A Modular and Distributed Readout, Control, and On-board Novelty Detection Architecture for ASTHROS using RabbitMQ

Paul Horton [1], Christian Thompson [2], Chris Groppi [1], Jose Siles [3], and Jonathan Kawamura [3]

[1] University of Massachusetts Lowell, Lowell, MA USA
[2] University of Toronto, Toronto, ON CAN
[3] NASA Jet Propulsion Laboratories, Pasadena, CA USA

Using **RabbitMQ**, we've developed the **ASTHROS** readout system as a **fully modular** collection of drivers and programs that integrate through our open-sourced library, **rmqtools**.

Overview

ASTHROS, the Astrophysics Stratospheric Telescope for High Spectral Resolution Observations at Submillimeter-wavelengths is a 2.5m telescope **balloon-borne observatory** that will make 3D maps of ionized gas in star forming Galactic and Extragalactic regions [1].

For **system readout**, we are utilizing **RabbitMQ** to broker a communications network that allows each device to seamlessly **share status and commands** between multiple concurrent systems.

To perform **on-board analysis**, we are testing different novelty detection methods on time-series calibration spectra in order to **highlight anomalous spectra** for a diagnosis downlink.

System Design

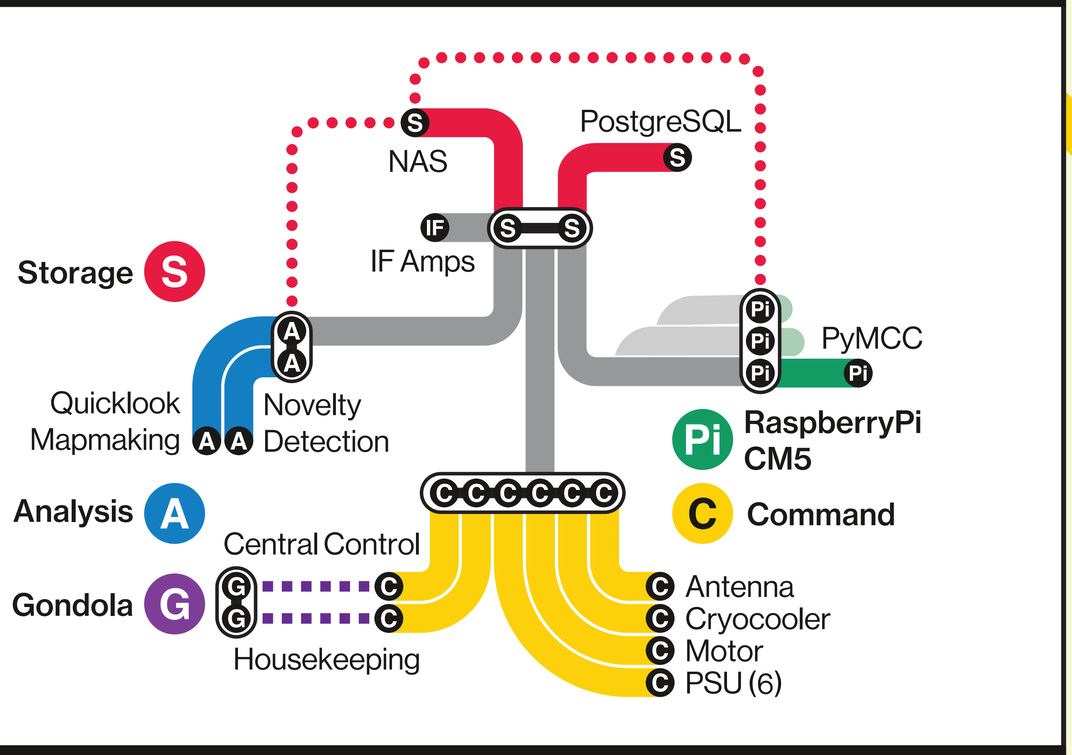
At the core of ASTHROS' readout architecture are **three main computers** that interact with the rest of the readout systems: **command, storage,** and **analysis.**

The **command** computer serves as the central coordinator for the system, **processing commands** and **managing telemetry** through a socket to the gondola. Key system devices, such as the power supplies and the antenna, are connected directly to this computer via serial to broadcast updates and await commands.

The **storage** computer **hosts a Network Attached Storage** (NAS) for network-wide ability to **store and retrieve science data**. The spectrometers, powered by Raspberry Pi Compute Module 5s (CM5), save spectra both locally and on the NAS for redundancy. Additionally, the storage computer **hosts the RabbitMQ** server, facilitating efficient **communication brokering** across the network.

The **analysis** computer, equipped with powerful processing capabilities, **generates quicklook maps** and **monitors spectral normalcy**. This computer will process incoming data and notify the command computer of any abnormal calibration spectra.

ASTHROS Network Map



RabbitMQ Network and rmqtools

We are utilizing RabbitMQ to handle communication across multiple processes and devices. RabbitMQ is an open-source Advanced Messaging Queue Protocol (AMQP) implementation that serves as a broker for incoming and outgoing messages by **routing messages to bound destination queues using an exchange** [2].

Using RabbitMQ's topic exchange, we implement a **publisher subscriber (PS)** pattern where devices can broadcast their status to bound queues for telemetry and monitoring overall system health.

Using RabbitMQ's direct exchange, we are able to perform **Remote Procedure Calls (RPC)** where commands and acknowledgements are sent between specific processes and devices through routing keys.

To make integration easy for missions, we developed the library **rmqtools**. Missions can design and implement system functionality for each component **independently** and then use **Python decorators** to expose commands and broadcast data by wrapping functions.

Implementing with rmqtools

ASTHROS utilizes an array of Pacific MicroCHIP Corp. (PMCC) P19800Bs to collect spectra. Up to four PMCCs are connected to a CM5 via SPI to control the spectrometers and collect data. We designed the Python drivers (PyMCC) for the PMCCs as a **standalone system**, agnostic of ASTHROS, for bench testing and usage beyond ASTHROS.

For commanding, we decorated key methods from the PyMCC drivers, such as initialization and spectral collection controls, with rmqtools to expose them as RPC methods in RMQ. Each PMCC and CM5 is given a unique name on the network so that Central Control on the Command computer is able to **address PMCCs and CM5s both individually and in groups**. Additionally, rmqtools handles the **threading** so that these calls aren't blocked by multiple processes talking with a single device.

For housekeeping, we create a new function that queries values from the PyMCC driver, like spectra count and integration settings. We decorate this function with rmqtools using an address and interval so that systems, like the on-board novelty detection system and the PostgreSQL database listener, can **get regular status updates** through the PS framework.

```
@rmq.handle_command("spec.start") ------ device
def start_spectra():
    ...
commander ------ rmq.send_command("spec.start")
```

References

[1] J. Pineda, *et al.*, "The Astrophysics Stratospheric Telescope for High Spectral Resolution Observations at Submillimeter-wavelengths, ASTHROS," in American Astronomical Society Meeting Abstracts, 2022.

[2] M. Dunne, *et al.*, "A comparison of data streaming frameworks for anomaly detection in embedded systems," in International Workshop on Security and Privacy for the Internet-of-Things (IoTSec), 2018.

Got a question?
Reach me at **paul_horton@uml.edu**

SPIE.