Team 24 Glorified Music Player:

Karaoke Machine using a MSP 432

Arley Trujillo
ENG CE 450 Microprocessors
Professor Giles
May 6th , 2017

Introduction

The name of my project was: Glorified Music Player. The concept behind it was to take the things Spotify does, play music, play random songs, fast forward, etc. and implement it in the MSP with my own twist. My twist was to also make a karaoke machine, where the lyrics of songs would play while the MSP is producing sounds. In one of my initial reports, I wrote that I wanted to implement a moving background image as well as display words, I ended up scrapping this idea. Therefore, the components of my final project ended up being: a music player with three songs; pause, fast forward, and slow down capabilities; display lyrics; an interactive screen; and a random mode selection.

Music Player and Karaoke

To have the MSP play music, I first defined the frequencies of every note (C3 through E7, including a rest). These frequencies were found in an example problem during Homework 2. Next I had to define how long a quarter note, half note, eighth note, etc. took to play. To figure out the "perfect" timing, I just had to play around with the numbers. I ended up defining a whole note as 120000 and just worked backwards or forwards from there. The next step was to define the songs and their lyrics. Each song had to have their own Beats, Notes, and lyrics defined in global arrays. The easiest way to do this was to just print out the music sheets of each song and going note by note and inputting them into the arrays. The three songs I used were: Piano Man, The USA Anthem, and Twinkle Twinkle Little Stars. These songs were used because they have simple note playing. It would have been more difficult to play a song with multiple parts to it (like a Star Wars Song) since the MSP could only play one note at a time. Any song that would be played with multiple hands in real life would have been difficult to implement.

To actually play the notes, I used Timer A0's output and connected it to a pin, P2.7. Every time a note is played, I play a rest (a pause), I delay the clock (a break between notes), and finally play the note using TimerA0. When the Karaoke mode is selected, I loop through the lyrics right after I pass a note to TimerA. This made it so that a note is played, and a lyric is displayed on the screen. This happened simultaneously because TimerA was running a 3 MHZ as well as the MSP's internal clock (which I set it to). In my code, each song has a line of code that looks like: delay_clock (songTwoBeats[iter]/30). This is line controls how long each beat is played. If I want a note to be played for a long time, I would change 30 to a lower number (from now on I will refer to this number as the "30" value). Fast mode would be the opposite. Each song has a different "30" because each song has a different tempo. Although, the USA Anthem and the Twinkle Twinkle share similar values, Piano Man is a slower song so I had to use 13 rather than 10 in normal mode.

When Karaoke mode is selected, each lyric has to be played and erased before the next edge of the clock. This means the Display has to be cleared and displayed after each lyric. This really slows down the music player and if the "30" value isn't high enough, each note would be played at an incredibly slow pace. That is why when every song is played on Karaoke mode, the "30" value is changed to 500, to compensate for this lag. Late in my project, I realized that by increasing the frequency of the MSP to something like 48 MHZ, this would have fixed the lag issue. However, this would have required to modify my entire project late into the process, therefore I made the decision to just compensate where it needed to be.

The songs could also be paused by pressing the top button on the board. The way this works is that when the the interrupt handler is triggered, TimerA is set on mode 4, which produces a square wave, basically pausing TimerA is place. If the pause button is pressed again, the song picks up from when it left off which means TimerA is set to mode 0.

Interactive Screen

The screen has three “positions”. The first position is the home screen, which is initialized at the beginning of the code, and displays the three songs and random mode. To toggle between songs, the bottom button is pressed. When the button is pressed, a text appears at the bottom of the screen indicating which song is selected. I do not clear the screen in this instant, I just print over the text that was previously there. Whenever the button is pressed I change the screen displayed, which occurs in the display_home function.

When you want to select a song, the joystick has to be pressed. When the joystick interrupt handler is triggered, it will change the home screen that is being displayed (a global variable). It also saves your selection in a global variable, which will be used in the main function to decide which song will be played. The next screen shows the different song modes. The same selection process is used, the bottom button to toggle between modes and the joystick press to select a mode. Once this is done, the final screen pops up which will either display lyrics or just display a white screen. The last functionality of the screen, is that to return to the main menu and start over the joystick has to be pressed. This will reset all the global variables and allow you to select a new song and mode.

Random Mode

When random mode is selected, it will randomly select a number between 1 and 3 (inclusive) for the song selection and then the same for the song mode. This will take you directly to the last screen skipping the second screen entirely.

Hardware/Miscellaneous

In my project, I used the joystick, both buttons, and a light that would display when it's okay to press the next button. This was an idea I thought of late in development, and couldn't get it to work

perfectly. Some of the miscellaneous stuff that I didn't include above, consists of initializing the buttons, joystick, the timers, and the display. This is all included in my code.

Bugs

Not all projects are perfect, and mine has some bugs. Like I mentioned earlier, displaying lyrics and playing the song really slowed down my performance. This is really noticeable when listening to Piano man in Karaoke mode. Even when I try to speed up note playing, it still isn't perfect like when in normal mode. This could have been fixed by increasing the frequency of TimerA and the internal clock of the board, but I just ran out of time. Furthermore, the buttons and joystick aren't debounced perfectly. This means that sometimes when the joystick is pressed, nothing happens and you are taken to the main menu. This is why I tried to implement the light that would tell you when it's okay to press the buttons, but I couldn't get it to work perfectly. Furthermore, the display glitches out sometimes. By this I mean, the screen doesn't clear when switching to the next screen. Hence, you'll have leftover text from the previous screen. I relate this problem to the fact that the frequency I use isn't high enough.

Moving Forward

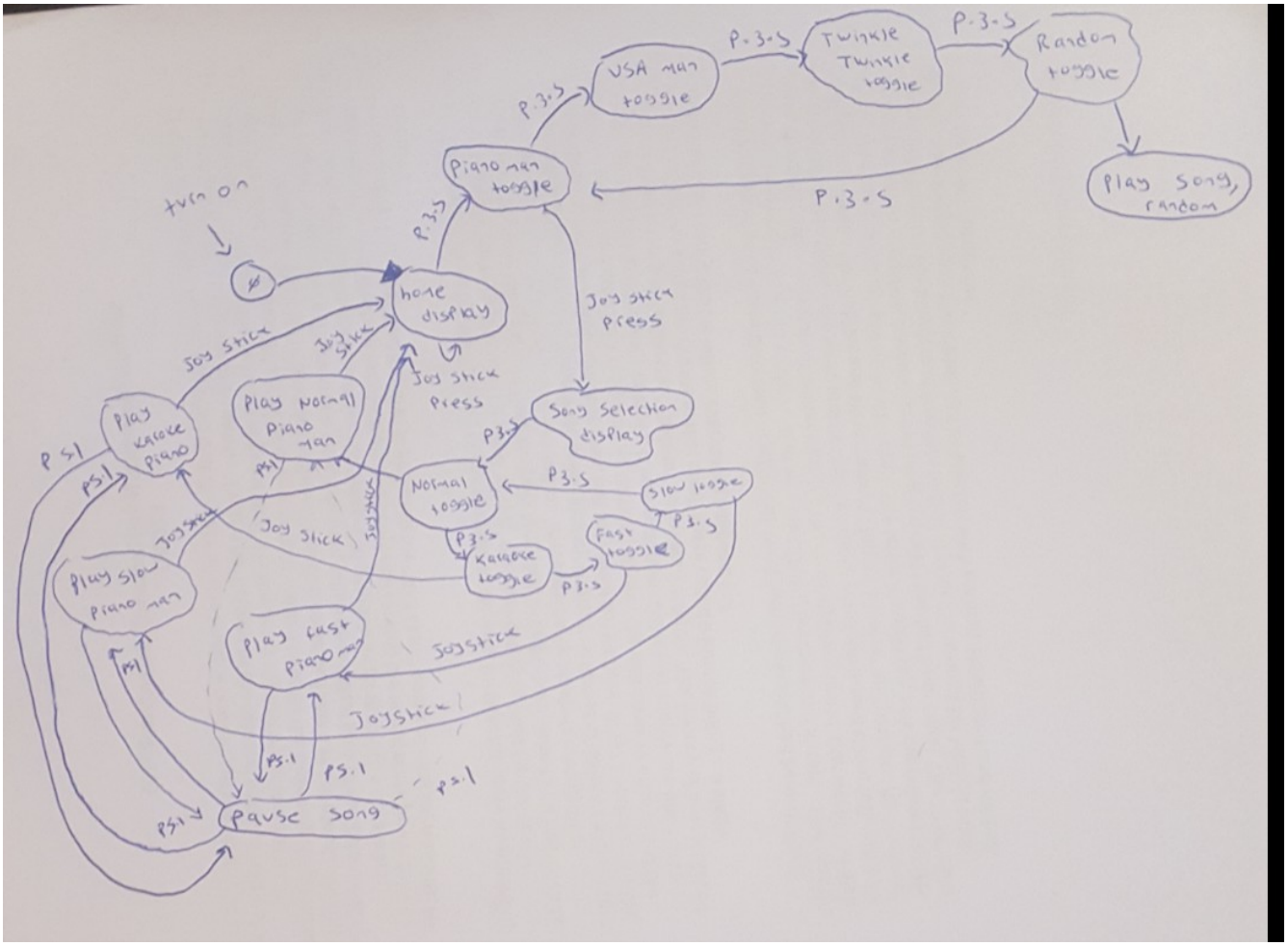
My next steps in this project would be to add more songs. Manually inputting every song was very time consuming and frustrating because there was so many components to it. Not every song is created equally so getting it just right was difficult. Expanding my song selection would be my first step moving forward. Also fixing the bugs would be a top priority. The fact that songs lag when displaying lyrics is an issue that should be addressed immediately. Besides that, everything else would just be an expansion of what I already have. Another group had the ability to make your own songs. I thought that idea was amazing and I would love to implement that into my project, but also add the ability to write your own lyrics.

Assessment of Success

I thought my project was an absolute success. I had no partner, I was worried I wouldn't be able to get anything to work a few days before demo, and I was feeling really insecure about my project when I saw other projects. However come demo day, I thought I had one of the better demonstrations/presentations. I had no bugs during the demo, I answered good questions, and I was very proud of my work. There are bugs that need to be fixed and I understand that my project was not the most difficult other there, but this doesn't take away from the hard work that I did and accomplished with very little help or knowledge.

FSM

Note that I only did the FSM for the Piano Man song. If I did it for the USA anthem and Twinkle Twinkle the image would be unreadable. The FSM for Piano man is the exact same for the other two songs. You can see the image in the next page.


turn on
home display
Piano man toggle
USA man toggle
Twinkle Twinkle toggle
Random toggle
Play song, random
P.3.5
Joy stick press
Joy stick
Play Normal Piano man
Play Karoke Piano
Play slow Piano man
Play fast Piano man
Song selection display
Normal toggle
Slow toggle
Fast toggle
Karoke toggle
Joystick
Pause song
P5.1