ICHNOS

A WEB APPLICATON FOR REAL-TIME DATA VISUALIZATION

F. Schirru[1], A. Tarchi[1], P. Castangia[1], E. Ladu[1]

[1] INAF - Osservatorio Astronomico di Cagliari, via della Scienza 5, 09047, Selargius (CA), Italy

Reviewer:
Dr. Andrea Melis
Dr. Alessandro Cabras

Anno 2026

Abstract

This technical report introduces ICHNOS, a Python-based quick-look web application designed to visualize both single spectra (obtained in position switching and nodding modes) and spatial distributions (continuum and spectroscopic maps) acquired at the Sardinia Radio Telescope using different receiver–backend combinations. After introducing the motivations behind this work, the report presents a technical overview of the software, highlighting its asynchronous architecture and its ability to handle real-time data streaming for different acquisition modes. This is followed by a detailed description of the graphical user interface and its interactive plotting capabilities. A test session aimed at evaluating the performance of the quick-look system with large datasets collected by the Sardinia Radio Telescope is finally presented, demonstrating the system's efficiency in realistic operational scenarios.

Contents

1 Introduction

Modern radio astronomical observations rely on extremely sensitive instruments capable of acquiring large volumes of data at high cadence and under complex operational conditions. In the case of next-generation radio telescopes, such as the Sardinia Radio Telescope (SRT), the variety of observing modes—ranging from continuum and high-resolution spectroscopy to pulsar and Very-Long-Baseline Interferometry (VLBI) measurements—requires constant and immediate monitoring of the quality of the acquired data.

In this context, a quick-look system plays a crucial role. It provides a timely and simplified visualization of the raw data, allowing operators and researchers to:

- Quickly verify the correct functioning of the acquisition chain, identifying in real-time any hardware, configuration, or pointing issues.
- Detect Radio Frequency Interference (RFI) contamination, unexpected variations in system noise, or acquisition chain anomalies that could compromise an entire observing session if not identified promptly.
- Optimize the ongoing observation by adjusting parameters such as bandwidth, integration time, central frequency, or scan mode.
- Reduce diagnosis times by limiting the need to analyze complete datasets or run complex calibration pipelines before uncovering potential problems.

Indeed, the increasing capacity of backends and operational bandwidths has created the need for tools that allow immediate decision-making without waiting for full data processing. In this context, a quick-look system does not replace scientific pipelines, but rather constitutes the first level of quality control, essential for ensuring the efficiency of the entire observing cycle.

For a multi-purpose radio telescope like the SRT, where heterogeneous and sometimes complex experiments may be carried out in parallel, a quick-look system is therefore a strategic operational component, capable of improving the reliability of observations, reducing antenna time wastage, and providing immediate awareness of the instrument status.

This report introduces ICHNOS, a web application developed for real-time visualization of *single* spectra, obtained, for instance, in Position Switching (PSW) and Nodding (NOD) [1] modes and *spatial distributions* (continuum and spectroscopic maps, hereafter *mapping mode*) from observations conducted at the SRT using single- and multi-feed receivers in combination with different backends (TOTALPOWER [2], SARDARA [3], and SKARAB [4]). The application is designed to fulfill two fundamental requirements:

- *Responsiveness to high data rates*: it must keep pace with the rapid production of sub-scans, generate plots in real-time, and ensure that the processing pipeline does not introduce delays or backlogs, even when handling high-resolution or wide-band data streams.
- *Immediate interpretability*: plots must be clear, readable, and easy to interpret, allowing users to quickly identify anomalies, trends, and operational conditions, and thus obtain an instant assessment of the ongoing observation.

To meet these requirements, ICHNOS operates in two main visualization modes.

In *single-spectrum* mode, ICHNOS is designed to display, for each sub-scan, the instantaneous bandpass, providing separate views for each polarization and, when applicable, for each feed. These capabilities allow observers to verify in real time the stability and quality of the bandpass across the full instrumental configuration, ensuring that all components—feeds, polarizations, and backend settings—operate as expected.

In *spatial distribution* mode (e.g., OTF, cross-scan, raster-scan maps), by correlating spectral power with antenna pointing coordinates, ICHNOS provides a real-time point-cloud representation of the scanned area. This feature allows for immediate verification of the spatial coverage and the detection of point-like or extended sources directly during the acquisition process, ensuring the scientific integrity of the mapping strategy.

ICHNOS does not handle advanced data visualization and analysis. Instead, these tasks will be addressed by a new tool called *Scan-Viewer*, which will be described in a separate report (Schirru et al., in preparation). In particular, for spectroscopic observations (e.g., taken in PSW and NOD modes), Scan-Viewer can generate processed plots such as $(\text{on} - \text{off})/\text{off}$, providing enhanced insight into the data that cannot be achieved through real-time monitoring alone.

This two-tier approach—combining immediate assessment with ICHNOS and detailed post-acquisition analysis with Scan-Viewer—ensures both operational efficiency and scientific rigor.

2 System Architecture and Implementation

ICHNOS is a full-stack web application designed for real-time monitoring of Flexible Image Transport System (FITS)[1] files produced by the acquisition instruments of the SRT. It is organized into three main components:

- **Server Core** (Flask/SocketIO): handles web connections and real-time data transmission to the frontend.
- **I/O Monitoring** (Watchdog): detects file system changes, including on network drives, to identify new or modified FITS files.
- **Data Processing** (AstroPy/Bokeh): parses FITS files and generates interactive visualization products, including spectral plots and spatial scatter maps.

The backend is implemented in Python with a multithreaded design, leveraging asynchronous execution. This design ensures low-latency feedback on scan coverage and receiver stability while optimizing computational resources. Frontend interactions use JavaScript and Bootstrap, with dynamic updates via the Socket.IO client library.

2.1 Configuration and Startup

System startup is managed by `app.py`, which can be executed in two modes: *debug* mode and *production* mode, determined by a command-line argument (`-d` for *debug* mode). The application implements a dynamic drive-discovery logic: it parses the `config.ini` file to identify all available storage units defined under the `[Drives]` section.

[1] See Appendix A for more details on the FITS file structure.

In *debug* mode, only the local directory is monitored. In *production* mode, ICHNOS automatically aggregates all paths starting with the `remote_drive` prefix. If the user is authenticated, the system appends the specific `<username>` subdirectory to each discovered drive path. This multi-path approach allows the application to simultaneously monitor data streams coming from different backends or storage nodes (e.g., from the SARDARA and SKARAB storage units) without additional configuration.

2.2 FITS Data Flow and Asynchronous Logic

The workflow is fully asynchronous and driven by file system events to ensure near-real-time processing.

2.2.1 Monitoring (Module `fits_watcher.py`)

The monitoring module uses Watchdog with a PollingObserver, selected for reliability on network file systems (NFS/CIFS) where native OS events may be unreliable. A single observer instance is used to schedule multiple monitoring tasks, one for each path identified during the startup phase.

The Watchdog-based monitoring thread scans the configured directories at regular intervals, detecting only complete and relevant FITS files by checking file extensions (`.fits` or `.fits#`) and ensuring that the file is closed before processing. Events for ready files are passed to a processing thread (`_safe_process_file`), while a global lock-protected set (`_processing_files`) prevents duplicate processing of the same file across multiple polling cycles.

2.2.2 Data Processing (Module `fits_processor.py`)

The data processor handles a basic scientific analysis and visualization. Metadata are extracted from primary headers and binary table extensions to retrieve critical parameters such as source information, observing band and frequency, spectral channels, and receiver feeds used to acquire the astronomical data.

For PSW and NOD modes, raw intensity data are averaged across all acquisition rows per channel to produce a representative spectrum for each polarization (LL, RR, LR, RL) and feed. For mapping modes (e.g., OTF, cross-scan, raster-scan), the module extracts antenna pointing coordinates (AZ/EL for Azimuth/Elevation or RA/Dec for Right Ascension/Declination) and correlates them with the integrated power of each sub-scan. These spatial data are streamed to the frontend to update the interactive point-cloud map.

Interactive plots are generated with Bokeh and served as dynamic web applications via the Bokeh Server, rather than being saved as standalone HTML files. The point-cloud visualization utilizes a dedicated `ColumnDataSource` that is updated via WebSocket streaming, allowing the map to grow dynamically as new sub-scans are incrementally processed. Extracted metadata and plot references are aggregated into a payload and transmitted to the frontend via Socket.IO.

2.2.3 Error Handling and Robustness

The data processing pipeline includes several mechanisms to ensure robustness in the presence of incomplete, corrupted, or temporarily unavailable data. File-level validation is performed prior to processing to ensure that FITS files are fully written before being accessed, preventing partial reads during ongoing acquisition. In addition, processing operations are encapsulated within exception handling blocks to prevent individual file-level failures from interrupting the overall monitoring workflow.

During execution, non-critical anomalies such as missing or temporarily inaccessible files are handled through controlled termination of the corresponding processing task, while diagnostic messages are generated for debugging purposes. Scientific computations are also designed to be resilient to invalid or missing values, using NaN-aware aggregation methods to avoid propagation of corrupted samples into derived products.

In case of inconsistencies in FITS structure or metadata, the system safely skips the affected file and continues processing subsequent acquisitions. This design ensures continuous operation of the pipeline, isolating failures at the level of individual files without compromising the stability of the real-time monitoring system.

2.3 End-to-End Data Flow

To complement the modular description of the system architecture, the overall data flow of ICHNOS can be summarized as a sequential pipeline from data acquisition to visualization. The process begins at the telescope backend, where the acquired signals are digitized and stored as FITS files. These files are then written to disk by the acquisition system and become available to the monitoring layer.

The Watchdog-based I/O monitoring module detects newly created FITS files on the configured storage paths and triggers the processing pipeline. Once a file is validated as complete, it is forwarded to the data processing module, where scientific extraction and aggregation are performed.

The resulting products, including spectra and spatial representations, are transmitted in real time via Socket.IO using WebSocket communication. Finally, the frontend browser receives and renders the updated visualization through interactive Bokeh components, enabling real-time inspection of the incoming data stream.

A schematic representation of this pipeline is shown in Fig. 1, illustrating the complete end-to-end flow from data acquisition at the telescope backend to real-time visualization in the web browser.

Figure 1: End-to-end data flow of the ICHNOS system, from data acquisition at the telescope backend to real-time visualization in the web browser. FITS files generated by the acquisition instruments are stored on disk, detected by the Watchdog-based monitoring module, processed by the backend pipeline, and streamed via WebSocket to the interactive frontend.

3 User Interface

The User Interface (UI) is designed as a reactive Single-Page Application (SPA) that updates in real-time through WebSocket communications (handled by Socket.IO) to immediately display data as FITS files are processed. The UI main content area, shown in Fig. 2, is divided into three sections that allow monitoring of the data flow and the visualization of spectra and spatial distributions:

- **Metadata Panel (Header Display)** - This section updates dynamically when a new FITS file is processed. Its purpose is to provide a quick summary of the acquisition parameters. The section reports the following items:

- **Source**: the name of the observed source, as provided by the observing schedule. It is acquired from the Primary extension of the FITS file.
- **RA [hms]**: the right ascension of the source, expressed in hours, minutes, and seconds (hms format). The value is extracted from the Primary extension of the FITS file.
- **Dec [dms]**: the declination of the source, expressed in degrees, arcminutes, and arcseconds (dms format). The parameter is obtained from the Primary extension of the FITS file.
- **LO [MHz]**: the local oscillator frequency, expressed in MHz. The value is extracted from the RF INPUTS extension of the FITS file.
- **BW [MHz]**: the bandwidth of observation, expressed in MHz. The parameter is obtained from the SECTION TABLE of the FITS file.
- **Receiver**: the receiver used during the data acquisition. It is extracted from the Primary extension of the FITS file.
- **Backend**: the backend used for data acquisition. It can take the following values: TOTALPOWER, SARDARA, SKARAB. The parameter is not explicitly recorded in the FITS file but is inferred from the FITS filename and from the type of spectrum recorded.
- **Channels [#]**: the number of channels (i.e. bins) of the recorded spectra.
- **Feed [#]**: the list of feeds relative to the receiver used. The value is extracted from the SECTION TABLE of the FITS file.
- **Mode**: the polarization mode of the acquired data. The UI displays DUAL (LL, RR) or FULL (LL, RR, LR, RL), depending on the spectrum type (SIMPLE, SPECTRA, STOKES) inferred from the SECTION TABLE extension of the FITS file.
- **Signal Type**: the type of signal recorded. According to the acquisition mode (mono or dual-feed) it assumes the following values: REFSIG, SIGNAL, REFERENCE, REFCAL.
- **Scan [#]**: the scan sequence number relative to the observation. The parameter is taken from the Primary extension of the FITS file.
- **Sub-scan [#]**: the sub-scan number relative to the current scan. The parameter is taken from the Primary extension of the FITS file.

- **Feed Selector (Filter)** - A dropdown menu that allows the user to select the specific feed to monitor. The software processes and displays data only for FITS files that contain information related to the selected Feed (e.g., Feed 0, Feed 1, etc.), reducing both visualization and processing load.
- **Plotting Area (Data Visualization)** - This area represents the core of the real-time monitoring system, designed to provide an immediate graphical representation of the extracted data. The plots, generated as interactive elements, ensure high performance and fluidity even with high spectral resolution datasets, such as the 65536-channel output of the SKARAB backend. The interface includes a comprehensive toolset for detailed inspection, featuring *Pan*, *Box Zoom*, and *Wheel Zoom* for high spectral resolution analysis, a *Reset* function to restore the original view, and an interactive legend to toggle individual signal lines.

  The visualization layout dynamically adapts to the FITS data type and the current observation mode:

Figure 2: The main content of the UI. Data were taken from project 26-23 (PI: E. Ladu).

– **Single Spectra Mode:** for SIMPLE, SPECTRA, or STOKES formats, the system automatically organizes data into dedicated tabs based on the acquired polarizations (e.g., LL, RR, or full Stokes parameters). This ensures a clean and separate analysis of different signal streams while maintaining global visual consistency. In case of NOD observations, the system integrates signals from two feeds into a single comparative visualization.

– **Spatial Distribution Mode:** during a map acquisition, the plotting area evolves into a dual-component diagnostic interface (see Fig. 3) designed to provide simultaneous feedback on both spatial and spectral data quality:

  1. **Map:** An interactive scatter plot where each point represents an individual integration (sampling) within the sub-scan. Each point is positioned according to its precise celestial coordinates (RA/Dec or AZ/EL) and color-coded based on its average power level. This provides an immediate visualization of the telescope's trajectory, the spatial sampling density, and the real-time detection of the source as the scan progresses.

2. **Real-time Spectral Monitor:** In this configuration, the system supplements the map with the average power spectrum for the current sub-scan. This facilitates the instantaneous identification of (bright) spectral lines or RFI.

To ensure maximum diagnostic precision, the system implements an adaptive synchronization mechanism. Whenever the observation mode changes or a new dataset is loaded, the interface automatically reconfigures axes (including the dynamic frequency scale in MHz), titles, and graphical scales. This ensures that the user always views information consistent with the latest telescope state, eliminating the need for manual intervention or session re-initialization.

The application is designed following a desktop-first approach and leverages Bootstrap 5 for responsive layout management. The interface is fully responsive in terms of structural layout across desktop and tablet screen sizes. While the system remains usable on mobile devices due to Bootstrap's grid system, the UI is not specifically optimized for mobile interaction, as the primary target use case is desktop-based scientific data analysis.

4 Performance Analysis of Data Processing

4.1 Performance Benchmarking and Data Flow Management

The performance analysis of ICHNOS was conducted by simulating the incoming data stream using a custom script that replicated the arrival of 10 FITS files recorded with the SKARAB backend in a monitored directory (see Table 1 for details of the FITS files). This setup accurately models the typical operational environment, where large FITS files are generated and archived at regular intervals. The simulated arrival cadence of 30 seconds was chosen to closely match the nominal data acquisition rate observed during real SKARAB operations. The substantial size and complex internal structure of the FITS files make them a relevant test case for evaluating the system's efficiency.

The average total processing time per FITS file—including file system read operations (i.e., after files are written to the monitored directory), data extraction, vectorized mean calculations, and Bokeh object creation and rendering—is reported in Table 2.

It is important to note that all performance measurements reported in this work were conducted on a local development environment and therefore reflect the computational performance of the ICHNOS pipeline rather than the full end-to-end performance of the SRT operational system. In particular, the actual operational throughput will be constrained by the data acquisition rate and network data delivery capabilities of the SRT.

4.2 Technical Discussion on Performance

The analysis of the profiling data confirms the high efficiency of the data processing pipeline. The total time required for a complete quick-look generation is less than 1 second, which is remarkable considering the need to load 908 MB FITS file and process approximately 113 million data elements. It should be noted that file transfer time (i.e., the time required to copy the original FITS file from the acquisition directory to the monitored directory) is not included in this measurement, as it is treated as a separate system-level process. For

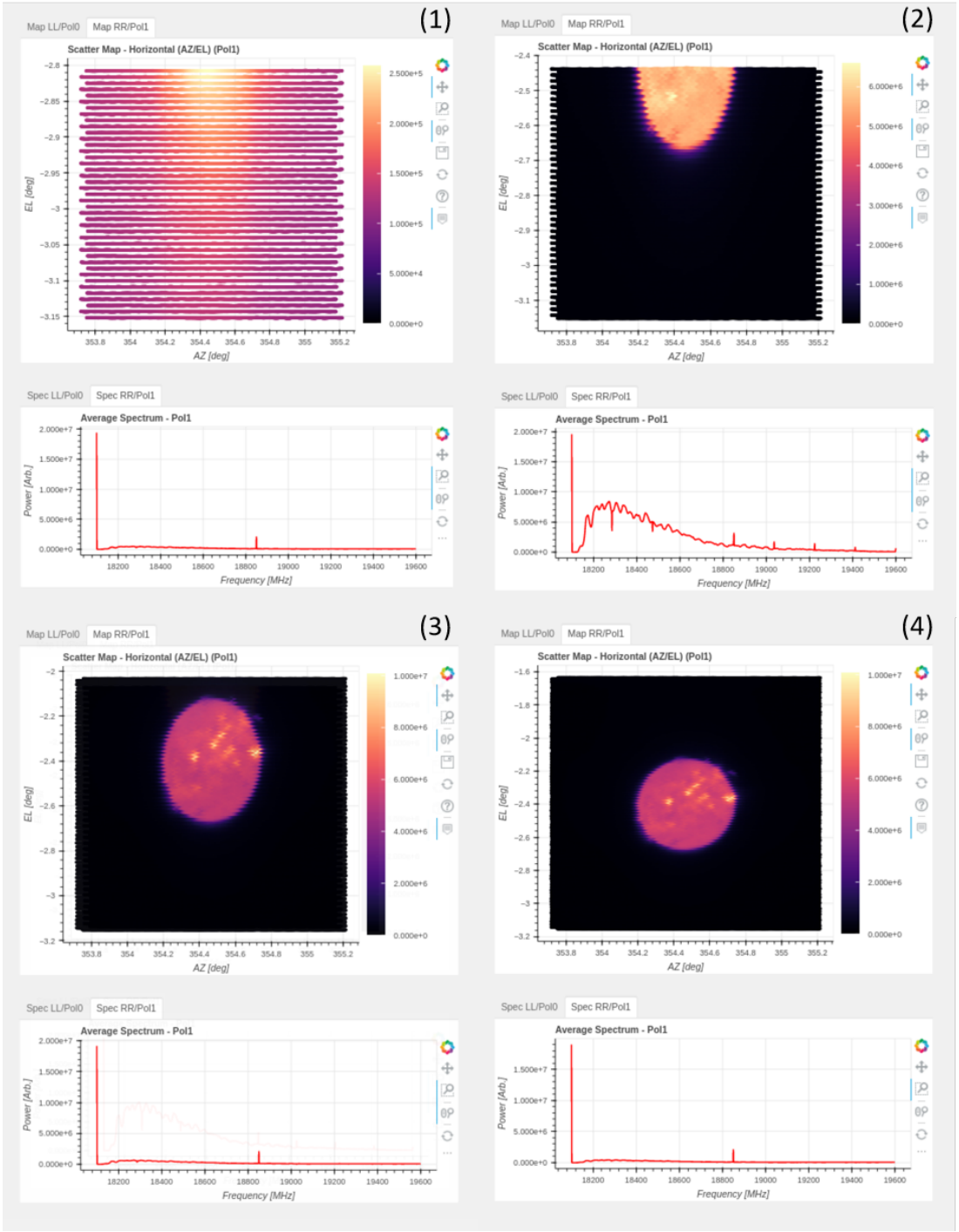

Figure 3: Representation of the processing pipeline for the dataset `20250314-130043-2-25-SUN_RA_K18` relative to the project 2-25 (PI: A. Pellizzoni). The four sub-images (1), (2), (3) and (4) illustrate the status of the radio map at different stages of the observation. The system dynamically updates the coordinates and the scaled power levels as each subsequent FITS file is ingested. This incremental visualization allows the operator to monitor scan quality, verify sky coverage, and assess the instrument's performance in real-time during the data acquisition process.

Table 1: Processing Metrics for a Single SKARAB FITS File ($\approx 908\,\text{MB}$)

Metric	**Value**	**Unit**	**Notes**
File Size	908328960	Bytes ($\approx 908\,\text{MB}$)	Substantial data volume
Data Structure	65536×1732	Channels $\times$ Rows	≈ 113 million elements
Arrival Cadence (Simulated)	30	seconds	Time budget for processing
Total Processing Time	**0.83**	**s**	

Table 2: Detailed Timing Analysis of the Processing Pipeline

Timer Phase	**Time (s)**	**Time (%)**	**Primary Function**
T1: Data extraction	0.3675	44.5%	Reading 908 MB from local file system
T2: Processing	0.2963	35.8%	Performing vectorized mean calculations
T3: Visualization	0.163	19.7%	Bokeh object construction and visualization
Total	**0.83**	**100.0%**	

completeness, the measured file transfer time for a ~ 900 MB FITS file is approximately 9.3 s, corresponding to an effective throughput of ~ 93 MB/s on the target storage subsystem.

The most time-consuming step is the T1 phase (see Table 2), accounting for 44.5% of the total execution time. This is primarily due to intensive disk input/output (I/O) operations associated with reading the full ~ 908 MB spectral array from the local file system into memory and performing data extraction. However, the use of vectorized operations is important in optimizing the subsequent computational step, ensuring that the temporal averaging (across 1732 rows) is executed with high CPU efficiency, thereby mitigating the computational cost of processing such a large dataset.

The memory footprint of the application was evaluated by monitoring the Resident Set Size (RSS) of the Python process during execution. The baseline memory usage remains stable at approximately 120 MB. When loading a 908 MB dataset, the system exhibits a peak memory usage of approximately 1.86 GB, resulting in an effective overhead of about 1.74 GB ($\sim 2.1\times$ the input file size). This behavior is consistent with the in-memory data representation, intermediate processing structures, and serialization required for WebSocket-based visualization. Despite transient spikes during processing, memory usage remains stable and no sustained memory growth or memory leaks are observed.

The measured processing time of 0.83 seconds is significantly lower than the simulated data acquisition cadence of 30 seconds. This large performance margin ensures that the quick-look system operates in a non-blocking regime and does not accumulate a processing queue under nominal load, even in the presence of large data volumes. This validates the system's robustness for near-real-time monitoring and data quality assessment.

5 System Requirements and Installation

To achieve the performance levels described in Section 4, ICHNOS requires a specific software environment. The application is designed to be cross-platform, but it is primarily optimized for Linux-based systems commonly present (or available) in control rooms of radioastronomical facilities.

5.1 Software Requirements

The application relies on a Python 3.8+ environment and is available upon request (repository currently not public). The core dependencies, extracted from the production environment, are categorized as follows:

- **Web Framework and Real-Time I/O**:
  - Flask (v1.1.4): A lightweight web framework used for request handling and routing.
  - Flask-SocketIO (v5.5.1) and python-socketio: Provide bidirectional real-time communication between client and server.
  - Simple-WebSocket and Tornado (v6.4.2): Provide the underlying transport layer for efficient data streaming.
- **Scientific Data Analysis**:
  - Astropy (v6.0.0): Used for FITS file parsing and celestial coordinate handling.
  - NumPy (v1.26.4) and Pandas (v2.2.3): Enable high-performance vectorized operations and data manipulation.
  - PyERFA (v2.0.1.5): Provides high-precision astronomical transformations.
- **Visualization and UI**:
  - Bokeh (v2.4.3): A Python visualization library used to implement interactive, server-driven web applications for real-time data visualization.
  - Jinja2 (v2.11.3): Template engine for rendering the dynamic frontend.
- **System Monitoring and Utilities**:
  - Watchdog (v2.1.0): Provides file system event monitoring, particularly for network-mounted directories.
  - SQLAlchemy (v2.0.40): Used for internal data handling and potential metadata persistence.

All dependencies are reported with fixed versions to ensure full replicability of the production environment.

5.2 Installation and Environment Setup

The recommended installation method is via a virtual environment to ensure dependency isolation. Installation steps are the following:

- **Environment Creation**
  Create and activate a Python virtual environment:

  ```
  python3 -m venv ql_env
  source ql_env/bin/activate
  ```

- **Dependency Installation**
  Install dependencies using the provided `requirements.txt` file, which contains the specific package versions:

  ```
  pip install flask==1.1.4 flask-socketio==5.5.1 astropy==6.0.0 \
              bokeh==2.4.3 numpy==1.26.4 watchdog==2.1.0 pandas==2.2.3
  ```

- **Configuration**
  Edit the `config.ini` file to set the correct monitoring paths and network port. This ensures that ICHNOS can access the FITS files and communicate properly within the SRT network.

- **Application Launch and Access**
  To start the application type:

  ```
  python app.py
  ```

  Once the server is active, the web interface can be accessed via a web browser at `http://127.0.0.1:5000/`. In a networked environment, replace `127.0.0.1` with the server hostname or IP address.

6 Final Remarks

The development and deployment of the ICHNOS web application represent a significant advancement in the operational monitoring capabilities of the SRT. By combining a fully asynchronous, multi-threaded architecture with modern web technologies, the system enables the real-time visualization and inspection of newly acquired data, effectively linking the acquisition pipeline with immediate scientific assessment. As a result, the software provides a valuable tool for the SRT, optimizing antenna time and ensuring high data quality standards.

Performance validation and benchmarking demonstrate that ICHNOS can ingest and process high-resolution FITS data (up to 908 MB per sub-scan) in less than 1 second. This efficiency ensures a non-blocking workflow, providing operators with near-instantaneous feedback even under the demanding data rates of the SKARAB and SARDARA backends.

Key achievements of the current implementation include:

- **Simultaneous Frequency and Spatial Monitoring**: Real-time visualization of high-resolution bandpasses alongside spatial point-clouds enables comprehensive assessment of both instrument stability and sky coverage.

- **Operational Robustness**: Polling-based monitoring ensures reliable performance across network-mounted file systems, making the tool resilient to the complexities of the SRT storage infrastructure.
- **Enhanced Decision-Making**: Interactive, scalable plots combined with critical metadata allow researchers to realize the presence of possible RFIs, hardware drifts, or pointing errors in real-time, significantly reducing the risk of compromising part of or entire observing sessions.

The current real-time implementation of ICHNOS imposes constraints aimed at ensuring minimal latency, which limit the simultaneous visualization of all feeds of a given receiver. To overcome these limitations, future developments will include the introduction of an offline analysis mode designed to complement the existing real-time capabilities. By relaxing real-time processing constraints, this mode will enable the simultaneous visualization of all receiver feeds and support expanded plotting capabilities, allowing more advanced and flexible data representations beyond the immediate needs of operational monitoring.

In addition, a complementary software tool called Scan-Viewer is being developed to support more advanced data visualization and analysis of spectral line data, particularly those acquired in NOD mode, including $(\text{on} - \text{off})/\text{off}$ plots. This tool will be described in detail in a forthcoming technical report.

This evolution will establish a two-tier workflow, combining real-time monitoring for immediate quality assessment with more comprehensive, quasi-real-time offline analysis. Such an approach will improve operational efficiency while enabling a deeper inspection and interpretation of the collected data.

A FITS Files in a Nutshell

FITS format is the international standard for the storage and exchange of astronomical data. It consists of one or more Header Data Units (HDU), each composed of:

- **a header**, made of 80-character ASCII records containing keyword–value pairs that describe the data;
- **a data unit**, which contains the actual data (images, tables, or spectra).

The standard FITS format adopted at the INAF radio telescopes consists of an initial HDU or *Primary* array [5]. Any subsequent HDUs that follow the Primary array are known as *extensions* and may represent:

- **Image**: an N-dimensional array of pixels;
- **Binary tables**: rows and columns of data in binary representation.

Fig. 4 shows the structure of a FITS file produced by the SRT using SARDARA[2], one of the backends available at the radio telescope and fully integrated into the control system.

Each header can contain mandatory, optional, or hierarchical keywords. Mandatory keywords include:

- `SIMPLE` (FITS compliance),
- `BITPIX` (bit depth),
- `NAXIS` and `NAXISn` (number and size of axes),
- `END` (end of the header)

Optional keywords include observational metadata (for instance, `SOURCE`, `SCANID`, `SIGNAL` in the Primary HDU), while hierarchical keywords (prefixed with `HIERARCH`) allow the representation of more complex structures, such as those defined by the ESO standards or used by specific instruments of the radio telescope.

The SECTION TABLE extension contains, among other things, information on the data type, the number of channels present in each recorded spectrum (e.g., 1024, 4096, 16384), the relevant frequencies, and the

[2]Also FITS files acquired with the TOTALPOWER and SKAARAB backends have the same structure.

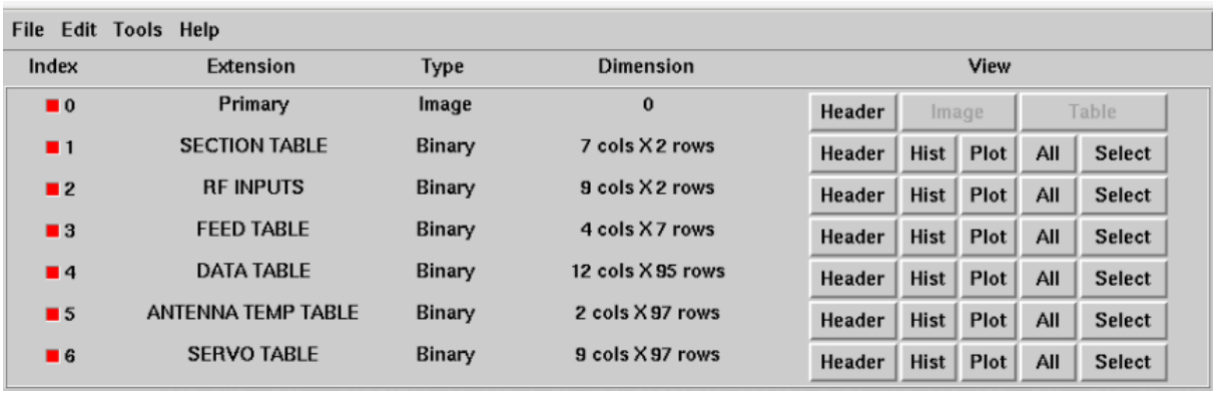

Figure 4: Structure of a FITS file as recorded with the SRT.

observation band. The rows correspond to the data columns present in the DATA TABLE extension, each of which contains a certain number of recorded spectra.

The data type reported in the SECTION TABLE extension can take three values: *simple*, *spectra*, and *stokes*. These values determine how the recorded polarization products are organized within the DATA TABLE and, consequently, affect the number of rows in the SECTION TABLE. In particular:

- the *simple* and *spectra* data types generate two rows per feed, meaning that the data for the LL and RR polarizations are separated into two different columns of the DATA TABLE.
- the *stokes* data type generates one row per feed, as all data for the LL, RR, LR, and RL polarizations are combined into a single column of the DATA TABLE.

The RF INPUT table contains mostly acquisition-related information, while the details regarding the feeds used to record the data (id, xOffset, yOffset) can be extracted from the FEED TABLE extension.

A `summary.fits` file is also generated and finalized at the end of each scan, containing additional keywords related to the observation. This file has not been described in this appendix, as it is not explicitly used within ICHNOS.

List of Acronyms

Acronym	Definition
CIFS	Common Internet File System
FITS	Flexible Image Transport System
HDU	Header Data Unit
NFS	Network File System
NOD	Nodding
OS	Operating System
OTF	On-The-Fly
PSW	Position Switching
RFI	Radio Frequency Interference
RSS	Resident Set Size
SARDARA	SArdinia Roach2-based Digital Architecture for Radio Astronomy
SKARAB	Square Kilometer Array Reconfigurable Application Board
SPA	Single-Page Application
SRT	Sardinia Radio Telescope
UI	User Interface
VLBI	Very-Long-Baseline Interferometry

References

[1] A. Tarchi et al., *Spectroscopic observations with the SRT using the NODDING technique*, OACA Internal Report N. 72, (2018)

[2] A. Scalambra et al., *PCB FS-TP Focus Selector e Total Power back-end*, IRA Internal Report N. 473/13 (2013) Interno IRA, n.473/13

[3] A. Melis et al., *SArdinia Roach2-based Digital Architecture for Radio Astronomy*, Journal of Astronomical Instrumentation, Vol. 07, No. 01, 1850004 (2018)

[4] A. Melis et al., *The SKARAB Board in the Framework of Single-Dish Radio Astronomy*, Journal of Astronomical Instrumentation, Vol. 13, No. 2, 2450008 (2024)

[5] S. Righini, A. Zanichelli, *FITS output format in DISCOS*, SRT-MAN-10000-003, Issue n. 4, (2018)