CodeForGoodTech 2026

Advanced Temperament System for Music Blocks v3

Project Proposal By

Srajan Gupta

Email Address	srajangupta1818@gmail.com
Github	https://github.com/srajang1805
Linkedin	https://www.linkedin.com/in/srajang15//
Phone	+91 9993281848
Resume	Srajan Gupta Resume
TimeZone	Indian Standard Time(IST) (GMT + 5:30)
Idea Link	GitHub Issue
College & Branch	IIIT Jabalpur CSE

DMP (CodeForGoodTech 2026) Proposal

Music Blocks Temperament System Refactor and Enhancement

Organization: Sugar Labs

Project: Music Blocks Temperament System Refactor and Enhancement

Contributor: Srajan Gupta

Mentors: Walter Bender, Devin Ulibarri

Abstract

This project proposes a **comprehensive refactor and enhancement of the temperament subsystem in Music Blocks v3 by backporting the robust temperament engine from Music Blocks v4 and extending the surrounding ecosystem to fully support both EDO and non-EDO tuning systems throughout the platform.**

The project will address long-standing architectural limitations that currently prevent consistent behavior across pitch stepping, chord generation, interval operations, and synthesis under alternate tuning systems. **In addition to core engine improvements, the project will integrate workflow-oriented tools such as the Scale & Mode Builder and a Temperament Visualizer to create a seamless educational experience.**

An initial prototype implementation has already been completed, including **working support for Pythagorean and Just Intonation tuning systems, integration of the Scale & Mode Builder directly into the Temperament Widget, lazy-loaded module initialization, and improvements to interval mapping reliability**.

The final outcome will be a **stable, extensible temperament engine that enables Music Blocks to become one of the few beginner-friendly music education environments with deep support for microtonality and alternative tuning systems.**

Introduction

Most beginner-oriented music software supports only **12-tone Equal Division of the Octave (12-EDO). While practical, this limitation hides the rich diversity of tuning systems used historically and culturally across the world**.

This proposal aims to solve these issues **by modernizing the temperament engine, improving interoperability across musical operations, and building an integrated workflow for exploring scales, modes, and alternate tunings.**

Problem Statement

Although Music Blocks includes a Temperament Widget and tuning-related APIs, the current implementation suffers from **architectural inconsistencies dating back to the early versions of the platform.**

These issues include:

- Partial or broken non-EDO support
- Silent fallback to equal temperament
- Incorrect interval synthesis
- Chord generation inconsistencies
- Pitch stepping inaccuracies
- Lack of integrated scale-building workflows
- Limited visualization tools
- Fragile internal mappings between interval systems

As a result, many advanced tuning systems appear available but are not reliably functional.

Current Limitations in Music Blocks v3

The existing implementation has several critical limitations.

Incomplete Temperament Definitions

Several tuning systems exist only as UI labels without complete mathematical backing.

Interval Mapping Errors

Internal interval naming mismatches cause failures during:

- Chord construction
- Step pitch operations
- Modal pitch generation

Hardcoded Equal Temperament Assumptions

Many synthesis paths assume 12-EDO behavior even when alternate temperaments are selected.

Workflow Fragmentation

The Scale & Mode Builder exists independently instead of integrating naturally with the Temperament Widget.

Lack of Visualization

Users cannot easily visualize:

- Frequency relationships
- Interval spacing
- Cent deviations
- Scale structures

Proposed Solution

The proposed project consists of five interconnected improvements:

Goal 1 Support any EDO temperament

Goal 2 Refactor custom pitch block

Goal 3 Support non-EDO temperaments

Goal 4 Integrate Scale & Mode Builder

Goal 5 Build Temperament Visualizer

Together, these changes create a unified and extensible temperament architecture**.**

Detailed Goals

Goal 1 — Supporting Any EDO Temperament

Objective

Generalize equal temperament calculations to support:

- 12-EDO
- 19-EDO
- 24-EDO
- 31-EDO
- arbitrary user-defined divisions

Goal 2 — Refactor Custom Pitch Block

Current Problem

The custom pitch block assumes fixed pitch relationships.

Planned Improvements

- Dynamic frequency resolution
- Temperament-aware pitch translation
- Consistent interval behavior

Goal 3 — Supporting Non-EDO Temperaments

Planned Additions

- Meantone temperament
- Extended Just Intonation
- User-defined ratio systems

Technical Improvements

- Ratio normalization
- Fractional pitch handling
- Accurate synthesis mapping

Goal 4 — Scale & Mode Builder

Planned Features

- Integrated modal construction
- Temperament-aware scale generation
- Interactive scale editing
- Visual interval feedback

Goal 5 — Temperament Visualizer Widget

Planned Features

- Circular pitch visualization
- Cent deviation display
- Ratio visualization
- Real-time tuning updates
- Interactive comparison tools

Goal 6 — Exporting and Importing Temperaments

Planned Features

- JSON-based temperament schema
- Import/export UI
- User presets
- Sharing capabilities

Implementation Plan

Phase 1 — Core Engine Refactor

- Audit the existing temperament engine and identify all hardcoded Equal Division of the Octave (EDO) assumptions across pitch generation, interval calculation, and synthesis logic.
- Backport stable temperament-processing logic from the v4 implementation into the current codebase while preserving compatibility with existing Music Blocks workflows.
- Refactor interval computation into a generalized ratio-driven processing pipeline supporting:
    - Arbitrary EDO systems
    - Just Intonation
    - Pythagorean tuning
    - Custom user-defined temperaments
- Normalize pitch and interval representations internally to eliminate duplicated conversion logic and inconsistent temperament mappings.
- Introduce centralized utility functions for:
    - Frequency calculation
    - Ratio normalization
    - Interval indexing
    - Pitch resolution
- Ensure backward compatibility for existing projects relying on standard 12-EDO behavior.
- Improve modular separation between temperament logic, synthesis logic, and UI-facing utilities to simplify future extensibility.

Phase 2 — Pitch & Synthesis Integration

- Refactor custom pitch blocks to consume the new generalized temperament engine instead of relying on fixed equal-temperament assumptions.

- Update pitch resolution workflows to correctly interpret intervals and scale degrees across non-EDO systems.
- Ensure synthesis pipelines generate acoustically accurate playback for arbitrary tuning systems and microtonal intervals.
- Standardize note-to-frequency conversion behavior across:
    - Playback engine
    - Keyboard interactions
    - Pitch widgets
    - Export systems
- Eliminate inconsistencies between visual pitch representation and actual synthesized frequencies.
- Add validation safeguards for malformed or unsupported temperament configurations.
- Optimize interval lookup and frequency generation performance for real-time interaction scenarios.

Phase 3 — UI Integration

- Improve interoperability between the Temperament Widget, Scale Builder, and pitch-related blocks.
- Refactor widget communication flows to ensure synchronized state updates between tuning systems and generated scales.
- Extend scale-building functionality to support arbitrary interval structures and user-defined temperament mappings.
- Introduce visualization tools for:
    - Interval relationships
    - Frequency distributions
    - Scale structures
    - Tuning deviations
- Improve responsiveness and usability of temperament-related interfaces for educational workflows.
- Ensure all UI components correctly adapt to dynamically changing tuning systems without requiring manual refreshes or resets.
- Reduce UI coupling with legacy pitch-processing logic through cleaner state abstractions.

Phase 4 — Import/Export System

- Design a structured serialization format for storing custom temperament definitions, interval mappings, and scale metadata.

- Implement import/export pipelines supporting persistence and sharing of user-created tuning systems.
- Add schema validation for imported temperament configurations to prevent invalid interval definitions or malformed scale structures.
- Ensure exported temperament data remains version-compatible with future engine updates.
- Build user-facing controls for:
    - Saving custom temperaments
    - Loading external configurations
    - Resetting default tuning systems
    - Managing saved presets
- Optimize parsing and serialization workflows for large or complex temperament configurations.

Phase 5 — Testing & Documentation

- Develop unit tests covering:
    - Interval normalization
    - Frequency calculation
    - Pitch conversion
    - Temperament serialization
- Add integration tests validating consistent behavior across synthesis, widgets, pitch blocks, and scale generation workflows.
- Test compatibility across:
    - Standard 12-EDO systems
    - Non-EDO temperaments
    - Custom user-defined scales
- Validate real-time synthesis behavior for microtonal and arbitrary interval playback.
- Prepare developer documentation explaining:
    - Core architecture changes
    - Generalized temperament processing
    - Serialization formats
    - Extension points for future contributors
- Create educational documentation and examples demonstrating:
    - Non-EDO systems
    - Custom scale creation
    - Temperament visualization workflows
    - Practical music theory applications within Music Blocks.

- 
Phase 1: Core Engine Refactor
add
Phase 2: Pitch & Synthesis Integration
add
Phase 3: UI Integration
add
Phase 4: Import/Export System
add
Phase 5: Testing & Documentation
add

Demonstration Work

- **Raised a PR (https://github.com/sugarlabs/musicblocks/pull/7317) implementing non-12 EDO temperament behavior during playback and pitch operations**

Non-EDO Temperament Support

Implemented:

- **Pythagorean tuning**
- **5-limit Just Intonation**

inside musicutils.js.

Added Features

- **Exact interval ratios**
- **Cent mappings**
- **Frequency relationships**
- **Robust interval ordering**

Examples include:

- **256/243**
- **3/2**
- **5/4**
- **15/8**

```
"just intonation": {
    name: "Just Intonation",
    description: "5-limit tuning based on whole number ratios",
    edo: null,
    pitchNumber: 12,
    generator: null,
    intervals: {
        "perfect 1": { ratio: 1, cents: 0 },
        "minor 2": { ratio: 16 / 15, cents: 111.731 }, // Pure diatonic semitone
        "major 2": { ratio: 9 / 8, cents: 203.910 },   // Major whole tone
        "minor 3": { ratio: 6 / 5, cents: 315.641 },   // Pure minor third
        "major 3": { ratio: 5 / 4, cents: 386.314 },   // Pure major third
        "perfect 4": { ratio: 4 / 3, cents: 498.045 }, // Pure perfect fourth
        // ...
        "major 7": { ratio: 15 / 8, cents: 1088.269 }, // Pure major seventh
        "perfect 8": { ratio: 2, cents: 1200 }
    },
    // ...
}
```

This ensures mathematically correct synthesis behavior.

Internal Mapping Refactor

Corrected mismatches between:

- interval keys
- intervalOrder arrays
- synthesis lookup structures

This eliminated regression issues affecting:

- pitch stepping
- interval synthesis
- chord generation

Srajan Gupta

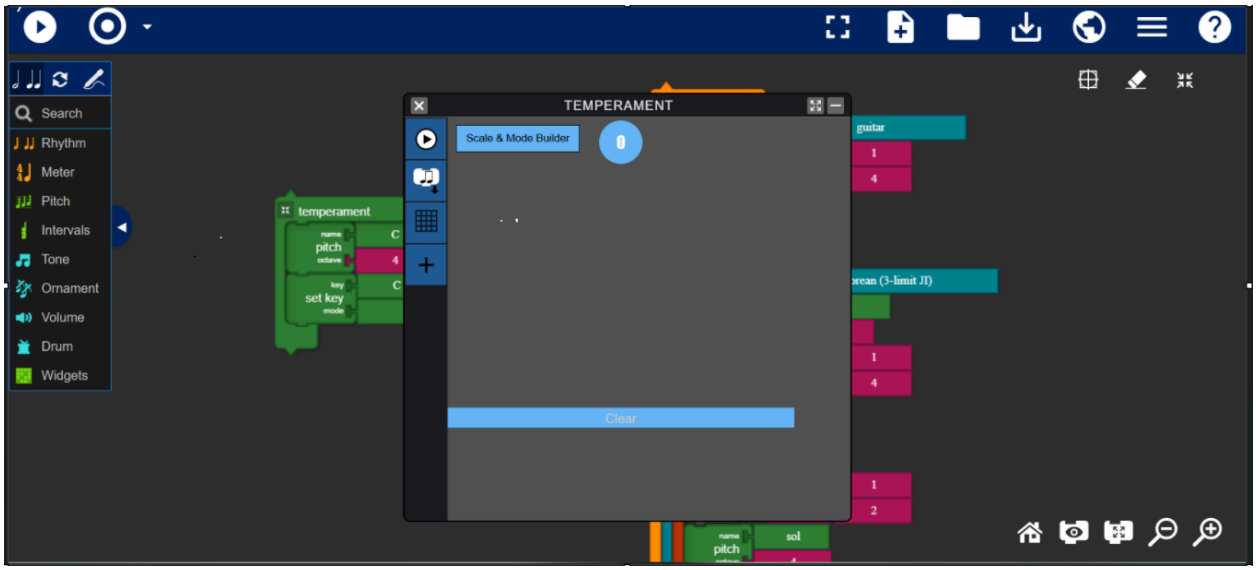
```
intervals: {
    "perfect 1": { ratio: 1, cents: 0 },
    "minor 2": { ratio: 256 / 243, cents: 90.225 },
    // ...
},
intervalOrder: [ "perfect 1", "minor 2", ... ]
```

Scale & Mode Builder Integration

Previously, the Scale & Mode Builder was isolated from the Temperament Widget.

Implemented:

- Direct widget integration
- Embedded launch button
- Lazy-loaded initialization
- Shared activity context



Lazy Loading Improvements

Used _lazyRequire to dynamically load ScaleModeBuilder only when required.

```
if (typeof _lazyRequire !== "undefined") {
    // Dynamically fetch the ScaleModeBuilder module over the network
    _lazyRequire(["widgets/scalemodebuilder"], function () {
        // Instantiate only after the script has successfully loaded
        if (that._logo.scaleModeBuilder === undefined || that._logo.scaleModeBuilder === null) {
            that._logo.scaleModeBuilder = new ScaleModeBuilder();
        }
        that._logo.scaleModeBuilder.init(that.activity);
    });
} else if (typeof ScaleModeBuilder !== "undefined") {
    // Fallback execution if the environment natively bundled it
    if (that._logo.scaleModeBuilder === undefined || that._logo.scaleModeBuilder === null) {
        that._logo.scaleModeBuilder = new ScaleModeBuilder();
    }
    that._logo.scaleModeBuilder.init(that.activity);
```

Stability Verification

The implemented changes currently preserve:

- **existing architecture**
- **existing APIs**
- **Jest test compatibility**

Core test suites remain stable and functional.

Technical Architecture

Primary files involved:

File	Responsibility
musicutils.js	Temperament engine and interval logic
turtle-singer.js	Pitch synthesis and playback
temperament.js	Widget UI
synthutils.js	Frequency generation
ScaleModeBuilder.js	Scale construction

The refactor will maintain backward compatibility while replacing fragile tuning assumptions.

Why I Am a Good Fit

I am a Computer Science student with experience in JavaScript, open-source contribution, debugging legacy systems, and architectural refactoring.

I have explored the Music Blocks codebase in depth and already contributed through a PR implementing non-12 EDO temperament behavior during playback and pitch operations.

I have also implemented temperament-related improvements, integrated UI workflow changes, and verified compatibility with the existing testing infrastructure. This prior work gives me direct familiarity with the project architecture, temperament engine, pitch processing pipeline, and related UI systems, allowing me to contribute immediately to the proposed implementation.

Deliverables

By the end of the project, Music Blocks **will include:**

- **Stable arbitrary EDO support**
- **Reliable non-EDO temperament synthesis**
- **Refactored custom pitch handling**
- **Integrated Scale & Mode Builder**
- **Temperament Visualizer**