**PITCHFINDER: LOGBOOK**

Redacted by Luis Herranz

**-15/03/2016**

* Began working in the assignment. This first phase will be part of the learning process and will allow gathering the fundamental skills in the manipulation of PICs devices.
* When mixing Led display and voice input the Leds seem to lose their "While button is pressed" functionality. They only trigger with each key press instead.
* At this point, we are studying the way we could incorporate the FFT in the application.

**-16/03/2016**

* Test with the FFT have begun. However we are trying to use al function incorporated in the RootTree library. However, no success was achieved Efforts focus on getting the FFT to work as is a required step in the pitch extraction.
* The Led problem was solved. An exception was required to solve the issue. There was no specification in what the application should do if both buttons were pressed at the same time.

**-17/03/2016**

* While trying to make the FFT work, we started getting a lot of Linker errors. The amount rised to an annoying count, to the point that the project was to the large amount of linker errors I have decided to start over again. This time we downloaded another project from MyStudySpace (Resource provided by Kingston´s University).
* The selected base for the project was “AudioAjuster”, which is based on a filtering system that works in the frequency domain. So it contains the FFT that we will be using.
* At the same point there is a second issue that we realize at this point. The FTT will originate a symmetric signal centered at the zero value. This means that both sides carry the same information and one of them could be dismissed. But at the same time, if we filter the negative part we will be halving the working frequency.

**-18/03/2016**

* For further testing we will be using an Android Application which allows outputting a tone of a certain frequency.
* During testing seems I didn´t do the frequency range limitation as it needed. The present version is trying to create an interval of 128 steps (corresponds to 7 bits) from the range 0 to 8000 Hz. Since we have filtered the negative Complex frequency representation we should be working from 0 to 4000 Hz instead.
* A first working version has been developed. Each of the three LEDs represent a different range of frequencies. Red represent low frequencies, from 0 to 1333 Hz. Yellow for mid-frequencies (1333 to 2666). Red for high Frequencies (2666 and higher).
* This version has a small Flickr at any of the states. So far I have not been able to determine the nature of this problem.
* There is a chance that the input introduces a little noise and this may cause that the other lights briefly activate for an instance. However it can be appreciated that the light that was supposed to be active bright for most of the time. Another possibility could be that the code contains a bug that causes the flickering lights.

**-26/03/2016**

* The code has been reorganized and revisited, but the flickering problem remains. The code could still be incorrectly implemented and there will be further revisions in the future.

**-02/04/2016**

* Today I acquired a second DSP Starter Kit to conclude if my previous device had some kind of defection. After setting up and updating the new equipment, the results after programing where neither the previous nor the expected. This raises plenty of worries regarding the code and will need testing with a third device. For now we consider the assestment as completed with the previous functioning version.