

LumiKey: Sheet-to-LED Piano Trainer

Authors: Chris Oh, Simon Lee, Gene Chang

Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

Abstract—LumiKey is a sheet-music-to-LED piano training system that converts printed sheet music into real-time LED guidance over a physical piano keyboard. The system captures sheet music through image upload, converts it into MusicXML using optical music recognition, transforms it into a structured event representation, and transmits the result via Bluetooth to an ESP32 microcontroller that drives RGB LEDs and evaluates user key presses. LumiKey supports beat and no-beat practice modes, provides low-latency feedback, and targets high pitch extraction accuracy and responsiveness to reduce the learning barrier for beginner pianists.

Index Terms—Optical Music Recognition, Embedded Systems, Bluetooth Communication, Real-Time Feedback, Music Education, Microcontroller

1 INTRODUCTION

Learning piano requires beginners to interpret musical notation while executing precise motor actions, creating a high cognitive load that can slow progress and cause frustration. LumiKey addresses this challenge by converting printed sheet music into real-time LED guidance mapped directly onto a physical piano keyboard.

The desired end-product is a low-cost, non-destructive retrofit system consisting of an LED overlay and sensing module mounted above an existing piano, paired with a lightweight software interface. A user uploads sheet music and receives time-aligned visual guidance and immediate feedback during practice. The primary application is guided learning for beginner and intermediate students in home or classroom settings.

From a social and economic perspective, LumiKey promotes equitable access to music education. Many guided-learning systems require specialized digital instruments or costly proprietary software. By operating on existing pianos, LumiKey reduces financial barriers while extending the lifespan of traditional instruments, supporting both affordability and sustainable reuse of hardware.

2 USE-CASE REQUIREMENTS

The use-case requirements define the functional and performance benchmarks necessary for LumiKey to serve as an effective educational tool. These requirements focus on the end-user experience, ensuring the system is reliable, responsive, and pedagogically sound.

1. **Image Upload Success:** The system must achieve a usable scan success rate of $\geq 90\%$ over 30 representative pages of sheet music. *Justification:* Reliability

is the cornerstone of user retention. If the initial step of digitizing music fails frequently, the barrier to entry becomes too high for the average student, leading to frustration and abandonment of the tool.

2. **Pitch Extraction Accuracy:** The system must correctly identify at least 90% of the pitches in a given piece of music. *Justification:* Accuracy is non-negotiable in a music education context. If the system provides incorrect guidance, it risks reinforcing "muscle memory" [4] errors that are difficult for a student to unlearn, thereby defeating the pedagogical goal.
3. **Low Latency Feedback:** The total perceived latency between a physical key press and the system's feedback must be ≤ 150 ms. *Justification:* Music is a time-sensitive discipline. A delay greater than 150 ms creates a "mismatch" between the user's tactile sensation [3] and the visual feedback, which is cognitively jarring and prevents the development of rhythmic fluidity.
4. **UI Interaction Responsiveness:** The User Interface (UI) must reflect changes and provide feedback within 200 ms of any user interaction. *Justification:* Following established Human-Computer Interaction (HCI) standards, 200 ms is the threshold for a system to feel "instant." [3] This ensures that the software does not feel sluggish during mode selection or file uploads.
5. **Varying Practice Modes:** The system must support specialized modes (Beat Mode, No-Beat Mode) and tempo adjustments (40–200 BPM) with state transitions occurring within 200 ms. *Justification:* Beginners require different levels of support as they progress. Providing a variety of modes allows for a scaffolded learning experience that adapts to the student's current skill level.

3 ARCHITECTURE AND PRINCIPLE OF OPERATION

The LumiKey architecture is designed to bridge the gap between static musical notation and real-time physical performance. The system operates on the principle of temporal synchronization, where symbolic data from sheet music is mapped to a discrete time-domain and then executed as a series of hardware interrupts.

3.1 Software Architecture

The software architecture is a unidirectional pipeline that emphasizes data integrity and timing precision. It is partitioned into three functional layers that handle the transition from pixels to digital events.

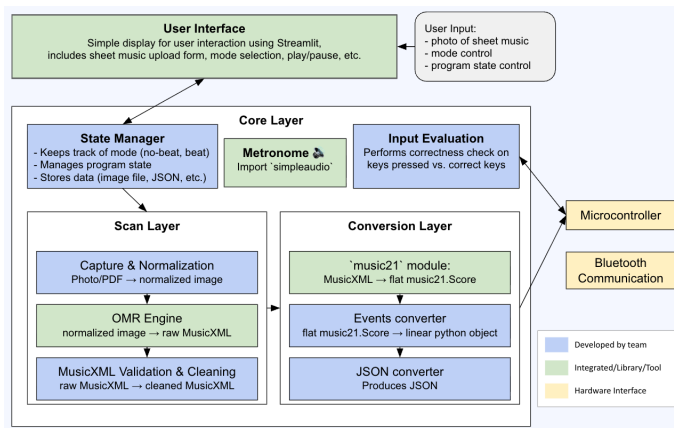


Figure 1: Block diagram of the software pipeline highlighting staged data processing, validation, and temporal synchronization from raw image input to serialized event output.

3.1.1 Symbolic Extraction (Scan Layer)

The Scan Layer functions as a preprocessing and post-processing wrapper around the external Optical Music Recognition (OMR) engine [1]. Prior to OMR execution, computer vision techniques are applied to normalize the input image, including grayscale conversion, adaptive thresholding, and geometric correction to reduce lighting variation and perspective distortion. The layer also estimates the staff space to provide a scale reference that improves pitch mapping robustness within the OMR process.

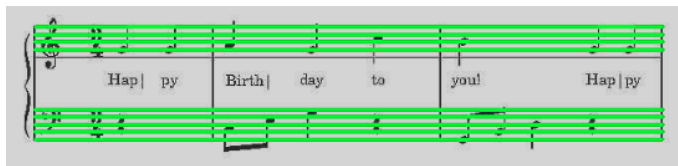


Figure 2: Detected staff lines (green) after preprocessing. The estimated staff spacing is used to calibrate vertical pitch mapping prior to OMR execution.

Following OMR execution, the generated MusicXML output [2] undergoes structural validation and cleanup to remove inconsistent metadata, verify measure integrity, and correct common symbol misclassifications, which are well-documented challenges in OMR systems [1].

3.1.2 Temporal Quantization (Conversion Layer)

The Conversion Layer is the mathematical engine of the system. Its primary responsibility is "Quantization"—the

process of converting musical durations (quarter notes, eighth notes) into absolute millisecond delays. The principle of operation relies on the formula $T_{delay} = \frac{60,000}{BPM \times PPQ}$, where PPQ (Pulses Per Quarter-note) defines the resolution of our system clock. This ensures that regardless of the musical complexity, the hardware receives a linear stream of timestamps that remain rhythmically consistent.

3.1.3 Execution Orchestration (Core Layer)

The Core Layer acts as the system's "Cerebellum," coordinating the flow of information between the user and the hardware. Its architecture centers on an asynchronous event loop. While one thread manages the UI rendering via Streamlit, a secondary background thread handles the Bluetooth SPP transmission. This prevents UI "hangups" from disrupting the data stream to the ESP32, ensuring that the hardware always has a filled buffer of upcoming notes to display to the student.

3.2 Hardware Architecture and Logic

The hardware architecture is designed for deterministic execution, ensuring that the visual LED guidance is precisely synchronized with the timing generated by the software layer, while also transmitting user key-press data back to the software for validation and real-time feedback.

3.2.1 Sensing and Feedback Loop

The input system operates on a variable resistance sensing principle. Each FSR functions as a transducer, converting applied mechanical force into a corresponding voltage change that can be measured by the ADC. The hardware architecture implements a centralized polling strategy in which the ESP32 scans the 16-channel multiplexer at a 1 kHz rate. This sampling frequency ensures compliance with the 150 ms end-to-end latency requirement by enabling near-instant detection of key strikes and rapid comparison against the expected note. The corresponding MIDI pitch is stored in a local buffer for validation and feedback processing.

3.2.2 Actuation Logic and Power Railing

The LED actuation architecture is driven by a single-wire digital protocol (WS2812B). To provide meaningful pedagogical feedback, the system implements a four-state color model: blue for left-hand notes, magenta for right-hand notes, green for correct key presses, and red for incorrect key presses. The ESP32 maintains a real-time state table for all 36 keys (three octaves), enabling synchronized visual guidance and feedback across the keyboard range.

The power system utilizes a standard 5V DC supply dedicated to the LED rail with a common ground shared with the ESP32. This architecture ensures sufficient power delivery for both the LEDs and the microcontroller while

isolating high-current LED switching behavior from sensitive analog circuitry. By stabilizing the 5V rail and maintaining proper grounding, the design minimizes noise coupling that could otherwise interfere with Bluetooth operation or introduce distortion into ADC readings from the FSR sensors.

4 DESIGN REQUIREMENTS

The design requirements translate high-level user needs into specific engineering targets for the software and hardware subsystems. These metrics are used to verify that the technical implementation supports the ultimate use-case goals.

LumiKey should support...		
Uploading Image Files	$P_{\text{conversion}} = P_{\text{preprocess}} * P_{\text{pipeline}} = (0.92)(0.98) \approx 0.90 \geq 0.90$	
Pitch Extraction Accuracy	$A_{\text{pitch}} = A_{\text{raw}} + \Delta A_{\text{pre}} + \Delta A_{\text{post}} = 0.87 + 0.02 + 0.01 = 0.90 \geq 0.90$	
Low Latency feedback	$T_{\text{total}} = T_{\text{sensor}} + T_{\text{processing}} + T_{\text{comm}} + T_{\text{LED}} = 25 + 30 + 25 + 50 = 130\text{ms} \leq 150\text{ms}$	
Basic UI Interaction	$T_{\text{UI}} = T_{\text{event}} + T_{\text{render}} = 100 + 80 = 180\text{ms} \leq 200\text{ms}$	
Varying Practice Modes	Timing Model + Accuracy	Responsiveness
	$T_{16}(B) = \frac{60}{4B}, \quad B \in [40, 200]$	$T_{\text{mode/bpm}} \leq 200\text{ms}$
	$ t_{\text{emit}} - t_{\text{ideal}} \leq 20\text{ms}$	$T_{\text{advance}} \leq 200\text{ms}$

Figure 3: Subsystem performance constraints derived from use-case goals to ensure overall system reliability and responsiveness.

- Uploading Pipeline Budget:** The overall scan success probability is modeled as $P_{\text{success}} = P_{\text{preprocess}} P_{\text{pipeline}}$, with internal targets $P_{\text{preprocess}} = 0.92$ and $P_{\text{pipeline}} = 0.98$, yielding $P_{\text{success}} = 0.92 \times 0.98 \approx 0.90$. *Justification:* By bounding cumulative failure probability across preprocessing and parsing stages, the system ensures that at least 90% of user-uploaded sheet music pages are successfully converted into usable MusicXML representations.
- Refined Pitch Extraction:** Pitch accuracy is modeled as $A_{\text{pitch}} = A_{\text{raw}} + \Delta A_{\text{pre}} + \Delta A_{\text{post}}$, where $A_{\text{raw}} = 0.87$, $\Delta A_{\text{pre}} = 0.02$, and $\Delta A_{\text{post}} = 0.01$, giving $A_{\text{pitch}} = 0.90$. *Justification:* Ensuring cumulative pitch accuracy of at least 90% guarantees that the LED guidance reflects the correct musical notes, preventing incorrect practice reinforcement and preserving instructional reliability.
- Latency Decomposition:** End-to-end latency is modeled as $T_{\text{total}} = T_{\text{sense}} + T_{\text{process}} + T_{\text{BT}} + T_{\text{LED}}$. With allocated budgets $T_{\text{sense}} = 25$ ms, $T_{\text{process}} = 30$ ms, $T_{\text{BT}} = 25$ ms, and $T_{\text{LED}} = 50$ ms, we obtain $T_{\text{total}} = 130$ ms ≤ 150 ms. *Justification:* Constraining subsystem delays through additive budgeting ensures that total feedback latency remains below 150 ms, maintaining perceptual synchronization between key presses and visual LED feedback.

- UI Performance Metrics:** UI latency is modeled as $T_{\text{UI}} = T_{\text{event}} + T_{\text{render}}$, with $T_{\text{event}} = 100$ ms and $T_{\text{render}} = 80$ ms, yielding $T_{\text{UI}} = 180$ ms ≤ 200 ms. *Justification:* Bounding interaction and rendering delays ensures that user interface responses occur within 200 ms, preserving a responsive and fluid interaction experience during practice.

- Timing Accuracy and Precision:** The sixteenth-note duration at tempo B is $T_{16} = 60/(4B)$. For $B = 200$ BPM, $T_{16} = 75$ ms. Timing deviation is bounded by $|t_{\text{emit}} - t_{\text{ideal}}| \leq 20$ ms. *Justification:* Limiting timing jitter relative to the minimum rhythmic interval ensures accurate LED actuation at high tempo, preserving rhythmic consistency and reliable beat guidance during playback.

5 SYSTEM IMPLEMENTATION

The LumiKey system is implemented through a distributed architecture that partitions high-latency computational tasks to a host PC while maintaining real-time deterministic control on the ESP32 microcontroller. This section details the software sub-components, hardware signal chain, and the integrated pipeline that links them.

5.1 Software Implementation

The software stack is organized into modular layers—Scan, Conversion, and Core—to facilitate parallel development and robust error handling.

5.1.1 Scan Layer and OMR Integration

The Scan Layer is implemented in Python and executes entirely on the host PC. Image preprocessing is performed using OpenCV (cv2), including grayscale conversion, adaptive thresholding, and perspective correction to standardize input quality prior to Optical Music Recognition (OMR).

OMR is executed locally using Audiveris, invoked through Python's `subprocess` module for automated batch processing. Audiveris outputs MusicXML files, which are then parsed using `music21` for structural validation and cleanup. Post-processing removes inconsistent metadata and enforces measure integrity to ensure compatibility with downstream event generation. All operations are performed locally to ensure reproducibility and eliminate external service dependencies.

5.1.2 Conversion Layer

The Conversion Layer processes cleaned MusicXML using the `music21` library. Scores are loaded [2] with `converter.parse()` and flattened to obtain a sequential note representation.

Note durations are converted to milliseconds using

$$T = \frac{60}{B} \times d,$$

where B is the selected tempo (BPM) and d is the duration in quarter-note units. Absolute onset times are accumulated iteratively to preserve ordering. Chords are decomposed into individual MIDI pitch events with shared onset timestamps.

Each event is serialized as a Python dictionary containing `pitch`, `onset`, and `duration`, then encoded using the built-in `json` module. Prior to BLE transmission, events are compressed into a compact comma-separated string format (e.g., 60,0,500;) to reduce parsing overhead on the ESP32.

5.1.3 Core Layer and Pipeline Integration

The Core Layer is implemented in Python and manages application state, user interaction, and communication with the embedded subsystem. The user interface is built using Streamlit, which handles file uploads, tempo selection, and practice mode configuration. Application state (current tempo, playback index, mode selection, and buffered event list) is maintained using Streamlit session state variables to ensure persistence across UI updates.

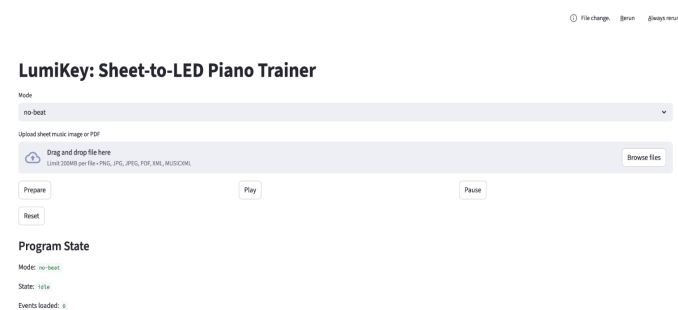


Figure 4: Streamlit-based user interface prototype showing mode selection, sheet music upload, playback controls, and program state display.

Bluetooth communication is implemented using the Bleak Python library for BLE client operation. Upon score conversion, serialized event strings are partitioned into fixed-size batches and transmitted to the ESP32 over a custom BLE characteristic. A lightweight acknowledgment protocol regulates transmission: the host sends a batch only after receiving a ready flag from the microcontroller, preventing buffer overflow and preserving event order. If an acknowledgment is not received within a timeout window, retransmission is triggered to maintain reliability.

Incoming key-press messages from the ESP32 are received asynchronously via BLE notification callbacks. These messages contain MIDI pitch identifiers corresponding to detected FSR activations. Reliable detection of human interaction is critical in embedded sensing systems [5]. The Core Layer compares incoming pitches against the expected event at the current playback timestamp and updates the UI state accordingly. Correct presses trigger visual confirmation, while mismatches update feedback indicators in real time. This bidirectional event handling

ensures synchronized interaction between the host software and embedded hardware.

5.2 Hardware Implementation

The hardware subsystem is designed to provide low-latency visual guidance and precise touch detection without modifying the user's piano.

5.2.1 Embedded Logic and Sensing

The system is centered on an ESP32-DevKitC V4 microcontroller, which manages a 36-key sensor matrix using Interlink FSR 402 sensors. To optimize GPIO utilization, the sensors are distributed across three CD74HC4067 16-channel analog multiplexers, allowing the ESP32 to scan all keys while minimizing ADC pin usage. The multiplexers share common select lines, while their outputs are connected to dedicated ADC inputs on the ESP32. The firmware implements a high-frequency polling loop to sequentially sample each channel, combined with software-based thresholding and debounce logic to ensure mechanical noise is not falsely interpreted as key strikes. The signals between components are illustrated in the block diagram shown in Figure 5.

5.2.2 Actuation and Power

Visual guidance is provided by a strip of WS2812B (NeoPixel) RGB LEDs mounted above the keys. The ESP32 controls the LEDs through a single high-speed digital data line, with each LED mapped to its corresponding MIDI pitch received from the software core layer. To ensure electrical stability and long-term hardware reliability, the system uses a dedicated 5V/4A power supply for the LED rail. A 2200 μ F decoupling capacitor is placed across the power rail to absorb transient current spikes caused by rapid LED switching, preventing voltage drops that could destabilize the ESP32, interfere with sensing accuracy, or disrupt BLE communication.

6 DESIGN TRADE STUDIES

LumiKey required several architectural tradeoffs balancing performance, cost, accessibility, and implementation complexity.

6.1 User Interface: CLI vs. Desktop GUI vs. Web-Based

We evaluated a CLI, local desktop GUI (e.g., Tkinter), and a web-based interface. A CLI was rejected due to poor accessibility for beginner users. Desktop GUIs required local Python environment management, increasing setup friction. Streamlit was selected because it enables a browser-based interface while maintaining Python-native development. This approach supports rapid iteration, low

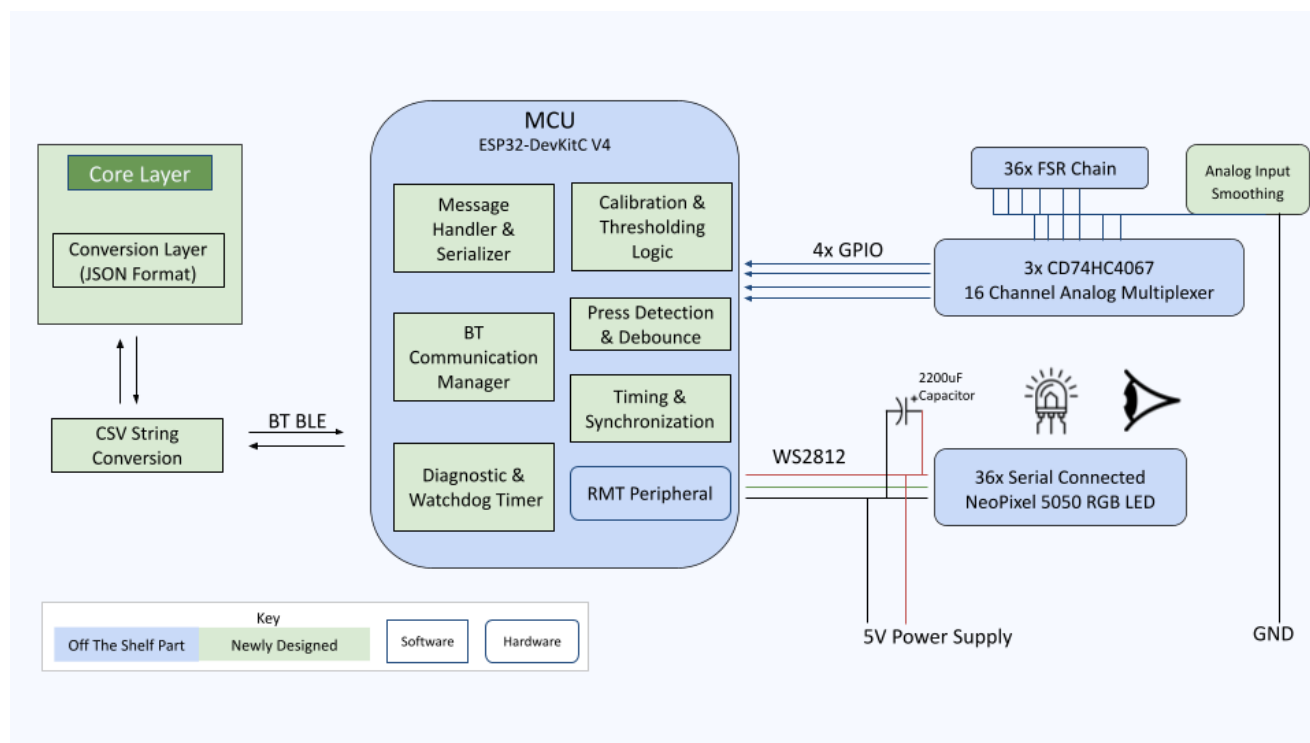


Figure 5: Block diagram of the LumiKeys system showing ESP32-based control, BLE communication, multiplexed FSR key sensing, and WS2812 LED actuation powered by a common 5V rail.

user setup burden, and seamless integration with the processing pipeline.

6.2 Data Transmission: Binary vs. JSON

For host-to-ESP32 communication, we compared binary encoding and JSON serialization. Binary encoding offers reduced packet size but increases development complexity and debugging difficulty when event structures change. Given that BLE throughput is sufficient for the event stream, JSON was selected for its readability, ease of logging, and rapid development iteration. Events are compressed prior to transmission to reduce overhead while retaining structured formatting.

6.3 Sensing Mechanism: IMUs vs. Hall Sensors vs. FSRs

We considered IMUs, Hall-effect sensors, and Force Sensitive Resistors (FSRs). IMUs would require one sensor per key and introduce I²C address contention, increased wiring complexity, and higher cost. Hall-effect sensing requires per-key magnet installation and precise alignment, increasing mechanical complexity and assembly time. FSRs were selected due to their thin profile, analog force output, low cost, and non-destructive placement on existing keys. This solution scales cleanly across multiple keys while preserving retrofit safety requirements.

6.4 System Scope: Custom Instrument vs. Retrofit

A custom MIDI-enabled keyboard would provide direct digital ground truth but significantly increases manufacturing cost and requires users to replace existing instruments. The retrofit overlay approach leverages existing acoustic or digital pianos, reducing cost, preserving instrument authenticity, and aligning with accessibility goals.

6.5 Public Health, Safety, and Societal Considerations

Public health and safety considerations were integrated directly into LumiKey's hardware and software design decisions. Electrically, the system operates entirely at regulated 5V DC levels, eliminating hazardous high-voltage exposure. A dedicated 5V rail supplies the LED array, while decoupling capacitors and common grounding reduce transient voltage spikes that could cause unpredictable behavior or sensor misreads. All exposed conductors are low-voltage and enclosed within insulated housing to minimize short-circuit or contact risk.

Mechanically, the retrofit architecture avoids drilling, adhesives requiring permanent bonding, or structural modification of the piano. The selected Force Sensitive Resistors (FSRs) are thin-film devices placed non-destructively on top of the keys, eliminating the need for embedded magnets, rigid brackets, or sharp components. This preserves user property and reduces risk of injury or mechanical stress.

to the instrument.

Software implementation decisions also reflect safety and privacy considerations. Optical Music Recognition and image preprocessing are executed locally using open-source tools, eliminating reliance on cloud-based processing and avoiding transmission of user-uploaded sheet music to external servers. The BLE communication protocol includes bounded buffering and acknowledgment logic to prevent synchronization errors that could produce misleading instructional feedback.

From a societal and economic perspective, LumiKey is designed as an accessible retrofit overlay rather than a proprietary instrument. Many guided-learning systems require specialized digital keyboards, which can impose financial barriers. By leveraging existing acoustic or digital pianos and using commercially available microcontroller platforms and open-source software (OpenCV, Audiveris, music21), the system reduces upfront cost and promotes reproducibility.

Environmental considerations further influenced the solution. Extending the functional capabilities of existing pianos reduces electronic waste associated with manufacturing new MIDI-enabled instruments [6]. The use of off-the-shelf components minimizes custom fabrication and associated material overhead.

Collectively, these considerations guided the adoption of a low-voltage embedded architecture, non-destructive sensing strategy, local processing pipeline, and modular retrofit deployment model.

7 TEST, VERIFICATION, AND VALIDATION

This section details the methodical testing procedures planned to verify that LumiKey meets its quantitative design and use-case requirements. Each test is designed to map specific inputs to expected functional outputs, while accounting for potential risk factors that may necessitate mitigation.

7.1 Image Conversion Verification

The objective of this test is to verify the robustness of the Scan Layer in translating physical media into a machine-readable format. To perform this, we will process a diverse dataset of at least 30 clean, printed sheet music pages through the full Scan Layer pipeline. The input consists of high-resolution photos or PDF scans of printed-out, clean sheet music, and the expected output is a structured MusicXML file that achieves a $\geq 90\%$ match with the original musical notation. During this behavior, the system must accurately identify staff lines, clefs, and basic note durations. We recognize that variable photo quality, poor lighting, or lens distortion can impede the OMR engine; mitigation involves using OpenCV normalization techniques such as adaptive thresholding and perspective correction.

7.2 Pitch Extraction Accuracy Validation

This test validates that the Conversion Layer accurately interprets the symbolic music data produced by the Scan Layer. The testing method involves an automated comparison of the generated JSON event list against a manually verified ground-truth MIDI representation of the same music. The input is the output from the Scan Layer (MusicXML) derived from clean sheet music photos, which should result in a JSON object containing note events and metadata that maintains a $\geq 90\%$ match with the ground truth. The conversion must correctly map MusicXML note values to their corresponding linear time-indexed MIDI pitches. Risk factors include complex rhythmic structures like triplets or dense polyphonic chords, which can lead to extraction errors.

7.3 User Interface Responsiveness

To ensure high-quality human-computer interaction, we must verify the speed and correctness of the Core Layer's frontend. This is achieved through manual interaction testing paired with software profiling to measure the time elapsed between a user action and the visual update. Using rapid sequential button presses, mode toggles, and image file uploads via the Streamlit interface as inputs, the expected output is the correct execution of the requested state change with UI latency updates $\leq 200\text{ms}$. The UI should remain fluid and provide immediate feedback for program state controls. A known risk is that invalid user inputs or heavy processing during file uploads could cause interface freezing.

7.4 Transmission and Feedback Latency

This test verifies the round-trip communication efficiency between the Core Layer and the physical microcontroller. The testing method uses a digital oscilloscope or high-speed software logging to time the round-trip of a "ping" command and subsequent "Ack" signal. The bi-directional data packets sent and received over the Bluetooth Communication channel serve as the input, with a successful test yielding a confirmed acknowledgement signal within $\leq 150\text{ms}$, ensuring no data is lost or corrupt during transit. The system must maintain a stable connection even during dense data bursts. However, Bluetooth timing jitters or buffer overflows on the ESP32 remain risks that can lead to missed notes.

7.5 Beat Mode Timing and Actuation

This validates the real-time performance of the integrated system in a pedagogical "play-along" scenario. We will use video analysis of the physical hardware synchronized with a metronome to measure the time-spacing accuracy of the LED guidance. Structured sheet music event streams at varying BPM settings are used as input, and the expected output is that LEDs actuate with accurate time

spacing relative to the tempo, with mode transitions occurring within ≤ 200 ms. The LED guidance should lead the user predictably according to the rhythmic requirements of the piece. Risks for this stage include timing jitters at higher BPMs, which can disrupt the player's rhythm.

8 PROJECT MANAGEMENT

The execution of LumiKey is managed through a rigorous schedule and a clear division of labor designed to balance the workload across specialized software and hardware domains, ensuring that all subsystems are integrated in time for the final demonstration.

8.1 Work Distribution

Tasks were assigned based on each team member's specific technical expertise and prior experience with the relevant stacks. This allows for parallel development of the software pipeline and the physical hardware interface, reducing the critical path duration. The distribution is as follows:

- **Chris Oh (Software):** Focuses on high-level logic and integration. His primary responsibilities include the development of the Core Layer and the Conversion Layer, managing the end-to-end pipeline integration, and overseeing the sheet scanner's interaction with the hardware. He is also responsible for unit test design and overall system debugging.
- **Simon Lee (Software):** Specializes in the data processing and algorithmic stages. His work centers on the Scan Layer, including image preprocessing and the OMR pipeline. Simon is responsible for MusicXML generation, data validation, and the development of quantization logic, providing essential support for the Core Layer's data requirements.
- **Gene Chang (Hardware):** Manages the physical and embedded implementation. This includes the development of the Hardware Layer, LED bit-stream control via the ESP32, and the integration of the force-sensitive resistor (FSR) input system. Gene ensures the low-level communication between the sensors, LEDs, and the software Core Layer is robust and low-latency.

8.2 Project Schedule

To manage the complexity of the project timeline, the team utilizes a detailed Gantt chart. This schedule forecasts major milestones—such as the completion of the OMR pipeline and the final hardware assembly—while tracking individual task progress. The chart is reviewed weekly to ensure that no particular teammate is overwhelmed and that the project remains on track for the final demo.

8.3 Bill of Materials

The full Bill of Materials is provided in Figure 7 at the end of the document to ensure readability.

9 SUMMARY

LumiKey delivers a high-accuracy, low-latency piano training tool. By adhering to the quantitative targets in Fig. 3, we ensure the system is both technically sound and pedagogically effective.

Glossary of Acronyms

OMR	– Optical Music Recognition
BPM	– Beats Per Minute
PPQ	– Pulses Per Quarter-note
FSR	– Force Sensitive Resistor
MCU	– Microcontroller Unit
SPP	– Serial Port Profile
ADC	– Analog-to-Digital Converter
BLE	– Bluetooth Low Energy

References

- [1] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. Marcal, C. Guedes, and J. Cardoso, "Optical music recognition: state-of-the-art and open issues," *International Journal of Multimedia Information Retrieval*, vol. 1, no. 3, pp. 173–190, 2012, doi: 10.1007/s13735-012-0011-2.
- [2] M. Good, "MusicXML for notation and analysis," in *The Virtual Score: Representation, Retrieval, Restoration*, W. B. Hewlett and E. Selfridge-Field, Eds. Cambridge, MA, USA: MIT Press, 2001.
- [3] J. Nielsen, "Response times: The 3 important limits," Nielsen Norman Group, Mar. 6, 1993. [Online].
- [4] B. Lafreniere, T. R. Jonker, S. Santosa, and M. Parent, "False positives vs. false negatives: Understanding their effect on user performance and experience," in *Proc. ACM Symp. User Interface Software and Technology (UIST)*, 2021.
- [5] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013, doi: 10.1109/SURV.2012.110112.00192.
- [6] V. Forti, C. P. Baldé, R. Kuehr, and G. Bel, *The Global E-waste Monitor 2020: Quantities, flows and the circular economy potential*. Bonn, Germany: United Nations University (UNU), 2020. [Online]. Available: <https://ewastemonitor.info/gem-2020/>

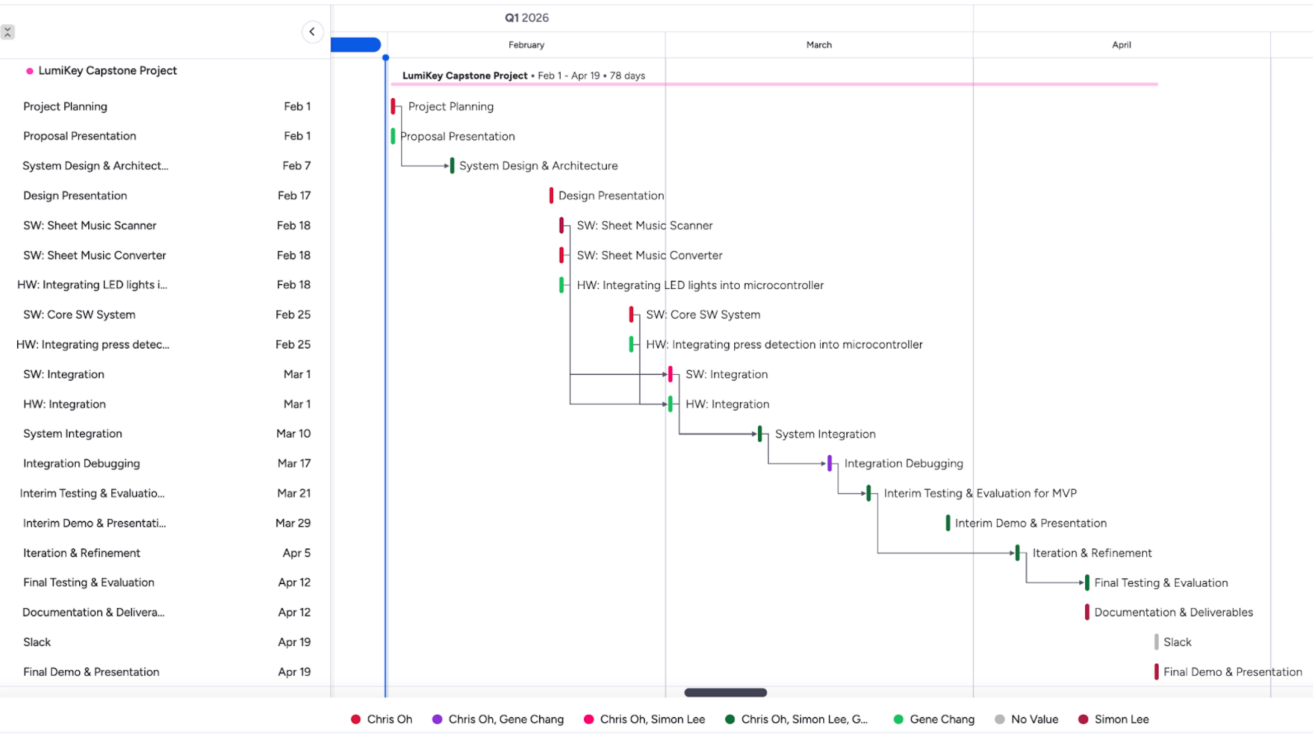


Figure 6: LumiKey Project Schedule and Work Distribution. This Gantt chart outlines the development tracks for software components (Scan, Conversion, Core) and hardware milestones (FSR matrix, LED control) across the semester.

LumiKey BOM					
Parts	Material	Manufacturer	Quantity	Cost per Unit	Subtotal
	NeoPixel Mini Button PCB - Pack of 5 Product ID: 1612	Adafruit Industries LLC	9	\$4.46	\$40.14
	2Pcs RP-L Film Pressure Sensor Pressure Detector Flexible	JESSINIE	20	\$12.95	\$259.00
	BNTECHGO 22 Gauge Silicone Ribbon Cable Flexible 4P Black 25 ft Flat Cable 22 AWG Stranded Tinned Copper Wire	BNTECHGO	1	\$14.98	\$14.98
	ESP32-DevKitC-32E Development Board.	Espressif Systems	2	\$11.00	\$22.00
	12Pcs CD74HC4067 16-Channel Analog Digital Multiplexer MUX Breakout Board Module CD74HC4067 CMOS Precise Module Compatible with DIY	Elecbec	1	\$9.99	\$9.99
Subtotal					\$346.11

Figure 7: Bill of Materials outlining the key hardware components used in the LumiKeys prototype, including sourcing information, quantities, and cost breakdown.