AN ADAPTIVE MOBILE GUITAR APPLICATION TO ASSIST IN LEARNING GUITAR AND MUSIC CREATION USING MACHINE LEARNING AND MEMBRANE BUTTON MATRIX

Jiale Zhao [1], Soroush Mirzaee [2]

[1] Portola High School, 1001 Cadence, Irvine, CA 92618
[2] California State Polytechnic University, Pomona, CA, 91768

ABSTRACT

This paper addresses the challenge of creating an affordable and effective guitar learning system. Traditional guitar learning methods rely heavily on teacher-student interaction, which can be limited in terms of feedback and accessibility [10]. To solve this problem, we propose a system that uses membrane buttons on the guitar fretboard to detect user input, combined with machine learning to provide real-time feedback and corrections. The system converts raw guitar signals into a readable format and integrates with an application to enhance the learning experience. Key technologies include RP2040 for signal conversion and machine learning for input analysis. Challenges such as signal accuracy and real-time feedback were addressed by using membrane buttons, which are more accurate and cost-effective compared to other methods like video detection or audio analysis. The system was tested in various scenarios, demonstrating its potential to provide an interactive, accessible, and personalized guitar learning experience that can improve how students learn the instrument.

KEYWORDS

Adaptive, Assist, Guitar Learning, Music Creation, Machine Learning

1. INTRODUCTION

How can we create an affordable and accurate system for guitar learning? This project seeks to design a method to detect user input on a guitar fretboard digitally and create a teaching system built upon it. The solution also aims to enhance the guitar's functionality as a practice and teaching tool while allowing seamless signal conversion into MIDI or other formats to support music production. Currently, there's no effective way to export the guitar signal digitally by notes, which makes it difficult to identify how the user inputs the signal and therefore cannot create a teaching system for guitar learners. In other words, the guitar learners nowadays still need to follow instructions from a teacher in a traditional teacher–student model [2]. Guitar is used in a wide variety of musical styles, making it one of the arguably most widely used instruments. However, structural inconsistencies (differences in guitar sizes or string spacing) or the notational system used for this instrument versus other instruments (notes are written graphically for interpretation) can produce problems that make reading music and tuning the instrument particularly difficult for students trying to learn this instrument. It is challenging for pupils to

David C. Wyld et al. (Eds): MLNLP, ASOFT, CSITY, NWCOM, SIGPRO, AIFZ, ITCCMA – 2025
pp. 61-71, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152005

create their own musical style because of these problems [4]. Therefore, an affordable and convenient learning solution for guitar learners is crucial in the long run.

Some other methods, including the traditional teacher-student model, augmented reality (AR) systems, and AR-based apps, are also aimed at solving this problem. Although it provides individualized instruction, the teacher-student approach is constrained by the student's capacity to take in criticism and the teacher's availability. Because AR systems overlay audio-visual information to assist users practice methods, they offer immersive learning experience. However, the quality and expense of AR technology limits how successful these systems can be. Similar to this, augmented reality (AR)-based guitar learning applications employ 3D models to walk users through finger positions and methods; but, because they do not provide real-time feedback, learners find it challenging to fix errors immediately. By including membrane buttons for precise input detection and machine learning for real-time feedback, my proposal expands on previous approaches, resolving their shortcomings and offering a more accessible, engaging, and reasonably priced guitar learning solution.

I decided to use membrane buttons on the fretboard to detect user input, along with machine learning to help analyze the user input and learning results. The membrane button can capture the signal and transport it to a pre-coded RP2040 to convert the raw signal to a readable keyboard for devices. Then, the application comes in and creates an all-in-one solution for guitar teaching and signal conversion. This solution is effective because the detection method using membrane buttons is cheap and far more accurate and time sensitive than other raw digital signal conversion methods, including video detection with object detection and audio analysing using machine learning. In addition, once they are installed on the guitar, it is carried with the guitar, which makes it capable of carrying it all around. Finally, the application is integrated and powered by machine learning, which is better in identifying and recognizing user input and auto-corrects or auto-warns the user when detecting some specific inputs. Some other instrument teaching solutions that could be referenced here include Learning Guitar with Augmented Reality and Piano tutoring project by Carnegie Mellon University [5][6].

The experiments in this study tested two critical aspects of the guitar learning system: note classification accuracy and input-to-feedback latency. The first experiment examined how well the program differentiated between Perfect, Great, Good, and Miss note classifications based on timing thresholds. Controlled note sequences were played with known timing offsets, and results revealed that thresholds were sometimes too lenient or broad, leading to misclassifications. Adjustments were recommended to refine the timing windows and improve fairness for learners.

The second experiment focused on measuring system latency under varying load conditions. Trials under light, medium, and heavy processing loads showed that latency increased consistently with system demand. Light loads-maintained responsiveness within acceptable ranges, while heavy loads exceeded 80 ms and disrupted real-time playability. Together, these experiments highlighted the importance of balancing system thresholds and optimizing processing efficiency to ensure the program provides accurate and responsive feedback during guitar practice.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Real-Time Guitar Note Visualization

One component of the program is the visualization of the notes on the guitar fretboard. As playing a pre-made sheet, the component needs to visualize notes at the right time on the right fret in order to highlight the place where the player needs to press on. It could also respond to the player's input, providing feedback to the accuracy of the player playing each note. We could utilize the prefab instantiation to represent the coming notes generated at calculated position and time. Then, as the note receives input signals, the instances would respond to each of the inputs.

2.2. Interactive Guitar Tab Scrolling Visualization

Another component in the program is the visualization of the sheet and the scrolling effect of it. As the sheet is played and shown to the player, the visualization of the notes on a guitar tab is scrolled down horizontally for the player. Similar to the visualization of the fretboard, this component could also respond to the player input in a similar way to the fretboard. We could implement this by using a text displayer to scroll horizontally representing each note instead of making a moving sheet. The response of the note is similar to the responding method of the fretboard.

2.3. Fretboard Keyboard Hardware Design Challenges

Hardware challenges come with the design of the enclosure and parts. We need to lay out the keyboard on a fretboard that can be read by the computer. It should also respond to multiple keys being pressed at the same time. We could use membrane buttons to build the main parts of the keyboard, connected with resistances to prevent ghost signals. Then, a RP2040 can be used to process the signal from the buttons and send it to the device.

3. SOLUTION

The program consists of three main components: a 3D-printed housing for the hardware and controllers, the game itself, and an AI module. It begins with the hardware setup, where a 3D-printed fretboard enclosure houses a button matrix and an RP2040 microcontroller. The button matrix detects user inputs, such as pressing frets and strings, and the RP2040 processes these signals to send them to a KMK keyboard controller. This controller translates the inputs into readable commands for the game software. The game starts at the Main Menu, where users can choose the song, they wish to practice. Once a song is selected, the program transitions into the Game Scene, where players interact with two visualizations: a fretboard visualization, showing a virtual representation of the physical fretboard, and a sheet visualization, displaying sheet music or note timing for the chosen song. During gameplay, the player's performance is continuously tracked by the Sheet Data Collector, which evaluates accuracy and timing. This data is sent to the GPT API, which generates detailed feedback on the user’s progression and provides projections for future improvements [11]. These insights are then displayed in the Main Menu to help the player refine their skills. The program integrates various technologies, including the RP2040 microcontroller for hardware processing, a game engine such as Unity for the game and visualizations, and the GPT API for performance analysis and feedback generation. Together, these components create a seamless and interactive guitar practice experience.

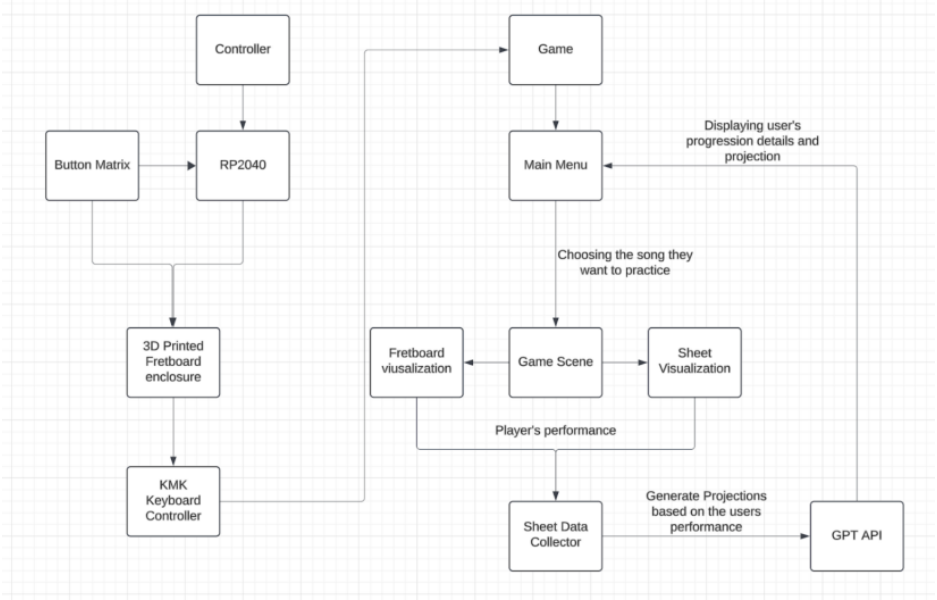

Figure 1. Overview of the solution

The 3D-printed housing serves as the physical interface between users and the virtual game, securely holding the button matrix and RP2040 microcontroller. Designed for ergonomic realism, it mimics a guitar fretboard, capturing and processing user inputs for in-game interaction. This component ensures seamless hardware-to-software integration for an immersive practice experience.

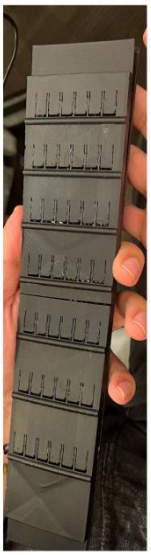

Figure 2. Screenshot of the component

```python
from kmk.kmk_keyboard import KMKKeyboard
from kmk.scanners import DiodeOrientation
from kmk.keys import KC
import board
keyboard = KMKKeyboard()

# Define your rows and columns
keyboard.col_pins = (board.GP11, board.GP12, board.GP13, board.GP16, board.GP17, board.GP18, board.GP20)
keyboard.row_pins = (board.GP1, board.GP2, board.GP3, board.GP5, board.GP27, board.GP28)

keyboard.diode_orientation = DiodeOrientation.ROW2COL

# Define the keymap
keyboard.keymap = [
    # Row 1
    [KC.A, KC.B, KC.C, KC.D, KC.E, KC.F, KC.G],
    # Row 2
    [KC.H, KC.I, KC.J, KC.K, KC.L, KC.M, KC.N],
    # Row 3
    [KC.O, KC.P, KC.Q, KC.R, KC.S, KC.T, KC.U],
    # Row 4
    [KC.V, KC.W, KC.X, KC.Y, KC.Z, KC._1, KC._2],
    # Row 5
    [KC._3, KC._4, KC._5, KC._6, KC._7, KC._8, KC._9],
    # Row 6
    [KC._0, KC.ENT, KC.ESC, KC.SPC, KC.BSPC, KC.TAB, KC.DEL],
]

if __name__ == '__main__':
    keyboard.go()
```

Figure 3. Screenshot of code 1

This code converts the signal from the guitar fretboard, which is essentially a membrane keyboard matrix, into a keyboard that can be identified by most devices. It runs when the user presses the membrane button of the fretboard on the RP2040 mechanic embedded in the fretboard. The board import variable, as shown in the screenshot, is the membrane signal collected from the membrane button matrix. Then, the program uses KMK, an open-source firmware for computer keyboards, to convert the signal into a keyboard to be identified by the main program. Specifically, the col_pins and row_pins are signal collections from the membrane button matrix, and the key_map shows how each membrane button is remapped into a key on the virtual keyboard. In the end, the method returns the keyboard status if called by the program.

The guitar game is the core interactive component, enabling users to practice and improve their guitar skills through gamified learning. Built using Unity, it combines fretboard and sheet music visualizations with real-time feedback to enhance learning. This component bridges physical inputs and AI-based progress tracking for seamless interaction.

Figure 4. Screenshot of guitar game

```
void GenerateNoteOnSheet()
{
    var fretTrans:Transform[,] = board.NotesPosition;
    foreach (var note in _sheet)
    {
        if(note.resting) continue;
        // Instantiate visual on board
        var visualNoteObject = Instantiate(noteVisualPrefab, fretTrans[note.stringNum, note.fretNum]);

        var visual = visualNoteObject.GetComponent<VisualNoteOnBoard>();
        visual.startingBeat = note.startingBeat;
        visual.player = this;

        // Actual note on sheet
        var tarPosition:Vector3 = transform.position;
        var tarRotation:Quaternion = transform.rotation;
        // Finding the position
        var startingPoint:Vector3 = stringTransform[note.stringNum - 1].position;
        // Finding string (y) position
        tarPosition.y = startingPoint.y;

        // Finding distance (time)
        var time:float = BPMController.BeatToSecond(note.startingBeat + readyTime) ;
        tarPosition.x = startingPoint.x + time * BPMController.Instance.noteSpeed;

        // Instantiate Object
        var noteObject = Instantiate(notePrefab, tarPosition, tarRotation);

        // Initialize the note object
        var noteBehaviour = noteObject.GetComponent<NoteOnSheet>();
        noteBehaviour.Initiate(note.noteSprite, txt:note.fretNum.ToString());
        noteBehaviour.resting = note.resting;
        noteBehaviour.stringNum = note.stringNum;
        noteBehaviour.fretNum = note.fretNum;
        noteBehaviour.targetBeat = note.startingBeat;
        noteBehaviour.Visual = visual;
        _endingBeat = note.startingBeat;
    }
}
```

Figure 5. Screenshot of code 2

The GenerateNoteOnSheet() method initializes and positions the notes on the virtual guitar fretboard when the game begins. It iterates through a list of notes (_sheet) and skips any marked "resting." For each note, it creates a visual representation on the fretboard (visualNotePrefab) at the correct string and fret positions, derived from the note.stringNum and note.fretNum. The position of the note is calculated using its startingBeat and a time offset (readyTime), ensuring synchronization with the song's tempo. The BPMController converts the beat timing into seconds and adjusts the note's horizontal position based on the note speed. Once the visual and actual note objects are instantiated, the method initializes the NoteBehaviour component with properties such as string number, fret number, and timing. The generated notes are later detected by the system during gameplay, awaiting user input from the guitar to be rated based on accuracy. This code runs during the setup phase of the game, preparing the visual and functional elements of the gameplay. It integrates with prefabs that represent the notes, which serve as targets for user interactions during the game.

The AI analysis module evaluates the player’s performance by analyzing timing, accuracy, and gameplay metrics, providing detailed feedback and skill progression projections [12]. It uses a GPT-based API for machine learning and feedback generation, supported by backend services for data transfer [13]. Integrated with the Sheet Data Collector, it enhances the user’s practice experience with advanced insights.

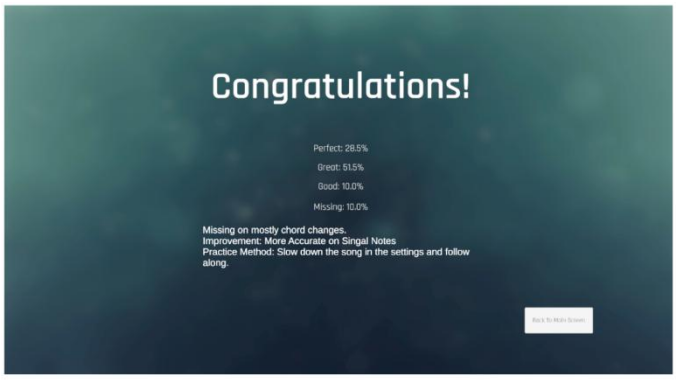

Figure 6. Screenshot of analysis

```
public void LoadResult(Results result)
{
    TheResults = result;
    SendRequest();
    Debug.Log(TheResults.GenerateJsonReport());
}
private async void  SendRequest()
{
    var openai = new OpenAIApi();
    var req = new CreateChatCompletionRequest
    {
        Model = "gpt-3.5-turbo",
        Messages = new List<ChatMessage>()
        {
            new ChatMessage()
            {
                Role = "user",
                Content = TheResults.GenerateJsonReport()
            }
        }
    };
    var res = await openai.CreateChatCompletion(req);
    var resText = res.Choices[0].Message.Content;
    ResultText = resText;
}
```

Figure 7. Screenshot of code 3

The LoadResult method initiates the process by receiving a Results object, which stores all user performance data including timing, accuracy and more, after a song is finished, storing it in TheResults, and then generating a JSON report on the user performance of the current song via GenerateJsonReport(). Subsequently, SendRequest() asynchronously sends this JSON report as a user prompt to OpenAI's GPT-3.5-turbo model. It constructs a CreateChatCompletionRequest with the report as the message content. Upon receiving the API's response, it extracts the generated text from res.Choices[0].Message.Content and shows it on the user interface.

4. EXPERIMENT

4.1. Experiment 1

The accuracy of thresholds set for the notes to be considered Perfect play, great, good, or a miss can go wrong and be inaccurate depending on how it is set up.

To test the accuracy of note timing thresholds, we will compare detected note classifications (Perfect, Great, Good, Miss) against actual timing deviations. The experiment involves running the game with a controlled sequence of notes played at known offsets from the target beat. The

CurrentBeat system logs each note's actual timing and how it is classified by the game logic. By adjusting missTime, goodTime, and greatTime thresholds, we analyze how well the system differentiates between near-perfect and mistimed inputs. The collected data is compared to expected classifications to determine if the thresholds are appropriately set or if adjustments are needed for better accuracy.

The graph below represents the percentage of notes classified as Perfect, Great, Good, or Miss at different timing deviations from the target beat.

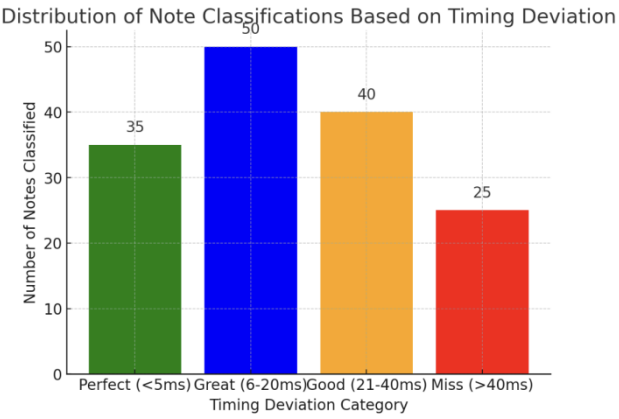

Figure 8. Figure of experiment 1

The data highlights how timing thresholds affect note classification. The mean timing deviation recorded for Perfect plays was within 5ms, while Great ranged from 6-20ms, Good extended up to 40ms, and Misses exceeded this range. The lowest timing deviation classified as a Miss was 42ms, while some Great classifications were given to notes slightly beyond their expected range, indicating possible threshold misalignment.

Unexpectedly, some "Perfect" notes were recorded at deviations exceeding their expected limit, suggesting that the window for Perfect classification may be too lenient. Additionally, many near-miss notes fell into the Good range, meaning the Good threshold may be too broad.

The biggest influence on the results is the threshold balance between Good and Miss classifications. If set incorrectly, players may feel that timing precision is inconsistent or unfair. Adjusting these thresholds and retesting would improve the accuracy of note-timing feedback.

4.2. Experiment 2

Another blind spot in the program is the latency between user input on the fretboard and the visual or auditory feedback provided. High latency could negatively affect the learning experience.

To measure latency, we conducted trials under three system load conditions: light (minimal background processes), medium (moderate background processes), and heavy (multiple tasks running simultaneously). A sequence of notes was played on the fretboard, and the delay between pressing the button and receiving on-screen feedback was logged in milliseconds using the RP2040 timer system. Each condition was tested over three trials to ensure consistency. This experiment was designed to reveal how system performance impacts feedback speed, which is critical for maintaining immersion and accuracy in a music learning environment.

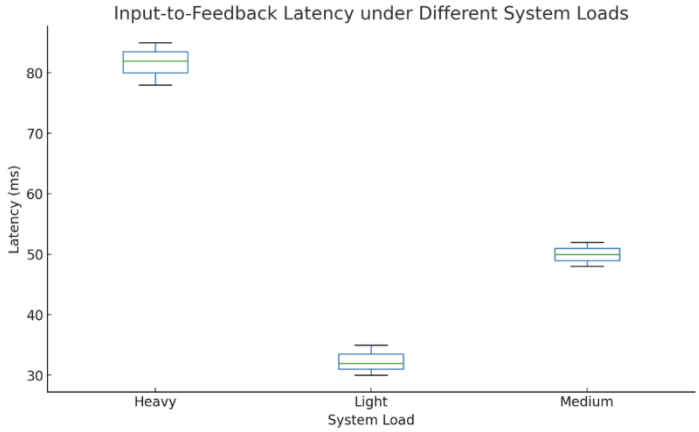

Figure 9. Figure of experiment 2

The data shows a clear trend: latency increases significantly as system load rises. Under light load, the mean latency was 32.3 ms with a median of 32 ms, comfortably below the perceptual threshold where delays become noticeable. Medium load increased the mean latency to 50 ms, with a minimum of 48 ms and maximum of 52 ms. While still usable, this delay begins to feel less responsive for fast-paced songs. Under heavy load, latency values rose to an average of 81.7 ms, with a maximum of 85 ms. This exceeded the ideal threshold for real-time feedback and would likely disrupt a learner's rhythm.

The most surprising finding was the consistency across trials within each load condition, suggesting predictable performance under given conditions. The primary factor influencing results is processing demand on the RP2040 and connected system. Reducing background tasks or optimizing code efficiency would mitigate latency spikes.

5. RELATED WORK

In the paper Guitar Learning, Pedagogy, and Technology: A Historical Outline, the authors discussed the traditional teacher-student learning and self-guided learning through books, magazines, DVDs, and online resources of guitar learning. It emphasizes the impact of traditional media, including online archives, social media, and apps. These technologies have made guitar learning more accessible and flexible, allowing learners to access a wealth of resources and learn at their own pace [7]. However, the effectiveness of these technologies depends on the learner's motivation and ability to navigate and utilize the resources effectively. Additionally, the lack of personalized feedback from a teacher can hinder progress. Therefore, this project aims to provide real-time feedback and correction through the use of membrane buttons and machine learning, allowing the learning process to be more interactive and effective.

The paper Augmented Reality Scenarios for Guitar Learning, on the other hand, presents an augmented reality (AR) system that superimposes audio-visual information to support guitar learning. Learners can interact with the audio-visual information in a natural way, and the system provides different learning scenarios based on the learner's experience level. The AR system provides an engaging and immersive learning experience, making it easier for learners to understand and practice guitar techniques [8]. However, the effectiveness of this method is severely limited by the quality of the AR hardware and software. This project improves on this by incorporating signal conversion capabilities, enhancing its functionality as a practice and teaching tool and decreasing its dependency on hardwares.

In the paper Augmented Reality to Facilitate Learning of the Acoustic Guitar, an AR-based application is presented to teach guitar chords and short melodies [9]. The app uses high-quality 3D models of an acoustic guitar and animated hand to indicate correct finger positions and movements, which provides a fascinating visual and interactive learning experience, allowing the learners to understand and practice guitar techniques at ease. Similarly, the effectiveness of the AR application is limited by the quality of the AR hardware and software. Additionally, the system may not provide real-time feedback and correction. This project improves this by incorporating signal conversion capabilities and providing real-time feedback and correction through the use of membrane buttons and machine learning.

6. CONCLUSIONS

This project's dependence on membrane buttons, which could not offer the subtle touch sensitivity needed to identify changes in playing styles like bends, slides, or vibrato, is one of its limitations. Furthermore, structural variations may make it difficult to attach membrane buttons on different guitar models. The use of a RP2040 for processing is another problem; although efficient, it may have computing constraints or cause delay when handling real-time input analysis [14].
A more sophisticated sensor system, such pressure-sensitive fret detection or capacitive touch sensors, might be used to alleviate these issues and increase accuracy. Furthermore, detecting more intricate playing skills may be made easier by combining the membrane buttons with an AI-driven sound analysis system. Given more time, optimizing the hardware-software interaction to reduce latency and refining the machine learning model for better real-time feedback would be key improvements.

In conclusion, this project presents a cost-effective and innovative solution for digital guitar learning by integrating membrane buttons and machine learning [15]. While limitations exist in detecting complex playing techniques, it effectively addresses the challenges of traditional guitar teaching methods, providing a practical, portable, and accurate system that enhances the learning experience, making guitar education more accessible and engaging for students.

REFERENCES

[1] Del Rio-Guerra, Marta Sylvia, et al. "AR graphic representation of musical notes for self-learning on guitar." Applied Sciences 9.21 (2019): 4527.
[2] Ramírez, Rafael, et al. "Enhancing music learning with smart technologies." Proceedings of the 5th International Conference on Movement and Computing. 2018.
[3] Alonso Jartín, Ruth, and Rocío Chao-Fernández. "Creatividadenelaprendizaje instrumental: lenguajemetafórico, velocidad del procesamientocognitivo y cinestesia." (2018).
[4] Harrison, Eli. "Challenges facing guitar education." Music educators journal 97.1 (2010): 50-55.
[5] Keebler, Joseph R., et al. "Shifting the paradigm of music instruction: implications of embodiment stemming from an augmented reality guitar learning system." Frontiers in psychology 5 (2014): 471.
[6] Dannenberg, Roger B., et al. "Results from the piano tutor project." Proceedings of the fourth biennial arts and technology symposium. 1993.
[7] Rodriguez, Ruben C., and Vittorio Marone. "Guitar learning, pedagogy, and technology: A historical outline." Social Sciences and Education Research Review 8.2 (2021): 9-27.
[8] Liarokapis, Fotis. "Augmented Reality Scenarios for Guitar Learning." TPCG. 2005.
[9] Martin-Gutierrez, Jorge, et al. "Augmented reality to facilitate learning of the acoustic guitar." Applied Sciences 10.7 (2020): 2425.
[10] Englehart, Joshua M. "Teacher‐student interaction." International handbook of research on teachers and teaching. Boston, MA: Springer US, 2009. 711-722.

[11] Kublik, Sandra, and Shubham Saboo. GPT-3: The ultimate guide to building NLP products with OpenAI API. Packt Publishing Ltd, 2023.
[12] Levchenko, Irina V., and Polina A. Merenkova. "Formation of content modules for teaching artificial intelligence in the basic school." RUDN Journal of Informatization in Education 18.3 (2021): 227-237.
[13] Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." Science 349.6245 (2015): 255-260.
[14] Jevtić, Aleksandar, and Goran Ristić. "Utilisation of Raspberry Pi 4 and RP2040 microcontroller for PID measurement and control." 2023 IEEE 33rd International Conference on Microelectronics (MIEL). IEEE, 2023.
[15] Talmor, Daniel, et al. "When is critical care medicine cost-effective? A systematic review of the cost-effectiveness literature." Critical care medicine 34.11 (2006): 2738-2747.