Piano Note Detector

Pothuri Venkata N S K Nikhil[1], Prabhanshu Chouhan[2]

[1] Integrated Circuit Design and Technology, Indian Institute of Technology Gandhinagar, Gandhinagar-382355, India

[2]Artificial Intelligence, Indian Institute of Technology Gandhinagar, Gandhinagar-382355, India

E-mail: 24110262@iitgn.ac.in, 24110263@iitgn.ac.in

***Abstract*—This paper discusses a method of detecting a note played in the 4th octave on a piano by employing Fast Fourier Transform (FFT). The method involves using KY-038 sound sensor for capturing the sound samples in the air, which later are processed in an Arduino Uno using the aforementioned method and displaying the results on a LCD screen.**

Please note that the results are best understood through viewing the video, rather than the report as it only shows pics of the result displaying.

Keywords — Fast Fourier Transform, KY-038 sensor, Note detection

I. Aim

To develop method to detect a note played in the 4th octave on a piano.

II. Introduction and Theory

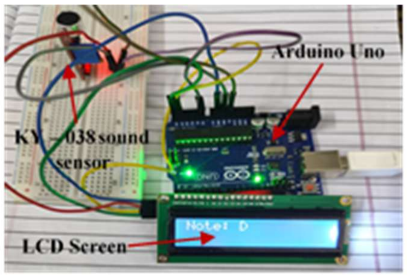

Fig. 1: Photograph of the note detection system

The prototype shown in Fig 1 is designed using a KY-038 sound sensor, 16 x 2 LCD screen and an Arduino Uno.

A. Data Collection

The KY-038 is an analog and digital sound detection sensor module, commonly used for detecting sound levels in the environment. It features an electret microphone that captures sound vibrations in the air and converts them into electrical signals. These signals are then amplified and fed into an LM393 voltage comparator circuit.

The module provides two types of outputs:

- Analog Output (A0): Outputs a variable voltage based on sound intensity.
- Digital Output (D0): Goes HIGH when sound exceeds a threshold level, adjustable through an on board potentiometer.

The presence of the potentiometer allows users to calibrate the sensitivity of the digital signal detection. The analog output can be used for more granular sound analysis, which is utilized in this project for approximating musical notes.

B. Data Processing

The data is then transmitted to the Arduino for processing. Hann windowing function, eq. (1), is applied to the inputs to avoid spectral leakage. This is done by multiplying a raised cosine curve to each input. Then average and root mean squared average of the samples is taken to determine, whether there is any amplitude in the data to perform DFT.

$$w(n) = sin^2\left(\frac{n\pi}{N}\right), n = 0, 1, \dots, N-1 \tag{1}$$

After that we apply Discrete Fourier Transform (DFT), but it is too slow. DFT takes samples from a signal and applies eq. (2) to breakdown the signal into sum of simple sinusoids with discrete frequencies.

$$X(k) = \sum_{n=0}^{N-1} x(n)\cdot e^{-\frac{j2\pi kn}{N}}, \quad k = 0, 1, \dots, N-1 \tag{2}$$

But, from analysis of the equation one can see that the computation is *O(N²)* complexity, and it will be very computationally expensive when dealing with lots of samples. This can be reduced to *O(N log N)* complexity using the Cooley-Tukey approach. In this approach, the equation is divided into even and odd harmonics for reducing the computation time, eq. (3).

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)\cdot e^{-\frac{j2\pi k(2n)}{N}} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)\cdot e^{-\frac{j2\pi k(2n+1)}{N}}$$

$$X(k) = E(k) + e^{-\frac{j2\pi k}{N}}\cdot O(k) \tag{3}$$

This recursive decomposition can be continued until the DFT's are of size 2 and then all sums can be added up. This approach is more commonly known as the Fast Fourier Transform (FFT). It works best with sample sizes of 2^n. So we used a sampling size of 128 samples, and processed the data.

C. Output Display

Then five dominant peaks are identified and correlated to musical note ranges. These ranges have a predetermined thresholds, to which we check the proximity of these peaks to their respective thresholds. The frequency closest to its respective threshold, is displayed on the LCD screen as the guess of the model.

III. Results

The model perfectly detected the notes played on the fourth octave with a response time of 0.2 – 0.5 sec. Fig. 2 displays the results of the experiment.

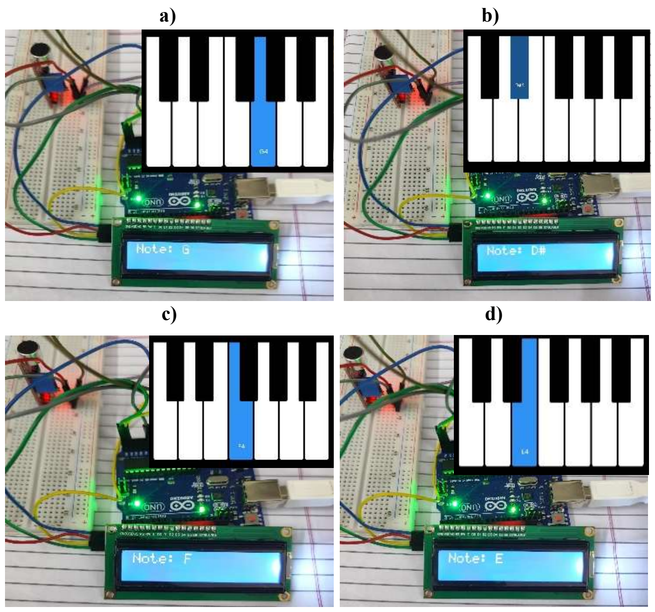

Fig. 2: a) Result of G note played on 4th octave, b) Result of D# note played on 4th octave, c) Result of F note played on 4th octave and d) Result of E note played on 4th octave

IV. Conclusions and Future Scope

In conclusion, this project effectively demonstrates the use of an Arduino, a basic sound sensor, and an LCD screen to detect and display musical notes. It highlights the practical application of real-time signal processing using FFT and note mapping techniques. Though there is room for improvement.

The project has few limitations, one of which is incapable of detecting two notes when played at once, complex chords and melodies. Applying machine learning can enhance the efficiency of the model, also try extending the idea to speech recognition and speech to text models. Future versions, can have an inbuilt Wi-Fi or Bluetooth module for transferring the data obtained to an app, so that it can be used as music transcription device.

Acknowledgement

We like to thank our course instructor, TA and peers for their invaluable support during the design and development of this project. We would also like to thank the lab staff for providing the resources useful for the project.

References

[1] Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90), 297-301.

[2] Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, 66(1), 51-83.

[3] M. Tahat, "EasyFFT - Frequency Transform for Arduino," *Instructables*.

[4] J. VanderPlas, "Understanding the FFT," *Pythonic Perambulations*, Aug. 28, 2013.