MIDI Interface Block
This block will interpret MIDI output signals from an electric keyboard and convert those serial data packets into a (parallelized) custom data format. It will propagate “Note on” and “Note off” signals into the system in this custom format, which will include an integer identifier for the key pressed or unpressed. Its functionality can be demonstrated by pressing keys on the electric keyboard and displaying the block’s output on the hex display. The musical scope is currently limited to a single octave.

Action Interpretation Block
This block will combine internal note data (i.e. the notes that should be played) with user input. It will thus determine (and output) data blocks that encode “events”. The events will be “note not played”, “note played correctly”, “errant note played”, “note released prematurely”, and possibly more as part of my stretch goals. These events will be passed back into the main game logic. The logic of this block will allow an appropriate tolerance for early or late key presses, since exactly timed presses are impossible for a user to perform. Its functionality can be demonstrated by outputting event encodings on the hex display.

Game Logic Block
This block will read pre-loaded note data from RAM and provide note information to the other modules at the times the receiving modules require (i.e. a few beats ahead for the action interpreter, many beats ahead for the display block). It will also receive event information from the action interpretation block and thereby create a score. Its functionality can be demonstrated by outputting the most recently read block of notes on the hex display, and by outputting the current score on the hex display. Again, only one octave will be considered.

Display Block
This block will create a “Guitar Hero” like experience, with notes streaming from the top of the screen to the bottom. There will be a clearly marked horizontal bar (or similarly indicated screen location) that indicates to the user that a note should be played. The notes will likely be relatively simple shapes (as in Guitar Hero), and will not necessarily be images stored in memory. The display block will also show the user his/her current score. It will furthermore provide the user with visual feedback about which notes they are pressing (Guitar Hero does something similar, by highlighting the keys that are pressed). This feedback will be in the form of highlighting or some sort of color change on the screen. Its functionality can be demonstrated via visual inspection on the computer monitor. Again, only one octave will be considered.

Stretch Goals
-Enable all modules to handle a keyboard multiple octaves wide.
-Build a “teaching” mode, in which a song can be played on the keyboard, and the song data will be interpreted and loaded into RAM. A “learner” user can subsequently play the game using this user-loaded song data.