ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR02

CONTROL, READOUT AND MONITORING FOR THE MEDIUM-SIZED TELESCOPES IN THE CHERENKOV TELESCOPE ARRAY

U. Schwanke*[1], T. Murach[2], P. Wagner†[2], G. Spengler[1]
for the CTA MST Project and
D. Melkumyan[2], I. Oya[3] and T. Schmidt[2]
[1] Humboldt-University, Berlin, Germany
[2] DESY, Zeuthen, Germany
[3] CTA, Heidelberg, Germany

Abstract

The Cherenkov Telescope Array (CTA) is the next-generation ground-based gamma-ray observatory. Its design comprises several ten imaging atmospheric Cherenkov telescopes deployed at two sites in the southern and northern hemisphere. The inclusion of various array elements, like large-sized, medium-sized and small-sized telescopes, instruments for atmosphere monitoring, etc, into the Array Control and Data Acquisition System (ACADA) poses a particular challenge which is met by an appropriate software architecture and a well-defined interface for array elements. This paper describes exemplarily how the interface is implemented for the Medium-Sized Telescopes (MSTs, 12 m diameter). The implementation uses the ALMA Common Software (ACS) as a framework for software applications facilitating the readout and control of telescope subsystems like the drive system or the pointing camera; the communication with subsystems takes advantage of the OPC UA protocol.

INTRODUCTION AND OVERVIEW

The Cherenkov Telescope Array (CTA) is the next-generation ground-based observatory for gamma-rays in the energy band between some 10 GeV and several 100 TeV. CTA will comprise two arrays of Imaging Atmospheric Cherenkov Telescopes (IACTs) located in the southern (Paranal, Chile) and northern (La Palma, Canary Islands, Spain) hemisphere, respectively. IACTs with different mirror areas (dubbed Large-Sized Telescopes (LSTs), Medium-Sized Telescopes (MSTs) and Small-Sized Telescopes (SSTs)) ensure proper detection efficiencies for gamma-rays over four orders of magnitude in energy. The stereoscopic observation of showers with numerous IACTs results in an angular resolution and flux sensitivity that constitutes a substantial improvement compared to current IACT arrays (like H.E.S.S., MAGIC, and VERITAS). In CTA's (initial) α-configuration, the northern installation will comprise 4 LSTs and 9 MSTs; 14 MSTs and 37 SSTs will be installed at the southern site.

The readout, control and monitoring of different array elements (telescopes and atmospheric monitoring devices) is the task of the Array Control and Data Acquisition (ACADA) system. The ACADA system is quite complex since it also needs to cope with the concurrent automatic operation of multiple IACT sub-arrays and the rapid re-scheduling of observations in response to science alerts generated internally or by external astronomical facilities. The ACADA software is based on an architecture designed using the Unified Modeling Language (UML) and Systems Modeling (SysML) formalisms [1]. The implementation of applications takes advantage of the ALMA Common Software (ACS [2]) framework.

It adds to the challenge for the ACADA system that the three telescope types (LSTs, MSTs, SSTs) use different optics designs, employ different auxiliary hardware devices (e.g. reference lasers, pointing cameras, light flashers) and implement different operational procedures (observations, calibration, monitoring). A key assumption of the software design is that ACADA can control any telescope without knowing its type and that all telescopes implement a common stateful behaviour via a finite state machine (FSM). This paper describes the software aspects of the interface between ACADA and a telescope. It uses the MSTs as the primary example and starts therefore with a description of the telescope hardware. The implementation of the software interface using ACS and the entire MST software system are detailed as well.

THE MSTS FOR CTA

Telescope Optics

The MSTs [3] are IACTs with a field of view (FoV) of about 8° in diameter and an effective mirror area of about 88 m^2. In the CTA installations, they will be arranged at a typical distance of $O(100)$ m from each other and will cover the central energy range between 100 GeV and 30 TeV. The tesselated mirrors consist of 86 hexagonal spherical mirror facets with a focal length of $f = 16.07$ m that are arranged on a sphere with a radius of curvature of $R = 19.2$ m. The telescope focal length in this so-called modified Davies-Cotton design is $F = 16$ m. The heart of the telescope is a camera[1] with a flat surface that is located in the telescope focal plane and uses about 1800 PMT pixels for the detection of Cherenkov light.

There are two Cherenkov camera projects differing in the number of pixels and the way of storing and processing PMT

* schwanke@physik.hu-berlin.de
† Software for Science (Berlin)

[1] In the following, the term *camera* without further qualification is used only for the Cherenkov camera.

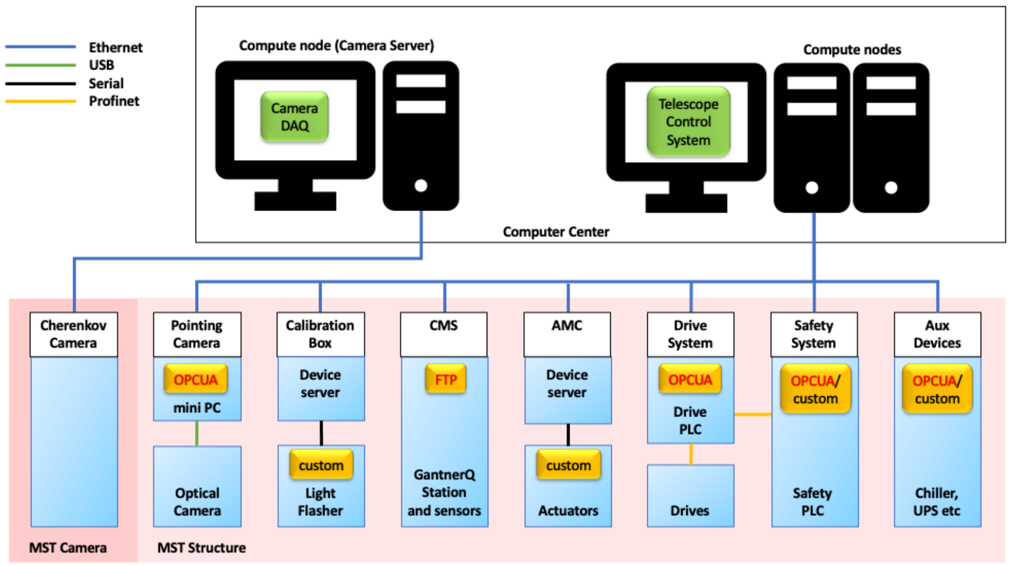

Figure 1: Breakdown of hardware and software systems involved in the operation of an MST. Rectangles denote hardware units. Protocols are denoted by yellow rectangles with rounded corners. See text for explanations.

signals. The MSTs in the southern CTA installation will be equipped with FlashCam cameras (1764 pixels, FADC-based data acquisition system [4]); NectarCAM cameras (1855 pixels, temporary storage of signals in an analogue ring buffer [5]) will be deployed in the north. It is beyond the scope of this paper to do justice to both camera projects. In the following, the focus will therefore be on the *MST structure* (everything but the Cherenkov camera). The description of the camera will be somewhat coarser and will only address those conceptual properties common to FlashCam and NectarCAM.

Telescope Subsystems

The Cherenkov camera is the most important subsystem of an MST. The acquisition of camera data will result in data streams of about 1 GB/s per MST. Next to the camera there are the following subsystems that have to be handled by the telescope control software (cf. lower part of Fig. 1):

- **Pointing Camera:** The position of the Cherenkov camera with respect to the MST structure and stars in the sky is measured by a wide-FoV (about $15^\circ \times 20^\circ$) optical camera. The pointing camera is located in the centre of the telescope mirror and roughly aligned with the optical axis of the MST. It images simultaneously stars in the sky and positional LEDs at the camera.
- **Calibration Box:** A light flasher[2] is mounted next to the pointing camera. It illuminates the Cherenkov camera with the aim of recording the response of the camera pixels to short light flashes. Equalizing the pixel response is part of the calibration of the Cherenkov camera.
- **Conditions Monitoring System (CMS):** Preventive maintenance of the telescope hardware shall be accomplished by the usage of a CMS [6] comprising $O(10)$ acceleration and displacement sensors along with a dedicated data acquisition system for the sensor readings. The sensors are mounted at components of the drive system (motors, gear units, worm shaft) and at locations in the telescope structure. Time-dependent changes of the sensor response to external (e.g. wind) or internal (e.g. well-defined telescope movements) excitations would be indicative of problems.
- **Automatic Mirror Control (AMC):** The 86 mirror facets have to be aligned such that light from a point source is imaged into a small spot in the telescope focal plane. For this purpose, the mirror facets have a triangular fixation, and two fixation points of each facet can be moved with the help of stepper motors (actuators) by a few centimeters. The proper actuator settings are typically found in a calibration process using images of stars in the telescope focal plane recorded by the pointing camera [7].
- **Drive System:** The MSTs have an azimuth-elevation mount. Telescope movements are facilitated by four elevation ($-20^\circ < \mathrm{el} < 91^\circ$) and two azimuth ($-270^\circ < \mathrm{az} < 270^\circ$) motors moving the corresponding drives based on encoders readings.
- **Safety System:** Safety of humans and equipment is ensured by a number of sensors, hardware switches (e.g. emergency buttons) and interlocks. The corresponding signals are captured and processed by a safety system which also communicates with the other subsystems.
- **Auxiliary Devices:** There are a number of smaller subsystems (e.g. a chiller for the cooling of the Cherenkov

[2] The description here and in the following refers to MSTs equipped with NectarCAMs.

camera and the telescope structure, an uninterruptible power supply (UPS)) that produce slowly varying sensor readings. The measurements comprise temperatures, pressures, humidities and rates of flow.

The drive system and the Cherenkov camera are vital for scientific observations. A failure of the pointing camera during observations would result in data sets with reduced pointing accuracy. It is expected that the mirror alignment obtained with the help of the AMC and the pointing camera is so stable that AMC only needs to be operated during the alignment or in emergency situations (e.g. to defocus the telescope to prevent damage due to the Sun). A CMS will be deployed and studied at the first MST scheduled for construction at the northern CTA site.

THE ACADA–TELESCOPE INTERFACE

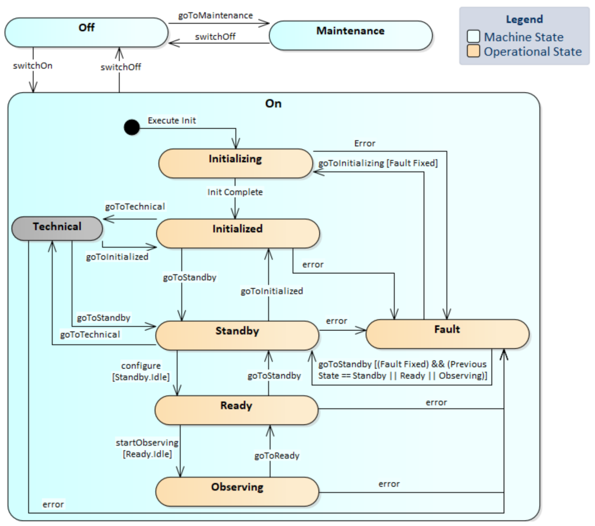

Figure 2: State machine of a CTA telescope. There are three machine states, of which the On state has several operational sub-states. Transitions between states are indicated by arrows.

The generic interface for CTA telescopes is primarily used by ACADA. This interface describes high-level methods that are primarily used to control telescopes (and other types of array elements, which will be omitted in the discussion below) and to obtain basic information about the state of the respective array element. The most important set of methods defined in the interface relate to state transitions of the array element. An incomplete list of relevant transitions is shown in Table 1. These transitions, together with the states and sub-states a telescope can assume, are also visible in Fig. 2. As can be seen from the table, these are high-level methods, and detailed control of e.g. individual hardware elements is not intended. The usage of configuration databases is assumed and reflected in certain interface method signatures. For example, both the telescope structure and the camera can load different configurations depending on the exact type of observations that shall be conducted. The respective

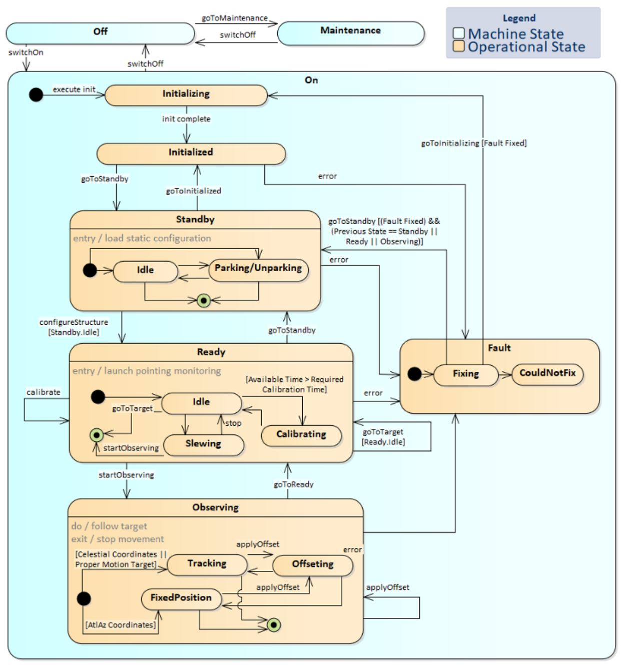

Figure 3: State machine of the structure of a CTA telescope. There are three machine states, of which the On state has several operational sub-states. Four of the operational states have sub-states themselves. Transitions between states are indicated by arrows.

Table 1: Selection of the most relevant telescope transitions. These transitions represent the core of the telescope interface. The last three transitions can optionally start observations with the camera. See Fig. 2 for a list of telescope states and allowed transitions.

Transition	Description
goToInitialized goToStandby goToReady goToTechnical	Puts telescope to the Initialized/Standby/ Ready/Technical state
startObserving	Starts observation of a previously defined target
goToSkyTarget	Lets telescope track a celestial coordinate
goToProperMotionTarget	Lets telescope follow a target by providing a trajectory
goToFixedPosition	Points telescope to a fixed position

interface method passes configuration IDs to a so-called TelescopeManager (cf. upper part of Fig. 4) which is the highest-level abstraction of a telescope. These IDs are then used to look up configuration parameters from a database.

For standard observations, the interface provides only methods for controlling the TelescopeManager. Access to subsystems, like the telescope structure or the camera, is not foreseen. For special-purpose operations the telescope

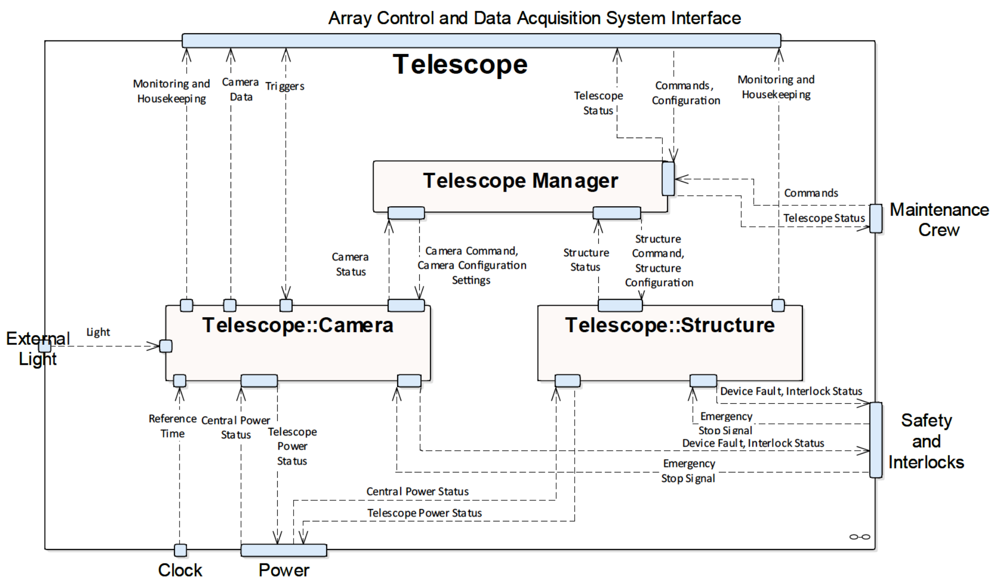

Figure 4: High-level view of the interfaces of the TelescopeManager, the camera and structure subsystems. When performing standard observations, ACADA only interacts with the TelescopeManager, which is in charge of supervising and controlling subsystems. When the telescope is in the Technical state, ACADA accesses and interacts with the two telescope subsystems directly. External devices and actors interacting with the telescope manager and subsystems are displayed as well. The exchange of commands and data is displayed through arrows.

can enter the Technical state. In this state, ACADA can request from the TelescopeManager access to an extended interface with direct control over the Cherenkov camera and the structure (CameraManager and StructureManager in Fig. 4). Also more detailed information about the telescope and subsystem statuses is available in this mode.

When maintenance work is ongoing at a telescope, the TelescopeManager shall reflect this by assuming the Maintenance state. The respective transition is initiated by a physical switch at the telescope. In this state, any remote control by e.g. ACADA is prohibited to ensure human safety. Obtaining monitoring data is still possible in this state, though.

The monitoring of telescope parameters and conditions is not part of the interface. Also the transfer of bulk data from the cameras is not specified through the interface discussed here.

The telescope interface is implemented from both sides. For the development of the user side, i.e. the ACADA system, a mock implementation of a generic telescope has been developed in Python. Telescope states, relevant transitions and subsystem transitions have been implemented to mimick realistic and configurable behaviour of telescopes. Threads are used to simulate long-lasting procedures and transitions, and callbacks are used as primary concept for notifying clients of the mock TelescopeManager about the progress of such operations. The mock implementation of the TelescopeManager is used in a variety of automated software tests of ACADA, such as verifications of high-level ACADA use cases.

The interfaces between the TelescopeManager or ACADA on the one side and the StructureManager and the CameraManager on the other side are also defined. As indicated above, ACADA can only access these components when the telescope is in the Technical state.

As shown in Fig. 3, a state machine has also been defined for the structures of CTA telescopes. The TelescopeManager (or ACADA) requests transitions between operational states. Internal states as well as transitions between them are handled by the StructureManager. Sub-states only need to be implemented if applicable.

Several states and sub-states can be entered by the telescope (or subsystem) autonomously. For example, calibration routines can be performed automatically by the telescopes if the telescope was instructed by ACADA to start observing at a given time and there is enough time in advance to calibrate subsystems. In this case, the StructureManager can decide to enter and leave the Ready.Calibrating state. Also in case an error occurred the TelescopeManager can attempt to fix the problem, and if successful, the telescope (or the respective subsystem) can return to a non-Fault state by itself.

READOUT, CONTROL AND MONITORING OF MSTS

Protocols, Standards and Data Formats

The data stream from the camera is dominating the bandwidth that the ethernet network (blue lines in Fig. 1) between an MST and the local computer centre on a CTA site has to provide. In the event of a camera trigger, the digitised PMT signals are transferred to a dedicated computer (*camera server* in the upper left part of Fig. 1) which caches all data for a period of about 1 s and processes the data further. Typically only events where at least one other telescope recorded useable camera data simultaneously are kept and stored on hard disk for offline analysis. The stereoscopy requirement (more than one telescope with data) is enforced by a central software array trigger (SWAT). The SWAT operates on trigger time stamps sent to it by all cameras in the array and informs the camera servers about data of interest. The transfer of camera data uses a highly-performant bulk-data transfer service that is based on the `ZeroMQ` messaging library and can also compress data [8].

All other data streams for readout, control and monitoring are fairly modest in size. The high number of different telescope subsystems and the need to upgrade their hardware and software over the anticipated lifetime of CTA (about 30 years) motivate, however, the prescription of standard protocols and mechanisms for the data transfer. For CTA, the combination of an ACS DeviceIO[3] with the OPC UA industrial communication protocol is the method of choice. OPC UA has been standardised by the OPC Foundation. It has growing support in industry and a number of software development kits exist. An implementation of an ACS DeviceIO for OPC UA has been made available to the telescope projects. To be practical, the following three options for the inclusion of a hardware device/telescope subsystem exist:

1. Devices that come with a vendor-side OPC UA server can be immediately integrated by means of the ACS DeviceIO for OPC UA.
2. For devices that communicate via a dedicated device protocal an OPC UA server needs to be developed. The server will translate between the device protocol and OPC UA and will be integrated as described above.
3. If the development of an OPC UA server is not an option a device with a dedicated protocol can be included by providing an ACS DeviceIO for this protocol.

Local Control Systems of the MST Structure

The usage of ACS is confined to the Telescope Control System (TCS, cf. upper right part of Fig. 1). The TCS provides the higher-level control and implements the telescope interface described above. The division of labour between the TCS and the software systems located at the telescope is motivated by safety aspects and the wish to minimize future changes of the software at the telescope. The communication between the TCS in the computer centre and an MST structure uses ethernet. The chosen hardware and software solutions are as follows (cf. lower part of Fig. 1):

- **Pointing Camera:** The pointing camera consists of an optical camera and a mini PC in a custom-built housing. The housing provides temperature stabilization and features a number of internal sensors. The camera and the mini PC are connected by USB; the mini PC has an OPC UA server for control and image readout.
- **Calibration Box:** The LED-based light flasher is managed by firmware on an 8bit microcontroller using a custom protocol. A device server is used to convert the serial communication to ethernet.
- **Conditions Monitoring System:** The data acquisition for the conditions montoring system uses a real-time board (Gantner Q.station) that digitises the arriving analogue sensor signals and buffers them. At present, the board sends the data to an FTP server in the computer centre. There are also ways to transmit data for storage.
- **Automatic Mirror Control:** The communication with the $2 \times 86 = 172$ actuators uses serial lines and a custom protocol. The serial lines are converted to ethernet with the help of an device server. Each actuator provides not only the current position of the stepper motor but also monitoring information (e.g. a temperature). There is the option to choose between several actuator settings stored in the firmware.
- **Drive System:** The drive system at the telescope needs no knowledge about astronomy and basically only deals with time, azimuth and elevation. A programmable logic controller (PLC) accepts precalculated tables with azimuth and elevations as a function of time in discrete time steps. The PLC code interpolates the information in the table and controls the drive motors such that the intended path in the sky is followed (e.g. to implement siderial tracking). The interface to the outside world is an OPC UA server.
- **Safety System:** The safety system is also a PLC that bundles and processes all safety-relevant information. The communication with the outside world proceeds via an OPC UA server. The Safety PLC offers some extra monitoring data (e.g. tables with motor torques at rates much higher than a few Hz) via a custom protocol. All PLCs in the system (Drive PLC, Safety PLC) are connected via the Profinet bus.
- **Auxiliary Devices:** The readout and monitoring of chiller and UPS will depend on the exact models that will be chosen in the future. It is envisaged that the information (e.g. sensor readings or status flags) will be made available as an OPC UA server.

THE TELESCOPE CONTROL SYSTEM

Structure of the TCS

For the implementation of an MST-specific TelescopeManager, the Java programming language has been chosen. All relevant interface methods have been implemented, and their functionality is verified by a set of unit and component tests. Focus is put on achieving a good test coverage. In the following, only the StructureManager will be discussed in detail since the camera software design depends on the deployed hardware (FlashCam or NectarCAM).

For the MST StructureManager, the common interface has been extended by a set of custom methods that are required to control those subsystems (e.g. the AMC and the

[3] An ACS DeviceIO provides a mapping between a node in a CORBA object (e.g. Property in an ACS Component) and a monitoring/control point in an external hardware device.

pointing camera) that are only available for MSTs. ACADA is not aware of these methods, but the TelescopeManager makes use of them for both commissioning and regular operations. The extended interface of the StructureManager is described by an appropriate IDL[4] file. Placing the extended interface in the StructureManager has the advantage that the full functionality of an MST structure is available to clients. This removes the need for clients to interact directly with subsystems controlled by the StructureManager.

Implementation as ACS Components

The TCS for an MST is implemented as a flat hierarchy of ACS components. The TelescopeManager (cf. Fig. 5) controls the CameraManager and StuctureManager and calculates its state as a function of the states of the two subsystems. The StructureManager interacts directly with the Safety System to interrogate the hardware state (interlocks, emergency switches) of the MST structure and includes this information in the computation of its own state. There are four basic groups of ACS components that aid the StructureManager to read out and control the MST structure hardware:

- **Pointing Camera:** The top-level ACS component for the pointing camera receives configuration information (rates and exposures for image acquisition) and launches threads for the acquisition of images from the pointing camera. A *Bridge* ACS component is used for mapping to the OPC UA protocol.
- **Automatic Mirror Control:** The top-level ACS component for the AMC deals with the AMC configuration (number and arrangement of actuators) and accesses an OPC UA server via a *Bridge*. The OPC UA server for the AMC is executed locally and uses a custom protocol for the actual communcation with the hardware.
- **Drive System:** The ACS component for the drive system forwards simple commands (e.g. pointing the telescope to a fixed position in azimuth and elevation) directly to the drives system hardware. For tracking of astronomical targets it generates the time-dependent trajectory in azimuth and elevation in time steps of about 250 ms. Pointing corrections[5] are applied through a a pointing model in this step. About 1000 trajectory steps (corresponding to 250 seconds of tracking) are initially sent to the drive PLC and buffered there. The buffer is topped off whenever necessary.
- **Auxiliary Devices:** The readout and control of the auxiliary devices is managed by ACS components coupled to a *Bridge*.

Clients and Graphical User Interfaces

ACADA is the most important user of the MST TCS. It regularly inspects the TelescopeManager state to check the availability of the TCS (and hence the telescope hardware) for standard obervations. For a typical night with observations ACADA prepares the telescopes well before dark time. In this preparatory phase, the telescopes leave the parking position and execute telescope-specific routines (attain the operating temperature for the Cherenkov camera, record calibration data). Observations of scheduled astronomical targets are conducted when the conditions are optimal (astronomical darkness, partial Moon). There is also a closing phase near dawn; it is complementary to the preparatory phase and ends with the telescopes returning to the parking positions.

In periods of telescope commissioning, extended calibration and trouble-shooting, expert personnel requires a more fine-grained access to the TCS. For this purpose, the TelescopeManager can be brought to the *Technical State* (cf. Fig. 2) and signals in this way that a remote control by ACADA is prohibited. There a then two mutually exclusive ways to control an MST: by means of a graphical user interface (GUI) and by means of scripts that manipulate the TCS directly. In both approaches, the TelescopeManager must finally leave the *Technical State* again so that ACADA can reclaim control.

Figure 6 shows a screenshot of the browser-GUI that can be used for the MST structure. The underlying server takes advantage of the same software technologies that are used for the CTA operator GUI. The server is written in Python as an application of the `Pyramid`[6] open-source web framework. It uses native Python features (`asyncio`) and the websockets library to implement concurrency and the communication with a client, respectively. Internally, the server gains access to ACS components of the TCS. The offered functionality comprises actions with the drive system (track astronomical targets, point the MST to a coordinate in azimuth and elevation etc) and basic tests with the AMC (actuator movements). A passive monitoring of basic telescope parameters (e.g. the current pointing direction, cf. Fig. 6) is available at all times while active control of the MST structure is only possible in the *Technical State*.

Client scripts (in Python) are the most flexible way to access the TCS. While ACADA works exclusively with the TelescopeManager the scripts can also approach the StructureManager, the CameraManager or any of the underlying ACS components. A typical example for the MST structure is a script implementing a procedure for mirror alignment. In this application, the script would use the standard interface of the TelescopeManager to initiate the tracking of a star. The extended interface of the StructureManager is then used to vary the orientation of the mirror facets (by means of the actuators) and to record images with the pointing camera.

ACKNOWLEDGEMENTS

The authors acknowledge the work of numerous members of the CTA consortium in the ACADA and MST Projects which helped to shape the MST and ACADA software. We gratefully acknowledge financial support from the agencies

[4] interface definition language
[5] Pointing corrections are needed whenever the azimuth and elevation of the drive system are not identical with the astronomical azimuth and elevation.
[6] https://www.trypyramid.com

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR02

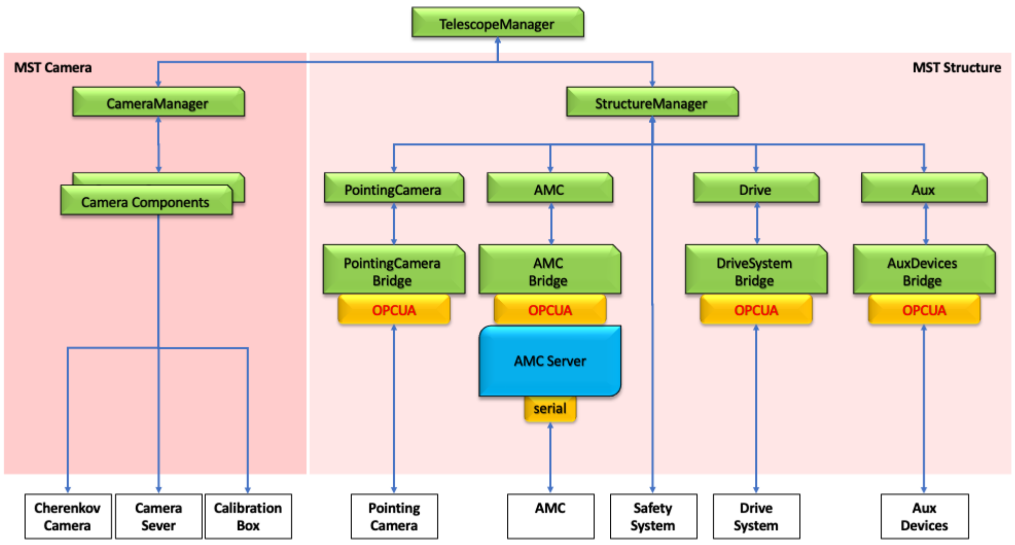

Figure 5: Breakdown of the TCS executed for each MST in the computer centre. Green boxes denote ACS components; the blue box with two rounded corners is an OPC UA server that runs locally. Arrows indicate both control and data transfer. The hardware at the MST that is accessed by the TCS is shown by the boxes at the bottom. Protocols are denoted by rectangles with rounded corners. The *Bridge* ACS components provide the mapping to OPC UA. Note that the number of ACS components for the Cherenkov camera depends on the camera project. See text for more explanations.

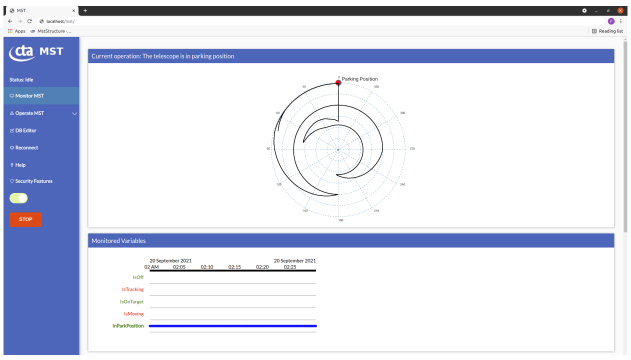

Figure 6: Screenshot of the browser-based GUI for monitoring and control of the MST structure. The black line illustrates the trajectory of the telescope pointing direction in azimuth and elevation.

and organizations listed here:
`http://www.cta-observatory.org/consortium\_acknowledgments`

REFERENCES

[1] I. Oya *et al.*, “Designing and prototyping the control system for the cherenkov telescope array,” *Journal of Physics: Conference Series*, vol. 1085, p. 032 045, 2018, doi:10.1088/1742-6596/1085/3/032045

[2] G. Chiozzi *et al.*, “The ALMA common software: a developer-friendly CORBA-based framework,” in *Advanced Software, Control, and Communication Systems for Astronomy*, vol. 5496, 2004, pp. 205–218, doi:10.1117/12.551943

[3] A. Schulz, M. Garczarczyk, L. Oakes, S. Schlenstedt, and U. Schwanke, “Building medium size telescope structures for the Cherenkov telescope array,” in *6th International Symposium on High Energy Gamma-Ray Astronomy*, vol. 1792, 2017, paper 080008, p. 080 008, doi:10.1063/1.4969029

[4] B. Bi *et al.*, “Performance of the New FlashCam-based Camera in the 28 m Telescope of H.E.S.S,” *arXiv e-prints*, paper arXiv:2108.03046, arXiv:2108.03046, 2021.

[5] T. Tavernier, J. F. Glicenstein, and F. Brun, “Status and performance results from NectarCam - a camera for CTA medium sized telescopes,” in *36th International Cosmic Ray Conference (ICRC2019)*, vol. 36, 2019, paper 805, p. 805.

[6] V. Barbosa Martins, G. Spengler, M. Garczarczyk, U. Schwanke, MST-STR Project, and CTA Consortium, “A Condition Monitoring Concept Studied at the MST Prototype for the Cherenkov Telescope Array,” in *36th International Cosmic Ray Conference (ICRC2019)*, vol. 36, 2019, paper 626, p. 626.

[7] T. Murach *et al.*, “Automatic mirror alignment for the medium-sized telescopes of the Cherenkov Telescope Array using the Bokeh method,” in *Ground-based and Airborne Telescopes VII*, vol. 10700, 2018, paper 107001U, 107001U, doi:10.1117/12.2313517

[8] E. Lyard, R. Walter, and C. Consortium, “End-to-end data acquisition pipeline for the Cherenkov Telescope Array,” in *35th International Cosmic Ray Conference (ICRC2017)*, vol. 301, 2017, paper 843, p. 843.