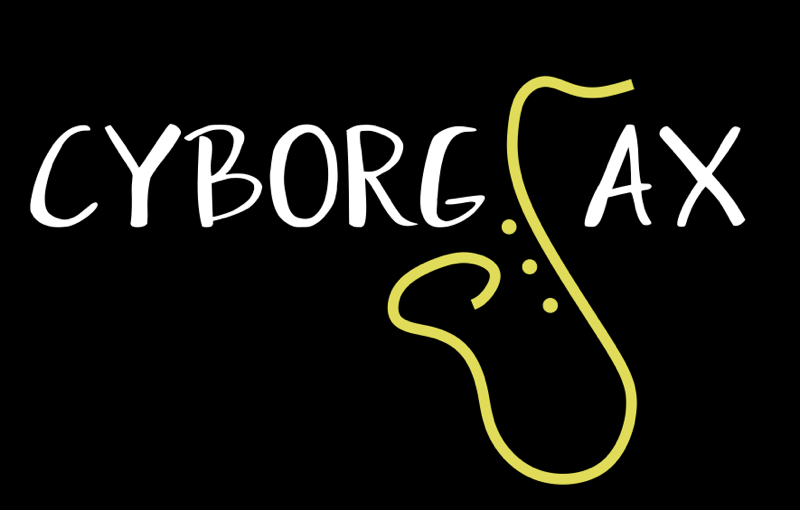
CyborgSax Test Report for Second Prototype

Team 7: Tariq Ahsan, Raymond Liang, Esiri Madagwa Jr., Qinglang Yu, Andrew Zannetos



Equipment and Setup

The main hardware equipment that was utilized during the prototype testing session consisted of a Teensy 3.6 microprocessor with an attached audio shield, a computer to program the Teensy, a 5V battery pack , two 8 x 8 LED matrices and an encoder knob. Several electrical cables were used to transfer power between the Teensy and the matrix, and a lavalier microphone with a TRS termination is used to transfer a real time audio signal to the Teensy audio shield.

The audio shield was attached below the Teensy microprocessor so that all the pins from the audio shield would align with the corresponding pins from the Teensy 3.6. The cables used to connect the LED matrix to the microprocessor were +5V, digital input and ground connections. These were attached to the Teensy’s +3.3V pin, GND pin, and pin 26 (assigned as the digital input in the code), respectively. The +5V and ground connections served to power the LED matrices while the digital input was used to control the matrix pattern and light intensity of the LEDs. There was also an XLR/TRS combo input that was soldered to the audio shield for the Teensy to process the input signal that the microphone was receiving and an encoder to control the amplification gain and the colour pattern on the LED matrices. There were 3 electrical wires for the user interface knob - to send clock, data and push button signals to the Teensy and a ground wire for those signals to pass through.

For the software setup, the latest version of the software was loaded to the Teensy microprocessor, which was used to make the real-time frequency spectrum patterns display in the matrices. When running the actual test, the microcontroller was powered with a battery pack so that it could receive the audio signal from the microphone and further output as a visual pattern. The microphone would take the audio signal and the microprocessor would perform FFT analysis on it to output FFT values. The LED lights will then light up in a range of different amplitudes based on the FFT algorithm and its FFT outputs. The encoder controlled the amplification factor of the FFT values by turning the knob, and the colour patterns were changed based off the push of the encoder button.

Measurements Taken

For the hardware measurements, it was determined the computer could communicate with the Teensy as the computer was able to identify the Teensy when plugged in. Utilizing our code, FFT values were recorded based on the microphone signal input. Based off the FFT values, the microphone was confirmed to work. The power cables, Teensy and audio shield were determined to have been soldered properly, as the LED matrices lit up based upon the corresponding FFT values (and algorithm) within the software. The code mapped out 16 columns, and assigned FFT bins (unique frequency bandwidths) to each column - should the FFT value be high for a certain frequency bandwidth, the corresponding column will light up a number of LEDs - the number of LEDs lit to FFT values is based on an algorithm that we created. This was consistent with our expectations of the code that we utilized to program the LED patterns. The rotary encoder was confirmed to work, as when turned clockwise, the gain of the FFT values would increase - and subsequently the number of LEDs lit up would increase. Also, the rotary encoder push function was confirmed to work, as when pressed, the matrix color scheme would change based on predetermined color presets coded on the Teensy.

In terms of software, the program was loaded successfully because the code worked as the LED matrices lit up to form a spectrum-equalizer pattern based off of the FFT function that was used for analyzing the sound input from the microphone. The LEDs had different colors to look like that of an SPL meter. The higher the FFT value for a column, the more LEDs lit in that column starting from the bottom and maxing out at the top. Also, when the encoder button was pushed down, a different color theme was displayed on the matrices. Different FFT values were displayed on the computer and were refreshed every 0.2 seconds, as did the LED matrix pattern.

For the debouncing feature test, the measurement numbers shown on the simulator did not get fluctuated and reacted based on the position the rotary encoder is rotated. In addition, the push counter only incremented once when the encoder was pushed down.

Improvements in the future

Even though the prototype presentation was a success, there is still more to be done to meet the client’s requirements for the final design. First, real time audio effects must be implemented (i.e reverb, pitch modulation, etc.). We also need the audio outputs for processed saxophone signal to be sent to speakers (XLR out, 3.5mm out). A UI must be implemented to control the various features of the project. A 4-position switch will be used to change between the various modes of the UI, and an extra encoder, totalling to two, will be used to control the various features. Finally, the casing of our project has to be completely redone to accommodate the 4-position switch, another XLR/TRS combo jack, an extra encoder knob, and the TFT screen.

Conclusion

The prototype presentation went according to the test plan that was written prior to the presentation. Even though everything is going according to plan so far, there will still be challenges in the next couple of months before ECE day when it comes to making improvements to our project. Our time for finishing this project is limited, as we only have around a few of weeks before our final prototype is due, so the team will be working hard to ensure that everything is in order for our final prototype testing.