OSC Extension - $99.00
Google Cloud Text to Speech API
Python

Implementation

2.1 Metric for Success

Our metric for success was making a fully functional system that can take eye blinks from the EEG and ultimately speak aloud the morse code translation of those blinks. Specifically, our original metric for success involved setting certain benchmarks for short and long blink lengths that could correspond to dots and dashes, respectively. We then hoped to implement a working OSC server to read the output from the EEG to be able to parse through that data in Python. After that, we hoped to set up a working translation program that translates the user's EEG input in real time, which would then be spoken aloud. To do this, we also wanted to implement a working UI that is relatively user-friendly. In the end, we adjusted our metric for success to use left and right blinks along with looking to the left and right to correspond with dots and dashes and the various spaces between letters and words for increased accuracy (described in more detail below).

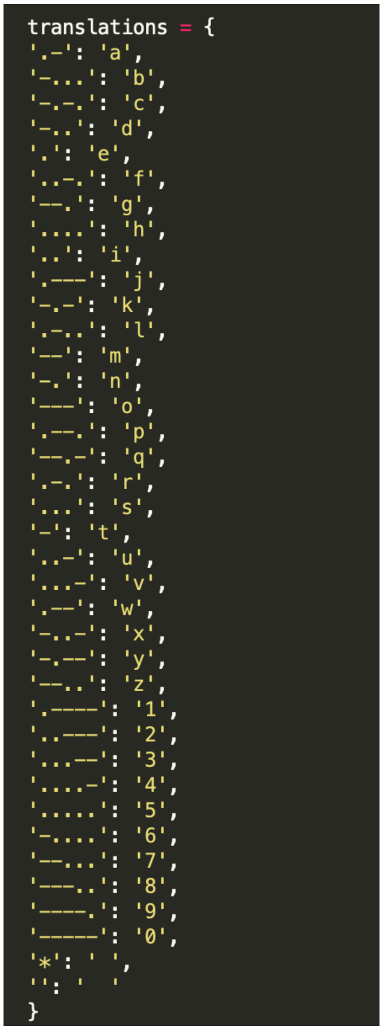
2.2 EEG/OSC Server Connection

We used the EMOTIV Epoc X EEG for our project, which came with a BCI-OSC extension to deliver raw data. OSC (Open Sound Control) streams are usually used to send audio between programs across different networks, but can also be used to transmit EEG data. The reason we decided to use an OSC server is because it was already implemented with the EMOTIV framework as the means to parse through the EEG data with another coding language (such as Python, Arduino, etc). We wanted to write the majority of our code in Python, so we set up an OSC server (which was also coded in Python) that could read from the EEG. With the python OSC package, this data could then be manipulated and translated in self-serving python scripts. The EMOTIV headset uses a UDP (User Datagram Protocol) packet in order to transfer information between different devices. By setting up an OSC-server on the local network, the server is constantly scanning to find packets that the EEG sends out. The packets just contain a simple string denoting a certain action, like the word "blink" if a blink is detected. The OSC server then takes this data and outputs it directly onto a text file.

After the data is sent to a file and read by our translation/speech script. As we worked with the EEG more and kept experimenting, we realized that using long and short blinks simply allows for a lot of error and the translations do not work well. We also realized that preset amounts of time for periods of no blinks to correspond to spaces between letters and words also did not work well. This was because the EEG was never consistent in how long short and long blinks were and it was extremely difficult to set an accurate benchmark. Additionally, trying to determine spacing between words and letters was also difficult because it takes the user time to look at what they should blink out to get their desired letter and this is not always consistent, depending on how familiar they are with the letter translation. After extensive testing, we realized that we simply could not get an accurate translation if we used left and right blinks and a certain amount of time of no expression for time between letters and words. Therefore, for simplicity sake and for more accurate and consistent translations, we decided to use left and right blinks to represent the dots and dashes in morse code, respectively. We also used looking to the left and looking to the right (both of which are simple for the EEG to detect), to represent a new letter and a new word, respectively. We output the user’s actions into a text file, encoding a left blink as “blinkL” or look right as “lookR”, so that we have a simple way to parse through our output when we are translating.

2.3 Translation Code

The latter half of our project engages in translating the output from the OSC server into morse code and then into text and then speech. This is done completely in Python. We first read in the output from the .txt file with the output of the OSC server in real time. Each line of the text file corresponds to either a left blink (a dot), a right blink (a dash), looking to the left (new letter), or looking to the right (a new word). We then create a morse string that is more simple to parse through, with a period (.) representing a dot, a hyphen (-) representing a dash, a right slash (/) representing a new letter, and an asterisk (*) representing a new word. We then take this morse string and convert it into English text. We first split this string up into the individual components between the slashes and then translate each individual letter using a simple dictionary in Python, where we map the relevant combinations of dots and dashes to the appropriate letters and numbers. This dictionary can be seen below.



```
translations = {
'.-': 'a',
'-...': 'b',
'-.-.': 'c',
'-..': 'd',
'.': 'e',
'..-.': 'f',
'--.': 'g',
'....': 'h',
'..': 'i',
'.---': 'j',
'-.-': 'k',
'.-..': 'l',
'--': 'm',
'-.': 'n',
'---': 'o',
'.--.': 'p',
'--.-': 'q',
'.-.': 'r',
'...': 's',
'-': 't',
'..-': 'u',
'...-': 'v',
'.--': 'w',
'-..-': 'x',
'-.--': 'y',
'--..': 'z',
'.----': '1',
'..---': '2',
'...--': '3',
'....-': '4',
'.....': '5',
'-....': '6',
'--...': '7',
'---..': '8',
'----.': '9',
'-----': '0',
'*': ' ',
'': '  '
}
```

Figure 6: Translation Dictionary

We translate this in real time, constantly updating our final output based on changes to the original text file. We then write this to another file which the Turbo Flask server can read from (elaborated on in 2.4a). After this, upon the click of a button by the user, we can call a function that takes the current English translation and converts it from text into speech using the Google Cloud Text to Speech library, which outputs an .mp3 file that we immediately read.

2.4 User Interface

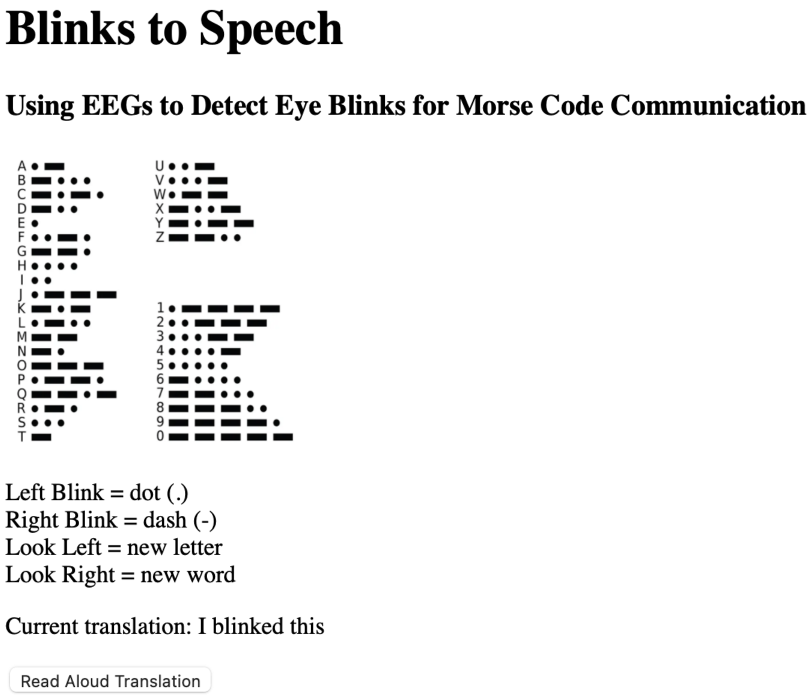
Our user interface was coded using a mixture of Python and HTML/CSS. We implemented a Flask server (2.4a), which had to be done in Turbo-Flask so that the page could update in real time without having to reload the page or user manipulation. We then used HTML to display the UI, with CSS for formatting.

2.4a Turbo-Flask

The translation script is outputted into a file where it can then be read by the Turbo-Flask server. The Turbo-Flask code consists of an inject load and update load function (described in 1.5d). The inject load function reads in fully translated words from the text file that updates in real time (described in 2.3) and sends it to the update load function. From here, the update load function takes the translation and renders it on the block template, where it can be displayed on the website. Before all of this happens, however, the threading is set up so this entire process can happen simultaneously with running the server.

Additionally, we have a button on our UI that allows the user to speak aloud the current translation, utilizing the Google Text to Speech library.

2.4b HTML/CSS



Blinks to Speech

Using EEGs to Detect Eye Blinks for Morse Code Communication

Left Blink = dot (.)
Right Blink = dash (-)
Look Left = new letter
Look Right = new word

Current translation: I blinked this

Figure 7: User Interface

The HTML code provides a template that is rendered by the flask server. The HTML code makes use of the block template to display the translation as it updates in real time, allowing the user to see the words they are forming in real time before they choose to speak it aloud. Additionally, a key displaying what left and right blinks and left and right looks correspond to is shown as well. There is also a diagram showing what dot and dash combinations correspond to what translation in international morse code. When the user is ready for the translation to be spoken aloud, they can click the button shown and the current translation will be read aloud from the computer.

2.5 Connection of Individual Systems

The flowchart below demonstrates how all of the individual parts listed above (2.1-2.4) come together.

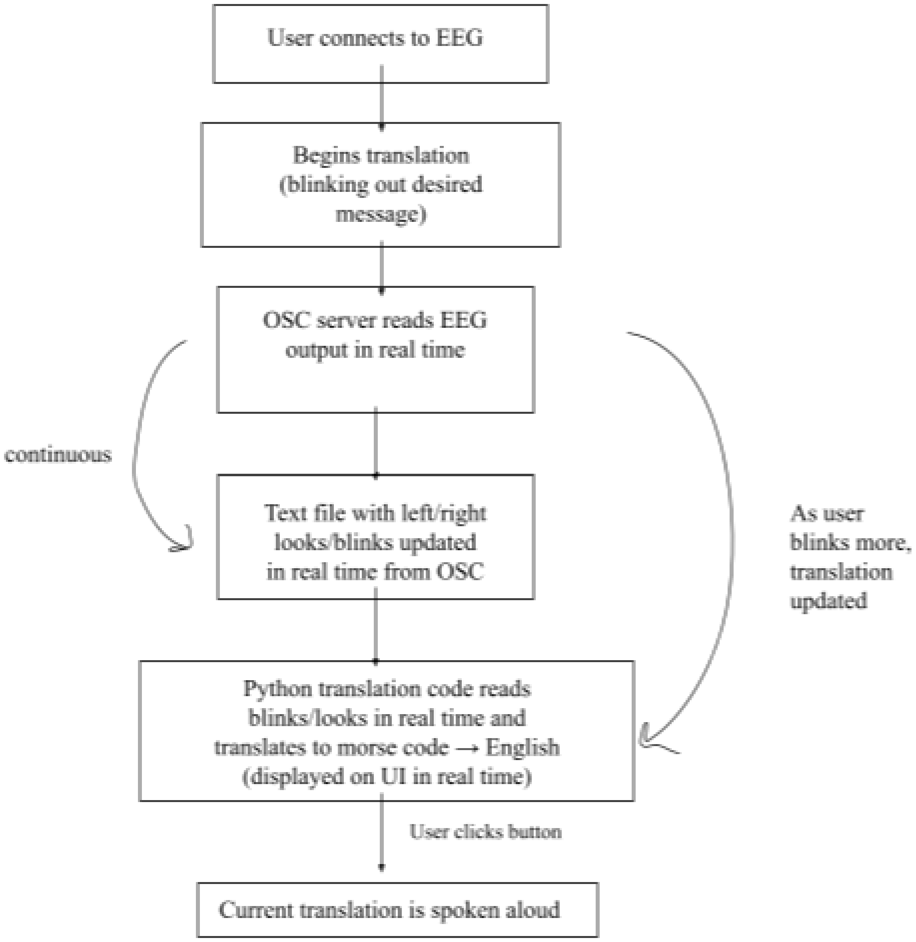


Figure 8: Flowchart of entire system

First, the user connects to the EEG and begins to blink out their desired message. Since the OSC server is running, it reads the output from the EEG as the user blinks. As the OSC server writes to a text file, our translation code updates the translation in real time as it reads from the text file. The translation code translates the blinks/looks to text, an almost instantaneous and simple operation. This translation is continuously updated and is sent to the Turbo Flask server, which outputs it onto the UI in real time so the user can see what is being translated. When the user clicks the appropriate button, the current translation is spoken aloud.

Results

Overall, we met our adjusted metric for success since we were able to implement a complete working system from start to finish that takes the users blinks from the EEG and outputs a translation. Although we were not able to meet our original goal of using short and long blinks to correspond to dots and dashes, our accuracy was much higher when we used left and right blinks and left and right looks. We are currently testing how accurate the translation is by doing more extensive testing with the EEG, but we have been able to get accurate translations a couple of times. As we conduct more tests, we can fine tune our project to better fit the needs of a potential user.

Future Steps

To improve this project, we hope to be able to translate shorter and longer blinks into dots and dashes, as is done in more conventional morse code communication instead of the left or right blinks that we have implemented now. We could also implement more complex ways of marking the spacing between letters and words (using different periods of times of rest) and possibly implementing the Farnsworth spacing method described in the background section of the paper above. We also would hope to improve the look and functionality of our UI to make it more accessible to our target audience.

References

Abo-Zahhad, M., Ahmed, S. M., & Abbas, S. N. (2015). A New EEG Acquisition Protocol for Biometric Identification Using Eye Blinking Signals. *International Journal of Intelligent Systems and Applications*, *7*(6), 48–54. https://doi.org/10.5815/ijisa.2015.06.05

American Speech-Language-Hearing Association. (2021). *Augmentative and Alternative Communication (AAC)*. ASHA. Retrieved October 14, 2021, from https://www.asha.org/public/speech/disorders/aac/

B Abd Rani, M. S., & Bt. Mansor, W. (2009). Detection of eye blinks from EEG signals for home lighting system activation. *2009 6th International Symposium on Mechatronics and Its Applications*. Published. https://doi.org/10.1109/isma.2009.5164828

Brazier, Y. (2019, July 10). *What is motor neuron disease?* Medical News Today. Retrieved October 14, 2021, from https://www.medicalnewstoday.com/articles/164342

Cerebral Palsy Alliance. (2016, September). *Eye-gaze control technology*. Retrieved October 14, 2021, from https://cerebralpalsy.org.au/our-research/about-cerebral-palsy/interventions-and-therapies/eye-gaze-technology-for-children-and-adults-with-cerebral-palsy/#1473736050897-5940a2c1-7648

Chang, W. D., Cha, H. S., Kim, K., & Im, C. H. (2016). Detection of eye blink artifacts from single prefrontal channel electroencephalogram. *Computer Methods and Programs in Biomedicine*, *124*, 19–30. https://doi.org/10.1016/j.cmpb.2015.10.011

Dahlburg, J. (2019, March 5). *A Writer's Spirit Soared With a Blink of His Eye*. Los Angeles Times. Retrieved October 14, 2021, from https://www.latimes.com/archives/la-xpm-1997-03-15-mn-38526-story.html

Fitzakerley, J. (2015). *Eyelid Movements*. University of Minnesota Duluth. Retrieved October 14, 2021, from https://www.d.umn.edu/%7Ejfitzake/Lectures/DMED/Vision/Optics/Blinking.html

Grinberg, M. *Turbo-Flask — Turbo-Flask Documentation*. https://turbo-flask.readthedocs.io/en/latest/. Accessed 2 May 2022.

International Telecommunications Union. (2009, October 3). *M.1677Â :Â International Morse code*. Retrieved October 14, 2021, from https://www.itu.int/rec/R-REC-M.1677-1-200910-I/

Maiese, K. (2020, September). *Locked-In Syndrome*. Merck Manuals Consumer Version. Retrieved October 14, 2021, from https://www.merckmanuals.com/home/brain,-spinal-cord,-and-nerve-disorders/coma-and-impaired-consciousness/locked-in-syndrome

Mayfield Brain & Spine Clinic, & Mandybur, G. (2018, April). *Electroencephalogram (EEG)*. Mayfield Clinic. https://d3djccaurgtij4.cloudfront.net/pe-eeg.pdf

Medeiros, J. (2015, January 13). *How Intel Gave Stephen Hawking a Voice*. Wired. Retrieved October 14, 2021, from https://www.wired.com/2015/01/intel-gave-stephen-hawking-voice/

Pinheiro, C. G., Naves, E. L., Pino, P., Losson, E., Andrade, A. O., & Bourhis, G. (2011). Alternative communication systems for people with severe motor disabilities: a survey. *BioMedical Engineering OnLine*, *10*(1). https://doi.org/10.1186/1475-925x-10-31

Snodgrass, R., & Camp, V. (1922). *International Morse Code* [Translation Diagram]. https://upload.wikimedia.org/wikipedia/commons/e/e9/International_Morse_code.png

SpinalCord.com, & Kuriakose, D. (2020, December 22). *Quadriplegia & Tetraplegia: Definition, Causes, Symptoms, and Treatment*. Spinal Cord, Inc. Retrieved October 14, 2021, from https://www.spinalcord.com/quadriplegia-tetraplegia

The National Archives. (1987). *Comdr. Jeremiah A. Denton, Jr. - Report from Inside a Hanoi Prison, 1966*. Eyewitness: American Originals from the National Archives. https://www.archives.gov/exhibits/eyewitness/html.php?section=8

Walker, Michael, "Designing the Haptic Interface for Morse Code" (2016). *Graduate Theses and Dissertations.* https://digitalcommons.usf.edu/etd/6600

War Department. (1939, November). *Basic Field Manual: Signal Communication*. United States Government Printing Office. https://archive.org/details/Fm24-5-nsia/page/n1/mode/2up

Yee, A. (2014, October 16). *Why Do Eye Muscles Function in ALS as Other Muscles Waste Away?* Scientific American. Retrieved October 14, 2021, from https://www.scientificamerican.com/article/why-do-eye-muscles-function-in-als-as-other-muscles-waste-away/