



# The MMT-POL Instrument Control System



C. Warner<sup>1</sup>, C. Packham<sup>1</sup>, T. J. Jones<sup>2</sup>, F. Varosi<sup>1</sup>, S. S. Eikenberry<sup>1</sup>, K. Dewahl<sup>2</sup>, M. Krejny<sup>2</sup>  
<sup>1</sup>University of Florida, <sup>2</sup>University of Minnesota

Instrument control system (ICS) suites are a continually evolving class of software packages that are highly dependent upon the design choices and application programming interfaces (APIs) of the observatory control system (OCS), as well as the hardware choices for motors and electronics. We present the ICS for MMT-POL, a 1-5  $\mu\text{m}$  polarimeter for the MMT telescope, in the context of being a transitional step between the software packages developed for facility class instruments at UF such as Flamingos-II and CanariCam and in preparation for 30m-class instruments.

Our goals for improving ICS suites are to make them (a) portable (compile once, run anywhere), (b) highly modular and extensible (through the re-use of common libraries), (c) multi-threaded (to allow multiple tasks to be performed in parallel), (d) smart, and (e) easy to use and maintain. An ICS should also be well-defined and use mature languages (we choose java and python) and common standards (such as XML and the FITS file format). We also note that as hardware moves away from serial communications to ethernet, the use of TCP sockets makes communication faster and easier. Below, we present our design choices for the MMT-POL ICS and discuss our reasons for these choices and potential issues that must be addressed for future ICS suites ready for thirty meter class instruments.

MMT-POL<sup>1</sup> is an adaptive optics optimized imaging polarimeter for use at the 6.5m MMT. By taking full advantage of the adaptive optics secondary mirror of the MMT, this polarimeter will offer diffraction-limited polarimetry with very low instrumental polarization. This instrument will permit observations as diverse as protoplanetary discs, comets, red giant winds, galaxies and AGN.

## A Transitional Step in UF Instrumentation Software

- MMT-POL is a PI-class instrument, allowing us to explore new ideas without the restrictions placed on facility-class instruments.
- Previous software packages developed for facility-class instruments had many inherent shortcomings:

- Developed mostly in C/C++ and required specific hardware architectures.
- Configuration changes sometimes required recompiling and/or rebooting of old VMS VAX machines.
- Single-threaded agents meant tasks could not be parallelized.
- Use of non-standard data formats and protocols.
- Slow serial communications.
- Non-extensible code becomes cluttered.

## Goals of the MMT-POL ICS

### Portable

- Use of Java and Python allow for code to be compiled once and run on any architecture.
- Ethernet TCP sockets allow agents and GUIs to be run from anywhere and connect to electronics and servers running in instrument rackmount.
- XML is used for all configuration files.
- Configuration changes are simple and do not require recompiling. Common changes, such as creating or modifying dither patterns, can be done live from the MMT-POL Java Engineering Console (MJEC) GUI

### Modular and Extensible

- Common tasks and data structures are combined into libraries (javaUFProtocol, javaUFLib, and javaMMTLib) used by both agents and GUIs.
- These libraries are extensible for re-use in future instruments.

### Multi-threaded

- Agents and GUIs are all multi-threaded so that multiple tasks can be performed in parallel.
- Performing tasks in parallel increases efficiency but synchronization must be carefully implemented to avoid collisions or deadlock.

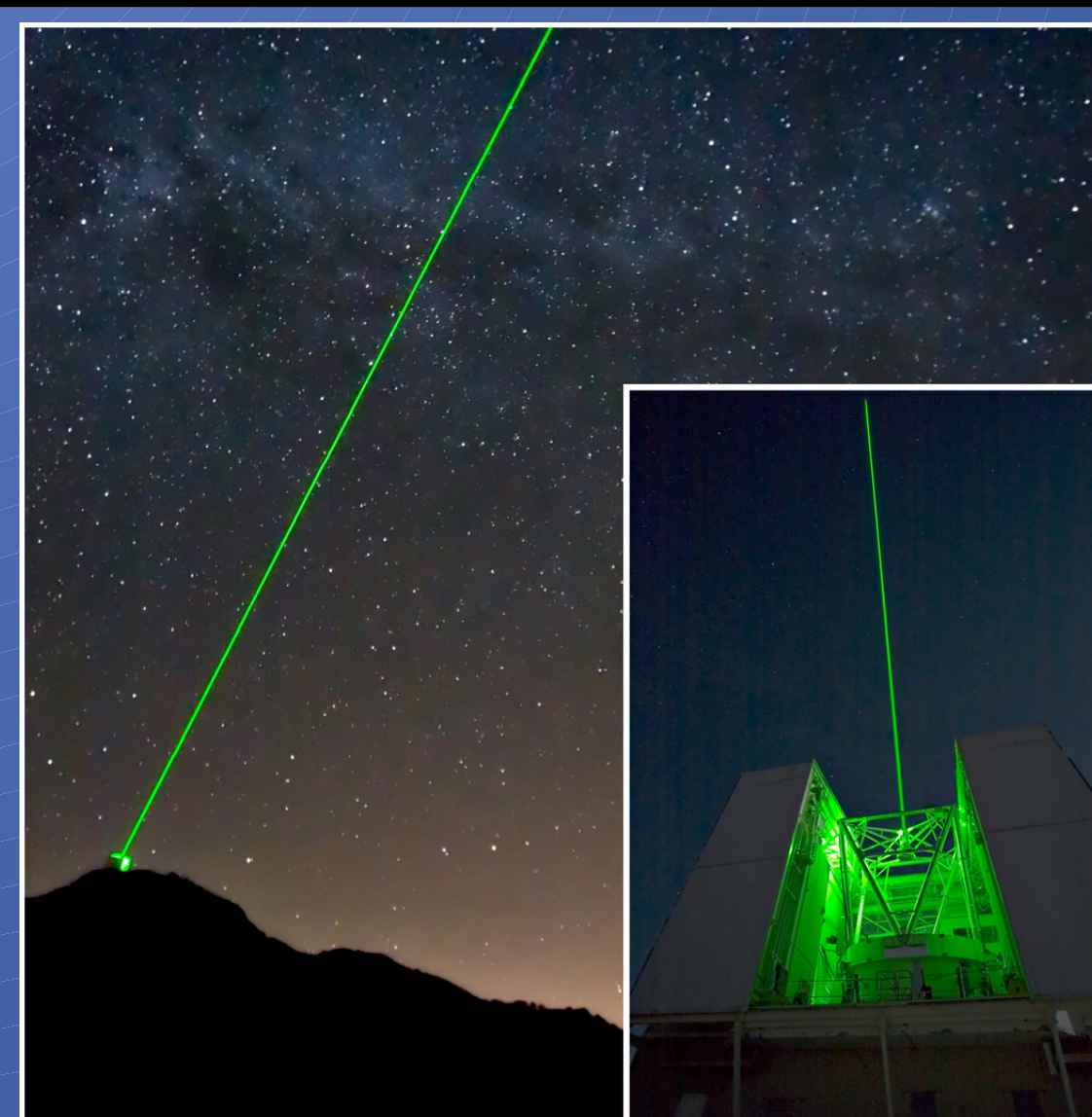


Fig. 1 – MMT Observatory in Arizona

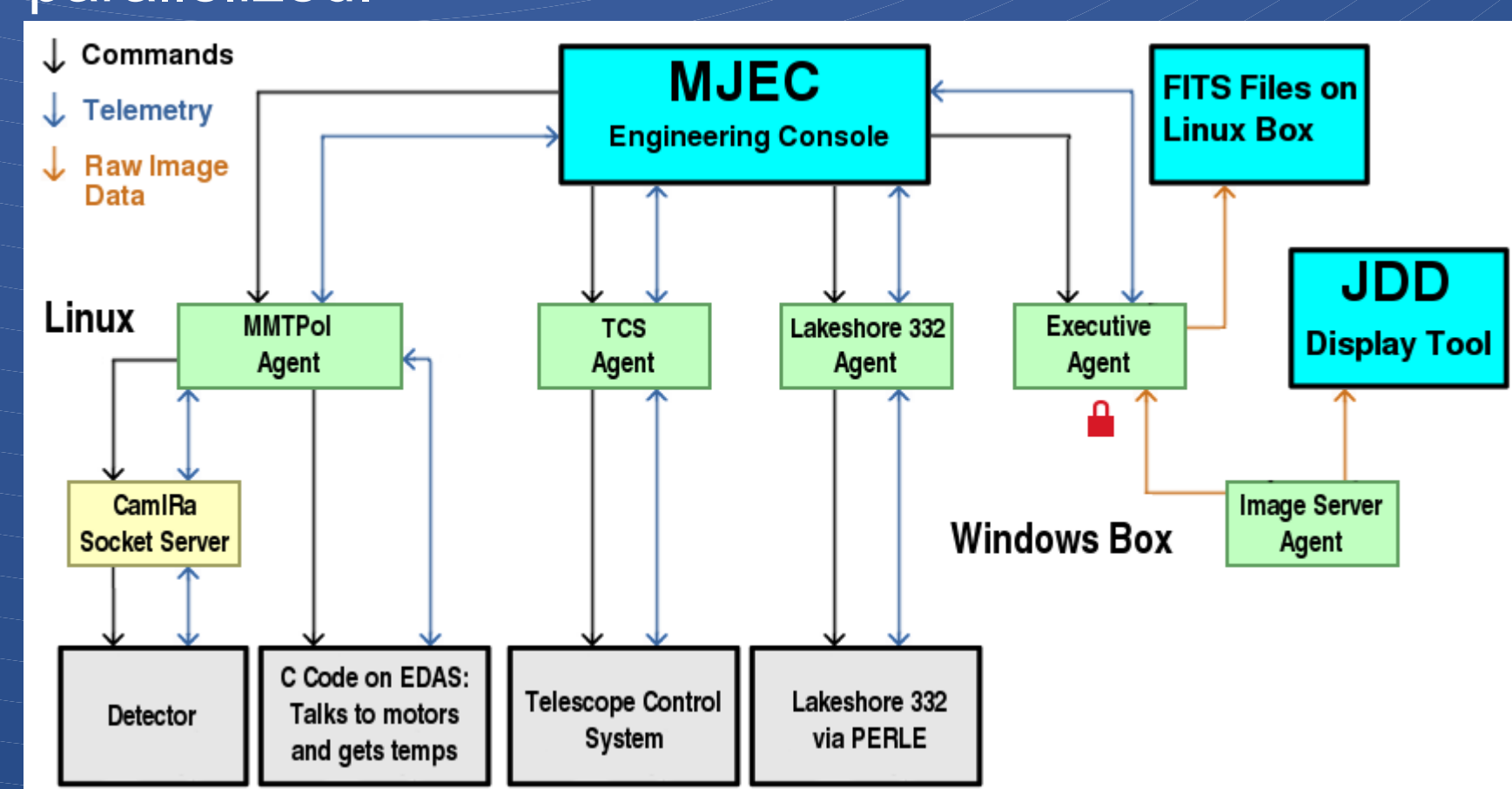


Fig. 2 - Schematic of the MMT-POL ICS

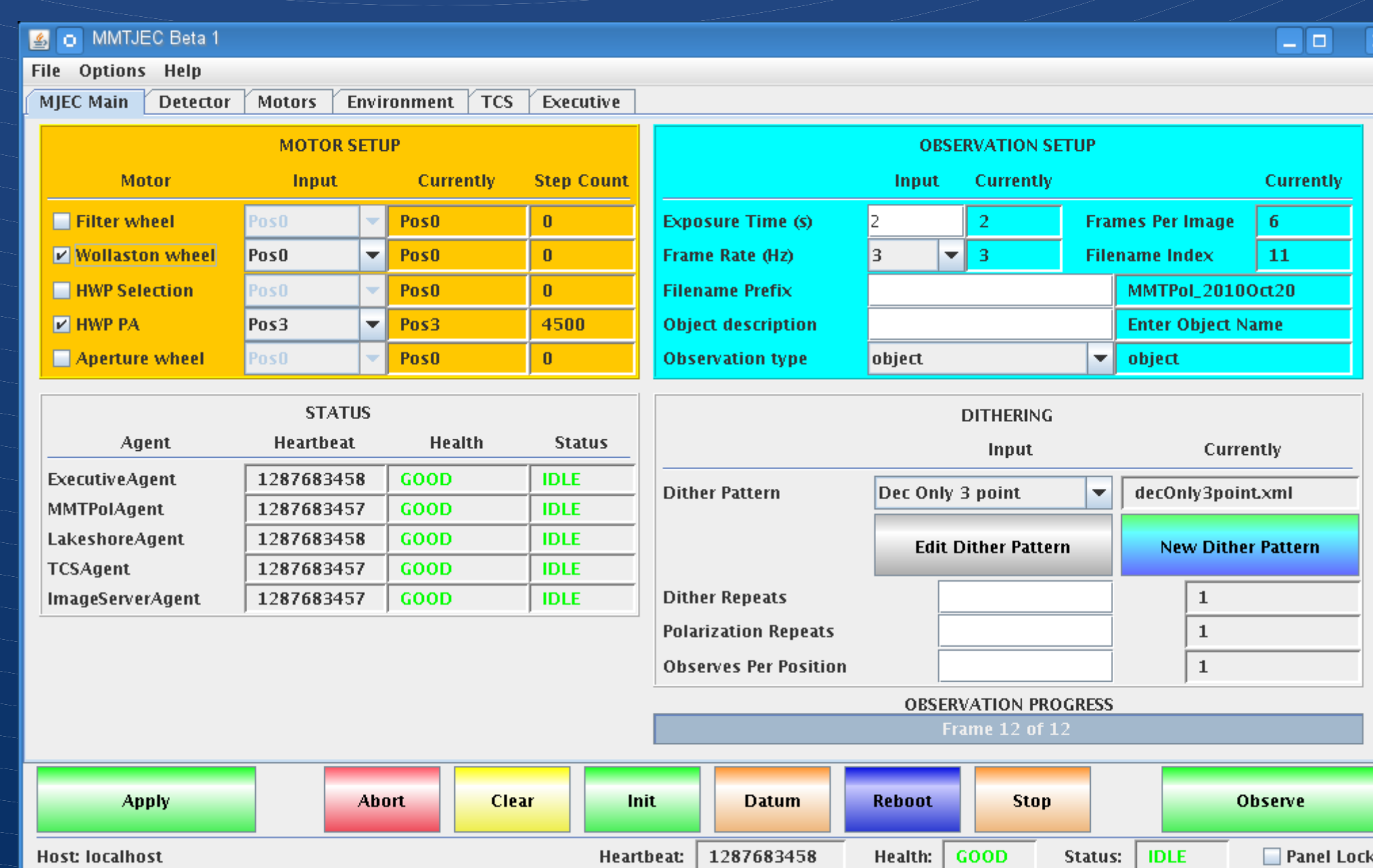


Fig. 3 - MMT-POL Java Engineering Console (MJEC)

## Goals (continued)

### Smart

- Agents and GUIs work together to automate complex tasks, such as observations.
- The executive agent has 14 active threads running during an observation to coordinate the various agents and GUIs.
- An observation set entails dithering the telescope to X positions, cycling through all 4 half-wave plate positions Y times at each, commanding the detector to take an image Z times at each position, and collecting telemetry data from the various agents to be added to the image header of each exposure.

### Easy to use and maintain

- Everything is run through MJEC. Its main tab (Fig. 3) displays all commonly changed parameters along with important health, status, and telemetry data.
- An easy-to-use quick look display tool (JDD) can connect to the image server from anywhere, providing a real-time view of the data. It can store multiple buffers and offers tools such as line cuts, statistics, arithmetic between buffers, zooming, various scaling algorithms, and color maps.

- Active threads in Executive Agent during an observation
- ▶ Thread 1: Main Thread – loop over client threads and check for action requests. Spawn off a new action thread to handle a new request.
  - ▶ Thread 2: Action Thread – coordinates observation by communicating with other agents and threads. Does the actual work.
  - ▶ Threads 3-6: Executive Client Threads that are full clients of other agents. These threads can send commands to other agents to facilitate in coordinating the observation (e.g., dithering, moving motors).
  - ▶ Threads 7-10: Executive Client Threads that are status clients of other agents. These threads listen to the Status Threads of the other agents and update the database when a new record is received.
  - ▶ Thread 11: Command Client Thread – is a client of MJEC or a script and listening to receive new commands (such as an Abort)
  - ▶ Thread 12: Status Client Thread – is a client of MJEC and constantly streams any new status info to MJEC.
  - ▶ Thread 13: Ancillary Thread – Once a second, this thread updates the heartbeat, polls any devices, updates database records, and tells the Status Threads to update their clients.
  - ▶ Thread 14: Listen Thread – listens for new clients to connect to the agent on its ServerSocket and spawns off a new Client Thread for each.

Fig. 4 – Thread diagram for Executive Agent

## Software Issues for 30m class Giant Segmented Mirror Telescopes (GSMTs)

- Each instrument at a GSMT would communicate with the Observatory Control System (OCS) through a middleware layer (software communications backbone, SCB).
- Agents and GUIs could be built upon existing common libraries developed for MMT-POL combined with observatory-defined APIs to communicate with the OCS.
- Big issue (1): Implementing APIs to use the SCB to communicate with the OCS.
- Big issue (2): Data Handling Client must be able to process vast amounts of data quickly (10 Gbit/s ethernet would be optimal) and use relational database management systems (RDBMs) to mine huge databases of meta-data.

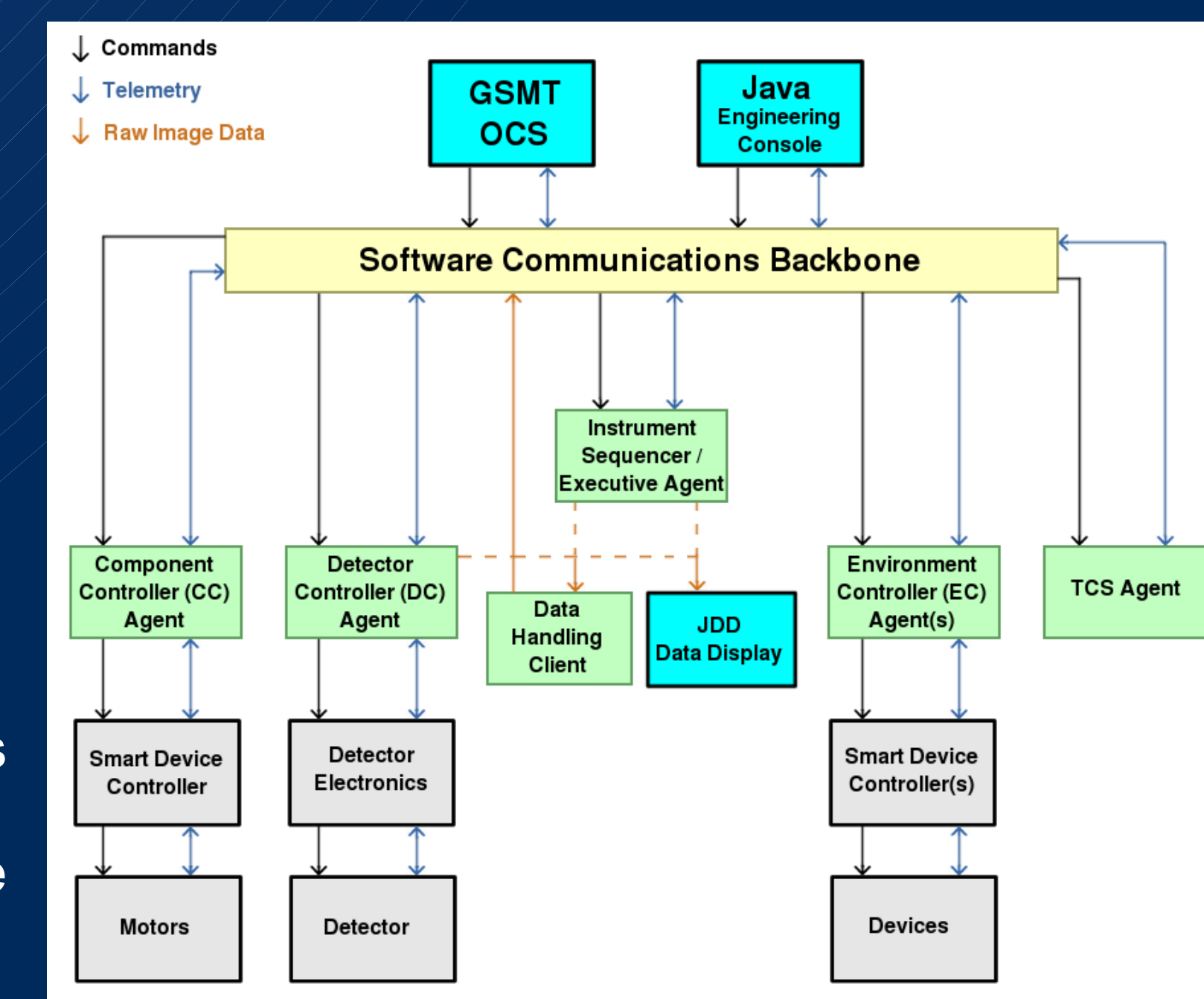


Fig. 5 – Potential design for a GSMT ICS

## References & Acknowledgements

<sup>1</sup>Packham et al (2010, SPIE, 7735E, 215P)