IoT Frequency Analysis System for Musical Instruments

Juan Cantu
Department of Electrical and Computer Engineering
The University of Texas Rio Grande Valley
Edinburg, TX, USA

***Abstract*—This project proposes the design of a real-time frequency detection system aimed at identifying the fundamental frequencies of brass instrument signals, such as those produced by a trumpet. Using a microphone as the input device and the DE1-SoC as the processing platform, this system will capture audio, perform frequency analysis using the YIN pitch detection algorithm, with future versions to support FFT-based spectral analysis, and transmit the detected frequency over a network. The project demonstrates an embedded IoT solution for musical signal monitoring and lays the groundwork for remote, connected audio analysis platforms.**

***Index Terms*—IoT, real-time signal processing, pitch detection, embedded systems, musical acoustics, DE1-SoC**

I. Introduction

Brass instruments like the trumpet produce sound through the vibration of the player's lips against a mouthpiece, generating tones with distinct fundamental frequencies and harmonic overtones. Detecting these frequencies in real time has practical applications in tuning, pitch tracking, performance analysis, and live signal enhancement.

With the rise of Internet of Things (IoT) technologies and embedded processing platforms, real-time audio analysis can now be implemented on low-cost, network-connected devices. This project presents an IoT-based frequency detection system designed for brass instruments, using a microphone and the DE1-SoC development board to capture, process, and transmit detected frequencies over a network. The goal is to establish a connected system for musical signal monitoring that bridges digital signal processing and IoT [3].

II. Theory

A musical note contains a fundamental frequency and harmonics. These frequencies are embedded in the time-domain waveform captured by a microphone.

To estimate the fundamental frequency, the system currently uses the YIN algorithm [1], implemented via the Librosa Python library [6]. YIN is a time-domain method that excels at tracking pitch in noisy or musical environments. It operates by identifying periodicity in the signal and is well-suited for detecting monophonic instrument tones in real time.

This pitch analysis is performed on the ARM processor (HPS) of the DE1-SoC, which records audio through a USB microphone and executes the Python-based detection pipeline. Detected frequencies are streamed to a remote GUI client over TCP/IP, enabling real-time visualization and logging.

III. Design

The implemented system consists of the following interconnected components designed to operate in real time:

- **Microphone Input:** Captures live trumpet audio and connects to the DE1-SoC's HPS via USB.
- **HPS (ARM Processor):** Hosts a Python-based server that captures audio samples, processes them using the YIN pitch detection algorithm from the Librosa library, and extracts the fundamental frequency in real time. This module is designed to transition to FFT-based analysis [2][4] in future versions for richer spectral insight.
- **Network Transmission:** The extracted frequency data, along with its corresponding musical note mappings, is transmitted via TCP/IP to a remote client on the same network.
- **Client Interface:** A graphical user interface (GUI) developed in Python with Tkinter receives the streamed data, displays live frequency and note information, and plots frequency over time. After each session, the client automatically receives a CSV data log and a PNG plot from the server for record-keeping and further analysis.

The complete interaction and data flow between the system components is illustrated in Figure 1.

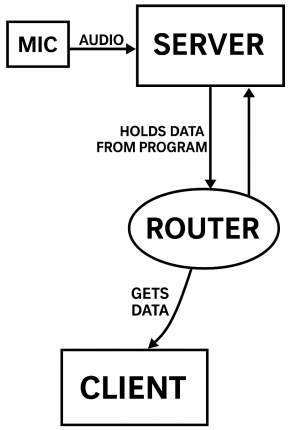

Fig. 1. High-level communication diagram: Audio is captured, processed on the DE1-SoC, and transmitted to a remote client over TCP/IP for real-time display and logging.

This modular design allows for easy substitution or enhancement of individual components, such as upgrading the

pitch detection algorithm or extending the GUI with new features. The architecture emphasizes low-latency communication and real-time responsiveness to support musical applications.

IV. Results

To evaluate the consistency of the frequency detection system, a trumpet note (G4, corresponding to concert pitch F4 at approximately 349 Hz) was played across multiple sessions. A total of $n = 30$ stable frequency samples were selected from one session for analysis. These correspond to the values shown in Table I.

The average measured frequency was $\bar{f} = 349.35$ Hz, with a standard deviation of $s = 10.78$ Hz. Using the t-distribution for a 95% confidence level with $n = 30$ samples [5], the margin of error (ME) was computed as:

$$ME = t_{0.025,29} \cdot \frac{s}{\sqrt{n}} \approx 2.045 \cdot \frac{10.78}{\sqrt{30}} \approx 4.03 \text{ Hz}$$

The 95% confidence interval for the detected pitch is:

$$\boxed{[345.32 \text{ Hz},\ 353.38 \text{ Hz}]}$$

This range includes the expected F4 pitch (349.23 Hz), demonstrating strong agreement between the system output and the target tone.

One sample (Sample 19, 292.31 Hz) deviated significantly from the rest. This is addressed in Section V, where we discuss potential causes and why the sample was retained.

TABLE I
Full Frequency Samples for Trumpet Note (G4)

Sample	Freq (Hz)	Concert	Trumpet
1	352.79	F4	G4
2	351.88	F4	G4
3	351.44	F4	G4
4	350.94	F4	G4
5	350.98	F4	G4
6	350.66	F4	G4
7	350.88	F4	G4
8	350.47	F4	G4
9	351.27	F4	G4
10	351.23	F4	G4
11	351.05	F4	G4
12	351.61	F4	G4
13	351.38	F4	G4
14	351.43	F4	G4
15	351.75	F4	G4
16	351.45	F4	G4
17	351.01	F4	G4
18	350.60	F4	G4
19	292.31	D4	E4
20	350.78	F4	G4
21	351.39	F4	G4
22	351.53	F4	G4
23	351.12	F4	G4
24	351.26	F4	G4
25	351.63	F4	G4
26	351.30	F4	G4
27	351.09	F4	G4
28	351.14	F4	G4
29	352.65	F4	G4
30	351.42	F4	G4

V. Discussion

The frequency measurements demonstrate that the system consistently detects musical pitch within a narrow confidence interval, validating its effectiveness for real-time pitch tracking. Minor variability in the results, such as the observed outlier at 292.31 Hz, can be attributed to realistic playing inconsistencies, like breath control and articulation. This sample was used to reflect the system's robustness under practical use.

The server, implemented in Python and deployed on the DE1-SoC's ARM processor, processed audio in real time using the YIN algorithm. Frequency data was transmitted over TCP/IP to a Python-based GUI client, which displayed pitch information and updated live plots with low latency.

Each session triggered a GUI reset and resumed cleanly on the next start, confirming the system's ability to support repeated recordings without restarting the client. The automatic transmission of CSV logs and frequency plots further demonstrates the reliability of the client-server pipeline.

Overall, the system successfully combines embedded signal processing with networked visualization to deliver a practical IoT solution for musical frequency monitoring.

VI. Future Work

Future work will focus on adding real-time pitch correction, harmonic enhancement, and noise suppression to improve tone quality. Adaptive filtering based on statistical analysis may also be used to better handle natural performance variability.

The IoT system will be expanded to support wireless transmission and integration with mobile or web-based dashboards. This will make the system more portable and accessible for musicians during practice or live performance.

VII. Conclusion

This project demonstrates a functional prototype for real-time trumpet pitch detection using an embedded IoT system. It captures live audio, extracts frequency data, and transmits results to a remote client with reliable performance.

The system accurately tracks pitch despite natural performance variability and provides automated logging and visualization. This foundation supports future DSP enhancements and shows the potential of combining embedded processing with networked audio analysis.

References

[1] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," J. Acoust. Soc. Am., vol. 111, no. 4, pp. 1917–1930, Apr. 2002.

[2] B. Gold, N. Morgan, and D. P. W. Ellis, Speech and Audio Signal Processing: Processing and Perception of Speech and Music, 2nd ed. Hoboken, NJ, USA: Wiley, 2011.

[3] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet, "Internet of Musical Things: Vision and Challenges," IEEE Access, vol. 6, pp. 61994–62017, 2018.

[4] E. Mandel, "Chord Recognition From DFTs of Down Sampled Audio Signals," in Proc. IEEE Western New York Image and Signal Processing Workshop (WNYISPW), Rochester, NY, USA, Oct. 2021.

[5] J. S. Bendat and A. G. Piersol, Random Data: Analysis and Measurement Procedures, 4th ed. Hoboken, NJ, USA: Wiley, 2010.

[6] B. McFee et al., "librosa: Audio and music signal analysis in Python," in Proc. 14th Python in Science Conf. (SciPy 2015), Austin, TX, 2015, pp. 18–25.