Federal University of Technology - Paraná - Brazil - UTFPR
Academic Department of Electronics – DAELN
Academic Department of Informatics – DAINF
Computer Engineering Course
Integration Workshop 3 (ELEX22) – S71 – 2025/1

Technical report

Melody: A musicalization tool for music teachers

Augusto Rosa – augusto.oliveira.rosa14@gmail.com
Bruno Emanuel Zenatti – brunozenatti@alunos.utfpr.edu.br
Fernando H. R. Caetano – caetanof@alunos.utfpr.edu.br
Nicolas Riuichi Oda – oda@alunos.utfpr.edu.br
Rafael E. Ishikawa Rasoto – rafaelrasoto@alunos.utfpr.edu.br
Rogério Slabiski Melo – rogerioslabiski@alunos.utfpr.edu.br

June 2025

Abstract

Melody is an embedded system designed to assist music teachers in introducing music theory to children in an interactive and engaging way, and also to teach musical theory to blind people. The project addresses the growing difficulty younger generations face when learning instruments due to frustration with steep learning curves and theoretical content. The system is based on a wooden board painted with a music staff, where children can place physical note markers and also an accessibility module to serve as an interface to the blind student. A camera and Raspberry Pi process these inputs, allowing the system to interpret and play the corresponding melody. It operates in three distinct modes: free play with looped melodies, guided placement based on preloaded songs via visual prompts, and auditory guided interaction where children identify and place notes based on pitch recognition. The project aims to bridge the gap between playful learning and musical education, offering a novel, inclusive, and teacher-endorsed approach to early music literacy.

1 Introduction

In recent years, educators have observed a growing disinterest among younger generations[2] in learning musical instruments and theory. Traditional instruments such as the piano or violin often require a prolonged and repetitive learning process, which includes memorization, fine motor skill development, and reading musical sheets. For many children, especially in a fast-paced digital age [1], the initial stages of learning music feel tedious and unrewarding. This leads to frustration and abandonment before they reach a stage where music becomes expressive and enjoyable. The decline in perseverance is compounded by a broader cultural shift that values instant gratification, making it increasingly

difficult for music teachers to maintain students' engagement through conventional pedagogical approaches.

Moreover, learning music theory presents even greater challenges for visually impaired individuals. Standard music notation is a visually dense and symbol-heavy system, which inherently excludes those who cannot rely on visual cues. Although Braille music notation exists, it has a steep learning curve, is rarely taught, and is often disconnected from mainstream teaching environments. As a result, blind students frequently lack access to inclusive tools that allow them to learn music theory and engage with musical sheets in a dynamic and intuitive way. This gap limits their ability to participate in formal musical education and often discourages them from pursuing music altogether.

To address the growing lack of engagement among children in learning music theory and the significant barriers faced by blind people in accessing traditional musical education, the Melody project proposes an inclusive and interactive embedded system. This system integrates visual, auditory, and tactile elements to create a dynamic learning experience centered around a physical board painted with a musical staff. Children will be able to place physical musical notes on the board, which are then detected by a camera and interpreted by a Raspberry Pi. The corresponding sound is played through a speaker, allowing users to both visualize and hear the melody they've constructed. In this way, Melody simplifies the abstract nature of sheet music into a playful, tangible, and immediate form of feedback.

The system also includes multiple modes to accommodate different learning approaches: a free-play loop mode that plays user-created melodies continuously, a guided mode that displays a pre-set melody for replication, and an auditory recognition mode where children place notes by identifying their pitch. These features aim to reduce frustration in the early learning stages while promoting experimentation and self-guided exploration. In addition, the system considers accessibility by integrating vibration motors and LED feedback, which can assist blind students in identifying notes and positions without relying solely on vision.

2 Project specification

2.1 Project overview

The system consists of a board with a blank musical sheet, on which students place physical musical notes. A camera positioned above the board captures the placement of the notes, and a Raspberry Pi processes the input to identify their position and translate it into sound, playing the corresponding melody through a speaker, as illustrated in 1. The system offers to the music teacher - through the web application - multiple modes of operation, including a free-play loop

mode and guided tasks based on preloaded songs[1], both visual and auditory. Additional features such as LED indicators and vibration feedback aim to support visually impaired users. The usability of the project is represented in 2.

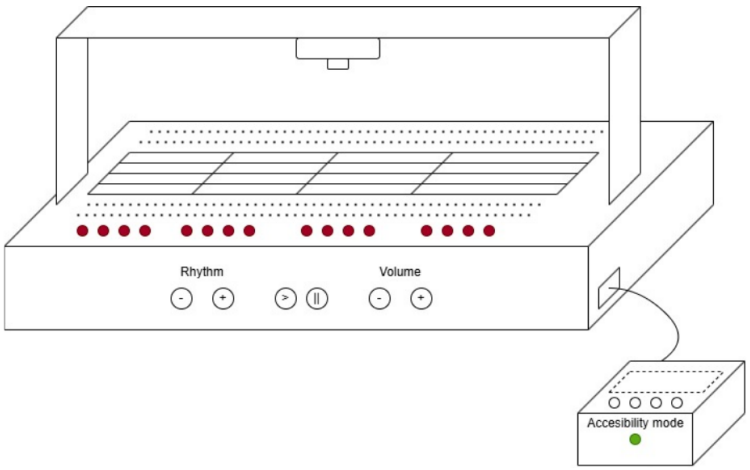

Figure 1: Melody concept

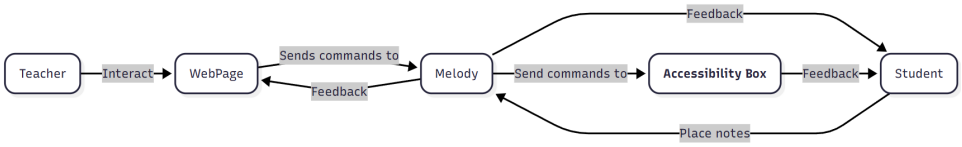

Figure 2: Melody interactions diagram

2.1.1 Operation mode: Creation

In Creation Mode, the Melody system allows children to explore musical composition by placing physical note pieces freely on the board's staff lines. As each note is positioned, the system detects its location using the camera and interprets the corresponding pitch, accidental and *tempo* based on its placement on the staff and also its color. The identified notes are then played sequentially in a continuous loop, creating a dynamic auditory feedback experience.

2.1.2 Operation mode: Hear the note

In Hear the Note Mode, the Melody system challenges students to recognize musical notes using only auditory cues. The system plays a single note through the speaker, and the student must identify it by pitch and place the corresponding physical note on the correct position of the staff. Once the note is placed, the

[1] Canon in D - Johann Pachelbel; Ode to joy - Johann S. Bach; Twinkle twinkle little star - Mozart.

system uses the camera to verify the position and color, determining whether the selected note, accidental and *tempo* match the one that was played. This mode reinforces ear training and helps develop pitch recognition skills, promoting a deeper internalization of musical concepts. It also offers an inclusive and engaging learning experience, particularly beneficial for students who rely more heavily on auditory perception, such as those with visual impairments.

2.1.3 Operation mode: Identify the note

In Identify the Note Mode, the system is used under the guidance of the teacher, who verbally instructs the student to place specific notes on the staff, including the pitch, accidental and the desired tempo (such as whole, half, or quarter notes). From that, the system works exactly like the Hear the Note mode, from the step when the student is supposed to place the note in the correct position.

2.2 Requirements

This section outlines the set of requirements defined by the Melody project team during the planning phase based on the resources at the team's disposal and the functionalities it should provide. They were categorized into functional and non-functional types, and further grouped according to the system's subsystems: software, mechanical, and hardware. Each requirement was designed to ensure the system meets its pedagogical, technical, and accessibility objectives.

The functional requirements (FRs) describe what the system must do, including image recognition of musical notes, continuous playback of melodies, and student guidance through lights and sound across different operating modes. The non-functional requirements (NFRs) specify constraints such as development language (Python and C), physical dimensions of components, and system responsiveness. In addition, a set of out-of-scope (Anti-requirements) were defined to clarify the project boundaries, ensuring a focused and feasible implementation. All requirements were evaluated for implementation risks, including whether the feature could be removed if team members were lost, and tracked for completion throughout the development process. During the project planning, the team had to evaluate possible risks and problems that could occur during the development, therefore, some requirements were marked as optional[2].

2.2.1 Mechanical requirements

The mechanical requirements of the Melody system define the physical characteristics, layout, and components necessary for proper user interaction, structural integrity, and accessibility. These requirements include the dimensions of the board, placement of note holes, integration of LEDs and accessibility modules, and the specification of 3D-printed components to support both visual and

[2]The optional requirements were marked with a *

tactile interaction with the system. The mechanical structure was planned to ensure usability by both sighted and visually impaired users. Table 1 lists all the initial requirements. Some of them, such as the measurements, were later modified to improve the user experience and facilitate project development.

Table 1: Mechanical Functional and Non-Functional Requirements

ID	Description
NFR 1	The system must have 305 drilled holes for positioning the notes on the musical sheet.
NFR 1.1	16 columns of holes for 4 measures with 4 beats each, 3 cm between them.
NFR 1.2	19 lines of holes for note pitch, spaced 0.03 m apart.
FR 1.3*	The system must have 1 drilled hole for the musical clef.
NFR 2	The system must have one bicolor LED (green/red) for each column (16 LEDs total).
FR 3	The system must have a board with the musical sheet on it.
FR 4	The system must allow the student to place musical notes on the drilled holes.
FR 5	The system must have an accessibility module for blind people.
FR 5.1	The accessibility module must connect to the main board by a 4-wire cable.
FR 5.2	The accessibility module must have a tactile feedback pin for each measure (4 total).
FR 5.3	The accessibility module must have a tactile feedback vibrating interface.
NFR 6	The board must have support to hold the webcam and light sources.
NFR 6.1	The support must be 0.90 m high.
NFR 7	The board must be 0.90 m wide, 0.90 m long, and 0.10 m high.
NFR 8	The system must have 3D-printed pieces for whole, half, and quarter notes.
NFR 8.1	Each note must have versions for sharp, flat, and natural accidentals.
NFR 9*	The system must have 3D-printed pieces for treble and bass clefs.

Continued on next page

Table 1 – continued from previous page

ID	Description
NFR 10*	The system must operate reliably under standard indoor lighting.
FR 11	The 3D-printed pieces must be colored according to the Notes Table.
NFR 12	The system must have pieces printed in the quantities listed in the Notes Table.
NFR 13	Half and whole notes must have raised areas for blind users.
NFR 14	Notes must not exceed 3 cm in width or length.
NFR 15	The accessibility module must be 0.10 m in length, width, and height.

2.2.2 Hardware requirements

The hardware requirements of the Melody system define all the physical and electronic components necessary for capturing input, processing data, and providing feedback to users. These requirements ensure the integration of cameras, LEDs, speakers, and microcontrollers in a reliable and responsive architecture. Special attention was given to local processing capabilities, user interaction through buttons and audio, and tactile feedback mechanisms to support accessibility. The hardware design also guarantees that all subsystems operate independently of cloud services, making the system self-contained and suitable for classroom environments. Additionally, the hardware-related requirements are detailed in Table 2.

Table 2: Hardware Functional and Non-Functional Requirements

ID	Description
NFR 1	The system must have a webcam for capturing images.
NFR 2	The system must have light sources for consistent image capture.
NFR 3	The system must have a speaker to output audio with 2.5 W.
FR 4	The system must allow control of volume (as a percentage of total power), tempo (60, 90, or 120 BPM), and play/pause using push buttons.
NFR 5	The system must perform all image processing locally on the Raspberry Pi 4 B.

Continued on next page

Table 2 – continued from previous page

ID	Description
NFR 6	The system must have one 5 V 2 A power supply for the Raspberry Pi and a second identical power supply for the LEDs, motors, and ESP32.
NFR 7	The ESP32 microcontroller must run firmware developed in C using the FreeRTOS framework.
NFR 8	The ESP32 must communicate with the Raspberry Pi via UART.
NFR 9	The ESP32 must control four SG90 servo motors that actuate the tactile feedback pins on the accessibility module (each pin with 0.10 m dimensions).
NFR 10	The ESP32 must control the vibration motor that actuates the vibrating interface.
NFR 11	The ESP32 must control the LEDs via two 1-to-16 74154 demultiplexers.

2.2.3 Software requirements

The software requirements define the functional behavior and technical implementation constraints of the Melody system's logic and interface. These requirements cover user interaction through a web-based interface, control over system modes and settings, note recognition through image processing, and the execution of each learning mode: Creation, Identify, and Hear the Note. Additionally, non-functional requirements specify implementation details such as programming languages, frameworks, and deployment platforms. Together, these requirements guide the development of a reliable, responsive, and user-friendly learning tool that supports music education in both visual and auditory contexts. All requirements related to the software side of the project are listed in Table 3.

Table 3: Software Functional and Non-Functional Requirements

ID	Description
FR 1	The system must have a web interface for the teacher.
NFR 1.1	The web interface must have a home page.
NFR 1.2	The system must start in stand-by mode.
FR 1.3	The teacher can choose the mode from the home page.
NFR 1.4	The system must have a dedicated page for each mode.

Continued on next page

Table 3 – continued from previous page

ID	Description
FR 1.5	The teacher must be able to control volume (percentage), play/pause, and enable/disable accessibility mode.
FR 1.6	The Creation mode page must allow tempo control at 60, 90, or 120 BPM.
FR 1.7	The Identify and Hear the Note pages must ask for a melody to be played.
NFR 1.8	The web interface must be implemented using JavaScript, HTML, and CSS.
FR 2	The system must use image recognition to identify and classify musical notes.
FR 2.1	The system must capture images of the board using the webcam.
FR 2.2	The system must classify note duration based on color into whole, half, or quarter notes.
FR 2.3	The system must classify pitch based on vertical position across 19 values.
FR 2.4	The system must classify note position in measure based on horizontal placement across 16 values.
FR 2.5	The system must classify the clef (treble or bass) placed on the board.
FR 3*	The Creation mode must continuously play the melody displayed on the board in a loop.
FR 3.1	In Creation mode, the system must light a green LED below the column of the note currently being played.
FR 4	The Identify and Hear the Note modes must guide the student in building the melody.
NFR 4.1	These modes must light a red LED to indicate where the student should place the note.
FR 4.2	The Hear the Note mode must replay the expected note every 3 seconds.
FR 4.3	The Identify mode must display the note name in the web interface for the teacher to dictate.
NFR 4.4	The red LED must remain on if an incorrect note is placed.

Continued on next page

Table 3 – continued from previous page

ID	Description
NFR 4.5	The red LED must turn green and move to the next note position if the placement is correct.
FR 4.6	After all correct placements, the melody must loop continuously.
NFR 4.7	The Identify and Hear the Note modes must each have 2 preloaded melodies.
NFR 5*	The software must be developed in Python and run on a Raspberry Pi 4.
NFR 6*	The Raspberry Pi 4 must communicate with external devices (computer or cellphone) via Wi-Fi.

2.2.4 Out of scope

To ensure a focused and feasible development process, the Melody project defined a clear set of out-of-scope requirements. These are features or functionalities that, while potentially relevant in broader applications, were deliberately excluded from the current version of the system due to time, complexity, or educational scope limitations. Identifying these boundaries helps to prevent scope creep and sets clear expectations for users and stakeholders regarding what the system will and will not deliver.

ID	Description
Anti 1	The system will not teach complete music theory, but will act as a support tool.
Anti 2	The system is not intended for children to use alone.
Anti 3	The system will not export sheet music or generate real-time scores.
Anti 4	The system will not include internet or cloud-based services.
Anti 5	The system will not support multiplayer or simultaneous multi-user interaction.
Anti 6	The system will not have a display.
Anti 7	The system will not play chords.

Table 4: Out of Scope Requirements

3 Project development

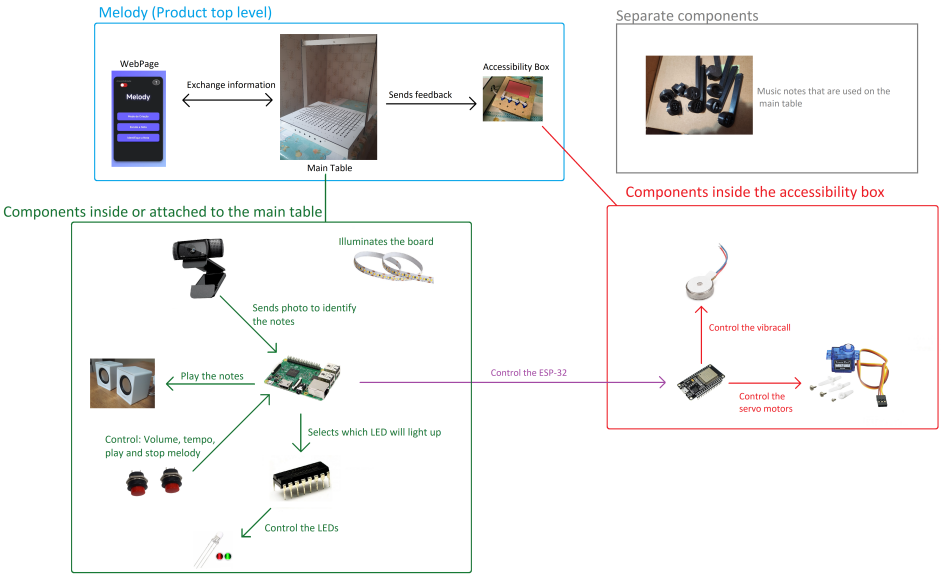

Figure 3: Melody development overview

3.1 Mechanical development

The mechanical structure of the Melody project was designed to be both functional and robust, providing a stable platform for all electronic components while ensuring an intuitive user experience. The design is centered around two main physical assemblies: the main board and an accessibility module.

The primary structure is a custom-designed main board with overall dimensions of 800 mm by 700 mm and a height of 68 mm, fabricated from a 6 mm thick sheet of Medium-Density Fiberboard (MDF), chosen for its durability and ease of machining. The board's dimensions and features were precisely defined in a software model before being cut using a laser cutter to ensure accuracy. This board serves as the interactive surface where users place physical note markers. It includes laser-cut holes positioned to represent the musical staff, along with openings for the system's physical interface, including six user-input buttons, an emergency reset button, and an access point for organized cable management. An integrated support structure holds the overhead camera at a variable height of 750 mm (± 20 mm).

The system's physical note markers, each measuring 30 mm x 96 mm, were 3D-printed with specific tactile features to make them distinguishable for visually impaired users. The differentiation is based on how accidentals (sharps and flats) are represented on notes of different durations. Quarter notes, which are represented as solid circles in standard notation, were printed as solid pieces; for

these, the sharp (♯) and flat (♭) symbols are indented into the surface (negative relief). Conversely, half and whole notes, which are hollow in standard notation, were printed with a hole through their center; for these notes, the accidental symbols are raised from the surface (positive relief). This design allows a user to first identify the note's duration by feeling if it is solid or hollow, and then determine its accidental by feeling for either a raised or an indented symbol.

In addition to the main board, the project includes a separate accessibility module designed to provide tactile feedback for visually impaired users. This module consists of a 100 mm x 100 mm x 110 mm MDF box with a wall thickness of 3 mm. The box, which was a pre-fabricated and glued unit, was subsequently modified using a laser cutter to create custom fittings for its components. These components include four servo motors and a vibration motor, which work in tandem to translate the system's visual LED feedback into tactile sensations, making the learning experience accessible to all users.

3.2 Hardware development

The system's hardware architecture is centered around a Raspberry Pi 4, which performs the main processing, and an ESP32 microcontroller, which manages the accessibility features. User interaction and feedback are handled through a webcam for note detection, a speaker for audio playback, and a matrix of 16 bicolor LEDs controlled by two demultiplexers for visual cues. The accessibility module, driven by the ESP32, consists of four servo motors for tactile pin feedback and a vibration motor. A UART serial connection enables communication between the Raspberry Pi and the ESP32. The entire system is powered by three independent 5V 2A power supplies to ensure stable operation for both the main board and peripheral components.

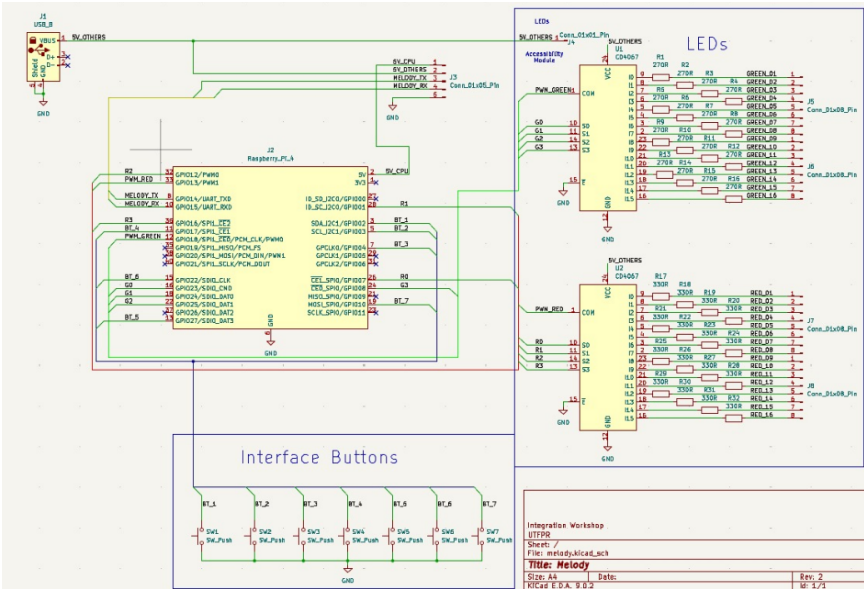

Figure 4: Melody hardware

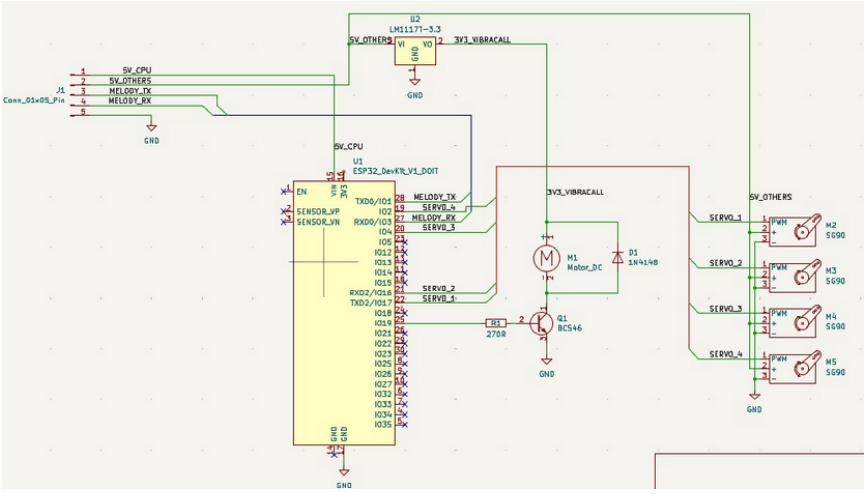

Figure 5: Accessibility box hardware

3.3 Firmware development

The project uses a dual-firmware approach to distribute tasks efficiently. The primary firmware is a Python application running on the Raspberry Pi, which orchestrates the entire system. It handles high-level tasks like image processing, sound synthesis, and the web server for the user interface.

A dedicated hardware module within this application manages all low-level interactions with the Raspberry Pi's GPIO pins. It directly controls the LED ma-

trix for visual feedback, offering functions to select specific columns and set their colors. This module also configures interrupt-driven event detection for the physical buttons (controlling volume, tempo, playback, and reset). When a button is pressed, a hardware interrupt triggers a callback function that immediately updates the system's state.

Additionally, the module manages the serial communication protocol used to transmit commands to the secondary firmware—a C program on the ESP32. This secondary firmware is dedicated to the accessibility module; it operates by listening for simple commands sent from the Raspberry Pi and executes real-time control of the servo and vibration motors.

3.4 Software development

The software architecture of Melody is designed around a client-server model, consisting of a Python backend that handles all core logic and a web-based frontend that serves as the user interface. This separation allows for a modular and scalable system. The backend is built using a modern web framework, which manages an asynchronous web server and communicates with the frontend in real-time via WebSockets.

3.4.1 System Architecture Overview

The software is divided into two main parts: the backend and the frontend.

- **Backend:** A Python application responsible for image processing, note detection, state management, sound generation, and hardware control. It runs on a Raspberry Pi and exposes an API for the frontend.
- **Frontend:** A web interface built with standard HTML, CSS, and JavaScript that runs in a browser. It allows users to interact with the system, select modes, and control parameters like volume and tempo.

Communication between these two parts is handled by a WebSocket connection, enabling the backend to push state updates to the frontend, ensuring the UI always reflects the current state of the application.

3.4.2 Backend Implementation

The backend is the core of the Melody project, containing all the logic for the system's functionality. It is organized into several key modules. In Figure 6, it is possible to see the state chart representing the overview of the system.

Web Server and API The backend web server, built using the FastAPI framework [4] and run by a Uvicorn ASGI server [3], handles two primary responsibilities: serving the static HTML, CSS, and JavaScript files for the user interface and

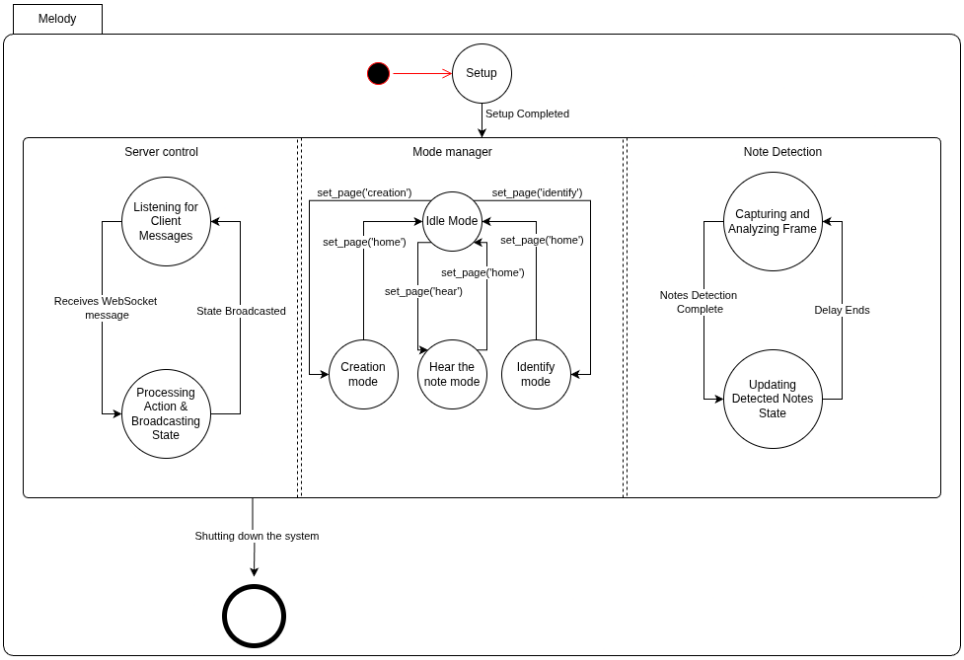

Figure 6: System's Statechart

managing communication. Communication is achieved via a WebSocket endpoint, which establishes a persistent, bidirectional channel between the server and the client. The API operates on a simple JSON-based messaging protocol. When a user interacts with the frontend (e.g., clicks a button), a message is sent to the server. The server receives this message, executes the corresponding function to update the system's central state, and then broadcasts the new state to all connected clients. This mechanism ensures that the UI always remains synchronized with the application's internal state.

State and Session Management It ensures that all components — from the user interface to the hardware LEDs — are synchronized and operate on the same information.

The core of this system is a centralized data structure that holds every important variable representing the application's current status. Whenever any of these values change, whether initiated by the user pressing a button on the web interface or a hardware button on the physical device, the central state is updated. Immediately after the update, this new state is broadcast over the WebSocket connection to all connected clients. The frontend interface listens for these broadcasts and instantly updates its components — for example, changing volume value on the UI or showing a new note — ensuring the teacher always sees a perfect reflection of the system's internal state.

Note Detection The note detection process is one of the most critical components of the software. To ensure the application remains responsive, this process runs in a dedicated thread. The detection pipeline involves several stages:

- **Image Calibration:** After startup, the camera feed is calibrated. This routine algorithmically detects the four corner holes on the board and applies a perspective warp transformation. This correction accounts for the camera's angle, producing a consistent, top-down view of the musical staff for reliable analysis.
- **Color Analysis:** The warped image is converted to the HSV color space to make color detection more robust. A set of color masks is then applied to isolate pixels corresponding to each of the colored note markers. To improve accuracy, morphological operations like dilation and erosion are used to reduce noise and fill small gaps in the detected color regions.
- **Note Parsing:** After identifying the predominant color in each grid position on the board, a parsing module translates this visual information into musical notation. It uses a predefined map where a combination of a color and its vertical position corresponds to a specific note pitch (e.g., "C4", "F5#") and duration, effectively converting the physical arrangement of notes into a digital melody.

Sound Generation The sound playing functionality programmatically synthesizes audio waveforms for each musical note based on its fundamental frequency and higher order harmonics. To create a more natural and pleasant tone, an ADSR (Attack, Decay, Sustain, Release) envelope is applied to the synthesized waveform. The final audio is then streamed to the system's speakers using a sound-playing library.

Application Logic and Modes The application's behavior is organized into a state machine, where each state is a distinct operational mode. A master controller, running in the main program loop, continuously checks the `page` variable from the central state to determine which mode should be active. Based on this variable, it executes the logic contained within the corresponding mode module. This design makes the system's logic modular and easy to manage.

Here is a more detailed breakdown of each mode's logic:

Idle Mode This is the default mode when the application is on the home screen. It is a passive state where the system is simply waiting for user input to navigate to one of the other functional modes. No active processing, like note detection or sound playback, occurs.

Creation Mode This mode allows students to freely compose music by placing physical notes on the board. The system continuously detects the note sequence and, when in the `playing` state, it plays each note with synchronized audio and LED indicators. If accessibility is enabled, the ESP32 receives the corresponding command. Playback follows the configured `tempo` and updates as users modify the notes.

The flowchart of the creation mode is shown in Figure 7.

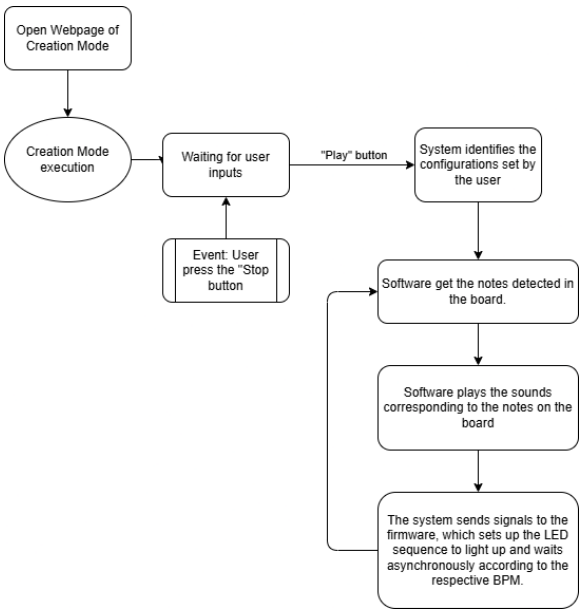

Figure 7: Creation Mode Flowchart

Identify the Note Mode This interactive game mode helps students practice note recognition. The system loads a predefined melody and displays each target note sequentially. It waits for the user to place a matching note on the board, using the detection module to evaluate input. Feedback is given through green or red LEDs, and, if accessibility is enabled, through membrane vibration. Upon a correct placement, the system advances to the next note in the melody.

The flowchart of the identify the note mode is shown in Figure 8.

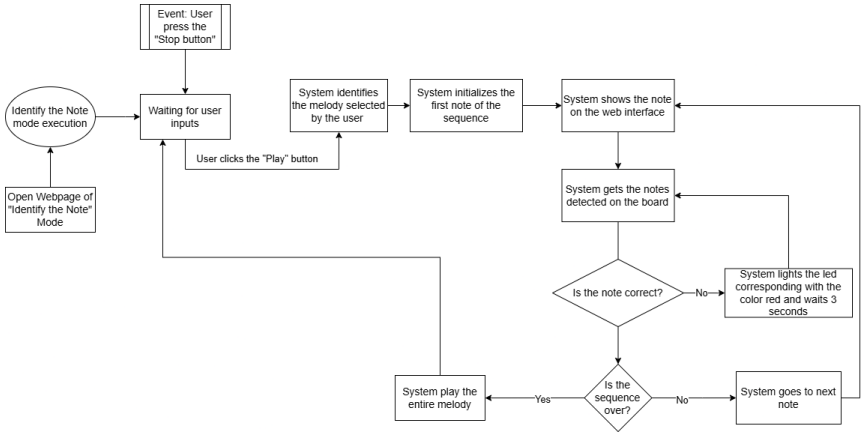

Figure 8: Identify the Note Mode Flowchart

Hear the Note Mode This ear-training game challenges students to recognize notes by sound alone. The system plays each note from a predefined melody using the sound generation module, without visual cues. The student must listen and place the corresponding physical note on the board. The system detects and evaluates the answer, providing feedback via LEDs and vibration (if accessibility is enabled). After a correct response, it advances to the next note in the sequence.

The flowchart of the hear the note mode is shown in Figure 9.

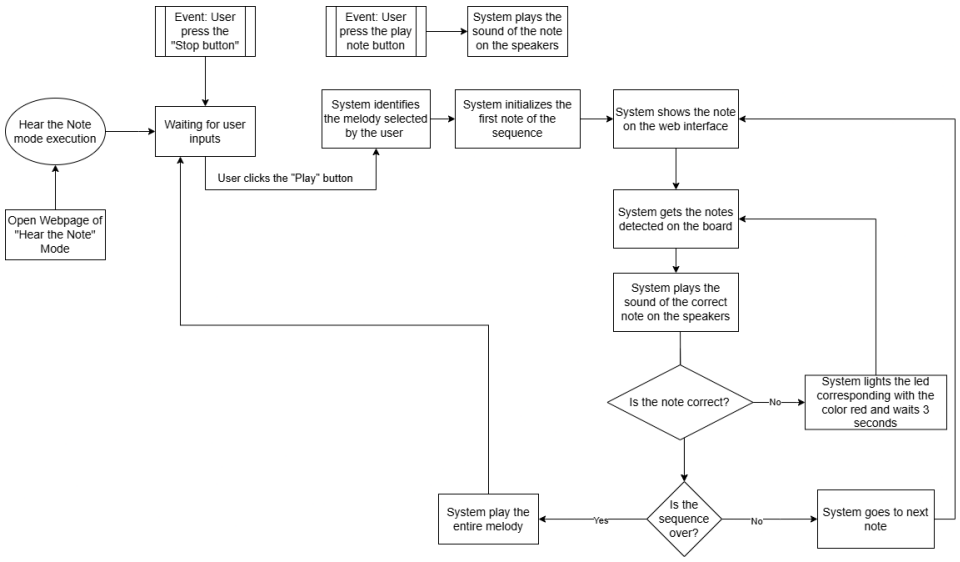

Figure 9: Hear the Note Mode Flowchart

3.4.3 Frontend Implementation

The frontend provides a simple and intuitive user interface for controlling the system. It is designed to be responsive and run in a web browser, allowing for easy access from various devices connected to the same network.

Communication Bridge At the core of the frontend is a JavaScript module that acts as a communication bridge. It establishes and maintains a WebSocket connection to the backend, enabling two-way communication. This bridge is responsible for two key processes:

- **Home Page:**
  - Navigation links to the three interactive modes.
  - A toggle switch for the accessibility mode.
  - A help button to instruct the teacher.
- **Creation Mode:**
  - Controls to start and stop playback.
  - Buttons for volume and tempo adjustment.
  - A navigation link to return to the home page.
- **Hear the Note Mode:**
  - A dropdown menu for melody selection.
  - A display area for the target note.
  - A control to play the sound of the current note.
  - Volume adjustment buttons.
  - A navigation link to return to the home page.
- **Identify the Note Mode:**
  - A dropdown menu for melody selection.
  - A display area for the target note.
  - Volume adjustment buttons.
  - A navigation link to return to the home page.

4 Results

4.1 Requirements completion

The Melody system successfully met all of the functional and non-functional requirements defined during the planning phase. Throughout development, the team prioritized user interaction, accessibility, and robust integration between hardware and software components. All three operation modes—Creation, Identify the Note, and Hear the Note—were fully implemented and validated in real-world testing scenarios.

In the software domain, the real-time communication between the frontend and backend through WebSockets ensured a fluid user experience. The backend effectively handled note recognition, sound generation, and system state management, while the frontend accurately reflected system status and allowed intuitive control of the music teacher over all modes and parameters.

On the hardware side, the LED matrix, buttons, and audio playback system responded as expected, with low-latency input recognition. Additionally, the accessibility module successfully converted visual cues into tactile feedback us-

Figure 10: Project assembled

ing servo motors and a vibration interface, enabling visually impaired users to interact with the system independently.

Mechanically, the board layout, drilled holes, and support for 3D-printed note pieces were precisely aligned with the musical staff, enabling reliable note detection and positioning. Both sighted and blind students were able to use the system in a classroom setting, achieving the pedagogical objectives of the project.

4.2 Budget

The initial budget estimation for the Melody project ranged from R$ 910,00 to R$ 1.195,00, considering all planned components, including electronics, mechanical parts, and materials for prototyping. The estimate included spare parts for critical components such as servo motors, ESP32, and the Raspberry Pi, aiming to ensure hardware redundancy and reduce project risks.

By the end of development, the total amount spent was R$ 1.133,11, which remained within the estimated range. This indicates effective financial planning and resource management by the team. The budget can be found in more details at the *blog* [5].

5 Conclusion

The Melody project successfully achieved its main objective of developing an educational tool to support music teachers in introducing music theory to children in an accessible and engaging way. Through the integration of hardware, software, and mechanical design, the system provides a playful and interactive experience that helps students understand the relationship between musical notation and sound. The inclusion of a dedicated accessibility module also

ensures that visually impaired students can participate equally in the learning process, reinforcing the inclusive nature of the solution.

From a technical standpoint, the project demonstrated robust implementation of real-time image recognition, audio synthesis, and responsive user interface design. The three operational modes —Free Play, Identify the Note, and Hear the Note — were developed and validated to meet pedagogical goals, allowing both guided and exploratory learning. The system's architecture, built around a Raspberry Pi with a Python backend and web-based frontend, offers a scalable and maintainable platform for future enhancements. Moreover, the team remained within the projected budget, with a final cost of R$ 1.133,11, below the upper limit of the estimated range. With all of the planned requirements fully implemented, Melody stands as a complete and viable prototype that could be extended into a classroom-ready product. Future iterations may include chord support, faster image recognition, better speakers or a more compact physical structure. As it stands, the system already presents meaningful pedagogical impact and aligns well with the goals of accessible and modern education.

References

[1] American Psychological Association. *Excessive screen time linked to behavior problems in children*. Accessed: 2025-06-25. June 2025. URL: https://www.apa.org/news/press/releases/2025/06/screen-time-problems-children.

[2] Association of Texas Professional Educators. *Generation Alpha: Educating the Most Digital Generation Yet*. Accessed: 2025-06-25. 2024. URL: https://www.atpe.org/News-Media/Magazine/ATPE-News-Summer-2024/Generation-Alpha.

[3] Uvicorn Project. *Uvicorn: The lightning-fast ASGI server implementation*. Accessed: 2025-06-25. 2025. URL: https://www.uvicorn.org/.

[4] Sebastián Ramírez. *FastAPI: The modern web framework for building APIs with Python 3.7+*. Accessed: 2025-06-25. 2025. URL: https://fastapi.tiangolo.com/.

[5] Augusto Rosa et al. *Melody - Integration Workshop 3*. https://github.com/IshikawaRasoto/integration-workshop-3. Accessed: 2025-06-25. 2025.