D4 - Synesthesia

Final Presentation

Abhishek Agarwal
Parth Maheshwari
Rachana Murali Narayanan

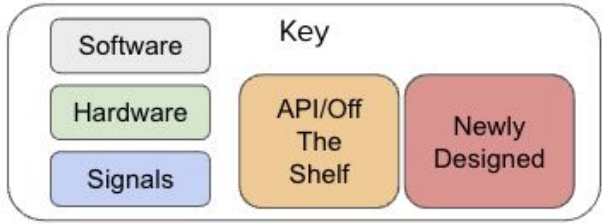
Use Case & Requirements

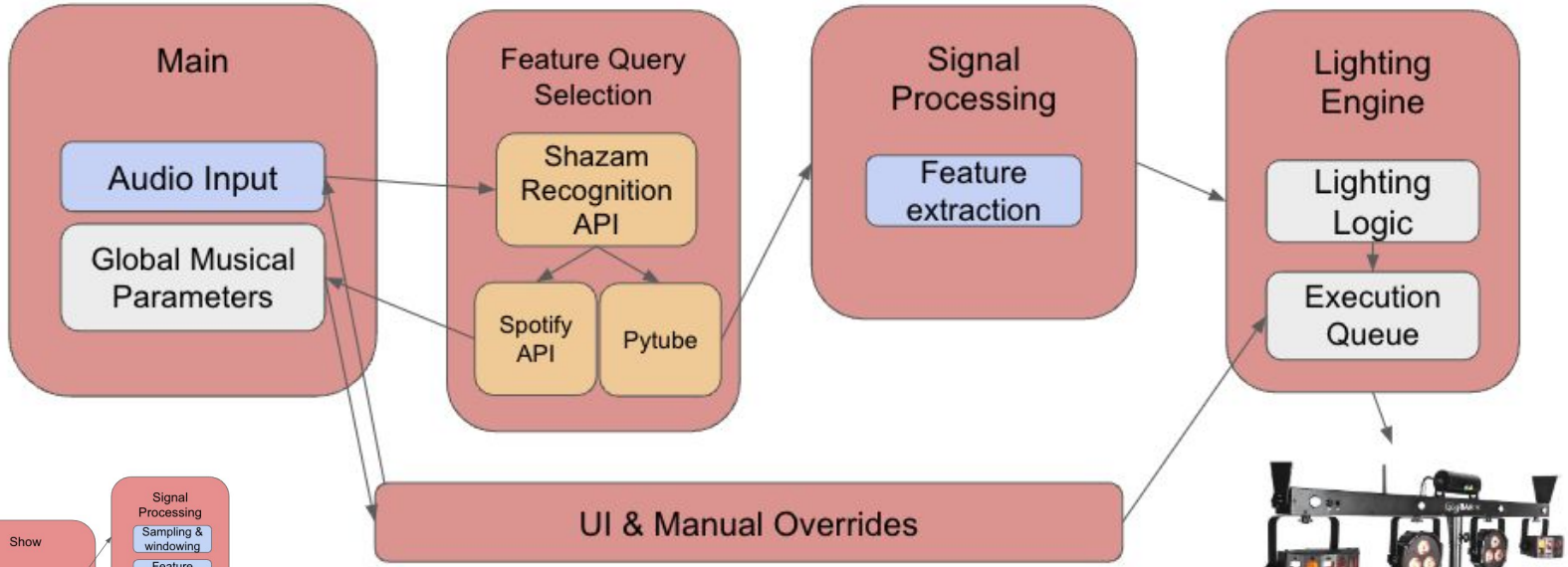
Automating light shows for performers using minimal equipment and signal processing

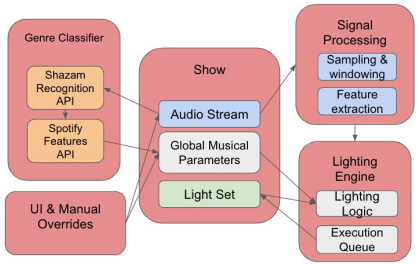
Our sandbox visualization

Requirement	Target Metric
Pre-processing latency	<10-15 seconds of the song
Setup time	< 5 mins of 1 hr performer setup time
Signal Processing	Extract > 90% of the auditory features of hand labeled audio file
Manual adjustments	< 3 adjustments per minute per song

Design vs Solution Approach



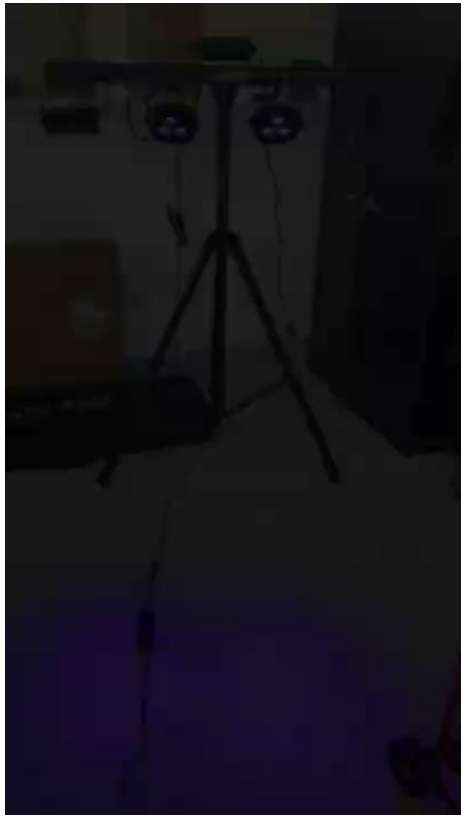




Demo Videos

Video 1

Changing brightness values of the light according to the amplitudes of a standardized soundtrack



Video 2

Detecting the beat timestamps in a soundtrack and changing functions on the beat

Testing and Validation

Subsystem	Tests	Testing Process	Results
Setup time	• Efficiency test • Processing load test	**Efficiency**: Ensure that the setup time for a performance is below 5 minutes **Processing load**: Ensure that the song recommender can generate recommendations and play for over an hour	***In Progress*** Group members take < 5 minutes. Additional participants must be tested ***In Progress*** We must finish song recommendation integration
UI and Manual adjustments	• Performance tests	**Performance:** Test on a sample audio with different genres and determine how many adjustments are needed	***Not Applicable*** User manual adjustments are the stylistic choice of the user and should not be limited
Lighting	• Functionality testing • Concurrency testing	**Functionality:** Blackout, SetColor, Fade, Rotate, Strobe, ColorCycle, Hold with different timers **Concurrency:** simultaneous threads and overwrites	***Pass!*** 100% successful with different hold intervals, and negligible delays

Testing and Verification

Subsystem	Tests	Testing Process	Results
Signal Processing	• Functionality testing • Stress testing (different audios, isolate features)	**Functionality:** Extracting features, hand labeled reference, different window frames/sampling rates **Stress:** isolating amplitude, frequency, and beats. Test distortion or clipping of files, load on Shazam-Spotify workflow	***Fail!*** Real time windowing unsuccessful with librosa ***Pass!*** 100% successful extraction
Feature Query	• Song recognition accuracy	**Recognition Accuracy**: We tested 15 different songs from Billboard's top 100 playlist and the song recognition was able to accurately identify each of them	***Pass!*** 15/15 songs were accurately identified within 5 seconds each
Integration Test	• UI-Light integration • Signal-Light integration	**UI-Lighting:** Real time synchronization stress test of user commands across the song, error handling when UI-Light connection breaks **Signal-Light:** Each set of signal parameters translated to lighting deterministically, high processing load of big files handled quickly	***Pass!*** 100% successful except on overlapping user commands at the UI

Testing : Initial Audio Metrics

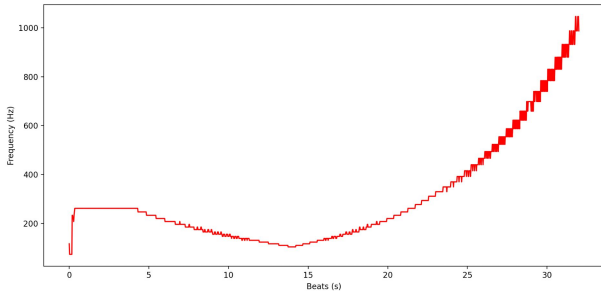


Figure 1: *Frequency against beats for a pitch modulated file*

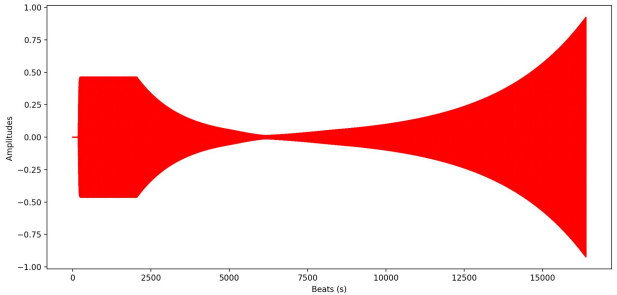


Figure 2: *Amplitudes against beats for a loudness modulated file*

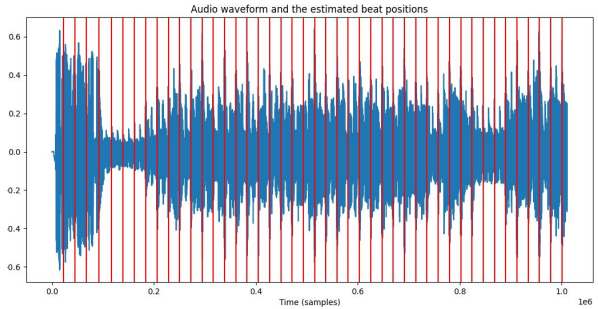


Figure 3: *Beat positions across time for a 20 second audio clip of “Teenage Dream”*

Testing : Processing Data

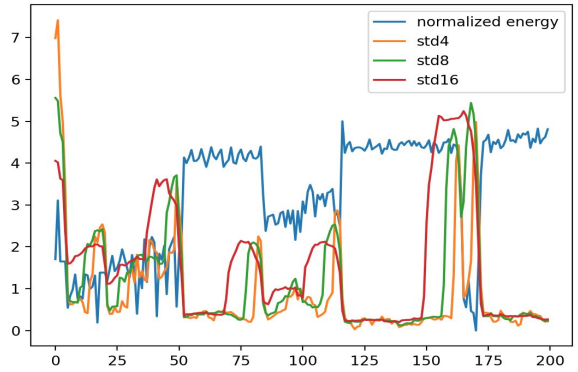


Figure 1: *Beat timestamps vs Standard Deviations in Energy for 4, 8, and 16 beat windows*

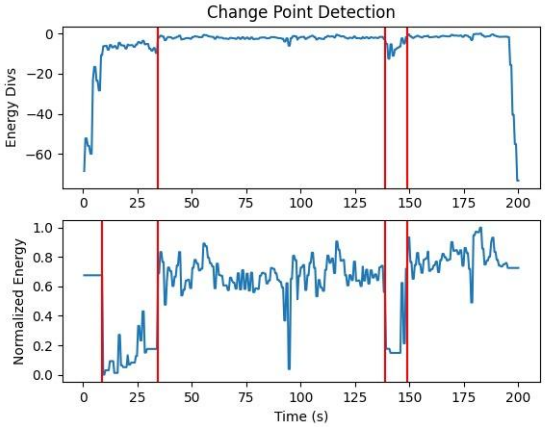


Figure 3: *Change point detection with normalized energies on "Teenage Dream"*

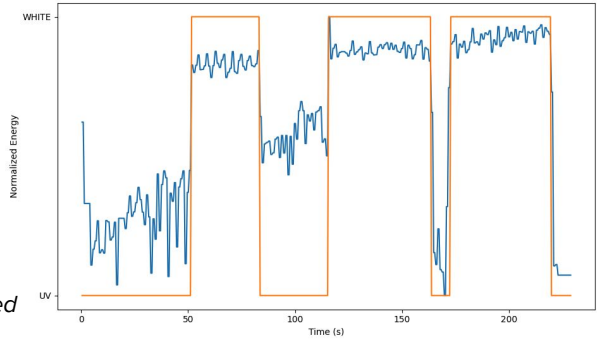


Figure 2: *Normalized Energy with Outlier Removal overlaid with strobe color decision*

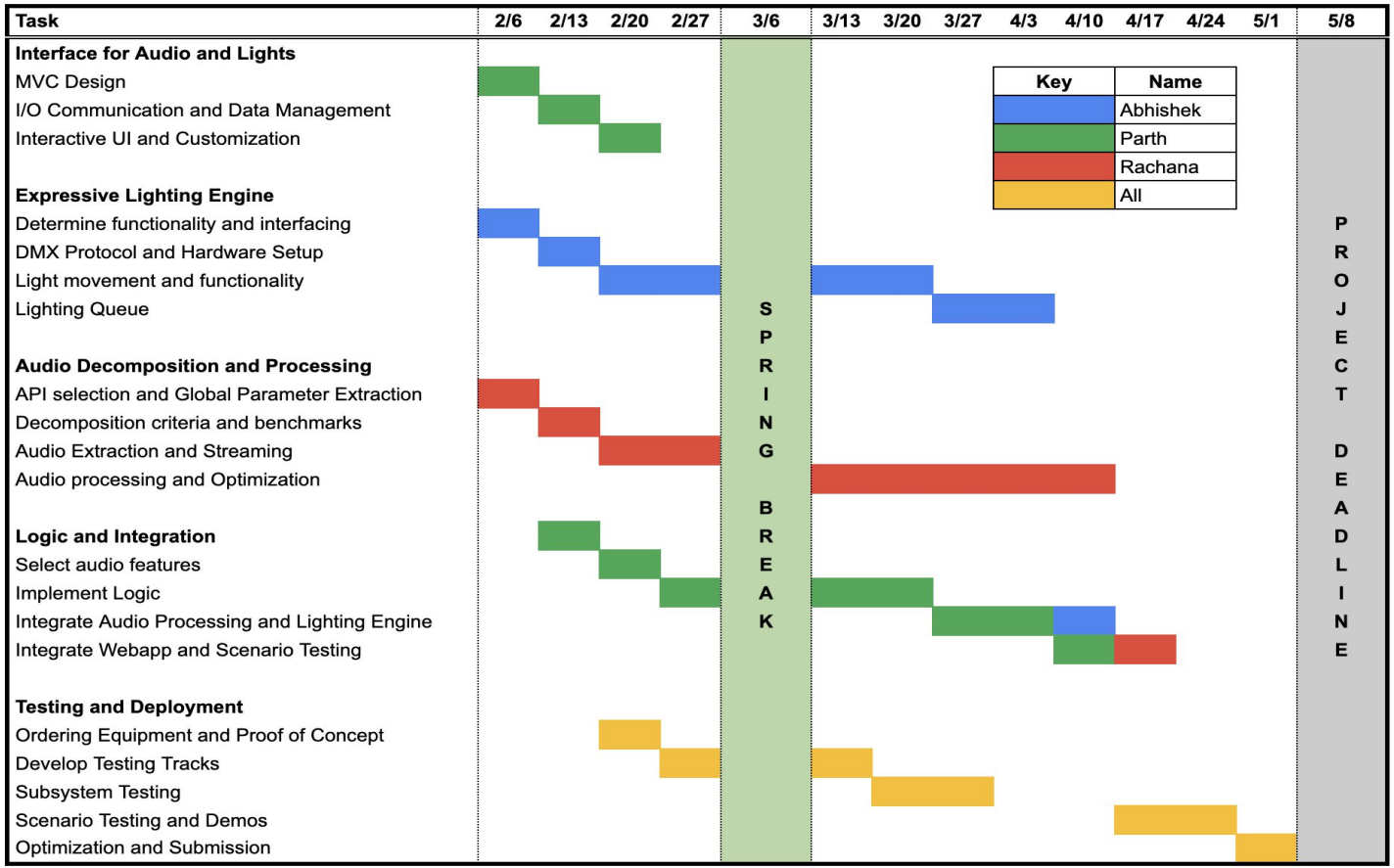
Trade Offs

- **ML Genre Classifier vs Shazam-Spotify recommendation system**
  - Genre classifier: Simplistic information on genres
  - Spotify: danceability, valence and other features
- **Linear command queueing vs Concurrent execution on threads**
  - Linear commands: Code simplicity
  - Concurrent execution: simple workflow + enhanced functionality
- **Deterministic vs probabilistic based elimination**
  - Deterministic: less variation + more hardcoded mappings
  - Probabilistic: creates unique light shows for the same audio
- **Better extraction of audio features vs real time chunk processing**
  - Real time: information generated on the fly with user streaming, look ahead is difficult
  - Pre-processed: Enables lookahead in song to anticipate song changes, but initial delay

System Performance and Metrics

Subsystem	Performance metric
Gigbar or Lighting Engine	Blackout, SetColor (RGB value), Fade, Rotate, Strobe, ColorCycle, Hold 100% functionality implemented - Channels used: 4 channels per light, 24 channels overall - Range: 0-255 for R, G, B, UV, and Rotation
Shazam-Spotify	5 second based song recognition with minimal latency Concurrent processes: 1. Shazam : 2 seconds + Spotify: 2 seconds 2. Signal Processing: ~7 seconds Singular songs, and medley songs (2 or 3 songs in a single audio file)
Signal Processing	70%-88% accuracy on Rhythm Extractor library, Essentia. High hand-inspected accuracy for energy detection where energy levels lined up with different parts of the song Pitch and amplitude detection lined up with test files created in Ableton
User Interface	Capacity to upload, search or record an audio file for synthesis ~2 second Ajax calls to dynamically refresh the page and update recommendations from Spotify to enhance user experience

Final Gantt Chart



Conclusions

- **Concurrent control** of different lights with user input is difficult
- **Mapping** signal processing parameters to lighting outputs is **subjective** and requires **aesthetic decisions**
    - Artist input and **feedback necessary** for custom shows
- Integration and testing breaks stuff – always **over-allocate time**
- **Teamwork** and **risk analysis** critical to the tremendous progress we made over the last **13 weeks**