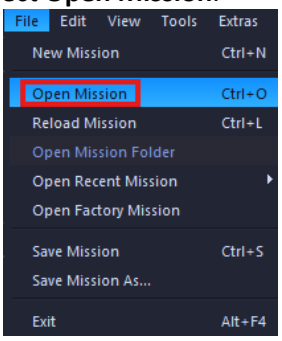
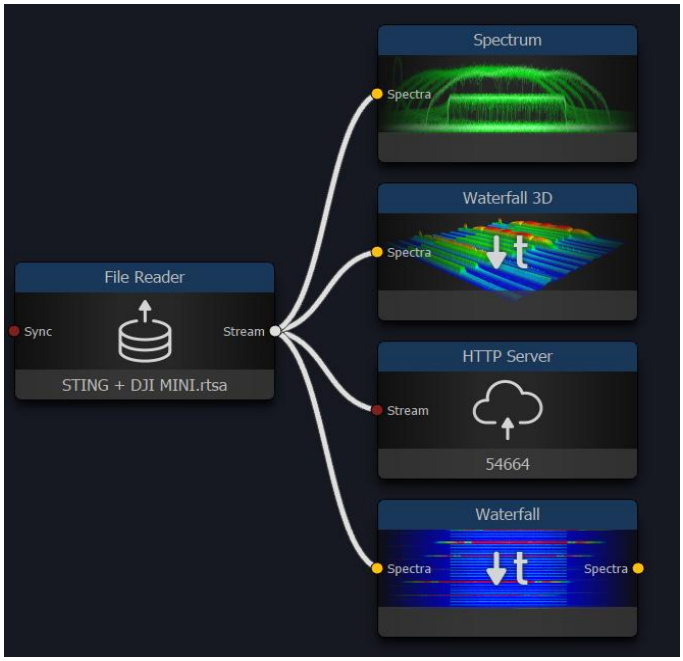
Data Transfer from Aaronia RSTA-Suite PRO software to MATLAB over HTTP server

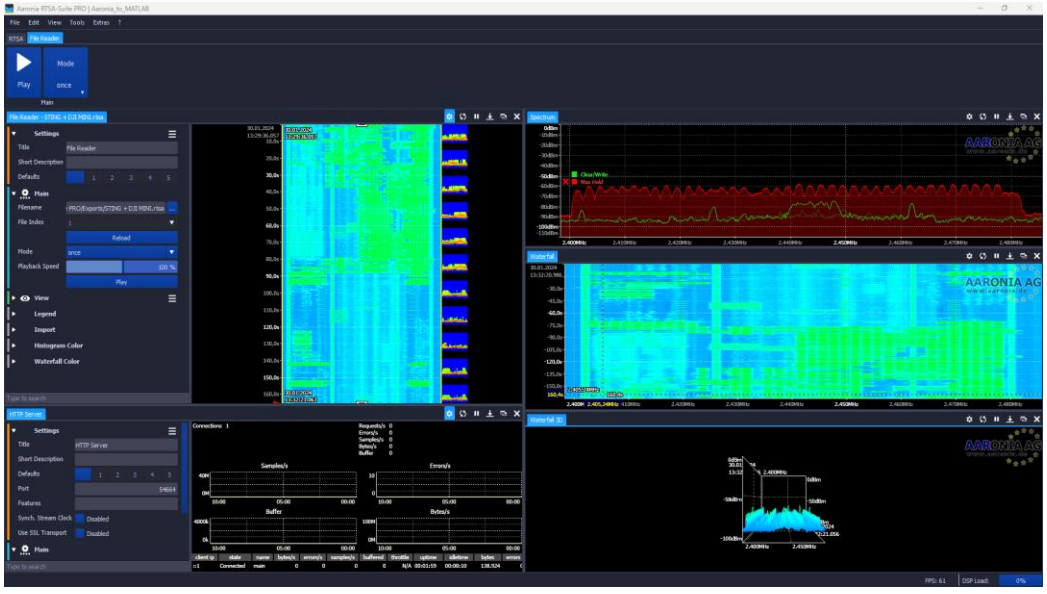
1. Launch the Aaronia RSTA-Suite PRO software.
2. Navigate to the menu bar, click on **File**, and select **Open Mission**.



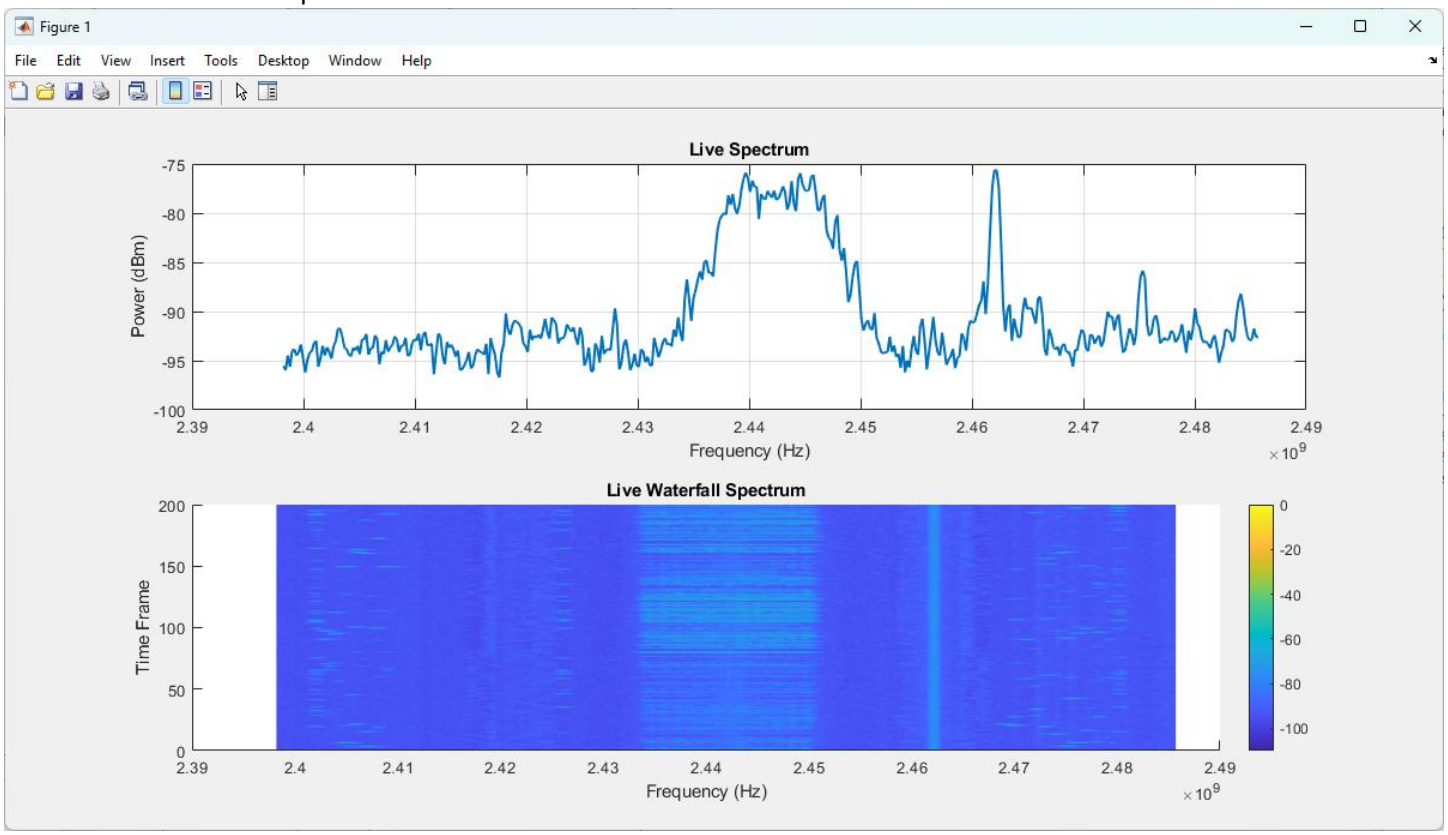
3. Locate and open the `Aaronia_to_MATLAB.rmix` file from the directory.
4. Below image shows the block diagram for this project file. It enables reading recorded data, while displaying waterfall and spectrum graphs within the Aaronia software, and simultaneously transmitting the data via an HTTP Server.



5. Overview of the Aaronia software while showing and sending recorded data

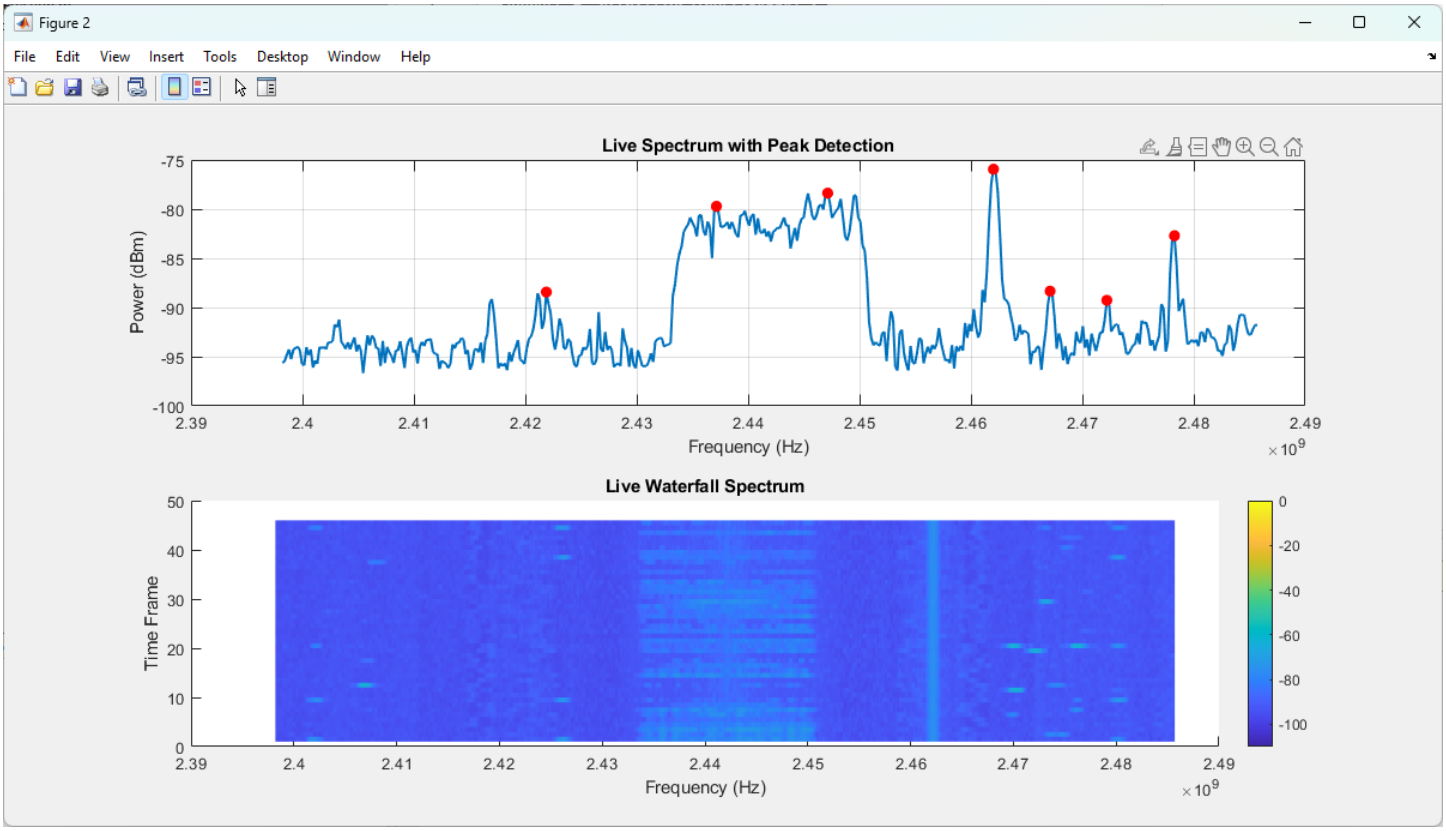


6. Open MATLAB and load the `Aaronia_to_MATLAB.m` script.
7. Execute the MATLAB script to run the code.



Peak Detection

1. Follow the same procedure for Aaronia software as given in the first part
2. Open MATLAB and load the `Peak_Detection.m` script.
3. Execute the MATLAB script to run the code.



This MATLAB code implements a **Live Spectrum Analyzer** designed to visualize and analyze real-time spectral data from a server. It features a **live spectrum plot with peak detection** and a **waterfall plot** to track the spectral data over time.

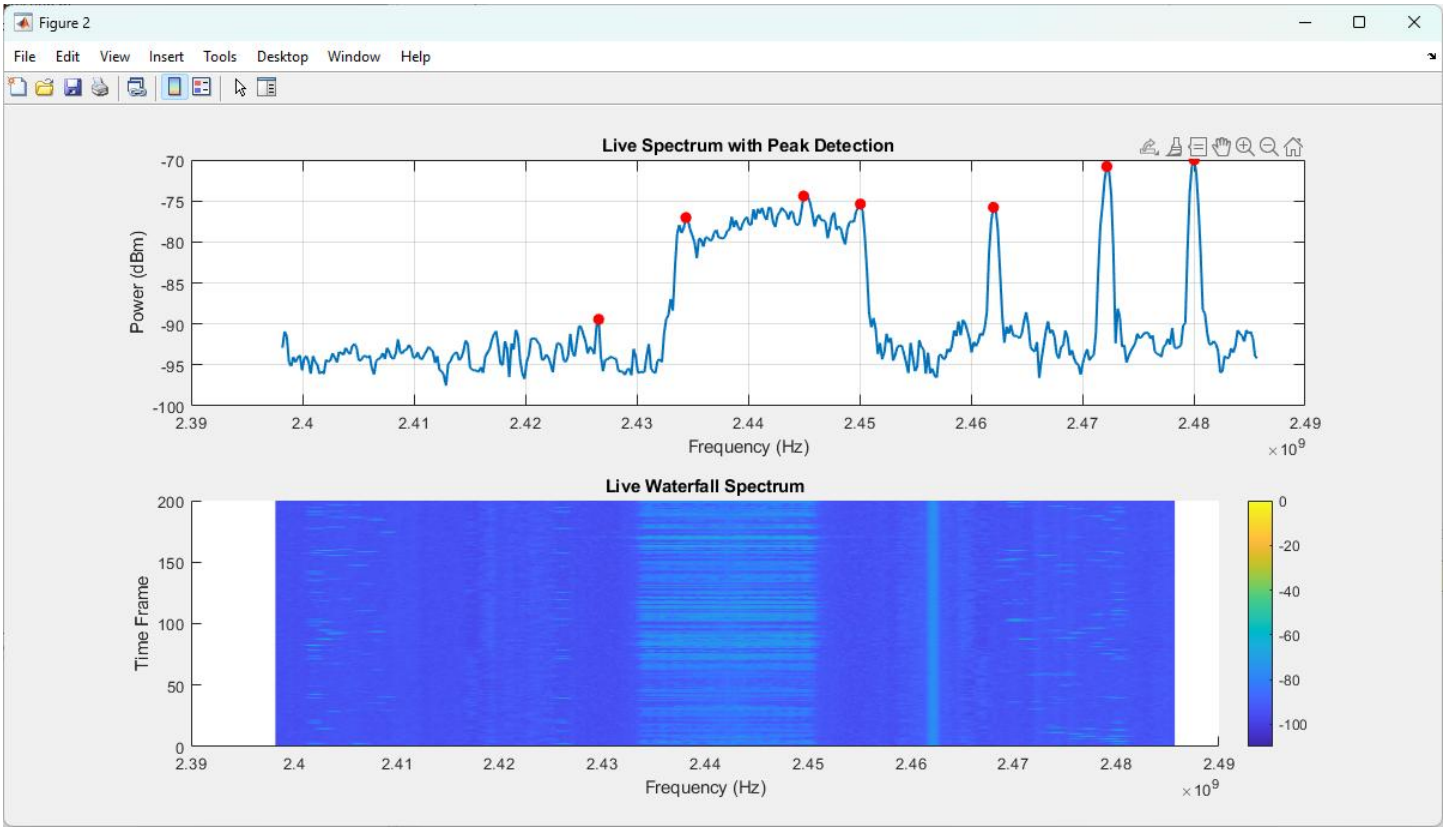
Algorithm Overview

1. **Initialization:**
   - Define server parameters (address, endpoint) and data acquisition settings (number of frames, pause duration).
   - Set up the frequency range for the spectral data.
   - Initialize plots:
     - Spectrum Plot: Displays the current power spectrum with detected peaks.
     - Waterfall Plot: Provides a heatmap-style visualization of spectral power over time.
2. **Live Data Collection:**
   - Start a loop for `numFrames` iterations to fetch and display data in real time:
     - Send HTTP Request: Use MATLAB's HTTP library to send a GET request to the server endpoint.
     - Receive and Process Response:
       - Check the status of the server response.
       - If successful, extract the spectral data (samples) from the server response.
     - Update Plots:
       - Spectrum Plot: Update the power spectrum with the latest data.
       - Peak Detection: Identify peaks in the spectrum using the `findpeaks` function based on:
         a. Minimum peak height.
         b. Minimum prominence.
         c. Minimum distance between peaks.
       - Waterfall Plot: Add the latest data to the waterfall plot for time-based visualization.
       - Shift the waterfall data to maintain a rolling history for continuous display.

3. Error Handling:
    - If an error occurs (e.g., communication failure), display a warning and gracefully exit the loop.

Channel Detection

1. Follow the same procedure for Aaronia software as given in the first part
2. Open MATLAB and load the `Channel_Detection.m` script.
3. Execute the MATLAB script to run the code.



Output:

```
>> Channel_Detection
Detected Channels:
OFDM channel approx 16.97 MHz detected at 2.443 GHz
Narrowband signal detected: 1.16 MHz at 2.462 GHz
```

This MATLAB code implements a **Live Spectrum Analyzer** that fetches spectral data from a server, processes it in real-time, and displays the data in two plots:
Spectrum Plot: Displays the current power spectrum with peak detection.
Waterfall Plot: Displays a heatmap-style visualization of the spectrum across multiple time frames.
The code also performs **post-processing** to detect and classify different types of channels (such as **OFDM** and **Bluetooth**) based on the frequency and bandwidth of detected peaks in the spectrum.

Algorithm Overview

1. **Initialization:**
   - Define the server address, endpoint, and parameters (e.g., number of frames, pause time).
   - Set the frequency range for the spectrum data collection.
   - Initialize the figure and subplots:
     - Spectrum Plot for real-time spectral data with peak markers.
     - Waterfall Plot for visualizing the spectrum over time (heatmap-style).
2. **Live Data Collection (Real-Time Loop):**
   - Start a loop to fetch spectral data from the server for `numFrames` iterations.
   - Data Fetching:
     - Use the `RequestMessage` and send function from the MATLAB HTTP library to send a request to the server and get spectral data.
     - Check Server Response: If the server responds successfully, extract the spectral data (power samples).

- Update Plots:
  - Spectrum Plot: Update the real-time spectrum plot with the new data.
  - Peak Detection: Use `findpeaks` to identify peaks in the spectrum based on minimum peak height, minimum prominence, and minimum distance between peaks.
  - Waterfall Plot: Update the waterfall plot to include the new spectrum data for visualization of changes over time.
- Shift Data for Continuous Display: As the loop progresses, old data in the waterfall plot is shifted to simulate continuous display.

3. Post-Processing (After Data Collection):

- After collecting the 200 frames of data, compute the average spectrum over all the frames (`avgSpectrum`).
- Channel Detection:
  - Use `findpeaks` on the average spectrum to detect the peaks (i.e., channels) present.
  - For each peak:
    - Measure channel width (bandwidth).
    - Classify the channel based on its width:
      a. OFDM Channel: If the width is greater than or equal to 15 MHz (commonly used for Wi-Fi).
      b. Bluetooth Channel: If the width is between 2 MHz and 15 MHz (commonly used for Bluetooth).
      c. Narrowband Signal: If the channel width is less than 2 MHz (could be any narrow signal).
    - Display Results: Print detected channels, including the frequency and type of the detected signal (OFDM, Bluetooth, or narrowband).

4. Error Handling:

- If there is an issue in data fetching (e.g., server communication failure), display a warning with the error message.