BeatBeam MDR
SDP15

Duncan Smith-Freedman, EE
Brandon Sprague, CSE
Brian Hickey, CSE
Daniel Bond, CSE

BeatBeam Team

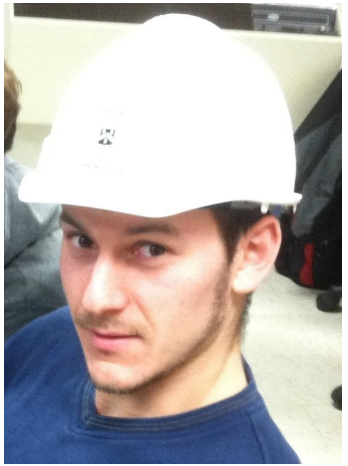
Daniel Bond
(Fearless) Team Leader

Board Interpretation, Musical Implementation

Brian Hickey

Networking, Wireless Connectivity

Duncan Smith-Freedman
Audio-to-Light Interpreter, Physical Design, Power



Brandon Sprague
Web Server, Web Application Desktop/Mobile Interfaces

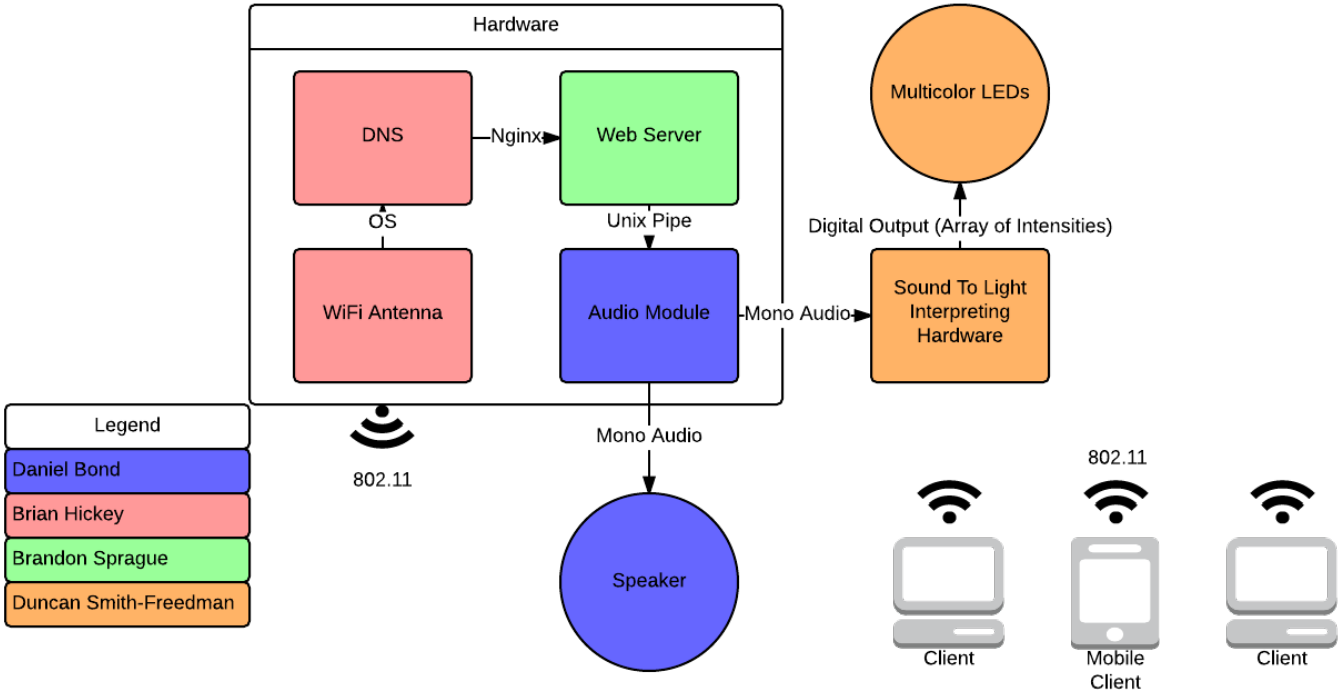
BeatBeam - Easy Music Creation for Amateurs

- High difficulty level with traditional methods of creating music - lessons and instruments take time and money that people don't have
- Other methods of music creation (i.e. rhythm games, GarageBand) still have creativity/complexity tradeoffs
- BeatBeam aims to be creative, easy, and fun

Reminder of System Requirements

- Users with no prior musical experience will be able to make pleasing music more than 90% of the time
- Groups of at least 20 people will be able to concurrently create music
- <25 ms delay for syncing game state across clients

System Block Diagram (with interfaces defined)



Gameboard Interpreter

Requirements

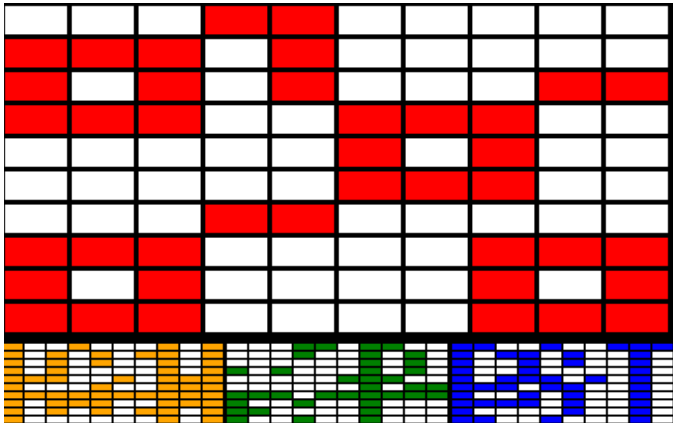
- Quickly processes frames of user input into desired music changes
- No “drastic” music alterations from minor grid changes (Prof. Salthouse)

Challenges

- Reproducible music from the same grid state
- Balancing room for creativity with protection against bad music

Solutions

- Reproducible music: allow first column to set static key of upcoming music sequence
- To limit complexity: additional columns are interpreted based on root note/chord (in progress)



Example Interface

Music Generator

Requirements

- Music must sound good to the user ~90% of the time
- Latency between receiving grid state and generating music must be low

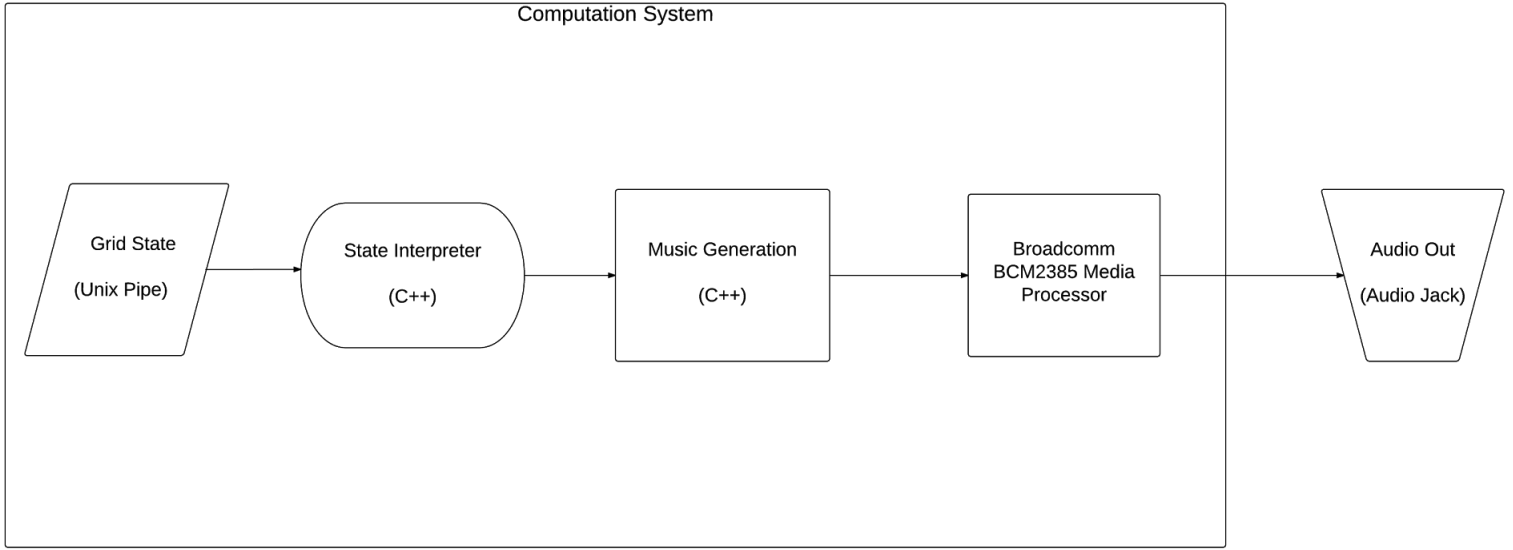
Challenges

- VST plugins more compute-intensive than desirable on the Pi
- Melopy Python library useful, but uses .wav files
- Still need close to real-time audio generation
- Writing to audio files and reading is very, very slow (unacceptable)

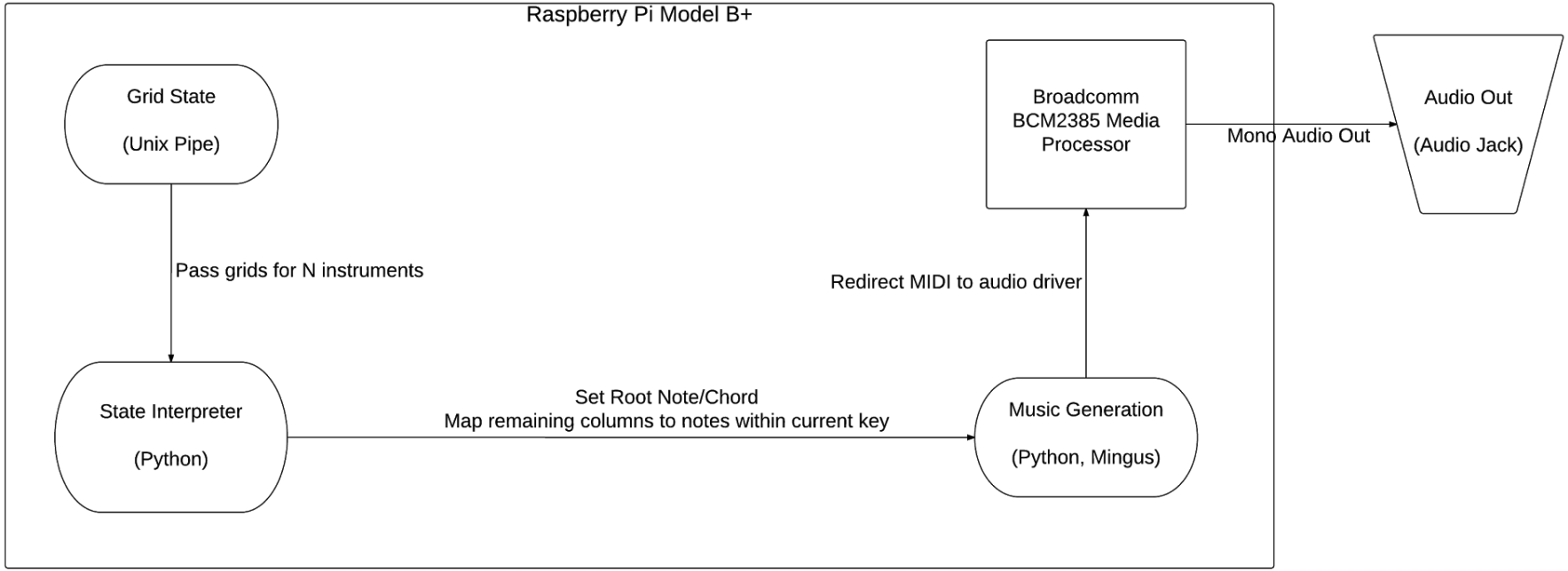
Solutions

- Use library with light-weight MIDI support (in progress)

Music Generator/Gameboard Block Diagrams (Before)



Music Generator/Gameboard Block Diagrams (After)



Audio-to-Light Interpreter

Requirements

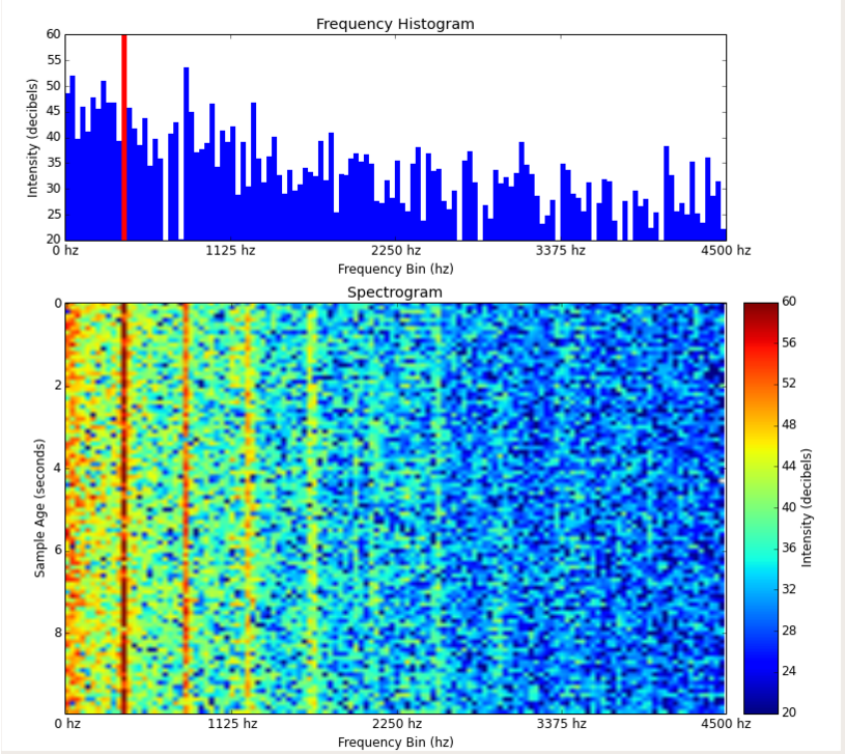
- Take FFT of audio signal and map frequency bins to LEDS
- Correctly average FFT size and correlate to LED intensity

Challenges

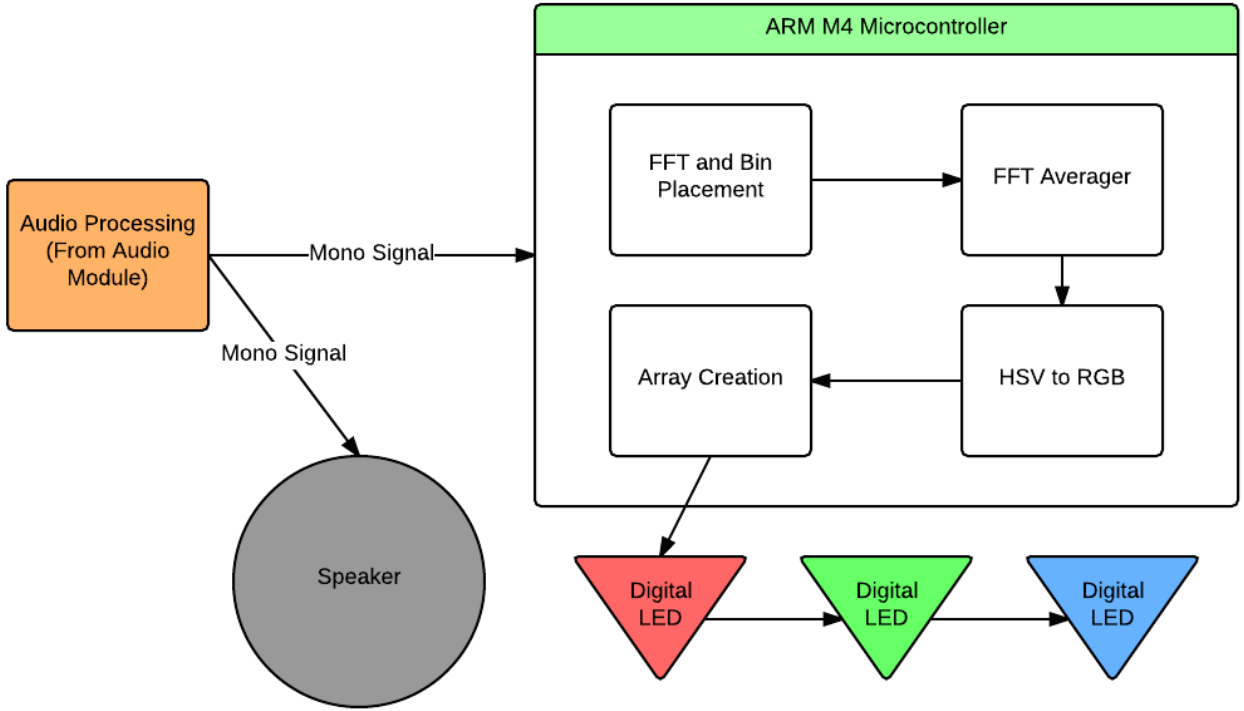
- Power output from microcontroller is too low to power LED array
- Powering system in final implementation

Solutions

- Use constant current LEDs with digital input



Audio-to-Light Interpreter

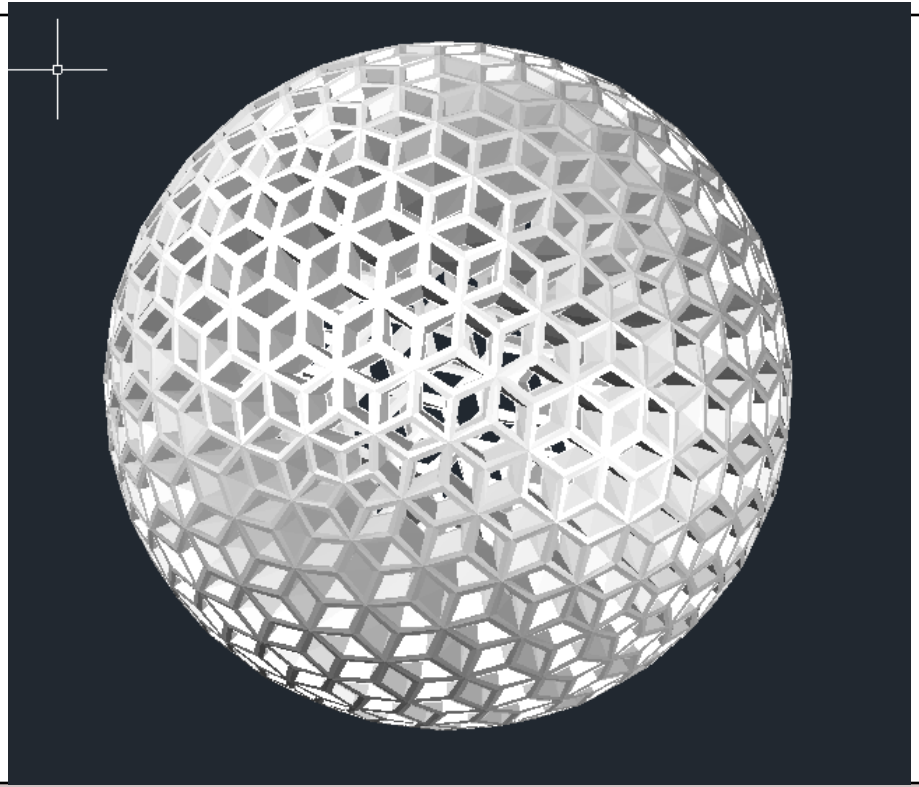


Central Node

Requirements
- Size to fit and secure Pi, microcontroller, and speaker
- Gaps on side for LED display
- 3D printable

Challenges
- Finding ideal design shape
- Durable
- Being able to line walls with LEDs

Solutions
- Cylindrical shape with screw caps
- Formiga P110 Printer



Network Interface

Requirements

- Central node that operates as access point (802.11 infrastructure mode)
- DHCP server assigns network addresses and static DNS address to clients
- DNS zone configured to redirect all web requests to central node
- Web server running on central node that hosts web interface for BeatBeam

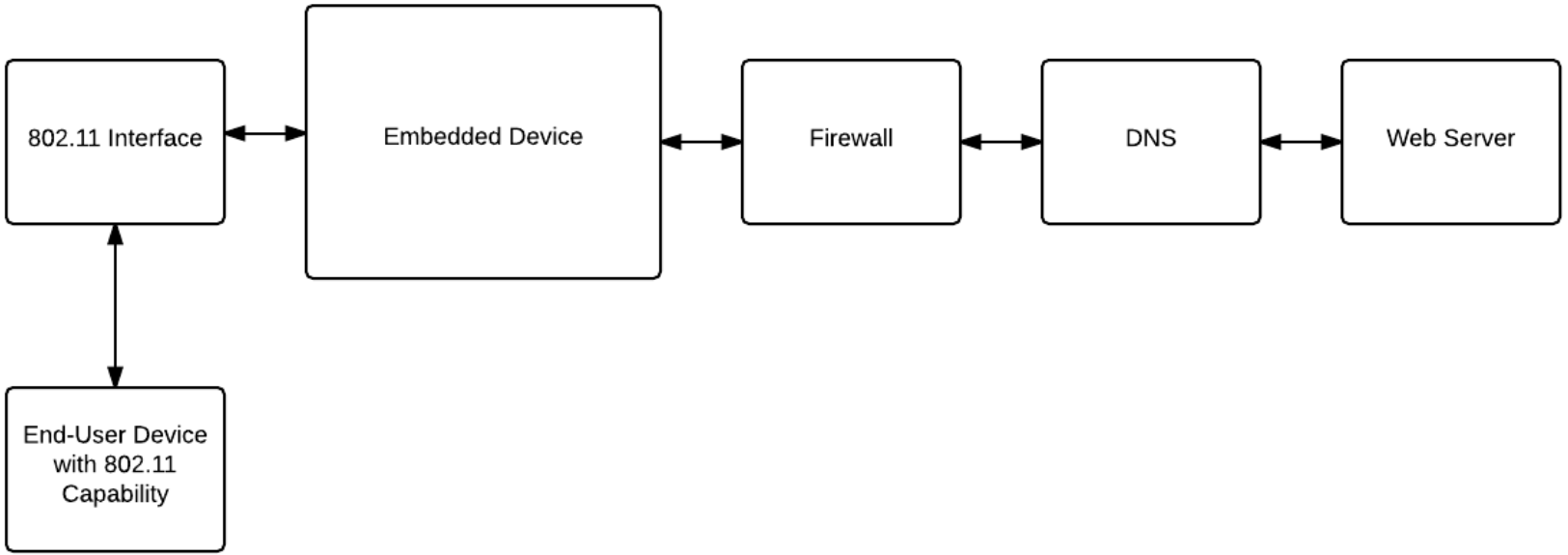
Challenges

- Driver issues involving Realtek hardware
- Access point service not starting correctly
- Proper DNS routing

Solutions

- Access point service currently being started automatically
- Chosen driver doesn't exactly match hardware but works nonetheless

Networking Component - Block Diagram

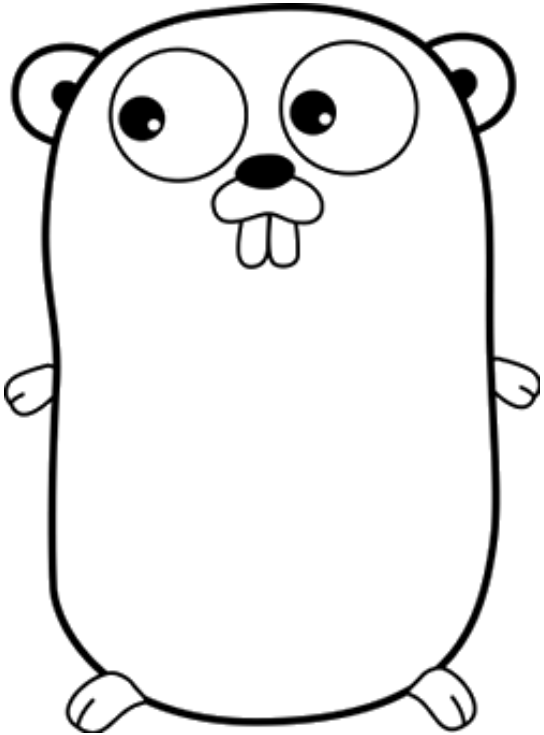


Web Server

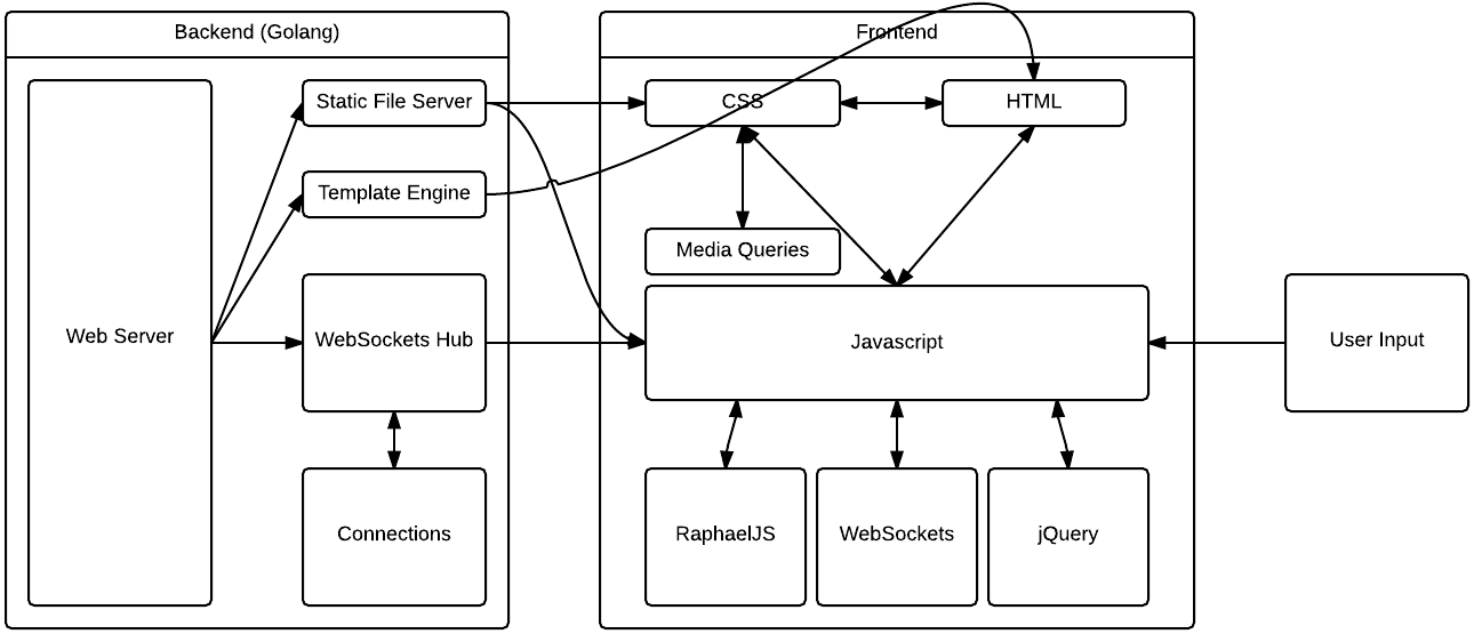
Requirements
- Low latency
- Serve web interface to clients

Challenges
- Synchronizing the game state across the clients
- Mobile-friendly site, quick and responsive on all form-factors

Solutions
- Using low-level synchronization primitives built into Go
- Use media queries and a responsive web framework, as well as CSS3 animations



Web Server Block Diagram



Team Responsibilities and Schedule

- Brian
  - Securing of the network connection and server(s) (Jan 15)
  - Load balancing scheme for multiple clients (Feb 5)
  - Latency and throughput testing (after data types are defined) (Feb 5)

- Duncan
  - Designing and printing central node (Jan 1, availability of printer)
  - Creating LED array (Jan 10)
  - Power Plan (Jan 20) and power implementation (Feb 3)

Team Responsibilities and Schedule

- Danny
  - Redesigned Music Generator leveraging Python and Mingus package (Jan 7)
  - Change Gameboard Interpreter to view first column as root note/chord (Dec 29)
  - Add support for Multi-instrument mode to Music Generator (Feb 20)
- Brandon
  - Completing Multi-instrument grid functionality (Jan 7)
  - Further integrated mobile functionality (gesture-based control) (Feb 20)

Individual Demonstrations

Questions?