CMPT 433 SPRING 2022
FINAL PROJECT
Audio Player

Contributors: Abay Kulamkadyr, Tarvinder Dhaliwal, Jaskaran Dhanoa, Khushwant Parmar

System Explanation:

Our system is an audio player implementation. It allows us to play audio files through the Beaglebone's audio output jack and manipulates the wav files by adding additional wave files on top.

Some features enabled by our system are a follows:

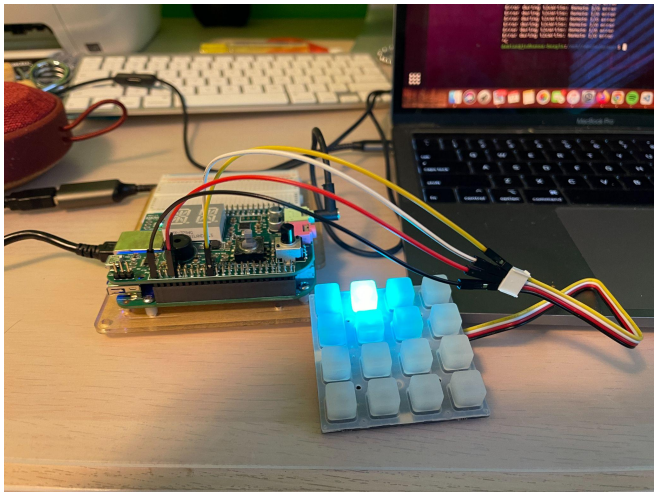
- playing wav files
- skipping through files using joystick and accelerometer
- Reversing and Forwarding through tracks
- Pause/Unpause feature
- Forwarding/Reversing while wav file is paused/playing
- control volume using the potentiometer
- display audio wav file progress on the 14 seg display
- Using embedded node server to control the features on the Beaglebone
- Playing/Pausing/Forwarding/Reversing buttons rendered through web server
- Displaying track currently being played on terminal
- Allow audio recording using a mic
- Displaying leds animation on the 4 x 4 matrix led display

How does it do it?

- playing audio files:
    - We use alsalib to directly interact with the hardware and write files to the audio buffer.
    - We separately push the data into 2 channels if it is stereo audio and 1 channel if it is mono audio
    - The playback audio is added to the audio buffer with a Direct Write
    - Background thread runs that plays sound tracks continuously.
- Parsing the wav files:
    - We used the dr_wav library(work of David Reid, https://github.com/mackron/dr_libs)
    - We use it to format the wav file into 32 bit signed int as our player uses that format
    - We created a data structure to hold the metadata for the samples and the wav file
    - We use 2 pointers to iterate through the samples if it is 2 channel audio and 1 pointer from the left channel if it is 1 channel audio and convert the file to the required format
- skipping through files using joystick and accelerometer:
    - Moving the Joystick left would go one track back, by accessing the list of tracks array back by one index
    - Moving the Joystick right would go one track forward, by accessing the list of tracks array forward by one index
    - By tilting the beagle bone in the x direction using the negative values it would go one track forward

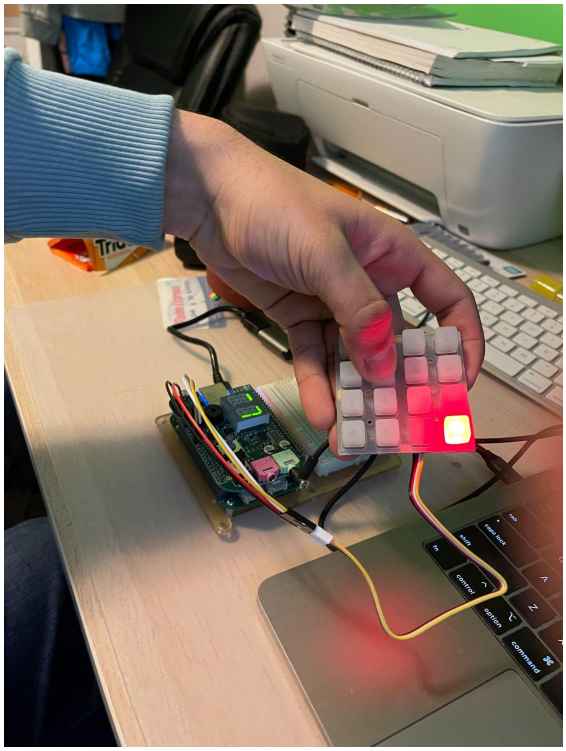
    - By tilting the beagle bone in the x direction using the positive values would go one track back
- control volume using the potentiometer:
    - Gets readings from the potentiometer using a thread
    - Converts these values to normalized voltage values
    - Maps the voltage values to PCM volume values from 0-100 and sets the volume
- display audio file progress on the 14 seg display:
    - It is used to display the current mode of the player(play, pause…)
    - It displays the progress through the track by lighting up segments as the track proceeds
- Using embedded node server to control the features on the Beaglebone:
    - It uses a js frontend file which takes in user interaction on the webpage.
    - This feeds information into the backend js server via a websocket.
    - The backend js server connects to the actual c code file via a UDP socket.
    - The server then emits messages through the socket to the c file which accepts udp packets and then executes the actual functions.

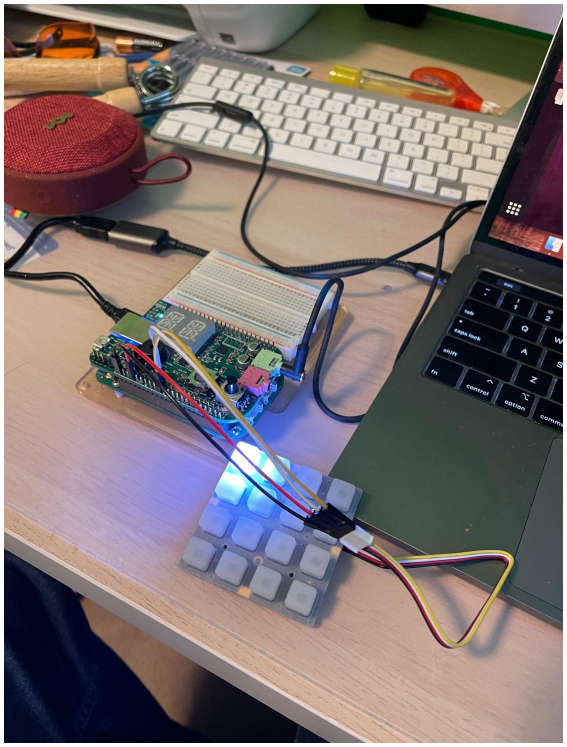
- 4 x 4 matrix led display animation and sound:
    - It is connected to the beaglebone on i2c2 bus using the following connection to the pins on the P9 header ·
        - GND pin to P9_1
        - VIN pin to P9_7
        - SCL pin to P9_19
        - SDA pin to P9_20
    - For writing values and reading values it uses the i2c registers on the Seesaw chip on the LED matrix.
    - The LEDs on the matrix are controlled by the registers under the module NeoPixel, specifically PIN, BUF_LENGTH, BUF, SHOW.
    - The buttons on the keypad are controlled by registers under the module Keypad, specifically KeypadEvent, KeypadEventCount, KeypadFifo.
    - Reading and writing values to the registers under the NeoPixel and Keypad modules lets the user light up LEDs when buttons are pressed.
    - We encoded a sound(wav file) and a color (RGB) to each of the 15 buttons on the matrix (16th button is for terminating the program, lights up red).
    - We created an API for the matrix that allows us to show ripple effects on the matrix and play sounds with different buttons.

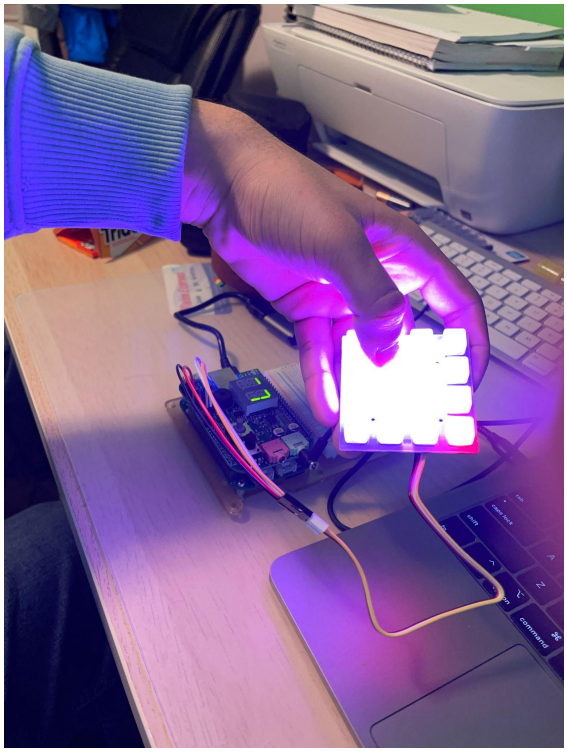


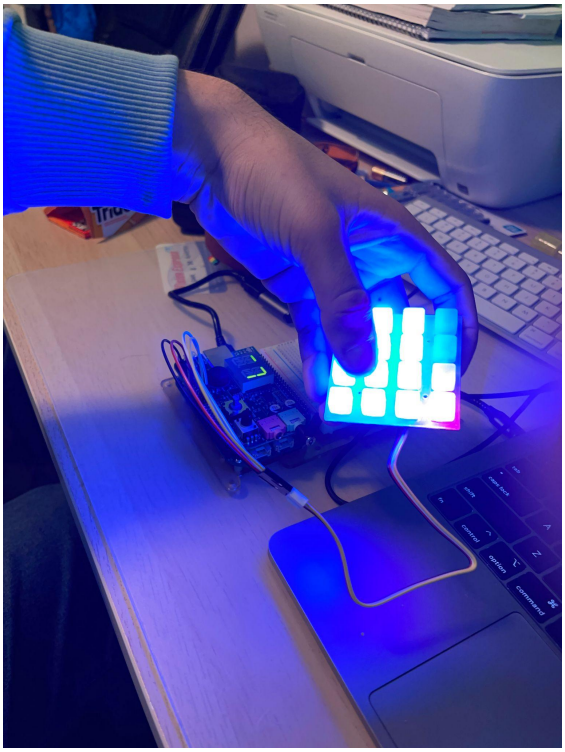
Total Circuit set up including Beaglebone, 4 x 4 LED matrix and output to a speaker.
This video showcases the LED matrix and the total system in action.
https://youtube.com/shorts/rrmhYxag-gM?feature=share

These pictures show the user interacting with the 4 x 4 LED matrix.
The red button is the termination button, by pressing that the user can end the program.
Each button on the LED matrix has a different color and wav file encoded to it.
While the program is executing, if a button is pressed it lights up the whole matrix with a specific color using a ripple effect and the particular encoded sound is mixed on top of the already playing track.









Not working features if any give info:
One of the most important hardware features we desired to implement was the Electret Microphone Amplifier. Sadly, none of the group members were able to use the microphone in any way which we intended to originally. The issues we faced were due to the values taken from the microphone that couldn't be loaded to a wav file. This was an important task, since our entire tracklist was played as wav files from the main thread.

Feature Table:

Description	*Host/Target*	*Comp*	*Code*	*Author(s)*	*Notes*
Web page	*T*	*5*	*JS / HTML,NodeJS*	*Khushwant, Jaskaran*	*Allows to control all project features from web server*
Joystick	*T*	*5*	*C*	*Tarvinder*	*Switches tracks, plays/pauses tracks, Reversing/Forwarding tracks*
Mic	*T*	*0*	*C*	*Jaskaran*	*Records audio, but could not work.*
Potentiometer	*T*	*5*	*C*	*Abay*	*Controls Volume*
Playback Audio through audio buffer with Direct Write	*T*	*5*	*C*	*Abay*	*Allows the user to play wav files via the Zencape's audio output jack*

Parser	***T***	***5***	***C***	***Abay***	***Enables to play any wav file with different metadata***
14 seg display	***T***	***5***	***C***	***Tarvinder***	***Display current mode of audio player (play, record) and audio file progress***
Accelerometer	***T***	***5***	***C***	***Abay***	***Enables changing tracks by tilting the board side to side***
4 x 4 LED matrix	***T***	***5***	***C***	***Abay***	***Pressing buttons plays sounds and plays RGB animation***

Extra Hardware Used:
4x4 Button with integrated LED matrix (with silicone keypad)

Extra Software Used:
For parsing we use a library **dr_wav.h** adopted from:
David Reid - mackron@gmail.com
GitHub: https://github.com/mackron/dr_libs
WAV audio loader and writer. Choice of public domain or MIT-0.
dr_wav - v0.13.5 - 2022-01-26
Under Public Domain License, Copyright 2020 David Reid.

Alsa sound library code:
https://www.alsa-project.org/alsa-doc/alsa-lib/_2test_2pcm_8c-example.html

AdaFruit Neotrellis 4x4 LED Board previous students' guide:https://opencoursehub.cs.sfu.ca/bfraser/grav-cms/cmpt433/links/files/2019-student-howtos/AdafruitNeoTrellis4x4LedButton.pdf

References:

Alsa sound library code:
https://www.alsa-project.org/alsa-doc/alsa-lib/_2test_2pcm_8c-example.html

AdaFruit Neotrellis 4x4 LED Board previous students' guide:
https://opencoursehub.cs.sfu.ca/bfraser/grav-cms/cmpt433/links/files/2019-student-howtos/AdafruitNeoTrellis4x4LedButton.pdf

Wav parsing library code:
https://github.com/mackron/dr_libs/blob/master/dr_wav.h

Sample wav files:
https://www.noiiz.com/sounds/samples/78800
https://www.noiiz.com/sounds/packs/1631
https://www.bensound.com
https://www.freesounds.info/production-music/feel-good-background-music/
https://www2.cs.uic.edu/~i101/SoundFiles/