TANGO INTEGRATION OF THE SKA-LOW POWER AND SIGNAL DISTRIBUTION SYSTEM

E. L. Arandjelovic[1*], U. K. Pedersen[1†], Observatory Sciences Ltd., U.K.
J. Engelbrecht[1‡], Vivo Technical, Cape Town, South Africa
D. Devereux[1§], CSIRO, Australia
[1]also at SKA Observatory, Jodrell Bank, United Kingdom

Abstract

The Square Kilometre Array Observatory (SKAO) is the world's largest radio telescope, currently being constructed on two sites: SKA-Low in Western Australia, and SKA-Mid in South Africa. The Power and Signal Distribution System (PaSD) is a key component of the SKA-Low telescope, responsible for control and monitoring of the electronic components of the RF signal chain for the antennas, and collecting the RF signals for transmission to the Central Processing Facility. This paper will describe how the PaSD is being integrated into the Tango-based SKA-Low Monitoring Control and Calibration Subsystem (MCCS) software, including the facility for a drop-in Python simulator which can be used to test the software.

Keywords: Tango, SKAO, Modbus, Telescope, Python.

INTRODUCTION

SKAO represents the next generation of radio astronomy, poised to transform our understanding of the universe. Spanning two continents, Australia and South Africa, this very large international initiative boasts two cutting-edge radio telescopes. SKAO's ambitions are wide-ranging and include exploring cosmic dawn: the formation of the first stars, galaxies and black holes, investigating dark energy and the acceleration of the expansion of the universe.

SKA-LOW

The SKA-Low is a radio telescope featuring over 130,000 log-periodic antennas, operating in the frequency range of 50 MHz to 350 MHz, with a total collecting area of 419, 000 m². Situated in the desert of Western Australia, the antennas are grouped into 512 Field Stations, distributed in three spiral arms radiating from a central core, allowing a maximum station-to-station distance of 65 km. SKA-Low's unique design employs wire-type antennas and advanced back-end technology for efficiency at low frequencies. It operates as a mathematical telescope, processing data, applying time-delays to align the phases of signals received from a certain direction, to form virtual beams that "point" the telescope in different directions without moving parts. The signals within the beam can then be searched for transient phenomena and timed.

Each Field Station of the SKA-Low telescope consists of a 256 antenna-element array to capture and amplify the signal from the sky. The Power and Signal Distribution System (PaSD), designed by Curtin Institute of Radio Astronomy (CIRA) takes care of powering the antennas, collecting the RF signals, and providing local monitoring and control [1].

PaSD ARCHITECTURE OVERVIEW

The PaSD system comprises 'SMART boxes' (SMART: Small Modular Aggregation and RFoF Trunk) which each connect directly to around 10 antennas to provide local monitoring and control, and one Field Node Distribution Hub (FNDH) per Field Station which distributes power to all the SMART boxes and provides an Ethernet-serial communications gateway, as well as additional local monitoring. Micro-controllers inside the SMART boxes and FNDH protect the equipment from damage by automatically turning off ports in response to current and temperature readings, thus separating the equipment protection concerns from any external control system.

All of the PaSD parameters that can be monitored or controlled are accessible via Modbus [2] registers in the FNDH and SMART boxes. A register map is published by the PaSD firmware manufacturers which describes the purpose and function of each register. These include read-only registers e.g. the Power Supply Unit (PSU) temperatures and output voltages, and read-write registers e.g. current trip thresholds. The first register of the FNDH and SMART boxes holds the corresponding register map revision number, which is incremented whenever this map has changed following a firmware update.

All communication to the SMART boxes is funnelled through the FNDH on a multi-drop serial bus using the Modbus ASCII protocol. A Field Node Communications Controller (FNCC) board in the FNDH handles the incoming Modbus packets, passes them on to the SMART boxes which each have a unique Modbus address, and routes responses back. The PaSD architecture is detailed in Fig. 1.

MONITOR, CONTROL AND CALIBRATION SYSTEM

The Monitor, Control and Calibration System (MCCS) is responsible for the monitoring and control of all the local hardware on the SKA-Low Field Stations. It monitors and aggregates hardware status, making this available to the Telescope Manager (TM) which provides the observation management interface for operators and scientists [3]. It also

* ela@observatorysciences.co.uk
† ukp@observatorysciences.co.uk
‡ jarrett@vivosa.co.za
§ drew.devereux@csiro.au

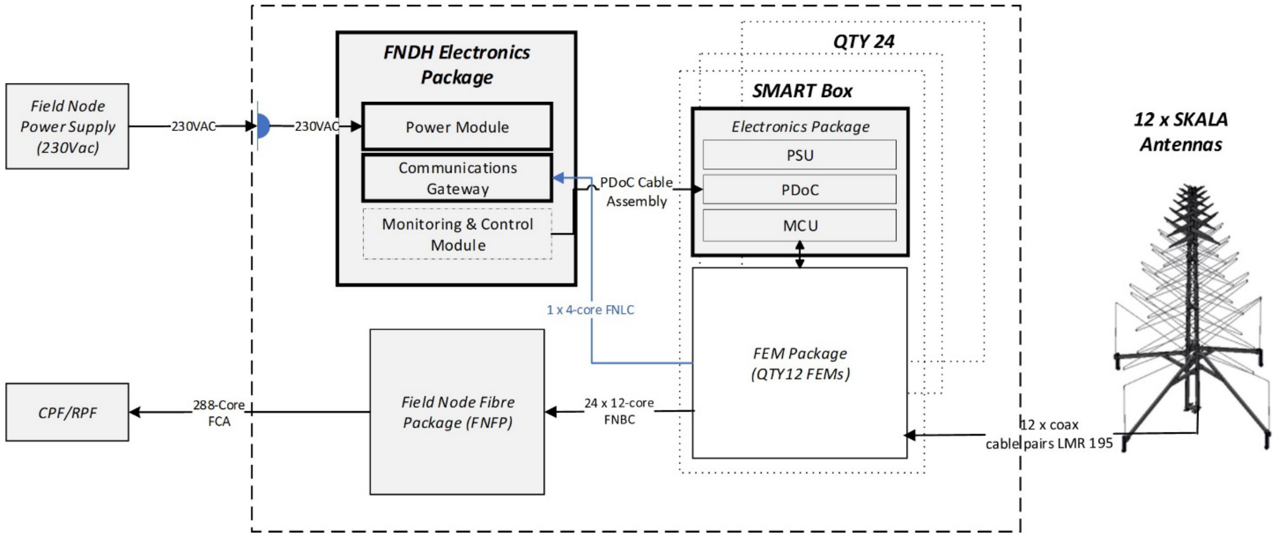

Figure 1: PaSD System Architecture.

includes the software to control and calibrate the Data Acquisition System. Based on the Tango Controls framework [4] and deployed as a containerized application, the MCCS is a complex product being developed by a distributed agile team. The following sections describe how the PaSD sub-component is being integrated into the existing MCCS architecture.

SOFTWARE DESIGN

The nature of the shared serial bus means that each SMART box must be polled in turn, to avoid collisions. This is achieved through a software design that makes use of a single client (Tango device) which communicates directly to the PaSD, named the **MccsPasdBus**. All communication is sent through this client, and a separate Tango device corresponding to each physical device (one per SMART box and an additional one for the FNDH) then proxies through **MccsPasdBus** and exposes the relevant attributes and commands for that device to the end user, maintaining a direct correspondence between hardware and Tango devices. An additional Tango device representing an entire Field Station also provides commands which will cascade down to its FNDH and SMART box devices.

The low level transport code in the **MccsPasdBus** device has been implemented using the PyModbus library [5]. This was chosen for a number of key reasons: it has a lightweight implementation with minimal dependencies (pyserial [6] being the only third-party dependency), it ships with a thorough test suite and example code, and supports the ability to add customizations. PyModbus exposes an API for all of the standard Modbus commands, including the ones needed for this application, namely: read holding registers (Modbus function code 03), write single register (function code 06) and write multiple registers (function code 16).

The Tango device code has been separated from the low-level Modbus communication through the creation of a PaSD Modbus API. This comprises two parts: a client side which has public methods used by **MccsPasdBus** to read and write attributes and execute commands, and a server side which provides a Modbus interface to a simulator used for testing purposes, as shown in Fig. 2.

Polling Model

A **PasdBusRequestProvider** class is used to determine what Modbus commands are sent to the hardware, and in what order. This keep tracks of what commands or register writes have been requested by the user (e.g. to turn a SMART box port on or off, or reset an alarm register), and also maintains a list of registers which need to be regularly polled so that the Tango device attributes can be updated to reflect the real state of the hardware. On each poll iteration, the request provider checks if a register write has been requested and needs to be forwarded down to the Modbus API client. In the absence of any command, it iterates through the FNDH and SMART box devices, requesting the next block of contiguous registers to be read.

SIMULATION AND TESTING

There are functional, integration and unit tests for all the components of the PaSD software implemented with pytest. The tests are run as part of the CI/CD pipelines on SKAO's Gitlab [7]. This infrastructure does not have access to the PaSD hardware, therefore the tests are run against a pure python **PasdBusSimulator** which communicates with the **MccsPasdBus** via a Modbus server. A Field Station is simulated consisting of a single FNDH simulator and up to 24 separate SMART box simulators.

The simulator functionality includes:

- Loading the port configuration (which FNDH port each SMART box is connected to, and which SMART box port each numbered antenna is connected to) from a YAML file.

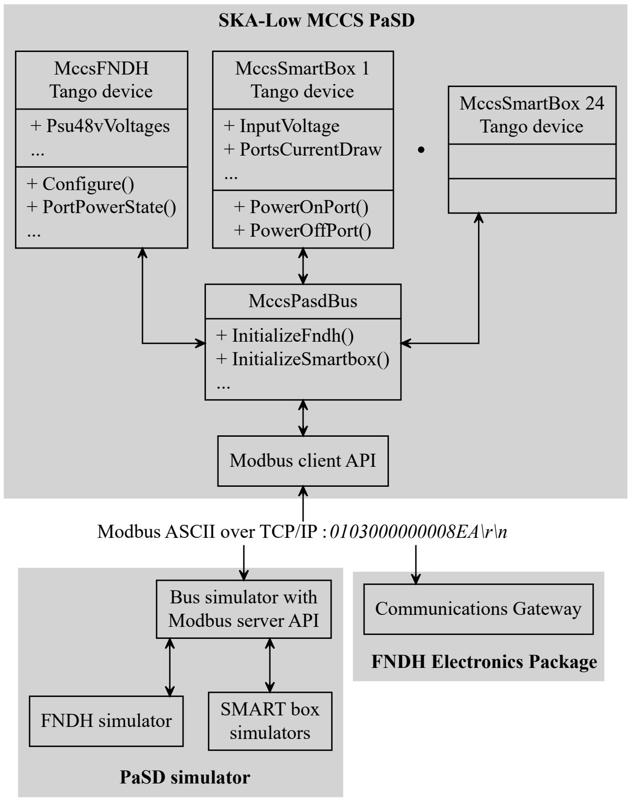

Figure 2: Monitoring, Calibration and Control System of Field Station PaSD subsystem (hardware or simulator).

- The coupling of SMART boxes to the power state of the FNDH port to which they are connected. In other words, if an FNDH port is turned off, the attached SMART box becomes unresponsive from MCCS's perspective.
- The ability to read/write all registers to be accessed during normal operation
- System status initialization, status transitions and their effects, e.g. forcing ports off after an alarm condition.
- Reporting the alarm and warning state of sensors, and setting their thresholds.
- SMART box port over-current breaker tripping and resetting.
- The ability for field technicians to override the devices' states for maintenance.

Limitations

All simulated attributes such as voltages, currents and temperatures are static and are not changed or randomized, as this would complicate unit testing. Fault conditions are only forced with function calls to the simulator within tests to check the behaviour of the system.

CONCLUSION

The PaSD instrumentation has been integrated into the Tango Control system through MCCS using the PyModbus module. Furthermore, a simulation of the PaSD Modbus interface has been created to enable further software development and CI testing without access to the actual hardware.

The solution has been tested at the SKA-Low Integration Test Facility (ITF) with a single FNDH and two SMART boxes available to developers. Initial testing on a larger scale system – the first full Field Station (AAVS3) – is currently underway.

ACKNOWLEDGEMENTS

This work fits into a very large project and, as such has been carried out in a collaborative fashion between individuals and teams working for or collaborating with the SKAO project. Acknowledgements go out to:

- The SKAO MCCS Team for patiently supporting us to bring up the complex development environment.
- C. McDonald at LOW ITF for enabling and supporting our access to the PaSD hardware.
- A. Williams at Curtin University, Perth, Australia whose in-depth explanations from his work with both simulating and implementing the client-side python utility for the PaSD was instrumental to get this integration work off the ground.
- M. Safta at Curtin University, Perth, Australia – credits for Fig. 1.

REFERENCES

[1] M. G. Labate *et al.*, "Highlights of the Square Kilometre Array Low Frequency (SKA-LOW) Telescope", *J. Astron. Telesc. Instrum. Syst.*, vol. 8, no. 1, p. 011 024, 2022. doi:10.1117/1.JATIS.8.1.011024

[2] M. I. I. A. Systems, "Modicon modbus protocol reference guide", Rep. PI–MBUS–300, 1996. https://www.modbus.org/docs/PI_MBUS_300.pdf

[3] Y. Gupta, V. Mohile, J. Kodilkar, R. Uprade, Y. Wadadekar, and S. R. Chaudhuri, "Telescope Manager for the SKA", *J. Astrophys. Astron.*, vol. 44, no. 1, 2023. doi:10.1007/s12036-023-09908-0

[4] Tango Controls, https://tango-controls.readthedocs.io/en/latest/overview/overview.html

[5] PyModbus, https://pymodbus.readthedocs.io

[6] pySerial, https://pyserial.readthedocs.io

[7] M. D. Carlo, M. Dolci, P. Harding, J. Morgado, B. Ribeiro, and U. Yilmaz, "CI-CD Practices at SKA", in *Proc. ICALEPCS'21*, Shanghai, China, 2022, paper TUBL04, pp. 322–329. doi:10.18429/JACoW-ICALEPCS2021-TUBL04