WaveTap ADS-B Tracking Quality & Analytics

Project Proposal – October 24, 2025

Blaine Harris

Abstract

WaveTap is a modular open-source software defined radio (SDR) framework for utilizing hardware such as RTL-SDR to capture and decode radio signals. The current implementation of the framework utilizes the open-source application *dump1090*, to capture and decode ADS-B messages. Figure 1 below illustrates the concepts of operations of functionality currently supported by WaveTap.

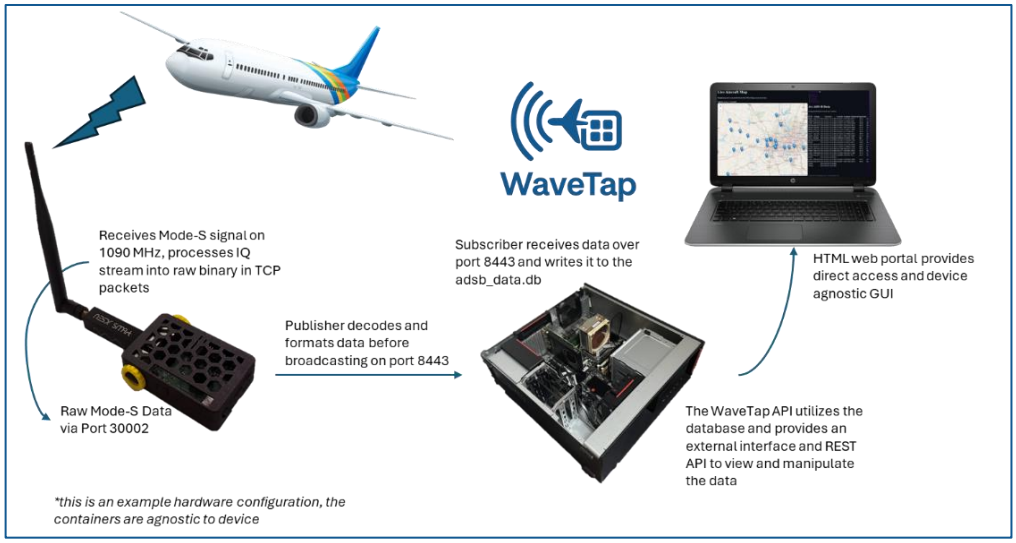

Figure 1: WaveTap Concept of Operations

This project will capture 3 major areas of metrics gathering, CI/CD pipeline metrics to ensure code quality and maintainability, internal application metrics to monitor WaveTap's operational performance, and external metrics capturing the performance of dump1090 at providing raw

decoded radio signals. Tools such as GitLab pipelines and 'cron' are planned to allow the automated execution of PyTest scripts to capture automated metrics. Analysis methods will include descriptive statistics, regression analysis, trend analysis, anomaly detection and frequency, and comparative analysis.

Measurement Implementation

The Goal, Question, Metric (GQM) approach was utilized to determine what metrics are planned to be recorded (represented below in Table 1). WaveTap is planned to be a *modular* open-source project, which requires extra emphasis on a clean and maintainable code base. This led to goals focused around code complexity, code performance, database performance, and reliability of the required dependencies.

Table 1: GQM Planning

Goal	Question	Metric
Ensure code quality & maintainability	Is the codebase style clear, consistent and well covered by unit tests?	Style violations (Ruff), unit test coverage (PyTest)
Prevent defects	Are new defects introduced between versions?	Unit test pass/fail counts
Reliable message Tx/Rx	Are dropped packets between components below the acceptable threshold?	Packet drop rate (dropped packets per hour)
Minimize message processing latency	Are ADS-B messages being constructed quickly and completely?	Time to construct message (ms), number of incomplete messages per hour
Maintain database performance	Does database performance maintain consistent or degrade as the database grows?	DB write/query latency (ms) vs. DB size (mb)
Validate external dependency reliability	Does dump1090 consistently start up and provide decoded binary packets?	Packets produced per second, MTTF, number of application crashes
Ensure operational stability	Are the system resources and uptime within acceptable limits?	CPU usage, WaveTap crash count

Data Collection

For CI/CD metrics, GitLab pipelines are planned to be configured to automatically run code style checks with Ruff, unit tests with PyTest, and coverage analysis for every commit and merge request. These pipelines will capture and store data for these metrics as part of the repository. For internal and external code metrics automated scripts using a combination of PyTest and `cron` will be utilized to schedule and execute automatic testing and data collection. This will allow large data sets to be generated and stored without human intervention. Frequent observation is anticipated on these metrics, therefore, these will be stored in local csv files or in SQLite databases for ease of implementation.

Table 2: Planned Metrics to Gather

CI/CD Metrics	Internal Metrics	External Metrics
- Code style (Ruff) - Style violations (Ruff) - Unit test Coverage (PyTest) - Unit test Failures (PyTest) - Unit test Passes (PyTest)	- Dropped Rx TCP packets - Dropped Tx TCP packets - Time to construct ADS-B message - Incomplete ADS-B messages - Database write time - Database query time - localhost vs remote host for services - CPU usage (RPi vs. Linux)	- MTTF (Mean Time to Failure) - Number of Packets Produced per Second - Number of application crashes

Data Analysis

Collected metrics will be analyzed using descriptive statistics to summarize performance indicators such as packer drop rate, ADS-B message construction time, and database query latency. Regression analysis will be used to track code performance and stability over commits and merge requests. Trend analysis will be utilized to track improvements or degradation in metrics such as unit tests pass and failures, ruff failures, and general application performance. As

defects are detected they will be manually tracked in an anomaly detection database as they are resolved or investigated. Given the short timeline of this project, this will most likely be a spreadsheet. This will support additional debugging data and provide additional traceability for defects. All findings from this analysis will be visualized using standard visualization tools such as Excel or Matplotlib in Python.

Follow Up Actions

Any detected anomalies that are not addressed during the duration of this project will be resolved in future code updates. The codebase can be refactored as needed based on the immediate feedback provided by automated GitLab pipelines reviewing code structure. If database slowdowns are recorded as the database grows or is hit with more queries, additional research will be conducted to investigate a more optimized data storage solution. Lessons learned will be documented in the final report and presentation.

Activities & Deliverables

This project takes place over the course of 10 weeks, this has been split into 6 discrete activities, and 5 deliverables that will be generated as part of the activities. Table 3 below outlines the activities planned for this project.

Project Repository: https://github.com/bwharris1125/CS7319_Project_WaveTap

Note: This software project builds on a project from course CS7319.

Table 3: Project Activity Breakdown

Activity	Activity Title	Description
1	Proposal Document	This activity includes the research and coordination needed leading up to the project proposal.
2	Literature Research	Researching software metrics, related applications, analysis methods, and modeling.
3	Research Report	This report will act as a summary of the research conducted and provide insight into with methodologies can be applied to the overall project.
4	Testing & Data Collection	Author CI/CD pipelines and PyTests to automatically evaluate the software suite and collect test data.
5	Data Review & Metrics Generation	Review test data and generate metrics.
6	Final Documentation	Reapproach material discussed in proposal and follow-up, research, and collected data to discuss analysis and results,
7	Project Presentation	Condense the final documentation into a visual presentation to discuss findings.

Project Schedule

The Project Schedule shown below identifies the timelines needed to complete all activities and the deadlines for the required deliverables.

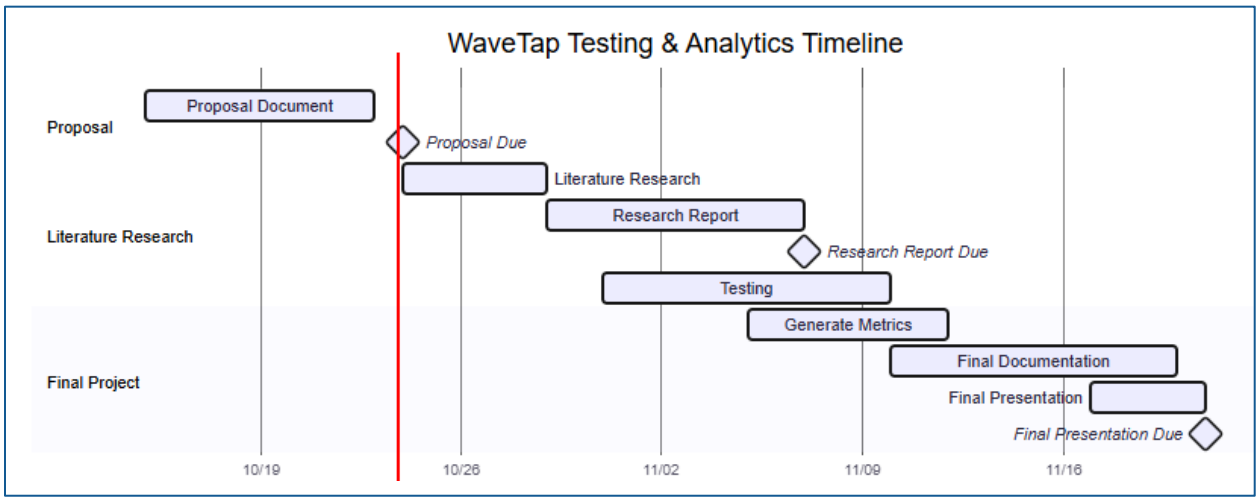

Figure 2: Project Timeline