Team A5 - *Hit it!* - Use Case

Hit It! is a drum-based rhythm game that aims to serve as a middle-ground between other rhythm gaming alternatives.

The Rhythm gaming market:

- Few rhythm games involve hardware
- Limited demographic appeal

Hit it! solves these issues:

- Small and portable hardware
- User inputted songs

ECE Areas:

- Hardware Systems
- Software Systems
- Signals and Systems

Use Case Requirements

Portable

Should be small enough to fit within an average backpack (<15 Liters)

Plays User's Music

Hit It! should be capable of generating beatmaps from user provided songs

Accurate Beatmapping

Generated beatmaps should align well with (>80% accurate) external beatmap software or our hand-calculated beatmaps

Latency

Latency between player input and game reaction should be < 70 ms

Ease of Use

Setup should be quick and easy (<1 min)

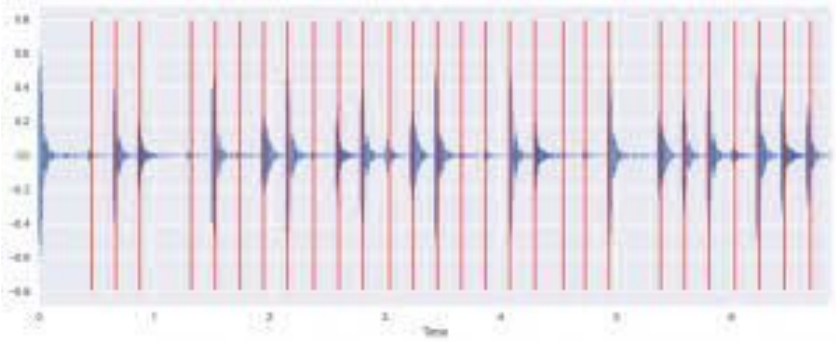
Input Recognition

The Drums used for player input should recognize the overwhelming majority (>99%) of hits by the player

Technical Challenges & Solution Approach - Software

Beat Tracking Timing

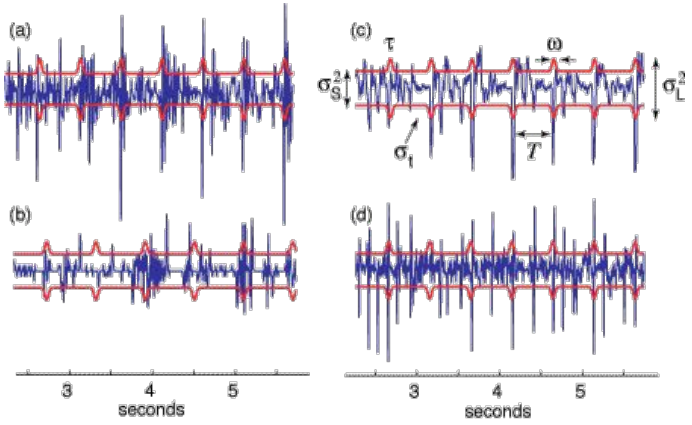
- Goal: Length of Audio Clip
- Challenges:
  - Slow Loop-Based Analysis of FFT
  - Decoding Noisy Audio Clips
  - Audio Source Separation
- Solutions:
  - Dynamic Programming to Limit Loops
  - Restricted User Inputs
  - Pattern Recognition
- Mitigation: Pre-determined beatmaps, Restricting User Inputs



Technical Challenges & Solution Approach - Software

Beat Tracking Accuracy

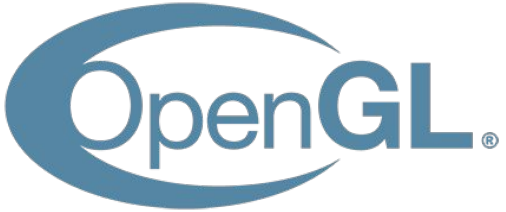
- Goal: 80% Accuracy between Externally Sourced Beat Maps and Auto-Generation
- Challenges:
  - False Detection of Other Noises
  - False Negatives
  - Changing Pacing of Audio Files
- Solutions:
  - ML with a pretrained model
  - Restricted User Inputs to Consistent Beats Used
  - Pattern Recognition
- Mitigation: Single Sound Audio Files Used



Technical Challenges & Solution Approach - Software

Aesthetics (GUI)

- Goals:
  - GUI updates at rate of 30 FPS.
- Challenges:
  - All GUI updating must occur within 33.3 ms
  - We aren't using a dedicated Game Engine
- Solutions:
  - 33.3 ms is a lax threshold to meet
  - Use of C++ and OpenGL
- Mitigation:
  - 24 FPS is the absolute lowest tolerable FPS
  - Usage of pre-created game structure to improve graphic development



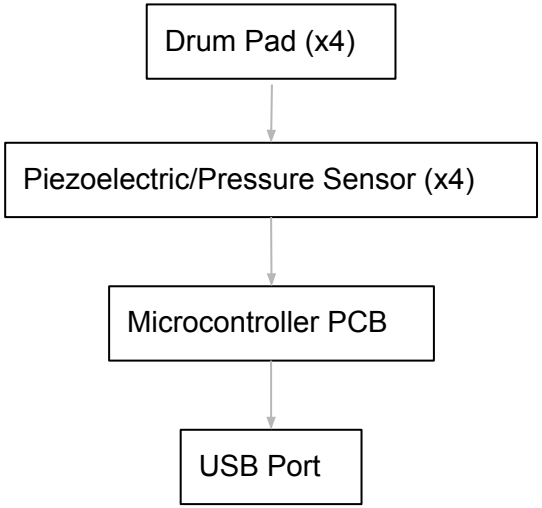
Testing and Verification - Software

Area	Metric(s)	Test
Beat Tracking Timing	Time for Creation of Beat Mapping is Length of Song (at most)	Use Python to track code execution time
Beat Tracking Accuracy	Generated beatmaps are more than 80% Accurate to testing/sample beatmaps	Two Tests: ● Beat overlap: Expanding each beat a finite amount in both directions, compare total overlap between maps ● Beat closeness: Instantaneous beat markers must be within ½ the song's average beat period (1/bpm) so that each beat is closer to its origin beat than any other beat.
Aesthetics (GUI)	> 30 FPS. Non-geometric graphics	FPS will be hardcoded. Software can measure FPS.

Technical Challenges & Solution Approach - Hardware

Latency

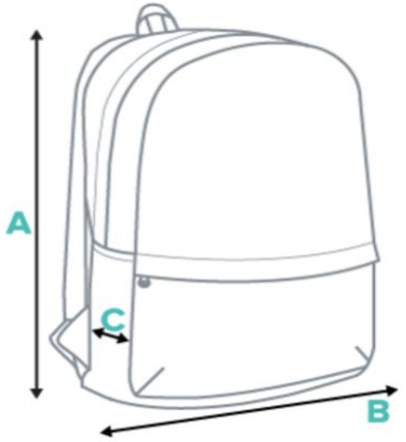
- Goal: 70 ms (Overall) and 37 ms (Hardware Only)
- Challenges:
  - Pressure Sensor Lag
  - Slow GPIO Detection
  - Slow Microcontroller Code
- Solutions:
  - Fast Sensor Response Time (< 5 usec)
  - ESP32 has 300 ns GPIO turning speed
  - High-Speed Clock (up to 240 MHz)
- Mitigation: Lower Requirement of 100 ms for MVP



Technical Challenges & Solution Approach - Hardware

Portability/Compactness

- Goal: 15 Liters = 15,000 cm^3
- Challenges:
  - Interconnected components may reduce freedom of movement
  - Wide spacing between drum pads
- Solutions:
  - Modularized Drum Pads can be unplugged
  - Wires reduce required surface area when storing
- Mitigation: Flat Drumming Pads, ESP32 has wireless capabilities



	US sizes (In)		Metric sizes (Cm)
A	16 7/8	A	43
B	13 3/8	B	34
C	4	C	10

Technical Challenges & Solution Approach - Hardware

Drum Recognition Accuracy

- Goal: >99% Hit Detection
- Challenges:
  - Large Pad Area to Small Sensor Area
  - Sensor Reliability
- Solutions:
  - Design of Drum Pads can mitigate issue
  - Our preferred sensor has high reliability (< 3% deviation)
- Mitigation: Use of buttons rather than sensors

Technical Challenges & Solution Approach - Hardware

Ease of Setup

- Goal: < 1 minute setup time
- Challenges:
  - Boot times of hardware components (microcontroller)
  - Many separate hardware components
- Solutions:
  - ESP32 can run programs on startup
  - Latch Connectors will simplify connections
- Mitigation: ESP32 has full wireless functionality (BLE & WiFi)

Testing, Verification, and Metrics - Hardware

Area	Metric(s)	Test
Latency	37 ms (Overall): 1 ms for Sensor, 1 ms for GPIO turn time, and 35 ms for Code	Probing via Oscilloscope
Portability/Compactness	15,000 cm^3 (Overall): 12 cm x 12 cm x 4 cm for each modular component (5 total, 2880 cm^3 total)	Physical Dimensions (Ruler) Practical Test (Backpack)
Drum Recognition Accuracy	>99% Recognition Rate	Repeated Trials
Ease of Setup	< 1 minute Setup Time (Overall): No more than 12 seconds per module (5 total)	Repeated Trials (Using random participants)

Schedule & Division of Labor