📖 Chapter: Spectrum Analysis of Captured IQ Data

1. What Are We Doing?

We are building the **spectrum analyzer component** of our FM receiver project.

- **Goal:** Convert raw IQ samples → FFT → spectrum plot.
- **Place in MVC:**
    - **Model:** Handles IQ data & DSP (FFT, normalization).
    - **View:** Spectrum visualization (Matplotlib/PyQtGraph).
    - **Controller:** Orchestrates reading & plotting.

This exercise is the *foundation* for later modules (FM demod, waterfall, real-time display).

---

2. Why Are We Doing It?

- To **see frequency content** of signals around our tuned frequency (e.g., FM @ 104.8 MHz).
- Spectrum plots show **where the power is** in frequency domain.
- Without spectrum view, you are “blind” — you have IQ data but no way to tell if you captured noise, FM, or Wi-Fi.
- Engineers use this to **debug antennas, SDR settings, interference**.

---

3. How Are We Doing It? (Step-by-Step DSP Flow)

Step	Action	Why	MVC Placement
1. Read Binary IQ	Load `.bin` file into numpy array	Raw SDR data is interleaved	`src/model/file_reader.py`

Step	Action	Why	MVC Placement
Data		8-bit unsigned samples (`I,Q,I,Q,…`).	
2. Normalize Samples	`(raw - 127.5)/127.5` → range `[-1,+1]`	Converts byte values to proper complex samples	`src/model/utils.py`
3. Convert to Complex	`I + jQ` form	DSP requires complex notation for FFT & demod	`src/model/file_reader.py`
4. Apply Window (Hanning)	Multiply samples before FFT	Reduces spectral leakage (sharp cutoffs create spurious tones)	`src/model/dsp_core.py`
5. FFT & Shift	Compute `np.fft.fft` then `np.fft.fftshift`	FFT → spectrum; shift → center DC at 0 Hz	`src/model/dsp_core.py`
6. Convert to dB Scale	`20*log10(	FFT	)`
7. Plot Spectrum	Frequency vs Power	Visual check of captured FM station	`src/view/spectrum_plot.py`

4. Mini-Tasks (Hands-On)

Task 1 — Inspect Raw Data

```python
import numpy as np

raw = np.fromfile("data/raw/Prac_FM_104_8MHz_20250820.bin", dtype=np.uint8)
print(raw[:20])
```

☞ *See the raw numbers (0–255). Confirm interleaved structure.*

Task 2 — Build Complex Signal

```python
iq = (raw[0::2] - 127.5)/127.5 + 1j*(raw[1::2] - 127.5)/127.5
print(iq[:5])
```

☞ *Check first 5 complex samples. Do you see floats ~−1…+1?*

Task 3 — FFT on Small Block

```python
N = 2048
fft_data = np.fft.fftshift(np.fft.fft(iq[:N]))
spectrum = 20*np.log10(np.abs(fft_data))
```

☞ *Plot it with Matplotlib, check frequency axis.*

Task 4 — Full Spectrum Plotter (MVP Module)

Write `src/view/spectrum_plot.py` that:

- Reads IQ via `file_reader`
- Calls `dsp_core.fft_block()`
- Plots frequency vs power

---

5. Summary

- You now understand the **why** behind each DSP step.
- Each step has a **place in MVC project**.
- Small tasks → build confidence → later scale to **real-time streaming** & **waterfall**.