Odd Semester (2024)

BINUS UNIVERSITY

BINUS INTERNATIONAL

Morse Code Training System

Student Information:

Surname:	**Given Name:**	**Student ID Number:**
Kusnadi	Clarissa Audrey Fabiola	2602118490
Sutiono	Rafael	2602174535

Course Code : COMP6100001 **Course Name :** Software Engineering

Class : L5AC **Lecturer** : Satrio Pradono, S.Kom., M.T.I.

Type of Assignment: Final Project Report

Table of Contents

I. Introduction

Numbers 23:23: “What hath God wrought” (“What God has done”), a quotation from the Bible was the official first Morse code message transmitted by the telegraph in 1844 (Silva et al., 2016).

w h a t h a t h G o d w r o u g h t

.-- .... .- - .... .- - .... --. --- -.. .-- .-. --- ..- --. .... -

Morse code is a method designed for encoding text characters used in telegraphy communication (Nalajala et al., 2016). Character alphabets and numbers encoded using sequences of two different electronic pulses, short and long. A dot “.” is a short pulse, while a dash “-“ is a long pulse (Cao et al., 2020). To be specific, the length of a dash is thrice that of a dot (Tiwari et al., 2022). While modern technology has mostly replaced Morse code in everyday use, it remains relevant in certain fields, such as military communications, emergency services, and navigation (You & Weng, 2024). This invention has revolutionized long-distance communication that transforms how governments, militaries, and business operated.

Letter	Morse
A	.-
B	-...
C	-.-.
D	-..
E	.
F	..-.
G	--.
H	....
I	..
J	.---
K	-.-
L	.-..

Letter	Morse
O	---
P	.--.
Q	--.-
R	.-.
S	...
T	-
U	..-
V	...-
W	.--
X	-..-
Y	-.--
Z	--..

Number	Morse
0	-----
1	.----
2	..---
3	...--
4	....-
5	.....
6	-....
7	--...
8	---..
9	----.

M	--
N	-.

Pramuka is the term for members of *Gerakan Pramuka Indonesia* (Indonesian Scout Movement), the national scouting organization of Indonesia. It involves students dressed in dark and light brown uniforms participating in the extracurricular school activity of scouting. *Pramuka* teaches students aged 7 to 25 years all about life skills, soft skills, and hard skills outside the school and family environments (Fatimah et al., 2018). One of its requirements is proficiency in Morse code, which can be quite challenging or intimidating to learn since it requires memorization for every letter and number. Traditionally, this learning method was done through the use of flashcards and notes, and this method lacks real feedback and interactivity, making them not quite engaging and effective for new learners.

Therefore, in the face of rapid advancements in technology, this can become a problem of the past. We can create a more accessible and engaging learning tool, with the advancement of the Arduino platform. Arduino is a low-cost and open-source I/O board that capitalizes on a large community that has flourished around its concept (Kondaveeti et al., 2021). Furthermore, parts used in Arduinos are affordable, lightweight, and compact, making them the perfect candidate for creating a portable and interactive Morse Code training device that provides instant feedback to the user and comes with features such as a decoder with a WPM counter and a practice mode that lets users practice typing in morse code (Waqar et al., 2019). All of it comes packaged in a simple and neatly designed box smaller than a shoe box.

Such a simple system designed to interact with learners can help reduce the steep learning curve for Morse Code learners, specifically aiming for a young audience of *Pramuka* scouts.

II. Literature Review

Morse Glasses is an IoT communication system based on Morse code created specifically for users with speech impairments. It works by translating the tracked patient's blinks into a generated speech. The sequence of Morse codes that have been encoded into alphabets and sentences are displayed on devices installed with Morse Glasses mobile application (Tarek et al., 2022).

Furthermore, a signature-based Morse code encoding for an LED-based optical layer 2 data communication was created to make wireless radio communications underwater. Traditionally, underwater communications use a physical connector that requires cables or Fiber optics, which can be quite a hassle since it could create hydrodynamic drag or end up entangled. Therefore, a solution was created for wireless communications underwater using LED, Morse code, and Arduino Microcontroller. It works by transmitting the received data from visible light using an LED that is controlled by the Arduino, turning it on and off based on the encoding systems specified by the Morse code standard. It is then decoded by a photodiode, matching the Morse code encoding with a created database (Lim et al., 2019).

Another Arduino-based Morse code project is a wearable triboelectric nanogenerator from waste materials for autonomous information transmission. Through a low-power electronic device, wearable autonomous communication is created to generate a Morse code. A customized LabVIEW program can be used by users to decode the Morse code, displayed on a computer screen that is controlled by an Arduino controller (Dudem et al., 2022).

The projects mentioned above focus more on the application of Morse code to improve communication in everyday life, rather than education. This makes our project different, as we aim to use our technology to create an interactive and accessible tool for students, specifically for the *Pramuka* program. Furthermore, our project will use the same encoding and decoding for Morse code in general. However, we will implement Morse code input through the use of buttons and visualize it on a screen with LEDs and buzzers to indicate the dots and dashes.

III. Methodology

A. Software Development Life Cycle (SDLC)

A software development life cycle is a structured process that is used to design, develop, test, and maintain a product development (Khan, 2021). The choice of method used for software engineering development projects is crucial. One of which is the Agile methodology, which has gained popularity as it has greater consistency in managing software engineering projects (Alaidaros et al., 2019). It is said to have a successful completion rate of 40%, in comparison to waterfall projects that only have a 15% rate (Mishra & Alzoubi, 2023). Furthermore, according to an Agile survey, it has been revealed that 95% of the respondents' projects using Agile methodology are successful (Herdika & Budiardjo, 2020).

It has been known for its responsiveness to changes, less paperwork, and rapid development, which can shorten the project's duration and ensure a proper timeline (Kumar & Bhatia, 2014). Several of its frameworks include Scrum, Kanban, XP, Behavior-driven development, Feature-driven software development, Dynamic Systems Delivery, and Crystal (Kumar & Bhatia, 2014).

There were several agile frameworks that we considered upon beginning our Morse code trainer project. One popularly used agile framework is Scrum, a rigid framework that makes use of sprints for progress tracking and reviewing. We went with the Kanban framework as it is less rigid and process-heavy than Scrum (Hron & Obwegeser, 2022), suitable for a two-person team. Another framework we considered is Extreme Programming, which heavily focuses on perfecting technical development and implementation (Sohaib et al., 2019). But we chose Kanban over it because Extreme Programming is software-heavy and we wanted to focus on both hardware and software without prioritizing one over the other.

Kanban offers a visual system for tracking and managing work efficiently through a unique tool called Kanban Board (Dalton, 2019). With the illustrative approach, the progress of the whole project can be easily understood at a glance (Striewe, 2019). Kanban boards include various columns that can be added according to one's project needs. Progress of the projects can be seen easily as the jobs sift across columns. Using the Kanban Board allows the team to focus on one goal at a time in an

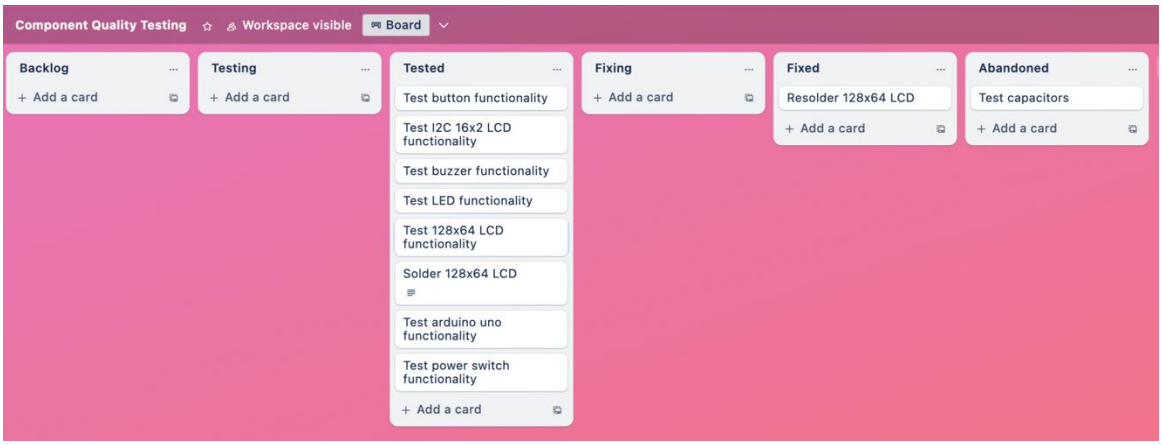
estimated time to avoid build-up in a phase or wasted time. Furthermore, it allows iterative software development, where the team can build the product step by step, allowing testing on each feature built for identification and correction (Saleh et al., 2019).

The Kanban Board is composed of lists and cards to organize the work items. Kanban lists are the stages of a process (columns), while Kanban cards are the specific tasks that need to be completed to finish a list item (McLean & Canham, 2018). Kanban cards include details like task name, description, who is responsible for it, and current status. To create our Kanban Board, we used trello.com. Since many tasks needed to be tested, we decided to split the Kanban Board into three sprints. Sprint is a time-boxed effort when a team needs to finish a certain amount of tasks (Sharma et al., 2021). Our sprints include:

- Component Quality Testing: A sprint to ensure the components' are not faulty by testing if their functionality is working or not. In this sprint, there are six lists: Backlog, Testing, Tested, Fixing, Fixed, and Abandoned.
    - Backlog: Uncompleted tasks that need to be done, similar to a to-do list.
    - Testing: Components that require validating or testing to ensure functionalities.
    - Tested: Components that have been completed and gone through the testing phase regardless of failure or success.
    - Fixing: Components that require hardware/software fixes.
    - Fixed: Components that have successfully undergone hardware/software fixes.
    - Abandoned: Components that were discarded since they were deemed to be unnecessary/beyond our scope.

  In the Component Quality Testing Board, we tested out all of the components that we had bought previously. This includes: buttons, LCD, LED, Arduino UNO, and a power switch. During this process, we faced some difficulties soldering one of the components, which is the

LCD 128x64 ST7920. Therefore, we had to re-solder it again and fix the previous badly done solder. Furthermore, we decided to discard the capacitors, since the buttons worked well without it. Therefore, we decided to abandon the capacitors.

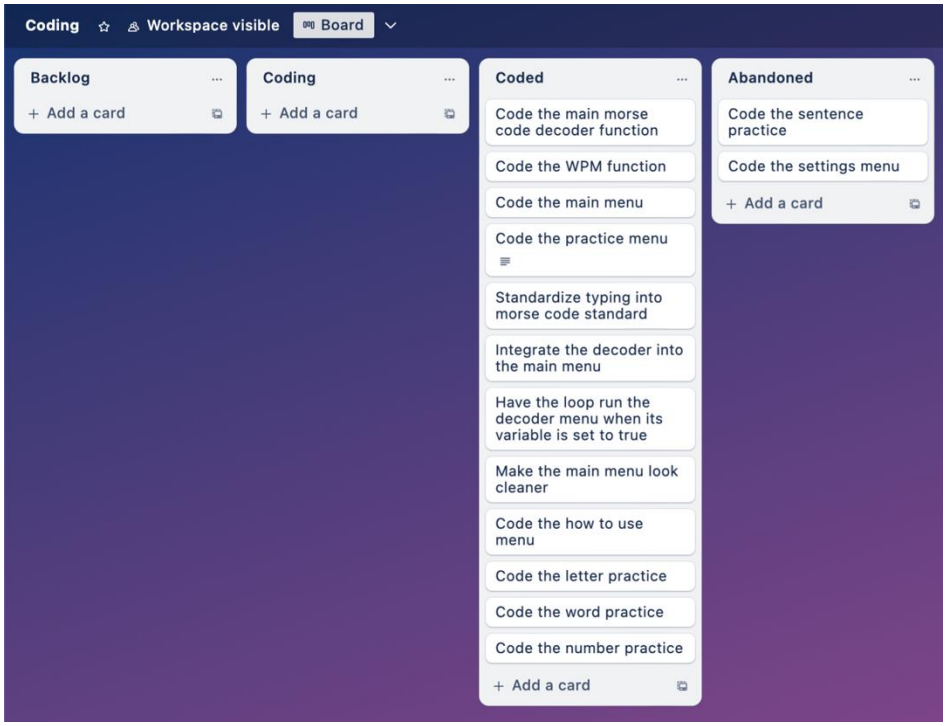


Sprint 1: Component Quality Testing Board

- Coding: A sprint to ensure that the code progresses in increments. In this sprint, there are four lists: Backlog, Coding, Coded, and Abandoned.
    - Backlog: Uncompleted tasks that need to be done, similar to a to-do list.
    - Coding: Coding tasks that are currently in development.
    - Coded: Coding tasks that have been successfully developed.
    - Abandoned: Coding tasks that were discarded since they were deemed to be unnecessary/beyond our scope.

In the Coding Board, we listed out all of the software components that needed to be coded. This includes: the main Morse code decoder function, WPM function, main menu, practice menu, and how-to-use menu. Our initial plan for the practice menu was to create a submenu within it that would allow users to choose to practice on letters, words, or sentences. However, when we tried to create a sentence submenu, the memory was too full, causing the system to crash. Therefore, we had to remove some features that we felt were redundant, such as the settings

menu. Furthermore, we decided to replace sentence practice with numbers to save memory.

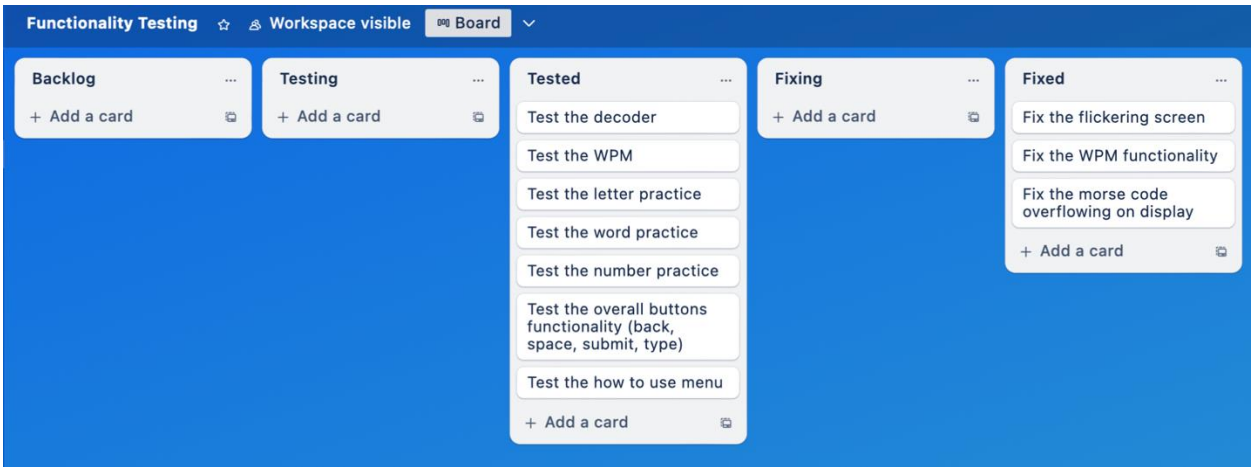


Sprint 2: Coding Board

- Functionality Testing: A sprint to ensure that the functionality of the device works as intended, and any software/hardware issues will be directed to the 'fixing' list to await being dealt with. In this sprint, there are five lists: Backlog, Testing, Tested, Fixing, and Fixed.
    - Backlog: Uncompleted tasks that need to be done, similar to a to-do list.
    - Testing: A list that contains software components of our device that need to be tested individually to make sure they function as intended.
    - Tested: Software components that have undergone the testing phase.
    - Fixing: Software components that require debugging and fixing because they do not work as intended.

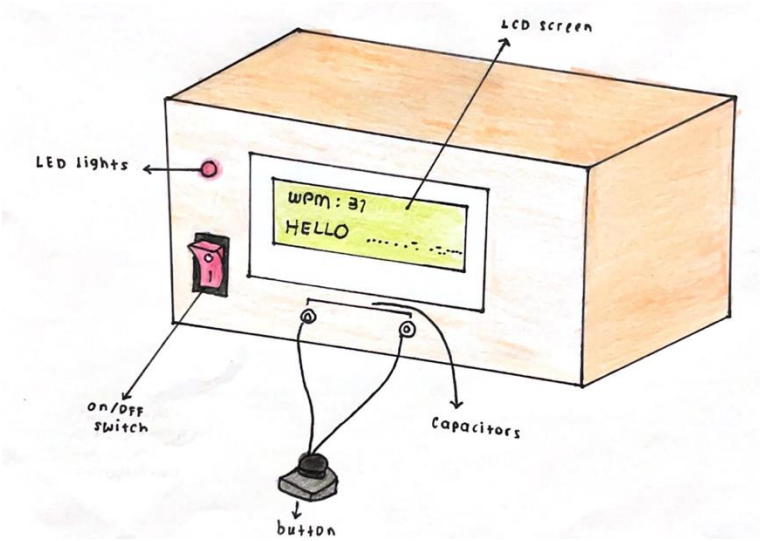
- Fixed: Software components that have already been fixed and successfully implemented.

In the Functionality Testing Board, we did a post-test on all of the functionalities that have been created from the tasks of the coding board. Through testing, several bugs and errors were found, such as screen flickering, incorrect WPM calculation, and Morse code overflowing the screen because it was too long. The flickering screen was caused by memory being too full, so the system did not run properly, therefore we modified some codes, such as changing String to Chars to reduce memory. Furthermore, the incorrect WPM calculation was caused by incorrect formula logic, which was then fixed by changing the logic. Finally, the overflowing Morse code was caused by the failure to implement its separation logic if it reached the end of the display. Therefore, the logic for that was created.



Sprint 3: Functionality Testing Board

B. Prototype



Initial Prototype Design of Morse Code Training System

The prototype of the Morse code training system can be seen above. Our team focused on designing a compactable and easy-to-use device for students to learn Morse code. The LCD screen is placed on the front center of the box to display the training system. The LED light indicator is placed on the top to represent the on-off sequence of the Morse code. Then, a switch is placed on the bottom to toggle the device on or off. Finally, a button is where the user can input their Morse code. All of the wiring components, including the Arduino, will be placed inside a box to make it neat and compact.

C. Materials Utilized

The following components were used to make the Morse code training system:

- Arduino UNO R3 (1 unit): The microcontroller board for handling inputs, outputs, and signal processing (Debele & Qian, 2020).
- USB cable (1 unit): To power up Arduino by connecting it to a power bank
- LCD Graphic 128x64 ST7920 (1 unit): To display the graphics (Yu, 2022).

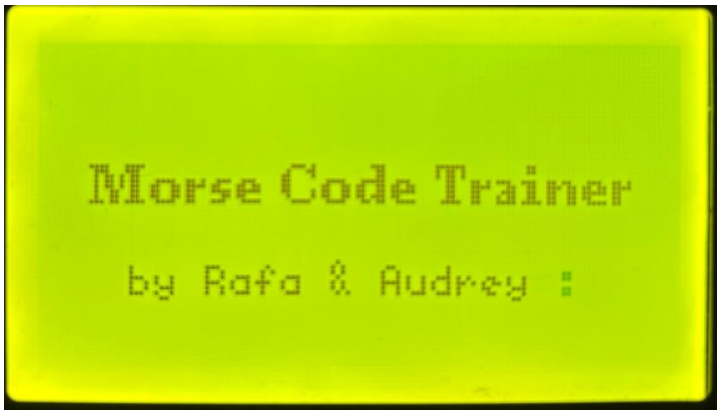
- RGB LED (2 units): To visualize the Morse code as a series of on-off lights and as a power indicator.
- Buzzer (1 unit): To produce beep sounds for dots and dashes.
- Aviation plug connector (1 unit): Large circular electrical connectors to connect the button modules to the Arduino board (Zhang et al., 2019).
- Ribbon cable (multiple): Used to pile together the cables used to connect the LCD screen (Smith et al., 2021).
- Jumper wires: Used in connecting individual pins of the buttons together, then connecting them to the aviation plug.
- PCB Dot Matrix: To screw the Arduino uno on the inside wall of the box.
- Resistor 1k ohm (2 units): To limit the flow of electric current for buzzer and LED (Astbury, 1935).
- Push button (3 units): Used by the user to navigate and use the Morse code training system.

D. Project Workflow

In this section, the logic of our code will be explained below. The program was coded in the Arduino IDE application using the programming language of C++. Arduino IDE is a software that can be used for digital control and data acquisition applications (Amestica et al., 2019). The full code to our project can be found in the repository below: https://github.com/rafaelsutiono/Morse-Code-Trainer

Libraries used for creating the program of Morse code training system include:

- U8glib v2: A graphical library to correctly functionate the LCD screen (Campbell & Jones, 2020).
- U8x8: To display characters by using the library fonts on the LCD screen.
- Wire: Allows you to communicate with I2C devices (Kurniawan, 2021).



Welcome Screen of Morse Code Training System

When booted up, the screen greets you with a welcome screen with our project title and our names on it, then it starts on the main menu, where there are three frames presented vertically.

The main menu can be navigated by pressing the red button, which goes down one highlighted frame each time it is pressed. To select a highlighted frame, the red button must be held down for 300 milliseconds and then let go.

Main Menu of Morse Code Training System

The first menu, the 'Decoder' menu, opens up the Morse code decoder, where the user is free to type anything they want. The user is greeted with a prompt to start decoding and when the user is ready, the user types in standard Morse code fashion using the red button, where each character being typed must have a small gap in between presses to register as a character. In our device, the user can press

the white button to register a word space. This is to accommodate users of different typing speeds, where different users might have different gap times in between words. On top of that, a WPM counter is displayed on top of the screen to track the user's typing speed in real-time. Finally, the user can press the blue button at any time to go back to the main menu.

The loop() function calls the main morseCode() function when 'Decoder' is selected, setting a variable decoderOpen to true which consists of the morseCode() function. Every dot and dash entered is written into a string variable, and when a gap long enough is detected in-between button presses, the string variable is read and matched against a list that contains the mapping of every text character to morse. A matching character is returned if it is in the list, if not a question mark is returned. This is the core of how the morse code decoding works in our device.

The WPM counter follows the PARIS standard (Guri, 2022). Our device follows morse code standard dot and dash durations; an int variable 'totalDots' is used to keep track of the sum total of dot duration entered in the decoder. Typing in a dot adds 2 to totalDot (one for the dot itself and one for spacing in-between morse characters). Typing in a dash adds 4 to totalDot (three for the dash, since a dash is three dot durations, and one for spacing in-between morse characters). As mentioned earlier, when a gap long enough is detected it returns a letter associated with the entered morse string. When this happens, totalDot is incremented by 3, since 3 dot durations are the standard for letter spacing in morse code. Additionally, when the white button is pressed (in order to give a word space), totalDots gets incremented by 4, since 7 dot durations are the standard for word spacing in morse code, and 3 of those dot durations are already accounted for in letter spacing.

The countWPM() function relies on a stopwatch that starts as soon as the user presses the red button for the first time upon using the decoder. The elapsed time is calculated in minutes and as such there is a statement to divide the elapsed time by 60000 as the default time format in C++ is in milliseconds. After that happens, totalDots is divided by 50 (the PARIS standard), and the resulting number is divided by the elapsed time in minutes (Va3kot, 2023). The final number is the WPM, and it is updated every second.

Practice Submenus: Letter, Word, and Number

The second menu, the ‘Practice’ Menu consists of three submenus: ‘Letter’, ‘Word’, and ‘Number’. The letter submenu is where users can practice entering single letters in Morse code, while the word submenu is where users can practice entering single words in Morse code. The Morse code function in the letter submenu is similar to the one used in the decoder, but without the character gap detection, since we are only inputting single letters (the result is automatically translated into a single letter). Lastly, the number submenu is where users can practice entering single numbers in Morse code.

The third menu, the ‘How to Use’ Menu when opened displays the first of two pages, wherein each page displays button functions in the decoder and practice menus respectively.

The entire process of the device is carefully placed in the singular loop function that the Arduino framework provides, with all three button reading functions placed first, followed by the morse code functions for each screen (during decoder or practice), and are each placed inside an if statement that only runs in the loop if the currentMenu variable is set to their respective screens. At the bottom of the loop function, the screen is always refreshed and updated.

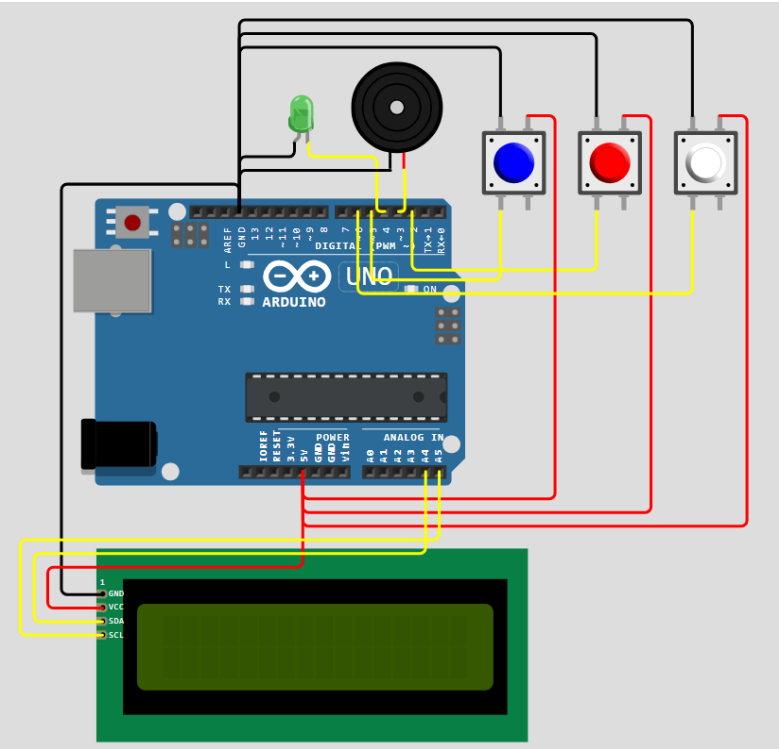
Measures were taken to reduce the memory used up on the flash memory, which mostly involves rendering frames and content only once or using loops so that the amount of drawStr functions are reduced. As such, functions to fix minor errors upon loading the decoder and practice word screen (a dash getting typed in

upon opening those menus) could be implemented after reducing the memory taken up.

IV. Results and Discussion

A. Hardware and Wiring Design

A circuit diagram is the foundation of electrical and electronic sciences, consisting of circuit components that specify the functionality of that circuit (Roy et al., 2020). Instead of using traditional methods like using paper to draw the circuit diagram, free online simulators in IoT education are available for free to visualize and test. One of which is Wokwi, a free online simulator that offers simulation for Arduino-based IoT projects (Atanasković et al., 2024). Wokwi is what our team decided to use to visualize our circuit diagrams, as it is free, easy to use, and provides lots of great examples. The picture below shows our circuit diagram for the Morse Code Training System created with Wokwi:



Morse Code Training System Circuit Diagram with Wokwi

Due to the unavailability of ST7920 128x64 displays on circuit makers, we replaced our LCD in the diagram with an I2C 16x2 LCD display. The wiring for our 128x64 ST7920 LCD uses 9 of its pins, listed as follows:

LCD pins	Arduino UNO Pins
GND	GND
VCC	5V
RS	PIN 10
R/W	PIN 11
E	PIN 13
PSB	GND
RST	PIN 8
BLA	3.3V
BLK	GND

Furthermore, the wiring for the buzzer is listed below:

Buzzer	Arduino UNO Pins
(+)	PIN 3
(-)	GND

The wiring for the buttons:

Red Button	Arduino UNO Pins
VCC	5V
OUT	PIN 2
GND	GND

Blue Button	Arduino UNO Pins
VCC	5V
OUT	PIN 5
GND	GND

White Button	Arduino Uno Pins
VCC	5V
OUT	PIN 6
GND	GND

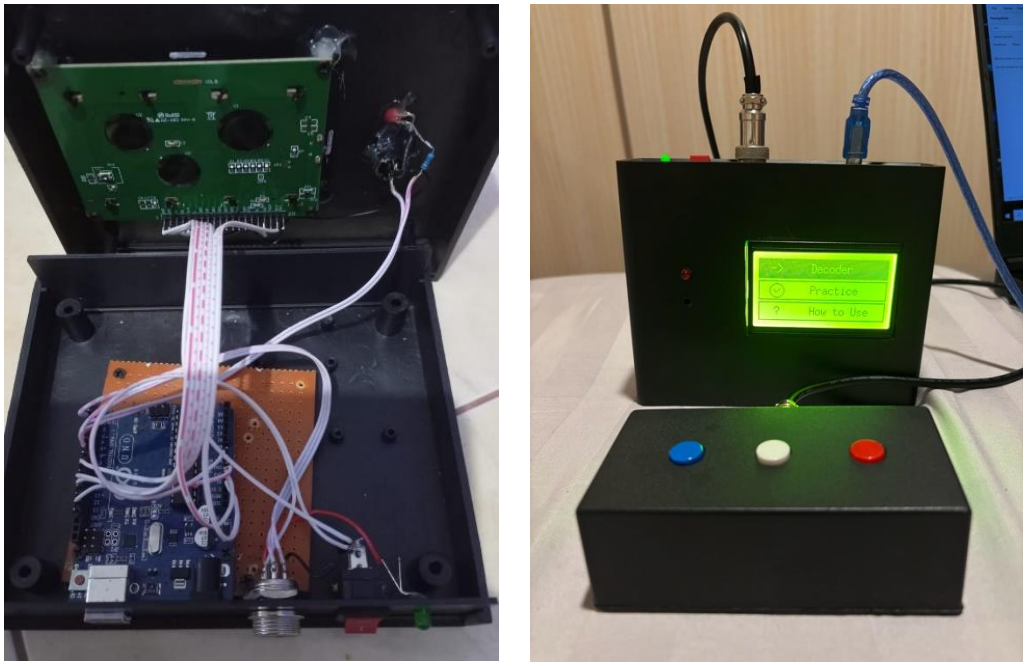
The wiring for the LED:

Red Button	Arduino UNO Pins
(+)	PIN 4 (Using 1k resistor)
(-)	GND

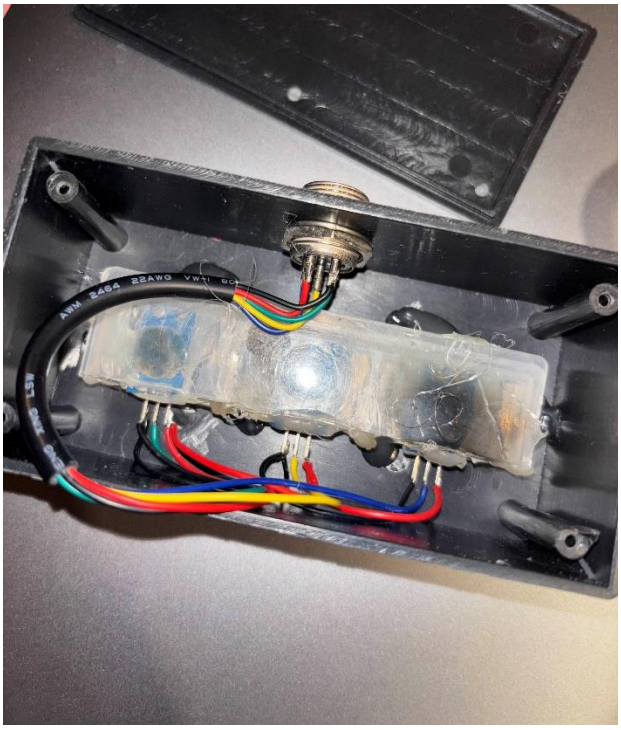
All 5V pins are connected to power switches to control the power on or off.

B. Final Product

The final design of the Morse code training system can be seen in the images below:



Final Product of the Morse Code Training System

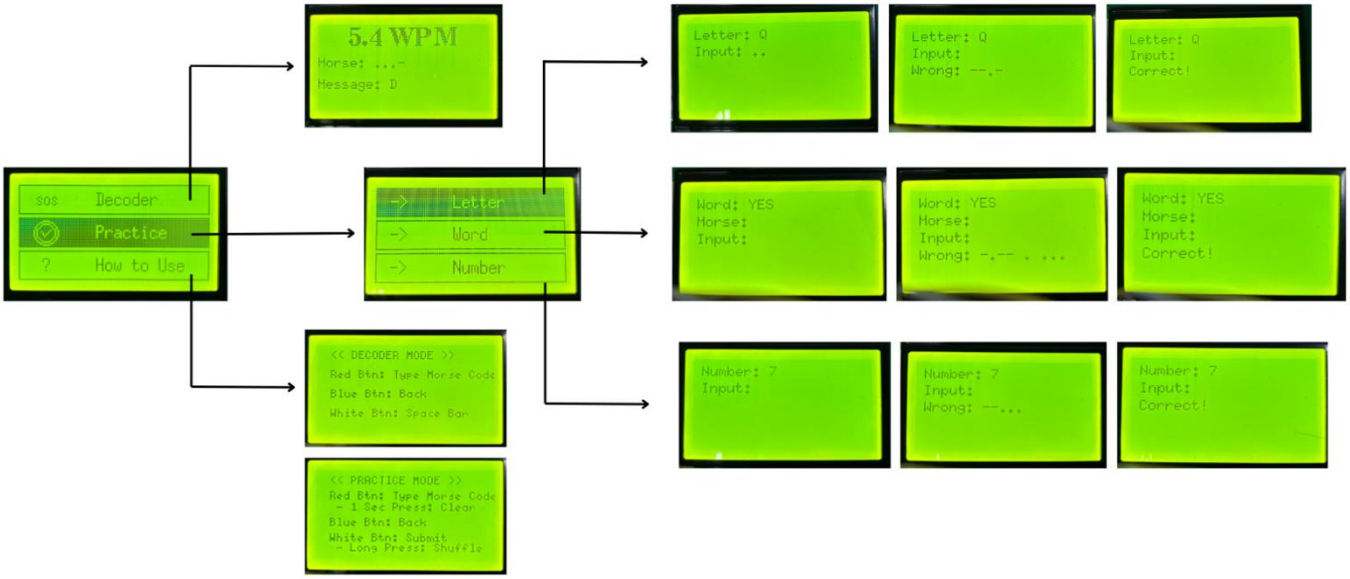


Inside the box where all three buttons are wired together

We have changed some of the design from the previously drawn prototype. Our final product comes in the form of a thin, lightweight, and compact box made out of PWC material, a durable composite material made from plastic and wood fibers. The box itself is custom-made and tailored to the dimensions of the hardware we used. Furthermore, we changed from using jumper wires to ribbon cables to make it more, which are soldered to connect the pins permanently. Like a monitor on a tabletop, the LCD screen is proudly displayed on the front center of the box, with the LED indicator placed on the left side of the LCD screen, lighting up according to the user's Morse code typing. A second LED is placed on the top side of the box to indicate that the device is powered on. Additionally, the top side also consists of the power switch, which is used to toggle the device on or off. The device itself is typically powered by a power bank. Meanwhile, the buttons used to navigate through and use the device are placed in front of the device, like a keyboard to a desktop computer. In making it, we realized that simply using one

button for all functionalities like back, space, and type, is not applicable. Therefore, we decided to use three different colored buttons, each with its own functionalities.

The final user interface (UI) navigation and interaction flow can be seen below:



Final UI Navigation and Interaction Flow of the Morse Code Training System

The final product of the Morse code training system consists of three main menus: Decoder, Practice, and How to Use.

- Decoder: Users can type Morse code freely, which is translated automatically. The typing speed is also calculated using the WPM Morse code formula.
- Practice: Includes three submenus. Users can practice Morse code by translating either letter, word, or number.
    - Letter: A randomized alphabet will be given to the user.
        - A correct answer: 'Correct!'
        - A wrong answer: 'Wrong: {the correct Morse code of the letter}'
    - Word: A randomized word will be given to the user.
        - A correct answer: 'Correct!'

        - A wrong answer: ‘Wrong: {the correct Morse code of the word}’
    - Number: A randomized number will be given to the user.
        - A correct answer: ‘Correct!’
        - A wrong answer: ‘Wrong: {the correct Morse code of the number}’
- How to Use: Includes two pages of simple manual instruction on how each of the colored buttons works.

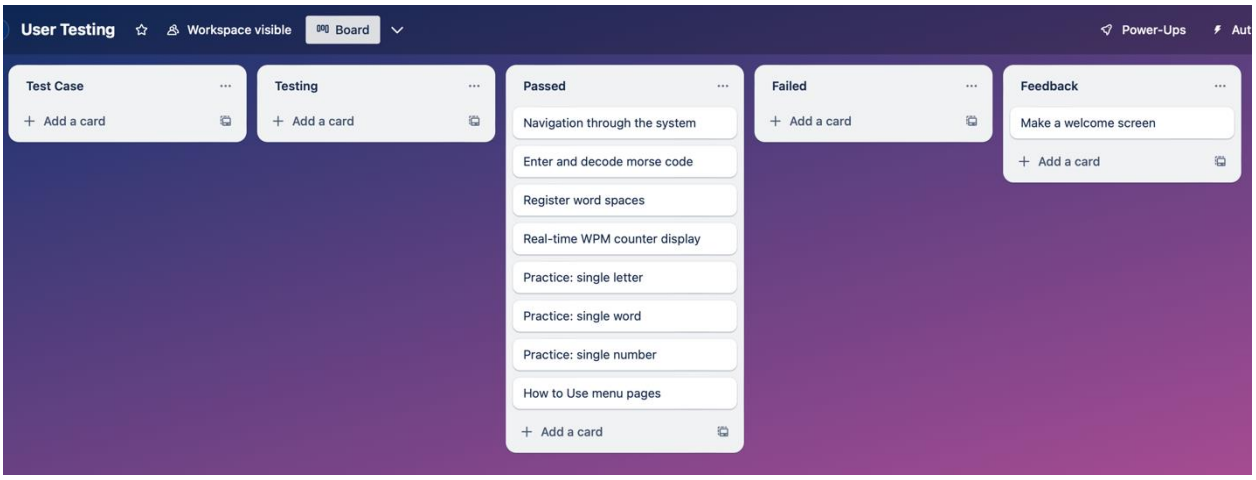
C. Evaluation and Impact

The Morse code training system project has been implemented and tested, successfully fulfilling the intended objectives of the system. The system will be effective in helping the teaching of Morse code, especially to young learners who participate in *Pramuka*.

User testing using our methodology, agile kanban board, was conducted to obtain additional feedback:

- User Testing: A sprint to test each of the functionalities on the user to gain direct feedback. In this sprint, there are six lists: Test Case, Testing, Passed, Failed, and Feedback.
    - Test Case: Includes test scenarios that need to be done.
    - Testing: Test Cases tasks that are currently being tested to the user.
    - Passed: Successfully tested cases with no issues.
    - Failed: Test cases that fail or there are issues encountered.

- Feedback: User feedback, suggestions, and insight gathered from testing (if any).



User Testing Kanban Board

After fixing the bugs found during functionality testing, to further make sure the refinement of our Morse code training system, we did a user testing to a 21-year-old student who is avid in learning Morse code. Through this, we gain feedback from the student to add an additional screen, which is the welcome screen, to greet the user. Other than that, other features that we have made work well. The student mention the system's ease of use and its interactive learning methodology. It has been said to boost both confidence and increase the learning rate in studying Morse code compared to traditional flashcards or paper-based methods. Additionally, its compact design makes it easy for students to carry around, so they can practice Morse code anywhere and anytime they want.

With our Morse code training system, the learning process can be fun and interactive. The incorporation of visual and auditory feedback to students can help them understand Morse code principles more deeply. Furthermore, the project used components that are easily accessible to ensure sustainability, which is Arduino and its components. Arduino's open nature allows others to replicate or build on the system, allowing innovation and collaboration among those in the tech community.

Overall, the effective implementation of the Morse code training system proves it can modernize and simplify Morse code education for *Pramuka*

students while promoting the integration of technology with traditional skill-building activities.

D. Limitations

Due to the Arduino Uno's limited flash/program memory of 32,256 bytes (Schubert et al., 2013) and static RAM of 2048 bytes, there were some functionalities that we proposed during the proposal phase of the project that could not be realized/fulfilled in the final product:

- Sentence practice mode was unfulfillable due to the large size of the existing code that is already taking up 99% of Uno's program space.
- Originally, the code's size exceeded the program space due to the number of words available in the random word list for generating words in the word practice mode. As such we had to greatly reduce the amount of words in the word list to just five words and simplify the way we render content.
- The originally proposed settings menu also could not be realized due to the limited space available.
- The practice letter section sometimes flickers when inputting a Morse code answer that is too long.
- With the way our Morse code decoder is programmed, it is difficult to have users decide their own dash length since there is a millisecond range for a button press to register as a dash. Without a settings menu where users can change those constants, users are forced to follow our pre-defined constants for dots, dashes, and long presses.
- The switch for powering on and off still has a problem controlling the LCD backlight. This is because the power supply comes directly from a USB port, which provides 5V. Meanwhile, the LCD backlight is powered through a different voltage source of 3.3V. Therefore, because of the separation of voltage, the LCD backlight remains ON when the switch is turned off.

E. Plans for Future Improvements

To address the limitations of the exceeded memory capacity, our team could transition from Arduino Uno to Mega increasing memory by 8x, and increasing the scope of usability. Many more new functions can be added this way, albeit at the cost of a larger and heavier box. An example could include adding over 100 words to the existing word list, and even the sentence practice module.

Additionally, a more well-designed UI/UX with animations implemented will also improve user experience should there be more program space, including cancelled features such as the settings menu. By having the settings menu, personalized settings could also be implemented, where users can decide their dash length in in to their liking. This way, it can further enhance their learning rate and experience.

Lastly, the power on and off button could be fixed by powering everything, specifically the LCD backlight, on the same power supply of 5V. Therefore, everything can be controlled by the switch in a uniform manner.

V. References

Alaidaros, H., Omar, M., & Romli, R. (2019). *A Theoretical Framework for Improving Software Project Monitoring Task of Agile Kanban Method* (pp. 1091–1099). https://doi.org/10.1007/978-3-319-99007-1_101

Amestica, O. E., Melin, P. E., Duran-Faundez, C. R., & Lagos, G. R. (2019). An Experimental Comparison of Arduino IDE Compatible Platforms for Digital Control and Data Acquisition Applications. *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, 1–6. https://doi.org/10.1109/CHILECON47746.2019.8986865

Astbury, N. F. (1935). The design, contruction, and use of resistors of calculable reactance. *Journal of the*

Institution of Electrical Engineers, *76*(460), 389–396. https://doi.org/10.1049/jiee-1.1935.0055

Atanasković, A., Dimitrijević, T., Ilić, N. M., & Čabarkapa, M. (2024). Empowering IoT Education Utilizing Free Online Arduino Simulators. *2024 59th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)*, 1–4. https://doi.org/10.1109/ICEST62335.2024.10639688

Campbell, T., & Jones, J. F. X. (2020). Design and implementation of a low cost, modular, adaptable and open-source XYZ positioning system for neurophysiology. *HardwareX*, *7*, e00098. https://doi.org/10.1016/j.ohx.2020.e00098

Cao, Q., Yu, C., Cheng, X.-F., Sun, W.-J., He, J.-H., Li, N.-J., Li, H., Chen, D.-Y., Xu, Q.-F., & Lu, J.-M. (2020). Polysquaramides: Rapid and stable humidity sensing for breath monitoring and morse code communication. *Sensors and Actuators B: Chemical*, *320*, 128390. https://doi.org/10.1016/j.snb.2020.128390

Dalton, J. (2019). Kanban Board. In *Great Big Agile* (pp. 187–188). Apress. https://doi.org/10.1007/978-1-4842-4206-3_36

Debele, G. M., & Qian, X. (2020). Automatic Room Temperature Control System Using Arduino UNO R3 and DHT11 Sensor. *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 428–432. https://doi.org/10.1109/ICCWAMTIP51612.2020.9317307

Dudem, B., Dharmasena, R. D. I. G., Riaz, R., Vivekananthan, V., Wijayantha, K. G. U., Lugli, P.,

Petti, L., & Silva, S. R. P. (2022). Wearable Triboelectric Nanogenerator from Waste Materials for Autonomous Information Transmission *via* Morse Code. *ACS Applied Materials & Interfaces*, *14*(4), 5328–5337. https://doi.org/10.1021/acsami.1c20984

Fatimah, D. D. S., Kurniawati, R., Tresnawati, D., & Ramdan, A. M. (2018). Designing instructional multimedia system for scout activities. *IOP Conference Series: Materials Science and Engineering*, *434*, 012061. https://doi.org/10.1088/1757-899X/434/1/012061

Guri, M. (2022). ETHERLED: Sending Covert Morse Signals from Air-Gapped Devices via Network Card (NIC) LEDs. *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 163–170. https://doi.org/10.1109/CSR54599.2022.9850284

Herdika, H. R., & Budiardjo, E. K. (2020). Variability and Commonality Requirement Specification on Agile Software Development: Scrum, XP, Lean, and Kanban. *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, 323–329. https://doi.org/10.1109/IC2IE50715.2020.9274564

Hron, M., & Obwegeser, N. (2022). Why and how is Scrum being adapted in practice: A systematic review. *Journal of Systems and Software*, *183*, 111110. https://doi.org/https://doi.org/10.1016/j.jss.2021.111110

Khan, N. A. (2021). Research on Various Software Development Lifecycle Models. In K. Arai, S. Kapoor, & R. Bhatia (Eds.), *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 3* (pp. 357–364). Springer International Publishing.

Kondaveeti, H. K., Kumaravelu, N. K., Vanambathina, S. D., Mathe, S. E., & Vappangi, S. (2021). A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations. *Computer Science Review*, *40*, 100364. https://doi.org/10.1016/j.cosrev.2021.100364

Kumar, G., & Bhatia, P. K. (2014). Comparative Analysis of Software Engineering Models from Traditional to Modern Methodologies. *2014 Fourth International Conference on Advanced Computing & Communication Technologies*, 189–196. https://doi.org/10.1109/ACCT.2014.73

Kurniawan, A. (2021). Arduino Nano 33 BLE Sense Board Development. In *IoT Projects with Arduino Nano 33 BLE Sense* (pp. 21–74). Apress. https://doi.org/10.1007/978-1-4842-6458-4_2

Lim, T. H., Haji Awang Damit, S. N., & Hazirah Azmi, N. A. (2019). Signature-Based Morse Code Encoding for a LED-Based optical Layer 2 data communication. *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 365–369. https://doi.org/10.1109/ICIEA.2019.8833886

McLean, J., & Canham, R. (2018). *Managing the Electronic Resources Lifecycle with Kanban*. *2*(1), 34–43. https://doi.org/doi:10.1515/opis-2018-0003

Mishra, A., & Alzoubi, Y. I. (2023). Structured software development versus agile software development: a comparative analysis. *International Journal of System Assurance Engineering and Management*, *14*(4), 1504–1522. https://doi.org/10.1007/s13198-023-01958-5

Nalajala, P., Godavarth, B., Raviteja, M. L., & Simhadri, D. (2016). Morse code generator using Microcontroller with alphanumeric keypad. *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 762–766. https://doi.org/10.1109/ICEEOT.2016.7754788

Roy, S., Bhattacharya, A., Sarkar, N., Malakar, S., & Sarkar, R. (2020). Offline hand-drawn circuit component recognition using texture and shape-based features. *Multimedia Tools and Applications*, *79*(41–42), 31353–31373. https://doi.org/10.1007/s11042-020-09570-6

Saleh, S. M., Huq, S. M., & Rahman, M. A. (2019). Comparative Study within Scrum, Kanban, XP Focused on Their Practices. *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 1–6. https://doi.org/10.1109/ECACE.2019.8679334

Schubert, T. W., D'Ausilio, A., & Canto, R. (2013). Using Arduino microcontroller boards to measure response latencies. *Behavior Research Methods*, *45*(4), 1332–1346. https://doi.org/10.3758/s13428-013-0336-z

Sharma, S., Kumar, D., & Fayad, M. E. (2021). An Impact Assessment of Agile Ceremonies on Sprint Velocity Under Agile Software Development. *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 1–5. https://doi.org/10.1109/ICRITO51393.2021.9596508

Silva, S., Valente, A., Soares, S., Reis, M. J. C. S., Paiva, J., & Bartolomeu, P. (2016). Morse code translator using the Arduino platform: Crafting the future of

microcontrollers. *2016 SAI Computing Conference (SAI)*, 675–680. https://doi.org/10.1109/SAI.2016.7556055

Smith, J. P., Mazin, B. A., Walter, A. B., Daal, M., Bailey, J. I., Bockstiegel, C., Zobrist, N., Swimmer, N., Steiger, S., & Fruitwala, N. (2021). Flexible Coaxial Ribbon Cable for High-Density Superconducting Microwave Device Arrays. *IEEE Transactions on Applied Superconductivity*, *31*(1), 1–5. https://doi.org/10.1109/TASC.2020.3008591

Sohaib, O., Solanki, H., Dhaliwa, N., Hussain, W., & Asif, M. (2019). Integrating design thinking into extreme programming. *Journal of Ambient Intelligence and Humanized Computing*, *10*(6), 2485–2492. https://doi.org/10.1007/s12652-018-0932-y

Striewe, M. (2019). Lean and Agile Assessment Workflows. In *Agile and Lean Concepts for Teaching and Learning* (pp. 187–204). Springer Singapore. https://doi.org/10.1007/978-981-13-2751-3_10

Tarek, N., Mandour, M. A., El-Madah, N., Ali, R., Yahia, S., Mohamed, B., Mostafa, D., & El-Metwally, S. (2022). Morse glasses: an IoT communication system based on Morse code for users with speech impairments. *Computing*, *104*(4), 789–808. https://doi.org/10.1007/s00607-021-00959-1

Tiwari, M., Kumar, G., Chambyal, M., & Jain, S. (2022). *Morse Tool—A Digital Communication Aid for Visually Impaired* (pp. 407–419). https://doi.org/10.1007/978-981-16-3675-2_31

Waqar, M., Inam, S., ur Rehman, M. A., Ishaq, M., Afzal, M., Tariq, N., Amin, F., & Qurat-ul-Ain. (2019). Arduino Based Cost-Effective Design and Development of a

Digital Stethoscope. *2019 15th International Conference on Emerging Technologies (ICET)*, 1–6. https://doi.org/10.1109/ICET48972.2019.8994674

You, X., & Weng, W. (2024). Real-Time Morse Code Decoding: Exploring Domestic SoC Applications. *2024 IEEE 4th International Conference on Software Engineering and Artificial Intelligence (SEAI)*, 296–300. https://doi.org/10.1109/SEAI62072.2024.10674174

Yu, H. (2022). Read and write methods for programmable chips. *2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 596–598. https://doi.org/10.1109/IPEC54454.2022.9777400

Zhang, M., Lu, Y., Li, X., Shen, Y., Wang, Q., Li, D., & Jiang, Y. (2019). Aviation Plug On-site Measurement and Fault Detection Method Based on Model Matching. *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 1–5. https://doi.org/10.1109/I2MTC.2019.8827118