Project Miku

Junha Choi 301291870

Yuhui Hu 301278249

Zhiheng Zhang 301277885

Junyuan Tan 301251535

Overview

Our system allows users to play a music rhythm game called “Miku”. The system takes audio input through BeagleBone Green, analyses audio data, generates a pattern in the 16x32 LED matrix and plays the audio through Beaglebone Green audio output at the same time. From the user’s point of view, the user plugs an audio player(phone) into Beaglebone Green, listen to Beaglebone Green audio output and play rhythm game which is displayed on the LED matrix by interacting with Xbox controller. If a player successfully clicks the corrected button when a bar reaches the bottom, the player will get one point. If the player misses a beat or misclick a button, the number of lives will decrease. When the lives drops to 0, the game is over.

Photo1. Game demo photo(without audio plugin)

Photo 2 & 3. Xbox Controller and LED display are mapped by color

Photo 4. Xbox Controller button to start a new round of the game

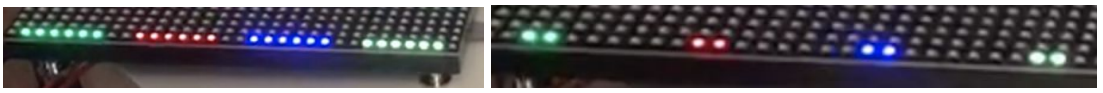
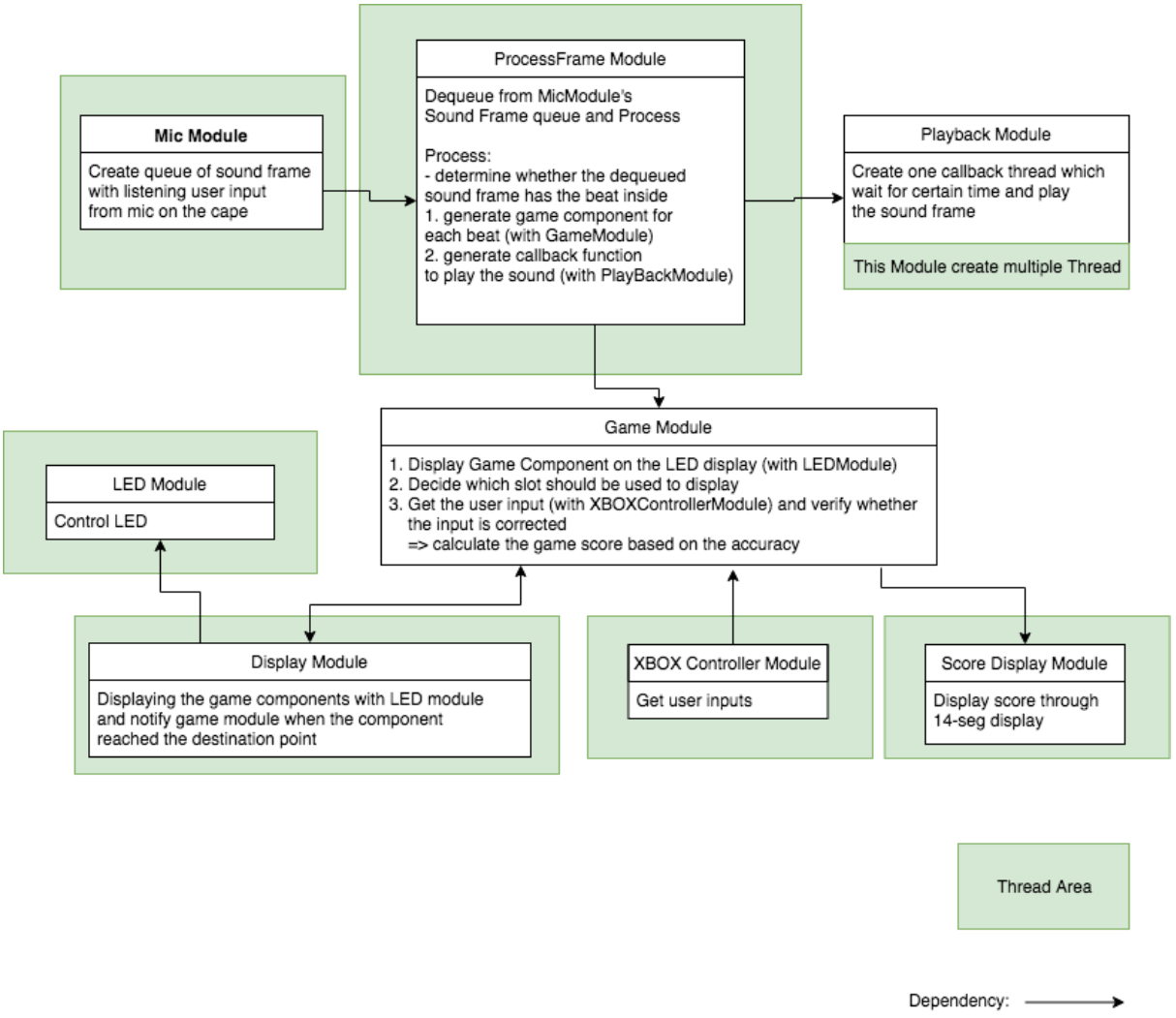


Photo 5. The width of LED bars indicates a user's life (number of chances to make mistakes)

Module Diagram



Module Descriptions

- Audio recorder

Audio recorder module uses ALSA libraries and receives wave signals from the 3.5mm audio input jack. It puts a data into a queue.

-PlayBack Module

Playback module creates a callback function which plays a given sound frame after a certain time of delay through 3.5mm audio output jack.

- ProcessFrame Module

ProcessFrame module dequeue from Audio recorder module's sound frame queue. With that dequeued frame, it produces callback function which plays the sound frame with Playback module and analyzes the sound frame to find whether there is a beat inside the sound frame with Aubio library. If it turns out there is, the module creates a game component with Game module. Otherwise, the module does not create.

- Game Module

Game module listens to ProcessFrame Module. There are 4 slots which store time. Whenever Game module receives a request to create a game component from ProcessFrame module, it will search for an empty slot and if there is, it will set that empty slot's time to the current time and ask Display module to display the game component.

Codes for this process:

```
for (int i = 0; i < XBOX_NUM_TYPE; i++){
      if (currentSlot == XBOX_NUM_TYPE) currentSlot = 0; // start all over from left hand side
      if (beatQueue[currentSlot] + delayTime < currentTime){
          // overdue, see it as an empty slot
            beatQueue[currentSlot] = currentTime;
            timeoutQueue[currentSlot] = true;

          // call led dropping
            Display_generateComponent(currentSlot);
            break;
       }
       currentSlot++;
}
```

When the game module receives input from XBOX Controller module, it will verify whether the current time is within the react time interval. If it is, the score increases. Otherwise, the number of lives decreases. If the number of lives drop down to 0, Game will be Over.

When the score changes, Game module calls Score Display module to display the score on 14-Seg display on BBG cape.

When the game is over and the module receives restart command from XBOX Controller, the module will start another round of the game without restarting the whole program. It uses mutex to lock the beat queue before any thread accesses it. This ensures thread safe.

- Display & LED Modules

LED module controls a 16x32 LED matrix with GPIO. Display module is in charge of managing game components which is being requested to display on the LED matrix. When Game module requests Display module to create a game component for displaying, Display module creates the game component and put it in a slot if there is an available slot.

Display module displays components in slots using LED module with updating slots ' information for displaying. When a game component reaches the destination (the bottom of the 16x32 LED matrix, it clears the slot which the component was at and informs Game module that it cleared the game component from the slots.

- XBOX Controller Module:

This module uses XBOX Linux driver. It keeps checking controller events. If users press a valid button, this module passes the input to Game module.

- Score Display Module:

When the score changes, Game module calls Score Display module to show the score on 14-Seg display on BBG.

Feature Table

description	completeness [1- 5]	code	notes
Audio input through the microphone jack	5	C (10%) C++ (90%)	We made use of ALSA library. (30% sample code, 70% us)
Audio output through the audio jack	5	C (10%) C++ (90%)	We made use of ALSA library. (30% sample code, 70% us)

Beat detection	3	C (80%) C++(20%)	We wrap aubio library so that we can use in C. (It is not working perfectly because we need a way to filter noise) (35% sample code, 65% us)
Led bar drops when there is a beat	5	C	We made use of GPIO. (referred Brian Fraser's code from previous courses)
Led life indicator to indicate how many chances user can make mistake.	5	C	The width of LED bars at the bottom represents the number of the left lives.
Score displaying by 14 segment display	5	C	We made use of I2C and GPIO (code used from assignment2)
Xbox controller input	5	C	We made use of Xbox Linux driver (10% sample code, 90% us)
Restart game	5	C	When the game is over, users can click a button on the controller to restart the game without restarting the entire program

Extra Hardware & Software

Hardware

1. LED matrix
2. Xbox One controller
    - the controller requires Xbox controller Linux driver
3. Audio source (phone, mp3)
4. Speaker and headphone

Software

1. ALSA library
2. Aubio library (https://aubio.org/)
3. Xbox controller driver

Challenges

A huge amount of time was spent unexpectedly (in order)

1. **Module design**

   Since it is a real-time system, we needed a good module design to allow modules to communicate in an efficient way. We tried our best to reduce dependency and coupling between modules. Each module works independently. Game module works like a controller and delivers information between modules.

   The system needs a way to ensure synchronization among all threads. We use the system time as the timestamp.

2. **Driver/library set up**
   - Compile library into .so
   - Choose between .so file for arm-linux or linux

3. **Use C++ libraries in C**

   Steps are included in How-To-Guide.

4. **Usage of beat detection library without understanding beat detection algorithm**
   - Performane issue in beat detection

Acknowledgments

1. Xbox controller linux driver setup guide
   https://github.com/Ostantric/xbox_controller_linux_driver

2. 16x32 LED matrix
   https://www.cs.sfu.ca/CourseCentral/433/bfraser/other/2015-student-howtos/Adafruit16x32LEDMatrixGuideForBBB-Code/test_ledMatrix.c

3. Aubio library's github repository
   https://github.com/aubio/aubio