SISTEM MONITORING FREKUENSI BERBASIS ALGORITMA *FAST FOURIER TRANSFORM* DAN *POWER SPECTRAL DENSITY*

S. Y. Anugrah[1], F. Hasan[1], W. A. Muttaqien[1*], M. Sari[1], I. P. A. Saputra[1], A. H. Dalimunthe[1], P. Kurniasari[1], D. Windisari[1]

[1]Teknik Elektro, Universitas Sriwijaya, Palembang
[*]*Corresponding author e-mail: muttaqien.wildan12@gmail.com*

ABSTRAK: Sistem pengolahan audio sering mengalami gangguan berupa tumpang tindih frekeuensi yang menurunkan kejernihan suara, maka dari itu perlu adanya pemantauan frekuensis sebagai sarana analisis. Sementara metode monitoring yang tersedia sering kali hanya terbatas pada aspek audio tanpa memberikan analisis kuantitatif terkait spectrum dan distribusi daya. Untuk menjawab permasalahan tesebut, penelitian ini berfokus pada rancang bangaun sistem monitoring melalui pengamatan frekuensi menggunakan algoritma *Fast Fourier Transform* dan *Power Spectral Density* yang mampu menampilkan visualisasi spectrum frekuensi sekaligus etimasi distribusi daya sinyal. Pengembangan utama dari penelitian ini adalah integrasi perangkat keras dan perangkat lunak yang sederhana namun efisien, diman sumber sinyal frekuensi diatur melalui Generator Frekuensi, kemudian disalurkan ke rangkaian *Band Pass Filter* untuk menyaring frekuensi tertentu, dan diteruskan ke modul ESP32 sebagai converter analog ke digital. Anda ingin memahami proses pengolahan data digital hasil konversi yang menggunakan algoritma FFT (*Fast Fourier Transform*) dan PSD (*Power Spectral Density*) dengan pemrograman *Javascript*, dan kemudian memvisualisasikannya menggunakan Chart.js. Cara kerja sistem secara menyeluruh meliputi input frekuensi melalui jack audio, penyaringan dengan BPF (*Band Pass Filter*), konversi ke data digital oleh ESP32, pemrosesan analisis dengan FFT dan PSD, hingga visualisasi dengan Chart.js. Hasil pengujian menunjukkan bahwa sistem bekerja stabil dengan error terbaik bisa mencapai 8,06%, resolusi frekuensi 8,20 Hz 39,06 Hz, latensi *end-to-end* 2436 ms, serta nilai SNR mencapai 4,17 dB pada ukuran sampel 256 & 64 dengan 5 kali ulangan pengukuran. Kebaruan penelitian ini terletak pada integrasi FFT–PSD berbasis ESP32 untuk monitoring spektrum frekuensi secara real-time dan berbiaya rendah yang dapat diakses melalui antarmuka web.

Kata Kunci: Band Pass Filter, ESP32, FFT, PSD, Monitoring Sinyal.

ABSTRACT: *The audio processing system often experiences interference in the form of overlapping frequencies that reduce sound clarity, making frequency monitoring necessary as an analytical tool. However, existing monitoring methods are generally limited to audio-only assessment without providing quantitative analysis of the spectrum and power distribution. To address this issue, this study focuses on the design and development of a frequency monitoring system using Fast Fourier Transform (FFT) and Power Spectral Density (PSD) algorithms, which are capable of visualizing the frequency spectrum and estimating signal power distribution. The main contribution of this research lies in the integration of simple yet efficient hardware and software, where the frequency source is generated using a Frequency Generator, filtered through a Band Pass Filter to isolate specific components, and then transmitted to an ESP32 module as an analog-to-digital converter. The resulting digital data are processed using FFT and PSD algorithms in JavaScript and visualized through Chart.js for interactive spectral representation. The overall system workflow includes frequency input through the audio jack, filtering using the BPF, digital conversion by the ESP32, analytical processing using FFT and PSD, and final visualization via Chart.js. Experimental results show that the system operates stably, achieving a minimum error of 8.06%, frequency resolutions of 8.20 Hz and 39.06 Hz, an end-to-end latency of 2436 ms, and an SNR of 4.17 dB at sample sizes of 256 and 64 with five measurement repetitions. The novelty of this research lies in the integration of an ESP32-based FFT–PSD system for real-time, low-cost frequency spectrum monitoring accessible through a web interface.*

Keywords:Band Pass Filter, ESP32, FFT, PSD, Signal Monitoring.

1 Pendahuluan

Berbagai gangguan audio seperti kebisingan, frekuensi tidak stabil, dan distorsi sering muncul pada proses perekaman maupun transmisi suara, sehingga menurunkan kualitas dan akurasi informasi. Dalam perangkat komunikasi maupun kegiatan industri dan penelitian, kesalahan pembacaan frekuensi audio dapat menyebabkan data tidak valid dan memengaruhi hasil analisis. Karena itu, monitoring frekuensi audio menjadi kebutuhan penting untuk menjaga kualitas komunikasi dan pengolahan sinyal. Pengembangan sistem monitoring yang handal dan mudah digunakan juga memberikan manfaat besar dalam kegiatan pembelajaran mahasiswa, khususnya di bidang telekomunikasi.

Dalam memenuhi kebutuhan tersebut, perlu adanya sitem monitoring yang mengimplementasikan sebuah algoritma, seperti penelitian yang dilakukan oleh Muhammad Hansyah Utama, Mohamad Fathurahman, dan Shita Herfiah berjudul *Rancang Bangun Monitoring Sinyal Radio VHF Band II Berbasis Graphical User Interface* [1] merancang sistem monitoring spektrum frekuensi radio yang portabel dan terjangkau untuk mendukung kebutuhan pembelajaran. Sistem tersebut menggunakan Raspberry Pi sebagai unit pemrosesan, RTL-SDR sebagai penerima sinyal, antena teleskopik, LCD touchscreen, dan speaker. Pada sisi perangkat lunak, algoritma FFT diimplementasikan untuk analisis spektrum, sedangkan teknik demodulasi FM digunakan untuk menghasilkan keluaran suara. Hasil pengujian menunjukkan sistem mampu mendeteksi sinyal siaran radio FM dengan nilai RSSI bervariasi antara –25,13 dBm hingga –56,15 dBm, dengan kualitas suara yang tidak selalu berbanding lurus dengan kekuatan sinyal. Kekurangan penelitian ini adalah hasil analisis masih terbatas pada penggunaan FFT dan demodulasi FM tanpa adanya algoritma yang membantu estimasi distribusi daya, serta media monitoring yang berbasis GUI sehingga belum mendukung fleksibilitas pemantauan melalui platform web yang lebih interaktif dan mudah diakses.

Pada penelitian [2] Memanfaatkan algoritma FFT sebagai inti dari proses analisis spectrum frekuensi suara untuk mengidentifikasi dan mengklasifikasi jenis suara manusia. Pada penelitian ini FFT berperan penting dala mengubah sinyal audio dari domain waktu ke domain frekuensi, sehingga sistem mampu mendeteksi frekuensi fundamental yang mewakili karakteristik setiap tipe suara. Dengan menerapan teknik *windowing* dan *peak detection* algoritma FFT memungkinkan proses analisi berjalan secara efisien dan *realtime*. Hasil dari penelitian ini menunjukkan bahwa metode berbasis FFT mampu mencapai tingkat akurasi identifikasi frekuensi antara 95,7% hingga 98,9%, serta akurasi klasifikasi jenis suara sebesar 81,2% dengan waktu pemrosesan hanya 5 milidetik. Kurangnya visualisasi spektrum secara interaktif serta ketiadaan fitur pemantauan frekuensi berkelanjutan menunjukkan bahwa masih ada ruang pengembangan menuju sistem monitoring frekuensi audio yang lebih fleksibel, komprehensif, dan mudah diakses oleh pengguna.

Penelitian [3] menjelaskan perancangan sistem monitoring spektrum yang bekerja dengan menangkap sinyal radio menggunakan RTL-SDR 2832U kemudian mengirimkan data ke komputer untuk diolah menggunakan aplikasi Visual Studio C#. Sistem monitoring ini memanfaatkan algoritma FFT untuk mengubah sinyal RF dari domain waktu ke domain frekuensi, sehingga pengguna dapat melihat pola spektrum AM, FM, dan trunking dalam bentuk grafik pada aplikasi desktop. FFT yang digunakan berfungsi untuk menampilkan puncak-puncak frekuensi dan memberikan gambaran intensitas sinyal, meskipun implementasinya masih bersifat sederhana tanpa teknik windowing seperti Hann atau Hamming yang biasa digunakan untuk mengurangi spectral leakage. Pengujian dilakukan dengan menangkap beberapa stasiun radio dan sinyal HT, dan hasilnya menunjukkan sistem mampu menampilkan spektrum dasar serta merekam audio yang diterima oleh antena. Namun demikian, penelitian ini masih terbatas pada monitoring sinyal RF berbasis perangkat RTL-SDR, belum menerapkan FFT dengan window Hann dan Hamming, tidak menghitung PSD, dan monitoringnya tidak berlangsung secara real-time berbasis website

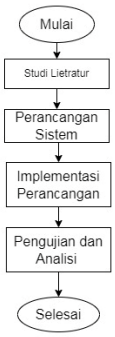
Untuk mengatasi keterbatasn penelitian sebelumnya maka penelitian ini mengombinasikan algoritma FFT (*Fast Fourier Transform*) dan PSD (*Power Spectral Density*). FFT ialah algoritmayang dapat digunakan untuk mentransformasikan sinyaldari domain waktu ke domain frekuensi [4]. Sementara itu, algoritma PSD berfungsi menghitung distribusi daya pada setiap frekuensi, sehingga hasil analisis tidak hanya bersifat visual tetapi juga kuantitatif .

Pada penelitian ini diamati frekuensi audio pada rentang 20–5.000 Hz [5]. Berdasarkan metode dan hasil pengujian pada sistem, ditetapkan hipotesis kuantitatif bahwa perangkat mampu mendeteksi frekuensi dengan resolusi minimum (Δf) sebesar 1 Hz, memiliki tingkat kesalahan pengukuran tidak lebih dari ±5%, serta bekerja dengan latensi pemantauan < 300 ms sehingga memungkinkan monitoring frekuensi audio secara *near real-time.*

Cara kerja sistem diawali dengan menerima inputan frrekuensidari generator frekuensi yang kemuduian frekuensi tersebut difilter oleh BPF (*Band Pass Filter*), kemudian dikonversi ke data digital oleh ESP32. Data tersebut diproses menggunakan FFT dan PSD melalui bahasa pemrograman JavaScript, lalu ditampilkandengan *library* Chart.js, sehingga hasil analisis dapat ditampilkan dalam bentuk grafik yang disiapkan. Dengan arsitektur ini, sistem monitoring mampu memberikan informasi analisis spektrum frekuensi audio secara lebih jelas, kuantitatif, dan mudah diakses.

2 Metode Penelitian

Perancangan sistem monitoring frekuensi ini dilakukan melalui tahapan metodelogis yang terstruktur, tahapan tersebut dapat dilhat pada Gambar 2.1 Diagrarm Tahapan Penelitian.



Gambar 2.1 Diagram Tahapan Penelitian

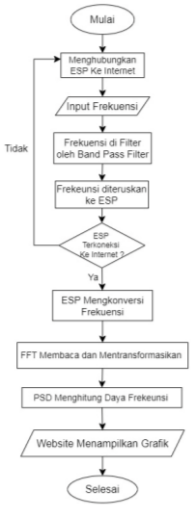
2.1 Alur Penelitian

2.1.1 Studi Literatur

Tahap awal penelitian dilakukan melalui studi literatur dengan menelaah berbagai jurnal, artikel, serta sumber referensi terpercaya yang membahas sistem monitoring frekuensi, pemanfaatan ESP32 sebagai konverter analog ke digital, penerapan algoritma FFT dan PSD, penggunaan filter band pass, serta pengembangan tampilan grafik dengan *Chart.js*.

2.1.2 Perancangan Sistem

Metode dalam perancangan sistem dirancang dengan penedekatan UML (*Unified Modelling Language*) untuk menafsirkan alur kerja sistem, pemodelan alur kerja dapat dilihat pada Gambar 2.2 berikut.



Gambar 2.2 Diagram Alur Kerja Sistem

Dalam perancangan sistem, hal utama yang harus dipastikan adalah koneksi ESP ke internet agar komunikasi data dapat berlangsung secara *online*. Setelah terhubung, sinyal frekuensi dari generator atau sumber audio melewati Band Pass Filter untuk memperoleh rentang frekuensi yang diinginkan, kemudian diteruskan ke ESP untuk dikonversi menjadi data digital. Data ini diproses oleh algoritma FFT untuk mengubah sinyal waktu ke domain frekuensi, lalu dihitung distribusi dayanya melalui PSD. Seluruh hasil pemrosesan tersebut kemudian dikirim dan ditampilkan pada website yang telah disiapkan.

2.1.3 Implementasi Perancangan

Pertama yang dirancang ialah rangkaian Band Pass Filter. Untuk merancang sebuah rangkaian Band Pass Filter membutuhkan bebrapa komponen elektronika yang ada pada Tabel 2.1 Tabel Kompnen Elektronika.

Tabel 2.1 Tabel Komponen Elektronika

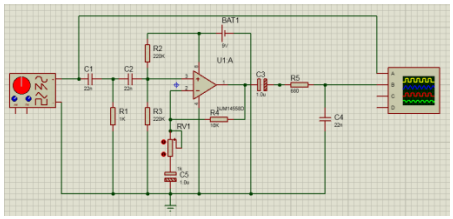
No	Nama	Jumlah	Fungsi
1	IC4558D (Op-Amp)	1	Penguat sinyal.
2	Resistor HPF (20KΩ) dan LPF (6.8KΩ)	2	Mengatur rentang band pass filter yang diinginkan.
3	Kapasitor Non-Polar 22nF	2	Mengatur rentang band pass filter yang diinginkan.
4	Resistor Feedback (10KΩ)	1	Sebagai umpan balik yang terhubung ke input pembalik.
5	Resistor Pembagi (220KΩ)	2	Membagi tegangan dengan sama rata.
6	Kapasitor Polar 10µF	2	Sebagai kopling untuk memblokir arus DC dari output Op-Amp.
7	Potensiometer	1	Sebagai gain.
8	Batre 9 Volt	1	Sebagai daya utama.
9	Breadboard	1	Media perakitan.
10	Kabel Jumper	-	Penghuung antar komponen.

Rangkaian BPF dirancang untuk melewatkan rentang frekuensi 362 – 1064 Hz. Spesifikasi utama BPF sebagai berikut:

- Tipe/orde: Orde dua implementasi melalui konfigurasi HPF + LPF atau Sallen–Key bandpass.
- Frekuensi cut-off bawah ($f_{c\,low}$): 362 Hz.
- Frekuensi cut-off atas ($f_{c\,high}$): 1064 Hz.
- Frekuensi tengah (f_c): $(f_{c\,low} + f_{c\,high})/2 =$ 713 Hz.
- Bandwidth (BW): $f_{c\,high} - f_{c\,low} = 702$ Hz.
- Quality factor (Q): $Q = \frac{f_c}{BW} \approx 1{,}02$.

- Op-Amp yang digunakan: IC4558D sebagai penguat dan buffer sinyal.

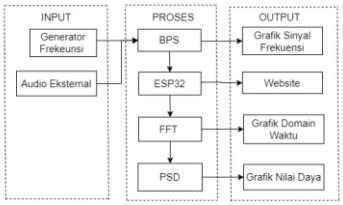
Sebelum merakit komponen menjadi rangkaian Band Pass Filter, rangkaian diubuat pada software proteus untuk ujicoba simulasi terlebih dahulu. Yang dapat dilihat pada Gambar 2.3 Simulasi Rangkaian BPF.



Gambar 2.3 Simulasi Rangkaian BPF

Tambahin penjelasan nilai frekuensi yang akan dilewatkan ialah 362–1064 Hz.

Untuk rancangan sistem diilustrasikan melalui Block Diagram yang dapat dilihat pada Gambar 2.3 Block Diagram.



Gambar 2.3 Block Diagram

Sistem menggunakan beberapa komponen perangkat keras, dan juga menggunakan Algoritma, diantaraya sebagai berikut:

a. Band Pass Filter
   Adalah rangkaian filter yang hanya akan melewatkan sinyal pada rentang frekuensi tertentu dan meredam frekuensi yang dilar rentang tersebut.
b. ESP32
   Merupakan mikrokontroler yang memiliki kemampuan pemrosesan tinggi serta konektivitas Wi-Fi. ESP32 berfungsi sebagai pegendali utama dalam proses data.
c. Generator Frekuensi
   Sumber sinyal frekuensi yang dihasilkan melalui situs *web* yang mampu memproduks frekensi tertentu.
d. FFT (*Fast Fourier Transform*)
   Merupakan metode komputasi efisien dari *Discrete Fourier Transform* (DFT) yang memungkinkan proses perhitungan transform spectrum dilakukan dengan waktu yang jauh lebih singkat.
e. PSD (*Power Spectral Density*)
   Adalah representasi otomatis yang menggambarkan bagaimana kekuatan dari suatu sinyal terdisribusi terhadap frekuensi. PSD juga dapat memperlihatkan seberapa besar energy atau daya yang terkandung pada setiap komponen frekuensi.

2.1.4 Pemrograman dan Integrasi Algoritma

Pada penelitian ini, Arduino IDE digunakan untuk memprogram ESP32, supaya dapat melakukan akuisisi sinyal analog dari Band Pass Filter, kemudian mengubahnya menjadi data digital melalui ADC 12-bit dengan resolusi 0–4095.

Data digital dikirim ke sisi klien menggunakan protokol MQTT dalam format JSON, yang berisi parameter sampling, data raw, dan metrik tambahan. Di sisi *JavaScript*, data diproses lebih lanjut untuk menghasilkan visualisasi domain waktu, FFT, dan PSD.

Agar pemrosesan sinyal berjalan konsisten, penelitian ini menggunakan parameter tetap sebagai berikut:

- Resolusi ADC: 12-bit (rentang nilai digital 0–4095).
- Tegangan referensi (V_{ref}): 3,3 V
- Sampling rate (f_s): 2,1 kHz & 10 kHz.
- Nyquist (f_N): $\frac{f_s}{2}$, 1050 Hz & 5000 Hz
- Jumlah sampel (N): 64 & 256 sampel per blok.
- Resolusi frekuensi (Δf):

$$\Delta f = \frac{f_s}{N} = \frac{2100}{256} = 8{,}2031 \text{ Hz}$$

$$\Delta f = \frac{f_s}{N} = \frac{10000}{256} = 39{,}0625 \text{ Hz}$$

- Latensi Sistem ADC (T_{ADC}):

$$T_{\text{ADC}} = \frac{N}{f_s} = \frac{256}{2100} = 121{,}9 \text{ ms}$$

$$T_{\text{ADC}} = \frac{N}{f_s} = \frac{256}{10000} = 25{,}6 \text{ ms}$$

Pada proses FFT, diterapkan fungsi window berupa Hann dan Hamming. Kedua window ini memberikan bobot yang bervariasi pada setiap sampel untuk mengurangi spectral leakage. Secara matematis, fungsi window tersebut dinyatakan sebagai:

- Window Hann:

$$w[n] = 0.5(1 - \cos(\frac{2\pi n}{N-1}))$$

- Window Hamming:

$$w[n] = 0.54 - 0.46\cos(\frac{2\pi n}{N-1})$$

Magnitude dihitung dengan persamaan berikut, untuk bisa ditampilkan ke *Chart.js*:

$$|\ X(k)\ | = \sqrt{Re(X(k))^2 + Im(X(k))^2}$$

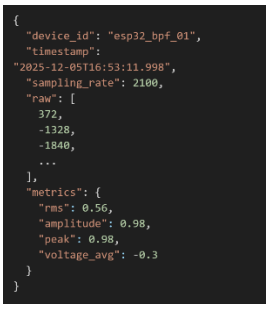
Algoritma FFT dihitung secara manual pada *JavaScript* menggunakan:

$$X(k) = \sum_{n=0}^{N-1} x[n](\cos(-\frac{2\pi kn}{N}) + j\sin(-\frac{2\pi kn}{N}))$$

PSD dihitung menggunakan metode periodogram sederhana:

$$PSD(k) = \frac{|\ X(k)\ |^2}{N \cdot f_s}$$

Data dikirim melalui MQTT dalam format JSON seperti Gambar 2.4 untuk memudahkan proses *parsing*:



Gambar 2.4 MQTT dalam Format JSON

Pada sisi klien, *JavaScript* dijalankan menggunakan VSCode sebagai lingkungan pengembangan. Sebuah *endpoint* MQTT dibuat untuk menerima data dari ESP32 secara *real-time* dan ditampilkan dengan menggunakan *library Chart.js*.

2.2 Metode Pengujian Sistem

Pengujian sistem meliputi akurasi pendeteksian frekuensi, resolusi spektral, rasio *signal-to-noise* (SNR), serta latensi *end-to-end* dari proses akuisisi hingga visualisasi. Menggunakan sinyal terkontrol dari Generator Frekuensi yang disalurkan melalui rangkaian BPF menuju modul ESP32 sebagai perangkat akuisisi digital.

2.2.1 Tujuan Pengujian

Tujuan pengujian ini adalah untuk memperoleh dan mengevaluasi parameter-parameter berikut:

1. Akurasi frekuensi terdeteksi, berupa selisih antara frekuensi acuan (f_{ref}) dan frekuensi hasil FFT.
2. Resolusi spektral, ditentukan oleh konfigurasi sampling rate dan ukuran sampel $\Delta f = \frac{f_s}{N}$.
3. *Signal-to-Noise Ratio* (SNR) yang berasal dari hasil PSD.
4. Latensi *end-to-end*, hanya dari akuisisi ADC → pengolahan FFT/PSD → pengiriman MQTT → visualisasi *Chart.js*.

2.2.2 Perhitungan Parameter Utama

SNR dihitung berdasarkan hasil PSD menggunakan pendekatan:

$$\text{SNR (dB)} = 10\log_{10}(\frac{P_{\text{signal}}}{P_{\text{noise}}})$$

dengan:

- P_{signal}: nilai PSD pada bin frekuensi utama,
- P_{noise}: rata-rata PSD pada bin selain puncak.

Latensi *end-to-end* (E2E) dihitung berdasarkan selisih waktu antara proses selesai dilakukan pada sisi perangkat (ESP32) dan waktu ketika data diterima serta divisualisasikan pada *browser*. Persamaan latensi *end-to-end* dirumuskan sebagai berikut:

$$\text{Latensi}_{E2E} = t_{\text{chart}} - t_{\text{esp32}}$$

Latensi yang dihasilkan pada sistem ini meliputi dari beberapa latensi berikut:

1. Waktu akuisisi ADC, dihitung sebagai:
$$T_{\text{ADC}} = \frac{N}{f_s}$$
2. Waktu komputasi FFT, yang bergantung pada platform pemrosesan (ESP32 atau JavaScript).
3. Waktu transmisi MQTT, yang dipengaruhi oleh stabilitas jaringan, jarak broker.
4. Waktu render grafik oleh Chart.js, yang umumnya relatif kecil dibandingkan komponen lainnya.

Pada salah satu sesi pengujian, *timestamp* yang terekam bisa untuk menghitung latensi, sebagai berikut:

$$t_{\text{esp32}} = 2025 - 12 - 05T15{:}50{:}06.620$$
$$t_{\text{chart}} = 2025 - 12 - 05T15{:}50{:}04.201$$
$$\text{Latensi}_{E2E} = t_{\text{esp32}} - t_{\text{chart}} = 2419 \text{ ms}$$

Nilai ini menunjukkan adanya waktu tunda yang signifikan dalam keseluruhan proses, di mana sebagian besar latensi disebabkan oleh internet publik.

2.2.2 Skenario Pengujian

Prosedur pengukuran disusun untuk memastikan proses akuisisi sinyal, pemrosesan spektral, serta visualisasi berjalan konsisten pada setiap sesi pengujian. Langkah-langkah pengukuran dilakukan sebagai berikut:

1. Menyiapkan Rangkaian BPF dengan rentang kerja 362–1064 Hz dihubungkan ke input ADC ESP32.
2. Konfigurasi Parameter Sampling ESP32 dikonfigurasi dengan parameter tetap:
   - Resolusi ADC: 12-bit (0–4095)
   - Sampling rate:
   $$f_s = 2{,}1\ kHz\ \&\ 10\ k\text{Hz}$$
   - Jumlah sampel per blok FFT:
   $$N = 64\ \&\ 256$$
   - Window: Hann & Hamming
   - Parameter ini ditetapkan untuk menjaga konsistensi antara seluruh pengukuran.
3. Pemberian sinyal dari Generator Frekuensi Generator Frekuensi disetel pada nilai frekuensi acuan (f_{ref}) di dalam BPF (362–1064 Hz), hingga

melewati batas BPF. Setiap frekuensi dilakukan uji coba 5 kali pengulangan.

4. Akuisisi dan Pengolahan oleh ESP32 ESP32 membaca sampel sinyal analog melalui ADC, menormalkannya, kemudian menghitung FFT untuk blok 64 & 256 sampel.
5. Penerimaan dan Visualisasi Data pada Klien Aplikasi *JavaScript* menerima paket MQTT, memproses nilai FFT dan PSD, lalu menampilkannya menggunakan *Chart.js*.
6. Pencatatan parameter pengujian Untuk setiap sesi, parameter berikut dicatat:
   - Nilai frekuensi acuan (f_{ref})
   - Frekuensi terdeteksi (f_{meas})
   - Resolusi spektral (Δf)
   - Nilai FFT/PSD pada puncak frekuensi
   - Perhitungan SNR
   - Latensi *end-to-end*

2.2.4 Perhitungan Error Relatif Frekuensi

Evaluasi akurasi estimasi frekuensi dilakukan menggunakan parameter *error* relatif antara nilai input dan nilai hasil pengolahan sinyal. *Error* relatif dihitung menggunakan rumus:

$$\text{Error Relatif (\%)} = \left| \frac{f_{\text{ref}} - f_{\text{peak FFT/PSD}}}{f_{\text{ref}}} \right| \times 100\%$$

Dengan:

- f_{ref}: frekuensi acuan hasil input,
- $f_{\text{peak FFT/PSD}}$: frekuensi hasil FFT atau PSD.

3 Hasil dan Pembahasan

Pengujian dilakukan dengan menguji sistem menggunakan *sampling rate* sebesar 2,1 kHz dan 10 kHz dengan jumlah sampel 256 & 64. Gambar 3.1 menunjukkan hasil uji coba sistem dengan menggunakan input sinyal dari Generator Frekuensi.

Gambar 3.1 Sumber Input Generator Frekuensi Online

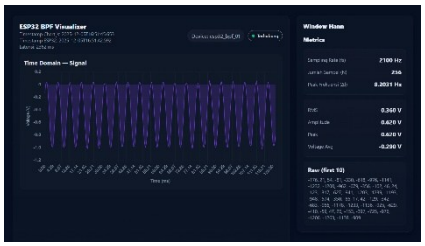
3.1 Hasil Pengujian

3.1.1 Visualisasi Hasil Pengukuran

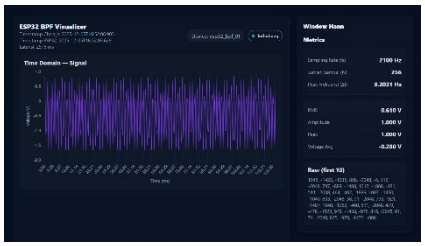
Pada bagian ini ditampilkan hasil visualisasi sinyal yang diperoleh dari sistem akuisisi menggunakan *Chart.js*. Visualisasi ini mencakup tampilan sinyal pada domain waktu, FFT, dan PSD.

3.1.1.1 Grafik Sinyal pada Domain Waktu (*Time Domain*)

Grafik yang ditampilkan pada Gambar 3.2 dan Gambar 3.3 memperlihatkan bentuk sinyal yang dihasilkan.



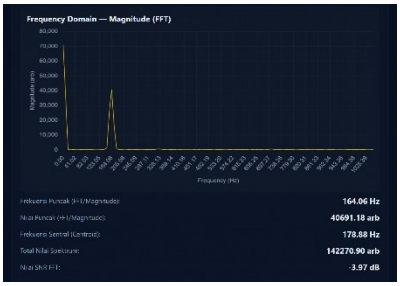
Gambar 3.2 *Time Domain* dengan Frekuensi 150 Hz



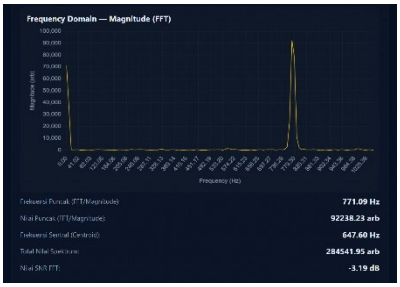
Gambar 3.3 *Time Domain* dengan Frekuensi 713 Hz

3.1.1.2 Grafik Transformasi *Fast Fourier Transform* (FFT)

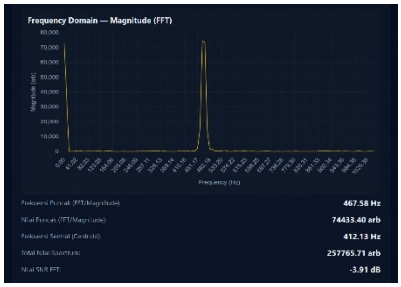
Grafik pada Gambar 3.4 hingga Gambar 3.6 menampilkan hasil transformasi *Fast Fourier Transform* (FFT) dari sinyal masukan.



Gambar 3.4 FFT dengan Frekuensi 150 Hz



Gambar 3.5 FFT dengan Frekuensi 713 Hz



Gambar 3.6 FFT dengan Frekuensi 1500 Hz

3.1.1.3 Grafik *Power Spectral Density* (PSD)

Grafik pada Gambar 3.7 hingga Gambar 3.9 merupakan hasil dari *Power Spectral Density* (PSD).

Gambar 3.7 PSD dengan Frekuensi 150 Hz

Gambar 3.8 PSD dengan Frekuensi 713 Hz

Gambar 3.9 PSD dengan Frekuensi 1500 Hz

3.1.2 Pengujian Menggunakan *Sampling Rate* 2,1 kHz dan 10 kHz

Pengujian dilakukan dengan menginput nilai frekuensi. Hasil pengujian dengan *Sampling Rate* 2,1 kHz dan jumlah sampel 256 dapat dilihat pada Tabel 3.1.

Tabel 3.1 *Sampling Rate* 2,1 kHz & Jumlah Sampel 256

$f_s = 2100\ Hz$ & $N = 256$					
No	Nilai Input (Hz)	BPF Amplitudo (V)	RMS (V)	FFT/DFT Frekuensi *Peak* (Hz)	*Error* (%)
1.	50	0,16	0,07	57,42	12,92
2.	150	0,43	0,23	164,06	8,57
3.	362	0,43	0,70	393,75	8,06
4.	500	0,83	0,52	549,61	9,03
5.	713	0,91	0,57	779,3	8,51
6.	900	0,93	0,57	992,58	9,33
7.	1064	0,9	0,55	935,16	13,78
8.	1500	0,83	0,49	451,17	232,47
9.	3000	0,6	0,33	902,34	232,47
10.	5000	0,49	0,27	812,11	515,68

Hasil pengujian dengan *Sampling Rate* 10 kHz dan jumlah sampel 256 dapat dilihat pada Tabel 3.2.

Tabel 3.2 *Sampling Rate* 10 kHz & Jumlah Sampel 256

$f_s = 10000\ Hz$ & $N = 256$					
No	Nilai Input (Hz)	BPF Amplitudo (V)	RMS (V)	FFT/DFT Frekuensi *Peak* (Hz)	*Error* (%)
1.	50	0,14	0,25	39,06	28,01
2.	150	0,26	0,49	234,38	36,00
3.	362	0,44	0,73	546,88	33,81
4.	500	0,52	0,86	742,19	32,63
5.	713	0,97	0,57	1054,69	32,40
6.	900	0,93	0,57	1328,13	32,24
7.	1064	0,9	0,55	1562,5	31,90
8.	1500	0,83	0,49	2187,5	31,43
9.	3000	0,6	0,33	4414,06	32,04
10.	5000	0,49	0,27	2656,25	88,24

3.1.3 Pengujian Menggunakan Window Hann dan Hamming

Pengujian ini membandingkan antara Window Hann dan Hamming pada *Sampling Rate* 2,1 kHz yang bisa dilihat di Tabel 3.3 berikut.

Tabel 3.3 *Sampling Rate* 2100 dengan Window Hann dan Window Hamming

$f_s = 2100\ Hz$ & $N = 256$					
No	Nilai Input (Hz)	Window			
		Hann		Hamming	
		FFT SNR (dB)	PSD SNR (dB)	FFT SNR (dB)	PSD SNR (dB)
1.	50	-5,81	-0,85	-5,66	0,04
2.	150	-2,4	2,51	-2,01	3,81
3.	362	-2,83	-0,12	-3,43	1,20
4.	500	-1,56	2,82	-1,06	4,17
5.	713	-2,86	-0,19	-2,76	0,18
6.	900	-3,01	0,06	-2,62	0,22
7.	1064	-3,1	-0,1	-2,74	0,19

8.	1500	-2,06	2,33	-1,74	3,53
9.	3000	-3,35	1,36	-3,02	2,28
10.	5000	-0,98	-0,16	-4,81	-0,40

3.1.4 Pengujian Perbandingan Jumlah Sampel 256 & 64

Pengujian ini dilakukan untuk mengetahui pengaruh jumlah sampel terhadap ketelitian estimasi frekuensi. Dua konfigurasi jumlah sampel diuji, yaitu 256 & 64. Pengujian dilakukan pada frekuensi 713 Hz, dapat dilihat pada Tabel 3.4 dan Tabel 3.5 berikut.

Tabel 3.4 Frekuensi *Peak* pada Pengujian Jumlah Sampel 256 & 64

Sampling Rate (Hz)	Jumlah Sampel (N)	BPF Amplitudo (V)	FFT/PSD Frekuensi *Peak* (Hz)	*Error* (%)
2100	256	0,97	1054,69	32,40
	64	0,84	1093,75	34,81
10000	256	0,91	779,3	8,51
	64	0,88	787,5	9,46

Tabel 3.5 *Signal-to-Noise Ratio* pada Pengujian Jumlah Sampel 256 & 64

Sampling Rate (Hz)	N	FFT SNR (dB)	PSD SNR (dB)	Nilai Puncak FFT (arb)	Nilai Puncak PSD (V^2/Hz)
2100	256	-3,04	2,34	102943,85	8230,60
	64	-1,9	1,72	22099,76	6707,35
10000	256	-2,86	-0,19	83930,07	19316,86
	64	-1,36	2,54	25276,88	9612,47

3.1.5 Pengujian Latensi *End-to-End* Sistem

Pengujian latensi dilakukan untuk dua konfigurasi berbeda, yaitu *Sampling Rate* 2,1 kHz (N = 256) dan Sampling Rate 10 kHz (N = 256). Masing-masing pengujian dilakukan sebanyak lima kali untuk mendapatkan nilai delay aktual antara data yang dikirim ESP32 dan data yang diterima aplikasi Chart.js. Tabel 3.6 dan Tabel 3.7 berikut menunjukkan hasil pengukuran.

Tabel 3.6 Pengujian Latensi pada *Sampling Rate* 2,1 kHz dan Jumlah Sampel 256

$T_{ADC} = 121{,}9$ ms		
Timestamp		Latensi (ms)
Chart.js	ESP32	
2025-12-05T 15:50:17.227	2025-12-05T 15:50:14.762	2465
2025-12-05T 15:50:12.033	2025-12-05T 15:50:09.481	2552
2025-12-05T 15:50:06.620	2025-12-05T 15:50:04.201	2419

Tabel 3.7 Pengujian Latensi pada *Sampling Rate* 10 kHz dan Jumlah Sampel 256

$T_{ADC} = 25{,}6$ ms		
Timestamp		Latensi (ms)
Chart.js	ESP32	
2025-12-05T 15:47:15.424	2025-12-05T 15:47:12.843	2625
2025-12-05T 15:47:20.628	2025-12-05T 15:47:18.027	2555
2025-12-05T 15:47:41.386	2025-12-05T 15:47:38.761	2581

3.2 Pembahasan

3.2.1 Pembahasan Bentuk Grafik Hasil Pengukuran

Pada bagian ini dilakukan analisis terhadap tiga bentuk visualisasi sinyal yang dihasilkan sistem, yaitu grafik domain waktu, grafik hasil transformasi FFT, dan grafik PSD.

3.2.1.1 Pembahasan Grafik Sinyal pada Domain Waktu (Time Domain)

Pada grafik domain waktu, terlihat perbedaan kepadatan gelombang berdasarkan frekuensi masukan. Pada sinyal 150 Hz (Gambar 3.2), bentuk gelombangnya tampak lebih renggang karena periode sinyal lebih panjang. Sementara itu, pada 713 Hz (Gambar 3.3), gelombang tampak lebih rapat dan osilasinya lebih cepat. Pola ini menunjukkan bahwa sistem mampu merekam perubahan amplitudo sinyal sesuai karakteristik frekuensinya.

3.2.1.2 Pembahasan Grafik Transformasi *Fast Fourier Transform* (FFT)

Pada grafik FFT untuk sinyal 150 Hz (Gambar 3.4), puncak magnitude terdeteksi di sekitar 164,06 Hz dengan nilai sekitar 40.691 arb, yang menunjukkan estimasi frekuensi masih berada dekat dengan nilai input. Hal serupa terlihat pada sinyal 713 Hz (Gambar 3.5), di mana puncaknya muncul pada 779,30 Hz dengan magnitude tertinggi sebesar 92.238,23 arb.

Namun, pada sinyal 1500 Hz (Gambar 3.6), puncak FFT justru muncul pada 493,19 Hz dengan magnitude 74.433,40 arb. Perpindahan puncak ini merupakan efek *aliasing*, karena frekuensi input 1500 Hz berada jauh di atas batas Nyquist dari *sampling rate* 2,1 kHz (Nyquist = 1050 Hz).

3.2.1.3 Pembahasan Grafik *Power Spectral Density* (PSD)

Pada grafik PSD untuk sinyal 150 Hz (Gambar 3.7), puncak spektrum muncul pada 164,06 Hz dengan nilai 6.122,28 V^2/Hz. Untuk sinyal 713 Hz (Gambar 3.8), puncak muncul pada 771,09 Hz dengan nilai 31.113,13 V^2/Hz. Sedangkan untuk sinyal 1500 Hz (Gambar 3.9), puncak PSD bergeser ke 467,58 Hz dengan nilai 20.542,77

V^2/Hz. Perilaku ini konsisten dengan FFT, di mana frekuensi di atas batas Nyquist.

3.2.2 Perbandingan Hasil Pengujian pada Sampling Rate 2,1 kHz dan 10 kHz

Pada bagian ini dilakukan analisis terhadap pengaruh perubahan *sampling rate* pada Tabel 3.1 dan Tabel 3.2 terhadap hasil estimasi frekuensi menggunakan FFT dan PSD. Dua aspek utama yang diamati adalah nilai amplitudo dan persentase *error* terhadap frekuensi input**.**

3.2.2.1 Perbandingan Amplitudo

Pada *sampling rate* 2,1 kHz, nilai BPF Amplitudo menunjukkan kenaikan yang cukup stabil dari frekuensi rendah hingga sekitar 900 Hz. Amplitudo meningkat dari 0,16 V (50 Hz) hingga mencapai kisaran 0,90–0,93 V pada frekuensi 713–900 Hz, kemudian kembali menurun pada frekuensi yang lebih tinggi, yaitu pada 3000 Hz (0,60 V) dan 5000 Hz (0,49 V).

Sementara itu, pada *sampling rate* 10 kHz, amplitudo cenderung lebih kecil pada frekuensi rendah (misalnya 0,14 V pada 50 Hz), kemudian meningkat hingga mencapai 0,97 V pada 713 Hz, mirip dengan pola pada 2,1 kHz.

Secara keseluruhan, perubahan sampling rate tidak terlalu mengubah bentuk respon amplitudo. Rentang amplitudo maksimum tetap terjadi di sekitar 713–900 Hz pada kedua konfigurasi sampling, menandakan BPF bekerja dengan baik karena rentang *filter* berada di 362–1064 Hz.

3.2.2.2 Perbandingan *Error* Input

Perbedaan yang paling signifikan antara *sampling rate* 2,1 kHz dan 10 kHz terlihat pada persentase *error* estimasi frekuensi.

Pada sampling rate 2,1 kHz, hasil pengujian menunjukkan bahwa estimasi frekuensi masih cukup akurat pada rentang input 50–900 Hz, dengan error berada pada kisaran 8–13%. Namun, ketika frekuensi input mencapai 1064 Hz, akurasi mulai menurun dan error berubah menjadi sangat besar. Setelah melewati batas Nyquist, seperti pada input 1500–5000 Hz, efek aliasing menjadi sangat dominan sehingga frekuensi puncak yang terdeteksi bergeser ke daerah yang lebih rendah, menghasilkan error yang sangat besar hingga ratusan persen.

Berbeda pada *sampling rate* 10 kHz, frekuensi input dari 50–3000 Hz masih jauh berada di bawah batas Nyquist (5000 Hz), sehingga sinyal tidak mengalami aliasing dan estimasi frekuensi tetap stabil. Meskipun *error* pada rentang ini berada di sekitar 31–36%, nilainya konsisten dan tidak mengalami lonjakan ekstrem. Namun, ketika frekuensi input mencapai 5000 Hz, yang berada tepat pada batas Nyquist, akurasi kembali menurun.

3.2.3 Perbandingan Window Hann dan Hamming

Pada bagian ini bertujuan untuk membandingkan performa Window Hann dan Hamming berdasarkan data pada Tabel 3.3. Secara umum, nilai SNR pada PSD selalu lebih tinggi dibandingkan pada FFT. Hal ini terjadi karena PSD melakukan proses *averaging* sehingga mampu mengurangi fluktuasi noise dan spectral leakage lebih baik.

Window Hamming secara konsisten memberikan SNR PSD yang lebih baik dibandingkan Hann. Pada 50 Hz, Hann menghasilkan PSD SNR -0,85 dB, sedangkan Hamming mencapai 0,04 dB. Pola serupa terlihat pada 150 Hz (2,51 dB menjadi 3,81 dB) dan 500 Hz (2,82 dB menjadi 4,17 dB). Bahkan pada 900 Hz, Hamming tetap lebih tinggi (0,22 dB dibanding 0,06 dB). Pada frekuensi lebih tinggi seperti 3000 Hz, Hamming kembali unggul dengan PSD SNR 2,28 dB dibanding Hann 1,36 dB. Meskipun pada 5000 Hz terjadi penurunan performa pada kedua window, Hamming tetap sedikit lebih baik dengan PSD SNR –0,40 dB dibanding Hann -0,16 dB.

Dari keseluruhan hasil, dapat disimpulkan bahwa metode PSD secara konsisten memberikan SNR lebih tinggi daripada FFT**,** dan Window Hamming memberikan performa paling stabil dalam menekan spectral leakage, terutama pada frekuensi menengah hingga mendekati batas Nyquist.

3.2.4 Perbandingan Jumlah Sampel 256 dan 64

Pada bagian ini bertujuan untuk melihat bagaimana jumlah sampel mempengaruhi kualitas estimasi sinyal berdasarkan data Tabel 3.4 dan Tabel 3.5. Pengujian dilakukan pada frekuensi 713 Hz menggunakan dua *sampling rate*, yaitu 10 kHz dan 2,1 kHz.

Pada *sampling rate* 2,1 kHz, penggunaan 256 sampel menghasilkan kualitas spektral yang lebih baik, dengan puncak FFT sebesar 83930,07 arb dan PSD 19316,86 V^2/Hz. Pada N = 64, nilai-nilai ini turun menjadi 25276,88 arb dan 9612,47 V^2/Hz. Penurunan puncak menunjukkan bahwa spektrum kehilangan ketajamannya akibat resolusi frekuensi yang lebih rendah.

Pola serupa terlihat pada *sampling rate* 10 kHz. Dengan 256 sampel, puncak FFT mencapai 102943,85 arb dan puncak PSD 8230,60 V^2/Hz. Ketika jumlah sampel dikurangi menjadi 64, SNR FFT memang sedikit membaik (–1,90 dB), tetapi puncak FFT merosot ke 22099,76 arb dan puncak PSD menjadi 6707,35 V^2/Hz.

Secara keseluruhan, analisis ini menunjukkan bahwa jumlah sampel memiliki pengaruh yang jauh lebih kuat terhadap nilai puncak spektral dibandingkan terhadap nilai SNR.

3.2.5 Analisis Latensi Sistem Pengiriman Data

Analisis ini fokus pada pengaruh sampling rate, jumlah sampel (N) dan latensi ADC terhadap delay *end-to-end* dari ESP32 hingga data ditampilkan pada aplikasi

Chart.js. Pengukuran dilakukan pada *sampling rate* 2,1 kHz dan 10 kHz.

Berdasarkan Tabel 3.6, maka didapatkan nilai rata-rata latensi untuk *sampling rate* 2,1 kHz sebagai berikut:

$$\frac{2465 + 2552 + 2419}{3} = 2478{,}6\ ms$$

Sedangkan pada Tabel 3.7, untuk *sampling rate* 10 kHz, rata-rata latensinya sebagai berikut:

$$\frac{2625 + 2555 + 2581}{3} = 2587\ ms$$

Hasil pengukuran menunjukkan bahwa sampling rate 2,1 Hz menghasilkan latensi *end-to-end* lebih rendah (2478,6 ms) dibandingkan 10 kHz (2587 ms). Perbedaan ini terutama disebabkan oleh meningkatnya beban data dan proses akuisisi pada sampling rate tinggi, ditambah latensi ADC yang lebih besar.

4 Kesimpulan

Sistem berhasil menampilkan dan menganalisis sinyal menggunakan BPF, FFT, dan PSD sesuai tujuan penelitian. BPF bekerja efektif pada rentang 362–1064 Hz dengan puncak amplitudo di 713–900 Hz dan peredaman memadai di luar band. FFT dan PSD memberikan estimasi frekuensi yang akurat selama sinyal berada di bawah batas Nyquist. Pada 2,1 kHz hanya frekuensi hingga 900 Hz yang stabil, sedangkan 10 kHz mampu mengukur dengan baik pada 50–3000 Hz.

PSD memberikan SNR lebih tinggi, dan Window Hamming menunjukkan performa terbaik dalam menekan spectral leakage. Jumlah sampel besar (N = 256) menghasilkan puncak spektral lebih tajam dibanding sampel kecil (N = 64). Dari sisi waktu, sampling rate 2,1 kHz menghasilkan latensi lebih rendah dibanding 10 kHz akibat beban data yang lebih kecil, selain delay tambahan dari jaringan MQTT.

Penelitian ini masih terbatas pada rentang filter, kondisi jaringan, dan pengujian laboratorium. Perbaikan dapat dilakukan melalui optimasi windowing, pemilihan sampel adaptif, dan pengujian pada kondisi lapangan yang lebih variatif.

DAFTAR PUSTAKA

[1] M. H. Utama, M. Fathurahman, B. Multimedia, T. Elektro, P. N. Jakarta, and K. Depok, "Rancang Bangun Monitoring Sinyal Radio VHF Band II Berbasis Graphical User Interface Abstrak," vol. 4, no. 1, pp. 445–455.

[2] I. K. Manik, I. G. Agung, and G. Arya, "Identifikasi dan Klasifikasi Suara Vokal Menggunakan Metode Fast Fourier Transform," vol. 3, pp. 815–824, 2025.

[3] S. Pratama and D. Tresnawan, "Sistem Monitoring Spektrum Akupansi Band AM , FM dan Trunking Menggunakan RTL SDR 2832U DVB-T Tuner Dongles Berbasis Visual Studio," vol. 6, no. 2, pp. 38–48, 2021, doi: 10.37253/telcomatics.v6i2.6343.

[4] L. Caroles, *HUBUNGAN NILAI MODULUS KEKAKUAN PADA ALAT MARSHALL TEST TERHADAP ALAT LIGHT WEIGHT DEFLECTOMETER (LWD) LABORATORIUM*. wawasan Ilmu. [Online]. Available: https://www.google.co.id/books/edition/HUBUNGAN_NILAI_MODULUS_KEKAKUAN_PADA_ALA/ghCmEAAAQBAJ?hl=id&gbpv=1&dq=Fungsi+Algoritma+Fast+Fourier+Transform&pg=PA30&printsec=frontcover

[5] F. A. ROSYID, S. LARASATI, and W. PAMUNGKAS, "Teknik Pengiriman File Audio Berbasis Software Defined Radio pada Kanal Komunikasi dalam Ruangan dengan Multicarrier OFDM," *ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 12, no. 4, p. 907, 2024, doi: 10.26760/elkomika.v12i4.908.