

# The History of WWI

## Taught By Morse Code



Three other people and Lincoln

(info redacted for privacy)

University of Colorado Boulder

Dec 8, 2025

## ABSTRACT

This project investigates, and aims to combat, declining history course engagement amongst middle and high schoolers. By presenting lesser-known WWI stories through an interactive medium, the project strives to improve student engagement and comprehension of historical events for students. The final product was a real-time Morse code translating telegraph that displayed messages entered by the user along with uniquely chosen WWI stories. The final design was chosen to focus on educating students through interaction and engagement with untold WWI stories shared through the telegraph. The telegraph key was mounted on a custom enclosure containing the electronics and an RGB LED visible through a clear acrylic panel. The upper surface included the antique key, control buttons, and a brief instruction sheet. Input durations were assigned to a dot or dash, translated into English, then displayed with their respective stories. Overall, the project was successful in increasing history engagement, with 100% of users in the target age range reporting they were ‘Engaged’ or ‘Very Engaged’. These results indicate that active and hands-on learning can be utilized to better educate students on history. Future work will focus on reducing the device’s size while preserving and improving the engagement and connection focused features.

## I. INTRODUCTION AND BACKGROUND

The Nations Report Card, in its 2022 article “U.S. History score continues decline begun in 2014,” highlights lower scores across America on the topics of culture, technology, and world role, showcasing that over forty percent of students are scoring below the NAEP basic level in their U.S. history classes (National Center for Education Statistics [NCES], 2023). This decline is slightly linked to the pandemic, but fundamentally reflects the way history is being presented in the classroom. Analysts and educators link the declining results to decreased classroom engagement and the common lecture-style learning methods failing to retain student interest (The Washington Post, 2023; EdSurge, 2022). A major reason interest is not kept and engagement decreases is, “because it sometimes feels like we are so far removed from the people it [history] happened to,” and if students do not see history as personally significant, they find no reason to learn about it. The older methods teachers use are not enough to captivate students in the fast paced ever-evolving digital world of the 21st century. New methods are needed to keep students willing to learn about the history of their nation and form a connection with it, and after all, “Those who cannot remember the past are condemned to repeat it” (Santayana, 2011, p. 284). The team’s target users are these students, in middle or high school, who are not engaged with the history of the country they live in and do not form deep and meaningful connections to the past.

In an attempt to meet these needs and better educate middle and high school students about history, the team first searched for products that attempted to meet similar user needs but had some shortcomings. LEGO Epic History and The Oregon Trail were two of the products the team found that try to educate its users on specific historical events, but have gaps spotted by the

team that could improve them further. For example, LEGO Epic History does not attempt to teach users a new skill while it is teaching about history, and The Oregon Trail does not give specific facts or details about the event it intends to educate on. These pitfalls that the team identified in these two products gave the team a better understanding of the current market and what the team could do in their own project to better educate and engage students.

Based on these holes in the market, the team created a product that shares with a user stories about a historical event. The team chose WW1 as the pilot topic because it was a massive war that shaped the world in which we live today. All over, the political, social, and technological landscape were transformed. With this in mind, the team also searched for products in the current market that the product could draw design inspiration from, but have a different intended audience and purpose from the team hoped to create. In the research the team found the Mini CW Morse Code Trainer and Morse Code Translator. This product intends to teach users Morse code, but does not educate users about the history that took place while it was in use. The team saw this as a clear market gap and opportunity. They decided to engage students through an interactive medium, the project strives to improve historical engagement and comprehension of WW1 stories. This engagement will serve to aid in the education of students on events that define history.

World War I offers a very powerful lens for improving engagement: it is a trove of bravery, technology, communication, and personal stories. However, the main methods for teaching about WWI are still through a lecture or the use of a textbook. Both of these are doubtless valuable, “but the ideal teacher is one who does not rely on the textbook alone” and “spoken word resources... can often be boring, lengthy, and hard for even the highest ability students to engage with” (Einhaus & Pennell, 2014, p. 34-35). The team will educate middle and high schoolers about the history of WWI, engaging them with the past and preparing them to make informed decisions in and about the future, through the use of morse code.

The team’s project is a modernized telegraph, incorporating an auto-translating feature and a live time digital display, supplementing education about the first World War. It functions as a tool to increase student interest, to draw them in; it inspires a tactile, hands on approach to learning, in contrast to traditional lectures. The morphing of a historical implement that actual soldiers used while fighting, with technology that abounds in the world of today is a unique blend that connects students individually with the history of their country.

Research shows that when students engage with hands-on, interactive learning tools, their retention and comprehension significantly improve (Freeman et al., 2014). The students get to learn about the functional technology that was the basis of communication and through this technology, they learn about history. The students' connection with WWI is immensely valuable, revealing how the world of today was shaped and the context that their lives take place in.

History sheds light upon the people and societies that inhabit the Earth. Greater knowledge brings greater opportunities for connection and relationships to be developed, increasing understanding and unity throughout the world.

## **II. DESIGN REQUIREMENTS**

The project design requirements are organized into two types, constraints and criteria, outlining the project scope into clear goals/requirements. Constraints are the must haves and objectives that are tangible and set. Criteria are the should haves, the abstract goals that the group wants to maximize into requirements. The constraints and criteria are described below with their corresponding requirements and a rationale of their importance in the project.

### **Constraints:**

- Timeline: The minimum viable product and CDR will be completed by 11/4. The Expo poster will be submitted by 11/7. The polished product will be finished by 11/15. Completing the product within the timeline allows the team to achieve a more polished, functional product. It is necessary that the team sets meetings and spaces out time spent on the project to meet these goals.
- Cost: The team's budget is \$75 per person. Minimizing cost ensures accessibility and affordability for a larger range of users.
- Size: The size of the product will be small enough to fit in a 26 inch wide x 16.5 inch high x 28 inch deep locker. Having a compact machine allows more accessibility and convenience for using in a classroom setting.

### **Criteria:**

- Social Impact: The product will be socially impactful and informative, ensuring it addresses an issue present in the world today: lack of engagement in the teaching of WW1. Addressing this issue allows current students to make more informed decisions for their future through connection with the past.
- Complexity: The product is elaborate and advanced while remaining appropriately challenging for the user, middle school and high school students. Maintaining the right level of complexity keeps the user engaged in the learning objective. If the product is the wrong level of complexity, the user may find themselves uninterested and not enjoying their interaction with the product.
- Interactive / Enjoyable: The product will have user interaction so physical touch will trigger a visual or audible output. Interaction increases engagement in the learning progress which will also contribute to the enjoyability of the product.
- Education of WW1: The product will educate students about WW1 through the use of facts and stories relating to a specific event from history. Including the educational content will help reach the team's goal of a deeper understanding of historical knowledge of WW1.

- Intuitive: The product needs to be intuitive to the user. Having a product that is quick and easy to pick up allows the user to prioritize *using* the product for education and enjoyment rather than having to learn *how* to use it.
- Visual and Audible Accessibility: The product will produce sound and have a visual feature. These indicators will allow users to know what they are doing better and enable a wider range of users that can engage with the product.

### III. DESIGN ALTERNATIVES AND PROCESS

The team developed an educational device aimed at helping middle and high school students connect with the untold stories of WWI, the team explored many designs. The team's early prototypes failed to balance visibility, interactivity, and clarity. Through iterative testing and structured brainstorming methods such as SCAMPER, the team made a design that invites users to engage directly with historical processes rather than simply observe them. The following section outlines the major concepts the team explored, and the reasoning that led us toward the open, transparent design that became the final product.

#### **Concept 1: Closed System With Electromagnet and LCD Screen**

The first design relied heavily on being enclosed. As seen in Figure 3.1, the system would be built into a closed box with only the bottom able to come out. When the user tapped the Morse key, the signal would be sent to an electromagnet that completed the circuit. This signal would then pass through the parser and appear on an LCD screen attached to the box. The box protected the internals, offered a professional appearance, and the large screen provided strong visibility. The electromagnet also added authenticity, reflecting the feel of an old telegraph.

As the team refined the design, several drawbacks became clear. The enclosed system made debugging or changing internal components difficult, and the lack of visibility prevented students from observing the technological workings of Morse code. The SCAMPER method reinforced these issues; although the original concept rated high in potential engagement, variants with interior panels scored significantly higher because they allowed students to form a more tactile, immediate connection. These findings revealed that the fully enclosed design limited educational value, pushing the team to explore new options for visibility and accessibility.

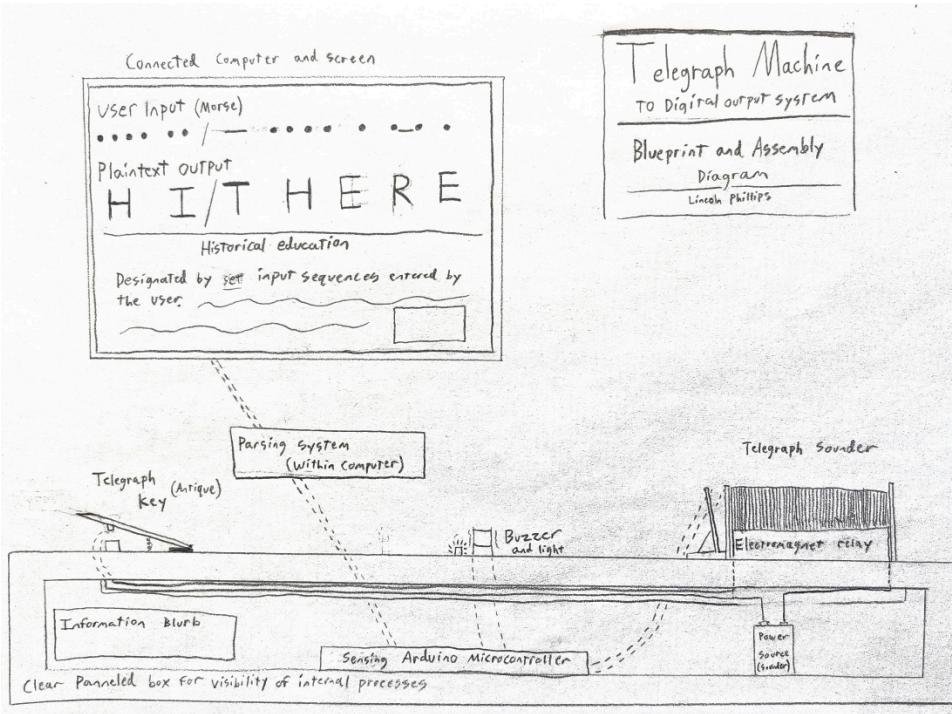


Figure 3.1: Original schematic sketch for telegraph, showcasing initial ideas.

### Concept 2: Open System with Computer Screen

The second concept removed the box entirely. All components—the Arduino, wiring, parser, input key, and display—were accessible and visible. The team replaced the oversized LCD with a computer screen, which maintained readability without overwhelming the structure. Several input variations were considered, including vertical straight keys, horizontal sideswipers, and vertical sideswipers; however, the original calibrated straight key remained the most accessible and historically accurate.

This concept rated highly through the SCAMPER method, scoring second highest overall. The open system performed well in user interaction due to its visibility and engagement. The team also evaluated multiple signal transmission mechanisms. Options like audio pickup or LED offered improvements for reaching a wider audience, while the original electromagnet proved less effective once assembled. The computer display received the highest rating in each output category. Although highly rated, this concept did not become the final design. Instead, it served as an important transitional step that clarified which features best supported usability, visibility, and student engagement.

### **Concept 3: Clear Panel Box or Diagrammed Interior**

In response to the earlier designs, the team chose to modify the box rather than remove it. One variant added a clear panel so users could view internal components, while another introduced an interior diagram explaining each part. Both options sought to preserve the structure of the box while improving visibility and ease of debugging.

The clear-panel idea offered moderate engagement potential but continued to restrict direct access. The diagram scored lowest, as students would see only a representation rather than the functioning system itself. These limitations revealed that visibility alone was not enough; accessibility was equally necessary. This stage made the team realize the system needed both a clear panel and a way to work inside the box. As a result, the final design incorporated a clear front panel combined with a hinged back wall for debugging access.

### **Design Selection:**

The team used multiple design selection matrices to determine the best concept to satisfy the constraints and criteria of the project. The team set up tables for voting for the weighting of each criteria, as shown in Table 3.2, as well as a table for voting for each design concept, as shown in Table 3.3.

Table 3.2: Team members' votes for criteria weighting. Total scores for each criteria correspond to the weighting for each category. Higher scores correspond to larger weights (i.e. “Educational” is weighted the highest, “Social Impact” is weighted the second highest, etc.).

Criteria	Person 1	Person 2	Person 3	Person 4	Total Score
Timeline	2	1	2	2	7
Cost	1	1	1	1	4
Size	2	1	1	2	6
Social Impact	6	6	5	5	22
Complexity	1	1	1	1	4
Interactive / Enjoyable	5	4	4	6	19
Educational	5	7	6	5	23
Intuitive	1	2	2	2	7
Accessible	2	2	3	1	8
<b>Total</b>	<b>25</b>	<b>25</b>	<b>25</b>	<b>25</b>	<b>100</b>

As shown in Table 3.2, each team member was allotted 25 points to put into each of the decided criteria. The more points a team member puts into a specific category, the higher they believe that category should be weighted. Voting for weighting of criteria ensured that each team member's voice was heard during the design selection process.

Table 3.3: Team members' votes for each concept. The sum of the votes for each concept multiplied by the weight for each category was added across each idea. This value is the total score, which was used to determine the best design concept to satisfy our criteria, with Concept 1 winning out over the others. The voting order is as follows:

Person 1, Person 2, Person 3, Person 4.

Criteria	Weight	Concept 1	Concept 2	Concept 3
Timeline	7%	5+4+3+4	5+4+4+5	5+5+4+5
Cost	4%	3+4+3+3	3+3+2+3	4+5+4+3
Size	6%	4+5+5+5	4+4+3+4	5+5+4+4
Social Impact	22%	4+5+4+3	4+3+3+4	3+3+3+2
Complexity	4%	5+5+5+5	3+3+4+3	3+3+2+2
Interactive / Enjoyable	19%	5+4+5+5	4+4+4+4	3+3+4+4
Educational	23%	5+5+4+5	4+5+3+5	3+3+4+5
Intuitive	7%	5+4+4+5	5+4+4+4	5+4+3+4
Accessible	8%	4+3+5+4	4+4+3+5	4+4+3+4
<b>Total</b>		17.62	15.62	14.30

Team members voted for each concept by criteria to determine the design idea that best satisfied the design criteria. Voting was done on a scale from one to five, with five being what a team member believed to be the best possible option(s) for that criteria, and one being the worst. As shown in Table 3.3, the team believed Concept 1 to be the best possible solution, as this design had the greatest total score.

#### IV. FINAL DESIGN

The final design of the product consisted of an electrical circuit and a RGB-variable LED within a containment box with a clear paneled face, which can be seen in Figure 4.1. On the surface of the box was an instruction sheet along with the antique telegraph key, and the space, backspace,

and calibration buttons. The electronics were connected to a computer running the parser with a display. The morse code inputs were shown upon the screen with live updates, accompanied by the English translation. When a specific unique word was sent, a story from WWI, one less known or acknowledged, would be displayed, educating the user on history through the use of the engaging telegraph.

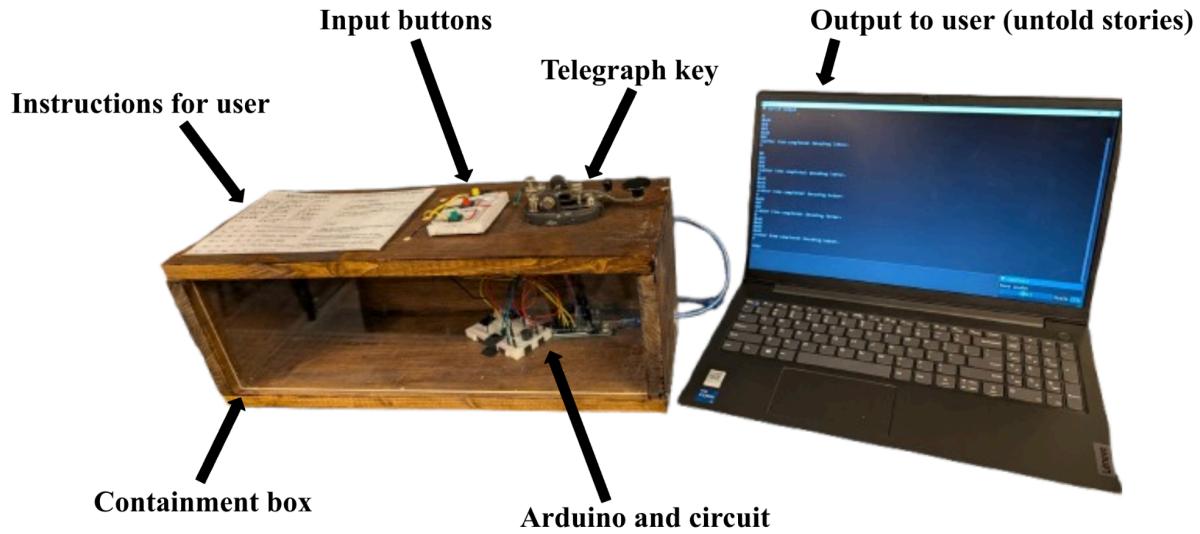


Figure 4.1: An intuitive and hands-on setup entices, engages, and educates the user.

In Figure 4.2, each component of the product is shown, as well as how each component connects to other parts of the product. The flowchart also shows who on the team lead the design of each part of the project.

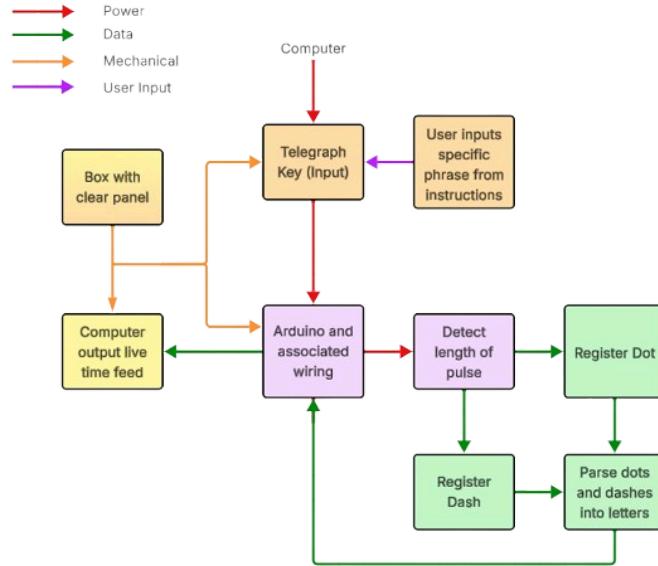


Figure 4.2: User input is received by the powered telegraph, the data is processed through the arduino and computer, then the educational, historical story from WWI is shared.

### Containment Box:

The containment box is a wood box with a clear front panel and hinged back panel. It's polished darker to give it a more polished, antique look. Its final design was beneficial to the telegraph in several ways. The clear front panel allows the user to see the internal mechanisms and wiring of the telegraph. The back panel that could be opened and closed allows team members to access the electronics inside, allowing for fixes to be made and also acting as a safety precaution. The box has a hole drilled into the side for the cable connecting the arduino and computer to run through, creating a more discrete look. Mounted to the box are the buttons and telegraph key. Providing a surface for the telegraph is the main purpose of having a box, creating a more unison and polished product.

### Electronic Wiring:

All electronic wiring is connected to an Arduino UNO, which receives user input. The Arduino is wired within the containment box, allowing for easy access to interactive components, such as the input buttons and telegraph key.

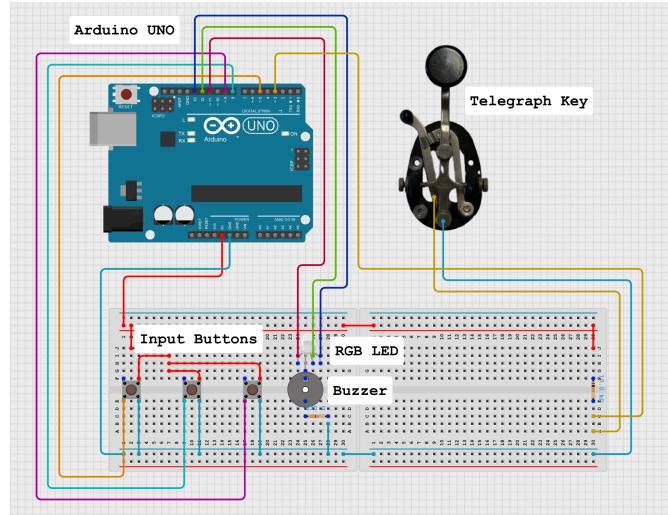


Figure 4.3: Labeled circuit diagram for Arduino and components. Various inputs from the user are interpreted by the Arduino UNO.

As shown in Figure 4.3, the Arduino is connected to multiple components, many of which are interactive for the user. The Arduino is able to take these inputs from these interactive components, like the telegraph key, and turn them into dots and dashes. This was done by setting up the pins on the Arduino board to receive input signals. Signals are also able to be sent from the input buttons in a similar fashion.

#### **Input and Calibration:**

The Arduino is set up to register multiple user inputs from the buttons and the telegraph key. The telegraph key itself serves as a device to register user input, as it is where morse code is sent through. There are also three input buttons that each serve a different purpose, the “space” button, the backspace button, and the reset/calibration button, as shown in Figure 4.4.

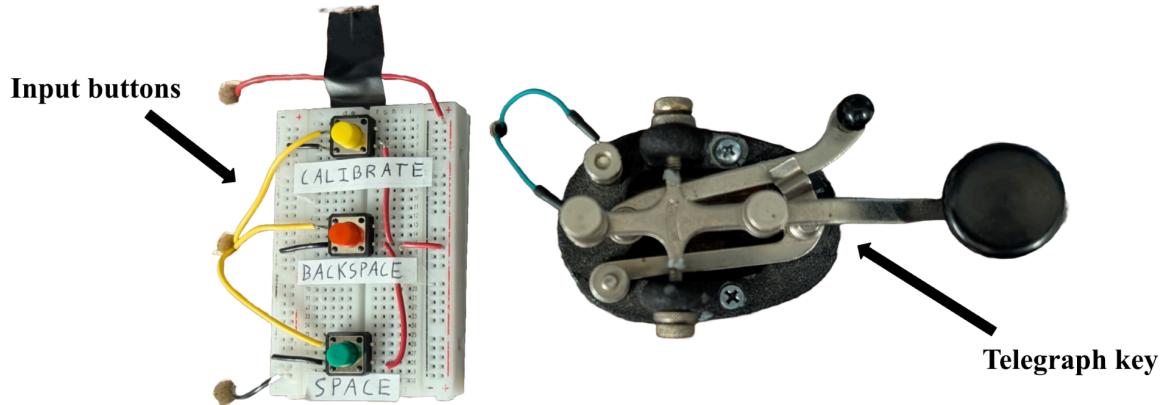


Figure 4.4: Tactile buttons to engage the user in learning about history.

Calibration allows the user to assign the amount of time they believe a dot and a dash should be, as people have different preferences as to how quickly they would like to input morse code. As seen in Appendix C, all of these buttons are coded similarly, as their button states are changed to on when they are pressed, and then changed to off after 120ms. Each of them have unique outputs to Serial. These outputs, besides calibration, are processed by the parsing system.

Calibration functions slightly differently, as it is done directly in Arduino. When the calibration button is pressed, the calibration phase starts. The first and second inputs during the calibration phase are the lengths of time the user believes a dot and dash should be, and those times are stored in the variables ‘short\_duration’ and ‘long\_duration,’ respectively. After these two inputs are received, the calibration phase ends, and the user is now able to input dots and dashes.

Whether the input is a dot or a dash is determined by the durations set during the calibration phase. If the length of time the user inputs into the key is less than the amount of time a dash is, determined by calibration, then the input is a dot. If the length of time is greater than the amount of time for a dash, then the input is a dash. These outputs are printed to Serial, and sent to the parser through the serial port.

The Arduino also controls the RGB LED and buzzer. These components are connected in series, so both turn on and off simultaneously. The LED is assigned to light up different colors based on the kind of user input. This was able to be done through the use of different pins on the Arduino board. These pins were assigned to the variables ‘redPin,’ ‘greenPin,’ and ‘bluePin.’ When input is received from one of the buttons, a signal is sent to the respective LED pin, turning the LED on and sounding the buzzer, providing visual and audible feedback for the user, improving the accessibility of the product.

### **Parsing System:**

The main parsing system is written entirely in the C++ language as a windows desktop application, running in Visual Studio 2022, fully visible in Appendix C. The logic of the parsing system is split into input recognition and the actual decoding process. The first step is connecting the Arduino UNO containing the calibration and input logic to the chosen code running device. Running at a 9600 Baud Rate and plugged into the computer, the Arduino serial port becomes linked with the C++ system.

For the input recognition sequence, all outputs from the arduino code are now read and stored in the string variable ‘input’. Two other main variables, in the form of a list of strings, ‘letter’ and ‘word’ are initialized. The ‘letter’ variable will store each individual instance of translation from Morse to English. The ‘word’ variable stores all of the accumulated letters recognized by ‘letter’ in a sequential list, until a changing input is received. The parsing system depends entirely on the six possible inputs received from the Arduino: ‘dot’, ‘dash’, ‘/’ (space), ‘-’ (backspace), ‘^r’ (calibration), ‘NA’ (no other input). The computer code runs a constant check of the ‘input’ variable against the chosen input representors, with a debounce designed to ensure an input is a new and not a continual single input. The base resting state input is ‘NA’, upon which the code breaks back to the ‘While (!done)’ loop (essentially a While True loop created to keep the parsing logic in a continuous cycle) and no update is broadcasted to the display. If the space input ‘/’ is matched, that is the cue to print the entire list of letters stored in ‘word’ to the display and reset the variable to an empty list so that a new word can begin to be generated. The opposite happens when the backspace input ‘-’ is received: the backspace removes the most recent letter addition to ‘word’ and sends a confirmation message of the deletion to the display. The last input, ‘^r’, representing the calibration phase, is purely aesthetic. The true processing of this sequence is computed in the Arduino process above and this input in the C++ code merely provides a quick confirmation that the calibration was successful then clears the screen, readying it for whatever new input may come next.

The two remaining inputs, ‘dot’, and ‘dash’, are instantly sent to the decoder. In the overall view of the parsing system, the decoder runs its inputs into a massive elimination bracket where each new dot or dash eliminates multiple options until only one possible letter remains which is then selected and added to ‘letter’ and subsequently, ‘word’. More specifically, the program seeks patterns along with paying attention to time. A standard time elapsed function is used and when this reaches a certain limit, the signal that ‘shouldDecode’ is now true is sent.

Once ‘shouldDecode’ is true, the program investigates which letter has been received. There are 13 letters that begin with a dot in International Morse Code and 13 that begin with a dash. One of those inputs then instantly eliminates 50% of the possible letters. The decoder then begins work. With nested ‘if’ loops, this process is repeated, eliminating possible letters over and over,

through recognizing the number of inputs received and what the dots and dashes narrow it down to. The final letter is then selected and defined as ‘letter’ which is sent to the visual display.

If the printed ‘word’ matches a custom list of strings exactly, then one of the unique messages will be sent to the output display in addition to the word. This entire process, entailing the inputting of a message, the decoding, and the resulting output are demonstrated in Figure 4.3.

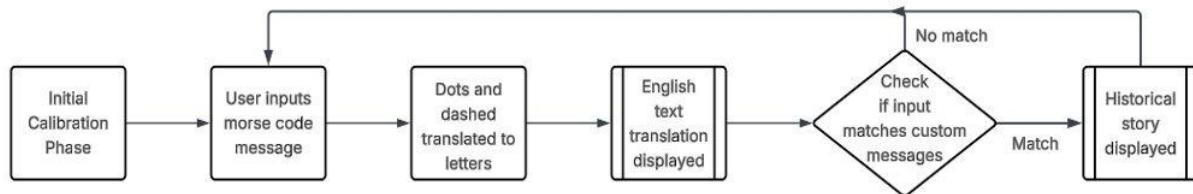


Figure 4.5: The C++ code interprets the user input, translates it to English, and displays an untold story, educating the user.

### **Visual Display:**

The other important part of the process shown in Figure 4.5 is the actual visual display showing the Morse code input, the English translation, and the historical story. To keep the display orderly and clean, the output is not printed to the terminal at all, instead it is sent into an entirely new graphic user interface generated for this express purpose. This was achieved through the use of the ImGui library, powered by Raw Win32 API + DirectX11. All outputs were sent to the GUI with the use of the ‘appendLog(“Text”)’ function. With the headers and source files named in Appendix D, ImGui generates the custom output window, frame by frame. It follows a standard terminal setup, as seen throughout Appendix E, with only a few notable variances: a unique font scale setup, the scrolling feature, and clear feature. The font scale, ranging from 0.5x to 3x, is represented with a slider float, in order that the size of all the text can be adjusted depending on the eyesight of the current user. Additionally, a scrolling function was created, that when an output would be displayed off-screen, but instead the topmost lines are deleted and the correct output is shown. Finally, there was a small clear button. With the implemented buttons, the only screen clear was triggered by the calibration button, so a way was needed to reset the visual field without beginning the entire calibration process and messing up the assigned preferences. This clear button triggers the ‘ImGui::End()’ function, which kills the display for a fraction of a second, then, as it is an iterative frame field, it is instantly regenerated as an empty field. The variables ‘letter’ and ‘word’ are also reset to prevent unwanted carryover.

### **Untold Stories:**

The final design of the product includes untold stories as an output option triggered by certain key phrases. The team chose to create six words that the user could input to be told a story. These stories dove into key figures in WW1, culture, lesser known battles, etc. The words were kept simple so that the design was kept beginner-friendly. For example, when the word “key” was

tapped out and entered, the display revealed the importance of Native Americans in WW1. This design element achieves the goal of education and engagement with the history of WW1. The idea of telling the untold or lesser known aspects of WW1 was chosen to captivate the user more instead of repeating boring history facts. To know what the key words were, and how to translate Morse code from English, a paper was provided for the user to reference.

## V. TESTING

To ensure an effective and robust product, the team did testing on the product's critical components. Two of the components prioritized in testing were the key and the translation system. Having a robust and working telegraph key was essential to the design of the product as user interaction and engagement were the main focuses of the project. Also necessary to the design was the translation system, which actually takes the dots and dashes inputted by the user and translates them to English, simultaneously providing custom untold stories of WWI.

To test these critical components, the team had 8 users of their target age group actually interact with the product with the same instructions that would be given at Expo. This helped assess the usability and intuitiveness of the product for the user while also testing the ability of the translation system to withhold a constant input of information. For the testing, each user was introduced to the product with an explanation of the product's goal and how it should be used. The users were provided a paper with a translation of morse code and the key phrases to make sure the goal of learning history of WW1 was being achieved rather than a focus on morse code itself. When getting the opportunity to use the product, the users were able to easily use the product, sometimes after a few tries. Through testing, the product held up and translated morse correctly, proving ready for the Expo.

Aside from testing the actual product, the testing of user engagement was essential to completing the project goal. The team decided to analyze user engagement through a poll after the users got to test the product. The students were asked what their level of engagement was with the product with the options being, not engaged at all, a little engaged, engaged, or very engaged. The final data showed that 63% and 37% of the users respectively reported they were engaged or very engaged, totalling a 100% engagement rate.

During the testing stage there were also safety considerations that had to be taken into account. The product does not have many aspects that could be considered unsafe so one of the group's biggest priorities was having neatly contained wiring/ electronics. This goal contributed to the final design of the arduino being enclosed in a box in which you can still see the internal mechanisms. The box also included the hinged side panel so that electronics could still be accessed for fixes and safety. For components not contained within the box, the group ensured that they were attached well to the surface of a part of the machine. With all these aspects taken into account, the product was proved viable and robust.

## VI. BUDGET AND BILL OF MATERIALS

The team's budget for this project was \$300. The team purchased multiple different items required for the construction of the containment box and the antique key, as well as a poster for the final Design Expo. Multiple items were received for free from various resources in the ITLL and manufacturing centers. These materials were needed to implement the appropriate electronic and wiring features.

Table 6.1: Itemized bill of materials for project, including free/donated materials demonstrating that the total cost of the project was within the allocated budget.

Component	Item	Cost
Arduino and circuits	Arduino UNO	\$0 (donated)
Arduino and circuits	Wires	\$0 (donated)
Arduino and circuits	Electronic components	\$0 (donated)
Box and manufacturing	Clear acrylic	\$5
Box and manufacturing	Wood	\$40
Box and manufacturing	Wood polish and stain	\$10
Box and manufacturing	Glue	\$0 (donated)
Box and manufacturing	Telegraph key	\$50
Box and manufacturing	Hinge	\$0 (donated)
Box and manufacturing	Handle	\$0 (donated)
Expo poster	Expo poster	\$30
<b>Total</b>		<b>\$135</b>

The majority of the budget went towards manufacturing the box; acquiring the wood and acrylic panel, as well as purchasing the telegraph key, as seen in Table 6.1. The total amount the team spent on materials and the poster for this project was approximately \$135.

## VII. CONCLUSIONS

Through a telegraph, a piece of technology that was heavily used during World War I, the team created a device to help educate students on the importance of history and engage them in WWI. The final product was successful in achieving this goal, as it was both interactive and

educational, which helps to keep students interested in learning about history. Despite the success of the project, the team's product would have benefitted from more stress testing, specifically the telegraph key. While the key was able to stand up against testing from team members, this component broke multiple times during the Expo. This showed the team that more user testing could have been done, and the key could have been even more robust. The solution for wiring the box could have been better thought out as well. Wires became disconnected from the circuit, meaning the team had to make internal fixes to the circuit throughout the Expo as well. This problem could have been avoided if more time was spent planning out the wiring of the box. In the future, better planning of components could have lessened the amount of issues that the team faced during Expo, and would have smoothed out the overall design process of the product.

A few specific changes could have been made to the product to improve its functionality. For example, less wood could have been used to create the box if the dimensions were smaller. This would reduce the cost spent on wood and manufacturing. The lower the cost of the product, the more cost accessible the product is for schools, allowing it to be better implemented into the classroom. Both the implementation of the telegraph key and the wiring of the box could have been better planned out, and more time could have been spent on both of these components. Also, with the main purpose of the product being education, for the future, it would be ideal to create more key phrases and untold stories that would inform the user. With more time spent on this research and the creation of more facts, the user would benefit even more educationally.

Teams working on similar projects should make sure to test their product with many different students in varying age ranges to make sure that their design is durable and able to hold up against students. The size of the product should be small enough to be reasonable to have multiple in a classroom. If the product doesn't require code to be running on the arduino and on the used device, space, time, and effort can easily be reduced.

The most important information to remember is that the product is for the users. It must use hands-on learning to be interactive and engaging, drawing students in and connecting them to history. When the students truly understand the past of their country, they can make wise decisions about the future and avoid what the famous philosopher George Santayana feared when saying, "Those who cannot remember the past are condemned to repeat it."

## REFERENCES

- Brouhard, A. (2024, April 3). *Most students think history is boring. Here's how we change that.* EdSurge.  
<https://www.edsurge.com/news/2024-04-03-most-students-think-history-is-boring-here-s-how-we-change-that>
- Einhaus, A.-M., & Pennell, C. (n.d.). *The First World War in the classroom: Teaching and the construction of cultural memory.* <https://ww1intheclassroom.exeter.ac.uk/>
- Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences of the United States of America*, 111(23), 8410–8415. <https://doi.org/10.1073/pnas.1319030111>
- National Assessment Governing Board. (n.d.). *2022 NAEP U.S. history assessment. The Nation's Report Card.* <https://www.nationsreportcard.gov/highlights/ushistory/2022/>
- Santayana, G. (2011). *The life of reason: Introduction, and reason in common sense* (Vol. 7). MIT Press.

## Appendix A: Untold Stories

As talked about in **Section IV**, there are certain phrases the user can tap out in morse code that will trigger untold facts about WW1. The team created six of these phrases and stories, which are listed below (utilized formatting visible with newline calls in Appendix D):

**SOS:** During WW1, people sent messages through a telegraph, the machine you are using now! SOS is a well-known distress signal in morsecode, meaning “save our souls” or “save our ship.” This signal was used extensively during WW1 by military vessels and aircraft. Its importance during the Great War was vital , the SOS signal alerted rescue teams of downed aircrafts or ships. This distress signal saved many lives.

**KEY:** Although few fully realize, the American Indians were key to the outcome of WW1. Many were drafted, but the majority volunteered to fight for America. Choctaw and Cherokee code-talkers participated in the Meuse-Argonne Offensive in the fall of 1918: the key battle that historian Geoffrey Wawro describes as having “cut the German throat”. “The North American Indian took his place beside every other American, in offering his life in the great cause, where as a splendid soldier, he fought with the courage and valor of his ancestors” -General John Pershing

**MAZE:** Not many people get a first-hand perspective from soldiers, a really important aspect of the WW1 living situation: Trench Life. Gilbert Williams, a WW1 private, wrote a letter describing his experiences: “Soldiers seem to spend three times as much time in the trenches than out,” “a veritable maze of trenches,” where explosives are going off constantly. The fighting was so strong that the dead could only be buried in the sides of the trenches. Trench life was gruesome and hard, yet the soldiers still fought on.

**ELSIE:** The story of Elsie Maud Inglis, although less known, had a huge medical impact in WW1. She suggested to the War Office the creation of a fully women medical unit on the Western Front. Initially rejected, she persisted and contacted the French, who supported her idea. Two auxiliary hospitals were sent to Russia and Serbia, subduing the Typhus epidemic.

**EAST:** The Battle of Tannenberg was an influential battle for Germany, occurring along their eastern front. The German military encircled and defeated the Russian Second Army, halting the Russian advance into German territory for the rest of WW1. They applied these same tactics effectively in future engagements. With pressure taken off the eastern front, Germany could focus more resources on the western front. The “Tannenberg Myth” led to a nationalistic belief in the steadfastness and military prowess of the Germans, even after the war had ended.

**LEWIS:** Sidney Lewis was the youngest authenticated British soldier in WW1. He enlisted at the age of 12 and fought in the Battle of the Somme in 1916, although he was eventually sent back home according to the demands of his mom. In WW1, it was not uncommon for young boys to

serve before they turned 18 despite the legal enlisting age being 18 years old. Underage boys might join for various reasons such as patriotism, recruiting pressure, or the desire for escape and adventure.

## Appendix B: ChatGPT PROMPT AND FEEDBACK

### *Introduction and Background and Design Requirements Feedback*

[Partial Draft \(link\) \[Coach Clarity prompt and feedback included\]](#)

### *Draft of Complete Report Feedback*

#### **Prompt:**

*TEAM: WALL-E | COURSE: GREEN 1400 (Retro Remix)*

*SUBMISSION TYPE: Full draft*

*PROJECT (1 line): User inputs morse code, and user receives code translation and lesser known stories and facts about WWI, focusing on education engagement*

*TENSE TARGET: past | VOICE GOAL: active (avoid passive)*

*CITATION STYLE: APA (use the same style for in-text and reference list)*

*PLEASE REVIEW (feedback only; no rewrites):*

- *Clarity, conciseness, completeness for reader understanding*
- *Organization/flow and single, consistent voice/tense*
- *Formatting & conventions: captions (tables above, figures below), numbering, cross-refs; equations centered with right-margin numbers and defined symbols; units/sig figs on tables/plots*
- *Captions: when a caption appears, check that it states the TAKE-AWAY and is CENTERED on the page*
- *References: in-text citations present where needed; style consistent with APA for both in-text and reference list*

*DELIVERABLES (use streamlined format):*

- 1) *Quick Summary (2–3 bullets, include the caption self-check)*
- 2) *Section-by-Section Feedback (≤4 bullets per section; fold in caption/equation/units/citation issues where they occur)*
- 3) *Next-Draft Checklist (up to 15 checkbox items; no rewrites)*
- 4) *Submission Readiness (Green/Yellow/Red + why + one next action)*

*DRAFT ATTACHED:*

[First Draft \(link\)](#)

#### **Feedback:**

- 1) Quick Summary

- What's working well: Very strong completeness for a full draft—every major required section is present, all core technical steps are explained, and your voice is consistently active and past tense. Your user-need argument is especially clear in the Introduction.
  - Biggest opportunity: Improve clarity + conciseness in places where long paragraphs mix multiple ideas (especially in Design Alternatives, Final Design, and Testing). Figures also need caption improvements and consistent cross-referencing.
  - Caption Self-Check: *Do your figure captions explicitly state the take-away message and are they centered on the page?*
- 

## 2) Section-by-Section Feedback

---

### TITLE PAGE

- Coverage is complete; formatting matches the template, but the title could be higher impact—just a note, no rewrite required.
  - Remove stray commas before student names to keep professionalism. - COMPLETED
  - Ensure all contact info formatting is consistent (font, alignment). - COMPLETED
- 

### ABSTRACT

- Consider tightening—some sentences combine too many ideas, making the flow harder for a new reader. - COMPLETED
- Align with rubric: briefly state method → key results → meaning → future recommendations. You cover these, but they appear in a different order. - COMPLETED
- Avoid unnecessary detail (e.g., listing box materials); abstract should stay high-level. - COMPLETED

## INTRODUCTION & BACKGROUND

- Paragraphs are long; consider splitting when you shift from user needs → market analysis → design gap to maintain reader clarity. - COMPLETED
  - Ensure consistent tense: a few sentences slip into present (“This recent decline is linked...”) while the target is past but present is acceptable if consistent. - COMPLETED
- 

## DESIGN REQUIREMENTS

- Good separation of constraints vs criteria. Criteria could be more measurable (e.g., “intuitive” could be linked to a target number of successful user trials), but do not add new values—just clarify measurability. - COMPLETED
  - Avoid mixing rationale with requirement statements—readers benefit when the “must/should” and the “why” are clearly distinct. - COMPLETED
  - No tables present → no table issues. - COMPLETE
  - Formatting: bullets are clear; consider consistent structure for sentences describing rationales. - ACCEPTABLE AND COMPLETE
- 

## DESIGN ALTERNATIVES & PROCESS

- Very complete narrative, but multiple paragraphs combine concept description + evaluation + decision-making; separating these within each concept would improve clarity. Completed
- Figures:
  - Caption needs to be centered (reader cannot rely on PDF auto-centering).

- Ensure the figure is referenced in the text by number *before* it appears. Completed
  - Voice: watch for passive moments (“was rated highly,” “were considered”). Aim for active when possible. COMPLETED
  - Flow is generally good, but Concept 2 and 3 discussions have overlapping justifications—could be tightened. - ADDRESSED and completed
- 

## FINAL DESIGN

- Very thorough explanation of every subsystem; the level of detail matches rubric expectations.
  - Figures 4.1 and 4.2:
    - Captions need explicit take-away messages (“This figure shows X so the reader understands Y”). - ACCEPTABLE
    - Ensure they are centered and referenced in the text. - COMPLETED
  - Some paragraphs combine construction, function, and rationale—these could be distinct ideas. Consider separating manufacturing steps vs function.
  - Consistency: maintain past tense when describing development (“The Arduino was set up to...” instead of present descriptions). - REVIEWED, POSSIBLE FUTURE EDITS, BUT ACCEPTABLE
- 

## TESTING

- Clear explanation of user testing and component testing; good connection to design requirements.
- The description of safety measures is correct but can be clearer by grouping physical vs electrical safety.

- Engagement statistics: include units (e.g., % is fine, but avoid decimals like “62.5%” unless meaningful).
  - No figures/tables here; fine.
- 

## BUDGET / BILL OF MATERIALS

- Table 6.1 needs a take-away in the caption (e.g., “This table shows the total cost and demonstrates the design met the \$300 limit”—don’t rewrite; just guide). - Captions fixed throughout.
  - Ensure the caption is centered and appears *above* the table.
  - Consider aligning all dollar values consistently (e.g., “\$0 (donated)” vs “\$0 donated”). - Not relevant, but checked anyway
- 

## CONCLUSIONS

- Strong connection back to goals and clear future work.
  - Paragraphs are long; could be split at natural transitions for readability.
  - Avoid causal overstatements—keep claims grounded (e.g., “proved ready” is acceptable but avoid implying universal success).
  - Tense is mostly consistent.
- 

## REFERENCES

- APA style requires:  
Author. (Year). Title. Publisher. URL (if applicable)

Several entries:

- Are missing publication years.
  - Use brackets or number labels (those are IEEE-style).
  - Include an organization name as an author inconsistently.
  - Some sources in the text may be repeated twice (NCES, EdSurge, Washington Post).  
Ensure each is listed once.
- 

## APPENDICES

- Code is long but well-organized.
  - Ensure each appendix starts on a new page and includes 1–2 sentences describing purpose (per rubric).
  - Some screenshots/code blocks may need cropping for focus. - There is nothing to crop for focus, the code is text, not an image
- 

## 3) Next-Draft Checklist ( $\leq 15$ items)

- Reference each figure by number in the text *before* it appears
- Add a take-away message to Table 6.1's caption and confirm caption is centered and above the table
- Ensure APA consistency for all in-text citations (author + year)
- Remove the IEEE-style bracket numbering in the References section and convert fully to APA format
- Merge or split overly long paragraphs in Intro/Background, Design Alternatives, Final Design, and Conclusions
- Check tense consistency—convert present descriptive sentences to past where appropriate
- Reduce passive phrasing in Design Alternatives and Final Design
- Clarify measurable elements of criteria in Design Requirements (without adding new numbers)

- Ensure all claims about engagement or education are supported by citations
  - Review cost table for consistent formatting (same style for dollar values and donated items)
  - Add introductory sentences to Appendices to explain what each contains
  - Verify that page breaks occur only where required (Title Page, Abstract, Intro start new pages)
  - Confirm consistent font and spacing across sections (especially title page and long code blocks)
  - Double-check that all references appearing in the list are actually cited in the text
- 

#### 4) Submission Readiness

Yellow — Solid draft with strong foundations, but citation style and figure/table captioning need correction to meet formal report expectations.

Next action to move to Green:

Fix APA formatting and update all figure/table captions with centered, take-away-focused text.

## Appendix C: Arduino code

```

// variables for buttons and led
const int buttonPin = 3; // dot / dash
const int resetPin = 7; // ^r (also starts calibration)
const int spacePin = 9; // /
const int backPin = 5; // -
const int redPin = 13; // red pin for led
const int greenPin = 12; // green pin for led
const int bluePin = 11; // blue pin for led

// button states
int buttonState = HIGH;
int prevButtonState = HIGH;

int resetState = HIGH;
int prevResetState = HIGH;

int spaceState = HIGH;
int prevSpaceState = HIGH;

int backState = HIGH;
int prevBackState = HIGH;

// time
unsigned long startTime;
unsigned long duration;

// initial values for duration before calibrating
unsigned long short_duration = 10;
unsigned long long_duration = 150;

// calibration state
bool calibrating = false;
int calibration_step = 0;
unsigned long cal_start = 0;

// calibration functions
// start calibration
void startCalibration() {
    calibrating = true;
    calibration_step = 0;

    Serial.println("^r");
    Serial.println("CALIBRATION STARTED");

    digitalWrite(redPin, HIGH);
}

```

```
digitalWrite(greenPin, HIGH);
delay(200);
digitalWrite(redPin, LOW);
digitalWrite(greenPin, LOW);
}

// calibration process
void processCalibration(int buttonState, int prevButtonState) {
    if (!calibrating) return;

    if (buttonState == LOW && prevButtonState == HIGH) {
        cal_start = millis();
        tone(bluePin, 600);
        digitalWrite(bluePin, HIGH);
    }

    if (buttonState == HIGH && prevButtonState == LOW) {
        noTone(bluePin);
        digitalWrite(bluePin, LOW);

        unsigned long dur = millis() - cal_start;

        if (calibration_step == 0) {
            short_duration = dur;
            Serial.print("DOT calibrated = ");
            Serial.println(short_duration);

            calibration_step = 1;

            digitalWrite(greenPin, HIGH);
            delay(150);
            digitalWrite(greenPin, LOW);
        }
        else if (calibration_step == 1) {
            long_duration = dur;
            Serial.print("DASH calibrated = ");
            Serial.println(long_duration);

            Serial.println("CALIBRATION COMPLETE");

            digitalWrite(redPin, HIGH);
            delay(200);
            digitalWrite(redPin, LOW);

            calibrating = false;
            calibration_step = 0;
        }
    }
}
```

```
        }

    }

// setup
void setup() {
    pinMode(buttonPin, INPUT_PULLUP);
    pinMode(resetPin, INPUT_PULLUP);
    pinMode(spacePin, INPUT_PULLUP);
    pinMode(backPin, INPUT_PULLUP);

    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);

    Serial.begin(9600);
}

void loop() {

    // read buttons
    buttonState = digitalRead(buttonPin);
    resetState = digitalRead(resetPin);
    spaceState = digitalRead(spacePin);
    backState = digitalRead(backPin);

    // reset button and calibration
    if (resetState == LOW && prevResetState == HIGH) {
        startCalibration();
        delay(250);
    }

    // calibration
    if (calibrating) {
        processCalibration(buttonState, prevButtonState);

        // update states and break after calibrating
        prevButtonState = buttonState;
        prevResetState = resetState;
        prevSpaceState = spaceState;
        prevBackState = backState;
        delay(1);
        return;
    }

    // register dot/dash
    if (buttonState == LOW && prevButtonState == HIGH) {
        startTime = millis();
```

```
tone(bluePin, 500);
delay(10);
digitalWrite(bluePin, HIGH);
}

if (buttonState == HIGH && prevButtonState == LOW) {
    noTone(bluePin);
    digitalWrite(bluePin, LOW);

    duration = millis() - startTime;

    // debounce
    if (duration > 30) {
        if ((duration > 11) && (duration < long_duration)) {
            Serial.println("dot");
        }
        else if (duration > long_duration) {
            Serial.println("dash");
        }
        else {
            Serial.println("NA");
        }
    }
}

delay(250);
}

// space button
if (spaceState == LOW && prevSpaceState == HIGH) {
    Serial.println("/");
    digitalWrite(greenPin, HIGH);
    delay(120);
    digitalWrite(greenPin, LOW);
    delay(60);
}

// backspace button
if (backState == LOW && prevBackState == HIGH) {
    Serial.println("-");
    digitalWrite(redPin, HIGH);
    delay(120);
    digitalWrite(redPin, LOW);
    delay(60);
}

// update states
prevButtonState = buttonState;
```

```
prevResetState = resetState;
prevSpaceState = spaceState;
prevBackState = backState;

delay(1);
}
```

## Appendix D: C++ parsing and display code

*Included ImGui headers:* imconfig.h, imgui.h, imgui\_internal.h

*Included ImGui source files:* imgui.cpp, imgui\_draw.cpp, imgui\_impl\_dx11.cpp, imgui\_impl\_dx11.h, imgui\_impl\_win32.cpp, imgui\_impl\_win32.h, imgui\_tables.cpp, imgui\_widgets.cpp

```
#include <algorithm>
#include <chrono>
#include <conio.h>
#include <iostream>
#include <list>
#include <string>
#include <stdio.h>
#include <thread>
#include <vector>
#include <windows.h>
#include <d3d11.h>
#pragma comment(lib, "d3d11.lib")

#include "imgui.h"
#include "imgui_impl_win32.h"
#include "imgui_impl_dx11.h"

bool readLineFromSerial(HANDLE hSerial, std::string& out) {
    char c;
    DWORD bytesRead;
    out.clear();
    while (true) {
        if (!ReadFile(hSerial, &c, 1, &bytesRead, NULL)) return false;
        if (bytesRead == 0) return false;
        if (c == '\n') break; //Space button
        if (c != '\r') out += c; //Reset button
        Sleep(1);
    }
    return true;
}

// Win32 setup code - From ImGui examples and implementation
static ID3D11Device* g_pd3dDevice = NULL;
static ID3D11DeviceContext* g_pd3dDeviceContext = NULL;
static IDXGISwapChain* g_pSwapChain = NULL;
static ID3D11RenderTargetView* g_mainRenderTargetView = NULL;

void CreateRenderTarget()
```

```

{

    ID3D11Texture2D* pBackBuffer = NULL;
    g_pSwapChain->GetBuffer(0, IID_PPV_ARGS(&pBackBuffer));
    g_pd3dDevice->CreateRenderTargetView(pBackBuffer, NULL,
    &g_mainRenderTargetView);
    if (pBackBuffer) pBackBuffer->Release();
}

void CleanupRenderTarget()
{
    if (g_mainRenderTargetView) { g_mainRenderTargetView->Release();
g_mainRenderTargetView = NULL; }
}

bool CreateDeviceD3D(HWND hWnd)
{
    DXGI_SWAP_CHAIN_DESC sd;
    ZeroMemory(&sd, sizeof(sd));
    sd.BufferCount = 2;
    sd.BufferDesc.Format = DXGI_FORMAT_R8G8B8A8_UNORM;
    sd.BufferUsage = DXGI_USAGE_RENDER_TARGET_OUTPUT;
    sd.OutputWindow = hWnd;
    sd.SampleDesc.Count = 1;
    sd.Windowed = TRUE;
    sd.SwapEffect = DXGI_SWAP_EFFECT_DISCARD;

    const D3D_FEATURE_LEVEL featureLevel = D3D_FEATURE_LEVEL_11_0;
    D3D_FEATURE_LEVEL obtained;
    if (D3D11CreateDeviceAndSwapChain(NULL, D3D_DRIVER_TYPE_HARDWARE, NULL, 0,
        &featureLevel, 1, D3D11_SDK_VERSION, &sd, &g_pSwapChain,
        &g_pd3dDevice, &obtained, &g_pd3dDeviceContext) != S_OK)
        return false;

    CreateRenderTarget();
    return true;
}

void CleanupDeviceD3D()
{
    CleanupRenderTarget();
    if (g_pSwapChain) { g_pSwapChain->Release(); g_pSwapChain = NULL; }
    if (g_pd3dDeviceContext) { g_pd3dDeviceContext->Release(); g_pd3dDeviceContext
= NULL; }
    if (g_pd3dDevice) { g_pd3dDevice->Release(); g_pd3dDevice = NULL; }
}

// Forward declared in imgui_impl_win32.h

```

```

extern LRESULT ImGui_ImplWin32_WndProcHandler(HWND hWnd, UINT msg, WPARAM wParam,
LPARAM lParam);

// Global helper to append to ImGui text buffer
static ImGuiTextBuffer g_logBuffer;
static bool g_scrollToBottom = false;

// Append a C-string or std::string to the GUI log
static void appendLog(const char* s)
{
    g_logBuffer.appendf("%s", s);
    g_scrollToBottom = true;
}
static void appendLog(const std::string& s) { appendLog(s.c_str()); }

// Replaces system("cls") clear function
static void clearLog()
{
    g_logBuffer.clear();
    g_scrollToBottom = true;
}

// More Win32 ImGui setup code
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    if (ImGui_ImplWin32_WndProcHandler(hWnd, msg, wParam, lParam))
        return true;

    switch (msg)
    {
    case WM_SIZE:
        if (g_pSwapChain && wParam != SIZE_MINIMIZED)
        {
            CleanupRenderTarget();
            g_pSwapChain->ResizeBuffers(0, (UINT)LOWORD(lParam),
(UINT)HIWORD(lParam), DXGI_FORMAT_UNKNOWN, 0);
            CreateRenderTarget();
        }
        return 0;
    case WM_DESTROY:
        PostQuitMessage(0);
        return 0;
    default:
        return DefWindowProc(hWnd, msg, wParam, lParam);
    }
}

```

```

// WinMain setup code
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE, LPSTR, int)
{
    // Serial port connection
    appendLog("Connecting to COM6...\n");
    HANDLE hSerial = CreateFileA(
        "\\\\.\\COM6", GENERIC_READ, 0, 0, OPEN_EXISTING, 0, 0);
    if (hSerial == INVALID_HANDLE_VALUE) {
        appendLog("Error: Could not open COM6.\n");
        return 1;
    }

    DCB dcbSerialParams = { 0 };
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
    if (!GetCommState(hSerial, &dcbSerialParams)) {
        appendLog("Error: Could not get serial parameters.\n");
        CloseHandle(hSerial);
        return 1;
    }

    dcbSerialParams.BaudRate = CBR_9600;
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;
    if (!SetCommState(hSerial, &dcbSerialParams)) {
        appendLog("Error: Could not set serial parameters.\n");
        CloseHandle(hSerial);
        return 1;
    }

    COMMTIMEOUTS timeouts = { 0 };
    timeouts.ReadIntervalTimeout = 50;
    timeouts.ReadTotalTimeoutConstant = 50;
    timeouts.ReadTotalTimeoutMultiplier = 10;
    SetCommTimeouts(hSerial, &timeouts);

    appendLog("Connected to COM6 at 9600 baud.\n\n");

    // Draw external display window
    WNDCLASSEX wc = { sizeof(WNDCLASSEX), CS_CLASSDC, WndProc, 0L, 0L,
        GetModuleHandle(NULL), NULL, NULL, NULL, NULL, L"ImGuiMorseWindowClass",
        NULL };
    RegisterClassEx(&wc);
    HWND hwnd = CreateWindowW(wc.lpszClassName, L"Morse GUI",
        WS_OVERLAPPEDWINDOW, 100, 100, 1280, 800, NULL, NULL, wc.hInstance, NULL);

    if (!CreateDeviceD3D(hwnd)) {

```

```

        CleanupDeviceD3D();
        UnregisterClass(wc.lpszClassName, wc.hInstance);
        appendLog("Failed to initialize Direct3D\n");
        CloseHandle(hSerial);
        return 1;
    }

    ShowWindow(hwnd, SW_SHOWDEFAULT);
    UpdateWindow(hwnd);

    // ImGui initialization
    IMGUI_CHECKVERSION();
    ImGui::CreateContext();
    ImGuiIO& io = ImGui::GetIO();
    io.FontGlobalScale = 1.0f;

    ImGui_ImplWin32_Init(hwnd);
    ImGui_ImplDX11_Init(g_pd3dDevice, g_pd3dDeviceContext);

    // Parser logic begins here
    std::list<std::string> word = {};
    std::list<std::string> letter = {};
    std::string input;
    bool shouldDecode = false;
    DWORD lastReceivedTime = GetTickCount();

    clearLog();

    bool done = false;
    MSG msg;
    ZeroMemory(&msg, sizeof(msg));
    while (!done)
    {
        // Physical forced exit through keyboard
        while (PeekMessage(&msg, NULL, 0U, 0U, PM_REMOVE))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
            if (msg.message == WM_QUIT) done = true;
        }
        if (done) break;

        // Various input logic
        if (readLineFromSerial(hSerial, input)) {
            if (input == "dot" || input == "dash") { // Normal input
                letter.push_back(input);
                lastReceivedTime = GetTickCount();
            }
        }
    }
}

```

```

        appendLog(input + "\n");
    }
    else if (input == "-") { // Backspace button
        if (!word.empty()) word.pop_back();
        appendLog("\nLAST INPUT REMOVED, PROCEED\n\n");
    }
    else if (input == "^r") { //Reset button
        letter = {};
        word = {};
    }
    else if (input == "DOT calibrated = ") {
        appendLog("Dot calibrated.\n");
        appendLog("-Input dash-");
    }
    else if (input == "DASH calibrated = ") {
        appendLog("Dash calibrated.");
    }
    else if (input == "CALIBRATION COMPLETE") {
        appendLog("Calibration complete.");
        appendLog("System ready momentarily");
        std::this_thread::sleep_for(std::chrono::seconds(2));
        clearLog();
        appendLog("Enter message:\n");
    }
    else if (input == "/") { // Print button
        appendLog("\n");
        for (const std::string& s : word) appendLog(s);
        appendLog("\n");
        std::list<std::string> sos = { "S","O","S" };
        std::list<std::string> key = { "K","E","Y" };
        std::list<std::string> maze = { "M","A","Z","E" };
        std::list<std::string> elsie = { "E","L","S","I","E" };
        std::list<std::string> east = { "E","A","S","T" };
        std::list<std::string> lewis = { "L","E","W","I","S" };

        if (word == sos) {
            appendLog("\nDuring WWI, people sent messages through a
telegraph,\nthe machine you are using now!\n\n");
            appendLog("SOS is a well-known distress signal in morse
code\nmeaning -save our souls- or -save our ship-\n");
            appendLog("This signal was used extensively during WWI by
military vessels and aircraft.\n\n");
            appendLog("Its importance during the Great War was vital \nas
the SOS signal alerted rescue teams of downed aircrafts or ships.\n");
            appendLog("This distress signal saved many lives.\n\n");
        }
    }
}
```

```

        if (word == key) {
            appendLog("\nAlthough few fully realize, the American Indians
were key to the outcome of WWI.\n\n");
            appendLog("-Many were drafted, but the majority volunteered to
fight for America\n\n");
            appendLog("-Choctaw and Cherokee code-talkers participated in
the Meuse-Argonne Offensive in the fall of 1918:\n");
            appendLog("The key battle that historian Geoffrey Wawro
describes as having -cut the German throat-\n\n");
            appendLog("'The North American Indian took his place beside
every other American, \n");
            appendLog("In offering his life in the great cause, \n");
            appendLog("Where as a splendid soldier, \n");
            appendLog("He fought with the courage and valor of his
ancestors.\n");
            appendLog("General John Pershing\n\n");
        }

        if (word == maze) {
            appendLog("\nNot many people get a first-hand perspective from
soldiers\non a really important aspect of the WWI living situation: Trench
Life.\n\n");
            appendLog("Gilbert Williams, a WWI private, wrote a letter
describing his experiences:\n");
            appendLog(" -Soldiers seem to spend three times as much time in
the trenches than out\n");
            appendLog(" -'a veritable maze of trenches,' where explosives
are going off constantly\n");
            appendLog(" -The fighting was so strong that the dead could
only be buried in the sides of the trenches\n\n");
            appendLog("Trench life was gruesome and hard, yet the soldiers
still fought on.\n\n");
        }

        if (word == elsie) {
            appendLog("The story of Elsie Maud Inglis, although less
known,\nhad a huge medical impact in WWI.\n");
            appendLog("She suggested to the War Office the creation \nof a
fully women medical unit on the Western front.\n");
            appendLog("Initially rejected, she persisted and contacted the
French,\n who supported her idea.\n\n");
            appendLog("Two auxiliary hospitals were sent to Russia and
Serbia, \nmajorly subduing the Typhus epidemic.\n\n");
        }

        if (word == east) {
            appendLog("The Battle of Tannenberg was an influential battle

```

```

for Germany, \noccuring along their eastern front.\n\n");
            appendLog("The German military encircled and defeated the
Russian Second Army,\n");
            appendLog("halting the Russian advance into German territory
for the rest of WWI.\n");
            appendLog("They applied these same tactics effectively in
future engagements.\n\n");
            appendLog("With pressure taken off the eastern front,\n");
            appendLog("Germany could focus more resources on the western
front.\n\n");
            appendLog("The -Tannenberg myth- led to a nationalistic belief
in the steadfastness and military prowess\n");
            appendLog("of the Germans, even after the war had ended.\n\n");
        }

        if (word == lewis) {
            appendLog("Sidney Lewis was the youngest authenticated British
soldier in WW1:\n");
            appendLog("He enlisted at the age of 12 and fought in the
Battle of the Somme in 1916.\n");
            appendLog("Although, he was eventually sent back home according
to the demands of his mom.\n\n");
            appendLog("In WW1, it was not uncommon for young boys to serve
before they turned 18,\n");
            appendLog("despite the legal enlisting age being 18 years
old.\n\n");
            appendLog("Underage boys might join for various reasons such
as:\n");
            appendLog("patriotism, recruiting pressure, or the desire for
escape and adventure.\n\n");
        }

        word.clear();
    }

}

// Letter decoding timeout. Changable, within code
DWORD now = GetTickCount();
if (!letter.empty() && (now - lastReceivedTime > 2000)) {
    shouldDecode = true;
    appendLog("-Letter time completed: Decoding letter-\n");
}

// Decoding logic
if (shouldDecode && !letter.empty()) {
    auto it = letter.begin();

```

```
if (it != letter.end() && *it == "dot") {
    ++it;
    if (it != letter.end() && *it == "dot") {
        ++it;
        if (it != letter.end() && *it == "dot") {
            ++it;
            if (it != letter.end() && *it == "dot") {
                appendLog("H");
                word.push_back("H");
            }
            else if (it != letter.end() && *it == "dash") {
                appendLog("V");
                word.push_back("V");
            }
            else {
                appendLog("S");
                word.push_back("S");
            }
        }
        else if (it != letter.end() && *it == "dash") {
            ++it;
            if (it != letter.end() && *it == "dot") {
                appendLog("F");
                word.push_back("F");
            }
            else {
                appendLog("U");
                word.push_back("U");
            }
        }
    }
    else {
        appendLog("I");
        word.push_back("I");
    }
}
else if (it != letter.end() && *it == "dash") {
    ++it;
    if (it != letter.end() && *it == "dot") {
        ++it;
        if (it != letter.end() && *it == "dot") {
            appendLog("L");
            word.push_back("L");
        }
        else {
            appendLog("R");
            word.push_back("R");
        }
    }
}
```

```
        }
    else if (it != letter.end() && *it == "dash") {
        ++it;
        if (it != letter.end() && *it == "dot") {
            appendLog("P");
            word.push_back("P");
        }
        else if (it != letter.end() && *it == "dash") {
            appendLog("J");
            word.push_back("J");
        }
        else {
            appendLog("W");
            word.push_back("W");
        }
    }
    else {
        appendLog("A");
        word.push_back("A");
    }
}
else {
    appendLog("E");
    word.push_back("E");
}
}
else if (it != letter.end() && *it == "dash") {
    ++it;
    if (it != letter.end() && *it == "dot") {
        ++it;
        if (it != letter.end() && *it == "dot") {
            ++it;
            if (it != letter.end() && *it == "dot") {
                appendLog("B");
                word.push_back("B");
            }
            else if (it != letter.end() && *it == "dash") {
                appendLog("X");
                word.push_back("X");
            }
            else {
                appendLog("D");
                word.push_back("D");
            }
        }
    }
    else if (it != letter.end() && *it == "dash") {
        ++it;
    }
}
else if (it != letter.end() && *it == "dash") {
```

```

        if (it != letter.end() && *it == "dot") {
            appendLog("C");
            word.push_back("C");
        }
        else if (it != letter.end() && *it == "dash") {
            appendLog("Y");
            word.push_back("Y");
        }
        else {
            appendLog("K");
            word.push_back("K");
        }
    }
    else {
        appendLog("N");
        word.push_back("N");
    }
}
else if (it != letter.end() && *it == "dash") {
    ++it;
    if (it != letter.end() && *it == "dot") {
        ++it;
        if (it != letter.end() && *it == "dot") {
            appendLog("Z");
            word.push_back("Z");
        }
        else if (it != letter.end() && *it == "dash") {
            appendLog("Q");
            word.push_back("Q");
        }
        else {
            appendLog("G");
            word.push_back("G");
        }
    }
    else if (it != letter.end() && *it == "dash") {
        appendLog("O");
        word.push_back("O");
    }
    else {
        appendLog("M");
        word.push_back("M");
    }
}
else {
    appendLog("T");
    word.push_back("T");
}

```

```

        }
    }
    else {
        appendLog("Error decoding input, check values");
    }

    letter.clear();
    shouldDecode = false;
    appendLog("\n");
}

// ImGui rendering
ImGui_ImplDX11_NewFrame();
ImGui_ImplWin32_NewFrame();
ImGui::NewFrame();

// Font scale setup
ImGui::Begin("Controls");
ImGui::Text("Font scale:");
ImGui::SliderFloat("Scale", &io.FontGlobalScale, 0.5f, 3.0f);
ImGui::SameLine();
if (ImGui::Button("Clear")) { clearLog(); }
ImGui::End();

// Scroll with new input or output
ImGui::Begin("Serial Output");
ImGui::BeginChild("ScrollingRegion", ImVec2(0, 0), false,
ImGuiWindowFlags_HorizontalScrollbar);
ImGui::TextUnformatted(g_logBuffer.begin());
if (g_scrollToBottom) {
    ImGui::SetScrollHereY(1.0f);
    g_scrollToBottom = false;
}
ImGui::EndChild();
ImGui::End();

// Render ImGui display
ImGui::Render();
const float clear_color[4] = { 0.1f, 0.1f, 0.1f, 1.0f };
g_pd3dDeviceContext->OMSetRenderTargets(1, &g_mainRenderTargetView, NULL);
g_pd3dDeviceContext->ClearRenderTargetView(g_mainRenderTargetView,
clear_color);
ImGui_ImplDX11_RenderDrawData(ImGui::GetDrawData());
g_pSwapChain->Present(1, 0); // vsync

Sleep(1);
}

```

```
// ImGui exit code
ImGui_ImplDX11_Shutdown();
ImGui_ImplWin32_Shutdown();
ImGui::DestroyContext();

CleanupDeviceD3D();
DestroyWindow(hwnd);
UnregisterClass(wc.lpszClassName, wc.hInstance);
CloseHandle(hSerial);
return 0;
}
```