# The H.E.S.S. central data acquisition system

A. Balzer[a,b], M. Füßling[b], M. Gajdus[c], D. Göring[d], A. Lopatin[b], M. de Naurois[e], S. Schlenker[f], U. Schwanke[c], C. Stegmann[b,a]

[a] *DESY, D-15738 Zeuthen, Germany*
[b] *Institut für Physik und Astronomie, Universität Potsdam, Karl-Liebknecht-Strasse 24/25, D-14476 Potsdam, Germany*
[c] *Institut für Physik, Humboldt-Universität zu Berlin, Newtonstr. 15, D-12489 Berlin, Germany*
[d] *Physikalisches Institut, Universität Erlangen-Nürnberg, Erwin-Rommel-Str. 1, D-91058 Erlangen, Germany*
[e] *Laboratoire Leprince-Ringuet, Ecole Polytechnique, CNRS/IN2P3, F-91128 Palaiseau, France*
[f] *CERN, CH-1211 Geneva 23, Switzerland*



**Abstract**

The High Energy Stereoscopic System (H.E.S.S.) is a system of Imaging Atmospheric Cherenkov Telescopes (IACTs) located in the Khomas Highland in Namibia. It measures cosmic gamma rays of very high energies (VHE; $> 100\,\mathrm{GeV}$) using the Earth's atmosphere as a calorimeter. The H.E.S.S. Array entered Phase II in September 2012 with the inauguration of a fifth telescope that is larger and more complex than the other four. This paper will give an overview of the current H.E.S.S. central data acquisition (DAQ) system with particular emphasis on the upgrades made to integrate the fifth telescope into the array. At first, the various requirements for the central DAQ are discussed then the general design principles employed to fulfil these requirements are described. Finally, the performance, stability and reliability of the H.E.S.S. central DAQ are presented. One of the major accomplishments is that less than 0.8 % of observation time has been lost due to central DAQ problems since 2009.

*Keywords:* DAQ, Data acquisition, VHE, Gamma ray astronomy, H.E.S.S.

**Contents**



## Introduction

The High Energy Stereoscopic System (H.E.S.S.[1]) is an array of Imaging Air Cherenkov Telescopes (IACTs) located in the Khomas Highland of Namibia. It is dedicated to the observation of very-high-energy (VHE) gamma rays [1]. IACTs detect Cherenkov light emitted by charged particles in cosmic-ray-initiated air-showers. The measured light distributions are then used to reconstruct the properties of the primary particle. More details about the IACT

*Email addresses:* arnim.balzer@desy.de (A. Balzer), anton.lopatin@fau.de (A. Lopatin)

[1] http://www.mpi-hd.mpg.de/hfm/HESS/

technique, and the state of gamma-ray astronomy in the TeV regime can be found in [2].

Currently, the H.E.S.S. experiment has entered so called "Phase II" with the inauguration of a fifth telescope [3, 4] on September 28$^{\text{th}}$, 2012. This upgrade of the H.E.S.S. Array will lower the energy threshold of the experiment from about 100 GeV [1] to about 30 GeV [5, 6] and improve the overall sensitivity by a factor of $\sim 2$ in the central energy range.

In the course of the upgrade, the central data acquisition (DAQ) system of the array [7] was also revised. The on-site computing cluster (next to the telescopes) was replaced by modern off-the-shelf hardware while the DAQ software was ported to this new environment and adjusted to meet the requirements of H.E.S.S. Phase II.

This paper gives an overview of the central DAQ system and the challenges that were met and overcome during the cluster upgrade. Unlike the DAQ systems of VERITAS [8] and MAGIC [9], the H.E.S.S. DAQ system is also responsible for array control and monitoring. We will therefore address these topics and also mention error propagation within the system, as well as the user interaction with the DAQ. Furthermore, the reliability and performance of the central DAQ are discussed. Finally, a summary of the important lessons learned during the process is given, which might be helpful to upcoming ground-based gamma-ray telescope arrays like the Cherenkov Telescope Array (CTA) [10].

## 1. Requirements

The requirements of a DAQ system for an experiment like H.E.S.S. can be divided into three groups. First, there are design goals that need to be met in order to achieve optimal scientific output. Moreover, there are technical requirements that need to be fulfilled by the DAQ. The last but not least group of requirements is about ensuring that the system is as user-friendly as possible.

### *1.1. Science requirements*

The expected flux of VHE gamma rays observable by ground-based IACTs like H.E.S.S. is very low compared to the expected background rate. The IACT arrays use stereoscopic observation of air showers to help reduce the high background rate of hadron-induced air showers as well as to improve the direction reconstruction of photon-induced air showers [1]. Therefore, the dead time of the whole array must not be increased by the DAQ. As a result, the number of telescopes that detect an air shower and are able to record the event is not to be reduced by the DAQ. Moreover, variations in the trigger rate such as fluctuations induced by night sky background or by transient phenomena (i.e. bursts) should not cause dead time due to DAQ related processing back-pressure.

The maximum hardware event readout rate is about 900 Hz with an event size of 4.5 kB for each of the H.E.S.S. I telescopes (CT1-4) and about 3.4 kHz with an event size of 10 kB for the fifth telescope (CT5) [11]. This leads to maximum data rates of approx. 50 $^{\text{MB}}/_{\text{s}}$ for the primary scientific data during routine operation. To account for other hardware devices on-site as well as some additional capacity for tests, the DAQ system is required to process at least 80 $^{\text{MB}}/_{\text{s}}$, thereby having sufficient throughput for network connections and storage facilities. The same data format, as decided by the H.E.S.S. Collaboration, is used for storage on-site as for high level tasks like event reconstruction and data analysis. This approach allows high level analysis algorithms and visualisation tools to be used on a DAQ level. Therefore, the raw data that are sent from the Cherenkov detectors have to be converted into a common data format at a very early stage, which requires additional computing power. Moreover, the common data format allows data that are identical to those recorded by the DAQ to be easily simulated.

On top of the mere acquisition of data, their integrity and quality needs to be verified in real time. Furthermore, the data should be checked and analysed by a preliminary analysis during observation. Due to the common data format it is possible to use the standard offline analysis software for this purpose. The array needs to be able to operate in different observation modes with different sets of telescopes. Several independent operation modes with disjoint sets of telescopes, so called *SubArrays*, should be possible simultaneously, for example, *observation* runs as well as *calibration* and *maintenance* runs.

The DAQ system needs to be able to respond to target of opportunity (ToO) alerts, like alerts from the Gamma-Ray Burst Coordinates Network[2], in real time. In order to allow for prompt follow-up observations, the DAQ should respond to such alerts as fast as possible (including a slewing of the telescopes to a new observation target) and should thus be highly automated. On top of that, the DAQ must be able to handle continuous data streams of other hardware components (for example monitoring data from weather stations) as well as to merge these streams with the data taken during observation.

The H.E.S.S. Array only takes data during the moonless part of a night when the weather conditions are good, i.e. no clouds are above the detector. To maximise the physics output of these time periods the scheduling of the observation targets should be automated and configurable remotely (i.e. from Europe where most host institutes are).

### *1.2. Technical requirements*

To facilitate the communication between the different devices, subsystems and computers, a common network infrastructure has to be established. This network should be based on robust, proven and commonly available technologies. This, in general, also holds for the hardware of the

[2]The Gamma-Ray Burst Coordinates Network (GCN) distributes information about the location of a Gamma-Ray Burst detected by various spacecraft `http://gcn.gsfc.nasa.gov/`

on-site computing cluster. So, whenever possible, standard off-the-shelf components should be used for equipment like computing nodes, storage servers and desktop PCs. This reduces the cost and simplifies the replacement of broken components.

To account for future developments and upgrades, the DAQ design should be scalable in terms of processing power, data throughput and storage capacity. Furthermore, it should be possible to incorporate new hardware devices easily, i.e. with no change to the general data storage and data transportation structures. All vital DAQ system components should be redundant to allow exchanging broken hardware components quickly to minimise loss of observation time due to DAQ problems. In case of a power cut the DAQ system should remain operational and prevent data corruption.

Each device in the array must be mapped to at least one controlling software process (*Controller*). More complex hardware devices may correspond to multiple *Controllers*. Furthermore, the configuration of the different pieces of hardware, as well as the DAQ itself, should be as flexible as possible. New hardware should be usable without the need to change major parts of the code (e.g. data transport and device monitoring). As a side effect of a flexible configuration, the array is more tolerant to missing or malfunctioning hardware.

To prevent damage to the telescopes, the Cherenkov cameras and other critical systems, error handling needs to be redundant and decentralised. Therefore, each subsystem (e.g. the camera or the drive system) is responsible for its own safety and fatal problems have to be handled directly in firmware. Only after immediate danger is averted, the devices inform the DAQ system of the error condition, which in turn takes appropriate actions, such as propagating the error to other subsystems or bringing the corresponding *SubArray* to a safe state. In order to avoid hazardous conditions for the hardware, slow control information from all the subsystems (e.g. device temperatures, positions or voltages) has to be monitored by the DAQ.

The H.E.S.S. Array is situated in a remote location where no cheap, reliable and fast internet connection is available. Therefore, the data cannot be streamed to the different institutes of the H.E.S.S. Collaboration. Instead, the data have to be shipped to the institutes in regular intervals via magnetic tapes. The DAQ provides the resources to store the data on-site for up to three months until the shipped data has been verified in Europe.

In general, the system should be as self-sufficient as possible. There must be no dependencies of DAQ components on remote network connections whatsoever, meaning data taking must not be interrupted by a failing internet connection. Furthermore, all relevant documentation and manuals need to be available on-site.

### *1.3. User requirements*

The H.E.S.S. DAQ should facilitate integration of new hardware and allow a quick response to hardware malfunctions without interfering with the rest of the array. The software should, therefore, be modular and support different hardware configurations.

The H.E.S.S. Array is operated by collaboration members (PhD students,...) during "Shifts" of one moon period. As a result, the array is operated by monthly changing non-expert personnel on-site (Shifters) with remote support by a team of subsystem experts working at the respective member institutes as well as a local shift expert responsible for the training of the Shifters at the beginning of each Shift. As a security precaution the access of Shifters to critical parts of the array has to be limited (for example configuration databases or low-level telescope movement), while subsystem experts must be able to perform remote maintenance. Moreover, the DAQ software should use stable releases that may not be altered by Shifters for observations, but should still allow subsystem experts on-site to develop and debug their software modules.

As subsystem experts are working at their home institutes most of the time and not on the H.E.S.S. site, a remote access to the various subsystems of the array must be available. Due to limitations in internet connectivity, these remote connections need to be simple. To further ease the maintenance of the array, a dedicated logging system with detailed information for Shifters and subsystem experts has to be part of the DAQ software. For testing purposes, and in case of malfunctioning hardware, the ability to manually override automated procedures must be given at all times.

Due to the non-expert personnel operating the array during a shift, manuals and documentation have to be available on-site at all times. This also includes detailed instructions about safety regulations and error recovery. The DAQ should also be operable in a semi-automatic way by a single Shifter (for security reasons at least two Shifters have to be present at all time). Moreover, the Shifters must be able to get fast feedback about the current status of the array during data taking and especially in the case of system errors. Easy to understand error reports and clearly described error recovery procedures are key to high-efficiency data taking. By design, the Shifters have the final say in deciding whether data taking continues or not after an error has occurred. This also holds true in H.E.S.S. for weather and atmospheric conditions: there is no automated response from the DAQ and the Shifters' decision is necessary.

## 2. Central DAQ hardware

The primary data flow starts in the Cherenkov cameras of the five telescopes. If an air shower event is seen by a telescope, i.e. the camera trigger decided to launch the read out of an event, the camera sends a trigger signal via optical fiber to the Central Trigger and begins to digitise the recorded image. The Central Trigger checks for coincidences in at least two telescopes, if no coincidence is

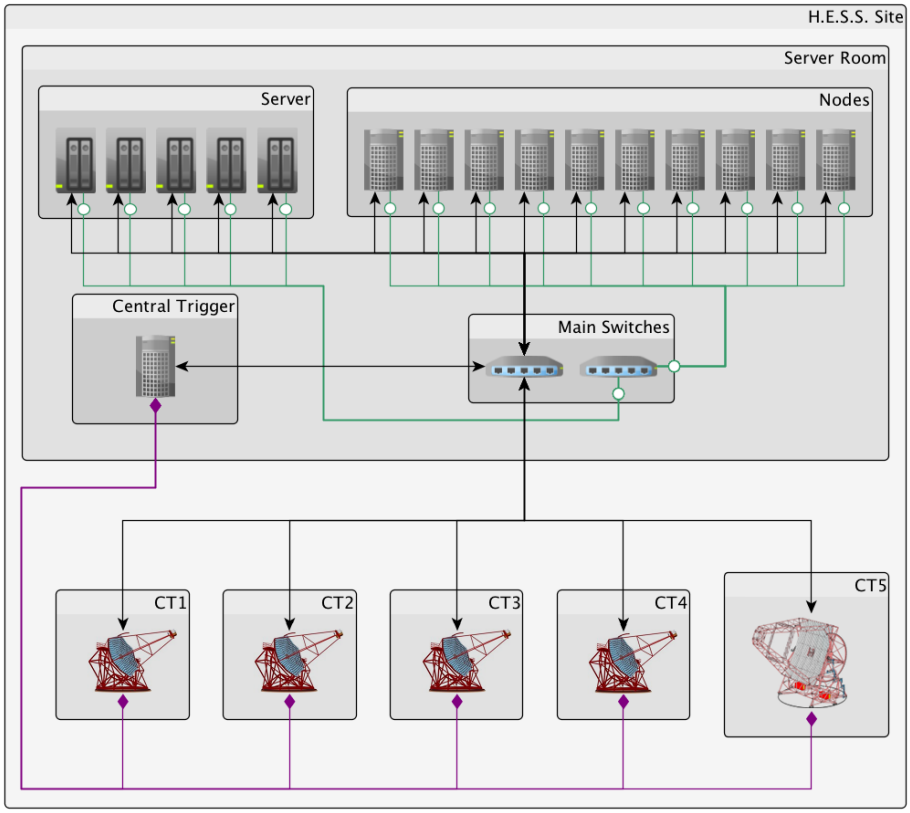


Figure 1: Scheme of the network layout on the H.E.S.S. site in Namibia. In the upper part, the server room with the five storage servers as well as the ten worker nodes is depicted. Furthermore, the Central Trigger as well as the main switches are shown. The Cherenkov Telescopes (CT) are labeled from one to five, with the last one being the H.E.S.S. II telescope. Black lines with arrows indicate gigabit data network ethernet connections while the green lines with circles represent a physically separated gigabit ethernet network for the mounting of the NFS and GlusterFS servers. The purple lines with the diamond shaped ends represent the direct optical fibre connections between the Cherenkov camera trigger sytems and the Central Trigger.

found it sends a "fast-clear" signal to the telescope, which will drop the event. In case at least two telescopes trigger within $\approx 80\,\mathrm{ns}$, the camera image is sent to the DAQ. A more detailed discussion of the H.E.S.S. II Central Trigger and camera readout can be found in [11, 12]. All the data recorded by the different cameras that belong to one event are sent via network to one node (see Figure 1) in the H.E.S.S. DAQ cluster. Furthermore, the data from the Central Trigger are sent to the same node, including a unique event number and GPS time stamp for each event. The received raw data are buffered in memory at the receiving node, converted to a common data format by that node and stored on file servers.

### *2.1. Network implementation*

The communication between the different components of the DAQ is done via a Gbit Ethernet network only. If a piece of equipment is not able to use Ethernet, a gateway server is used, e.g. a serial COM-Server. Several Gbit switches connect all hardware with each other on the H.E.S.S. site via two 48 Port switches as the central communication hub inside the server room. Long distances are covered using Gbit media converters and optical fibers. The connection between the camera trigger systems and the Central Trigger is a direct optical fibre connection and not part of the normal data network. This is done to achieve minimal signal latency to reduce the over all dead time of the array. The average mean dead time during normal observation is $\approx 8\,\%$ for the whole array ($\approx 7\,\%$ for CT5 and $\approx 9.3\,\%$ on average for the others). For the dead time of the camera electronics see [11]. Another exception are the distributed file systems (see Section 2.2) used within the H.E.S.S. DAQ which are accessed via an additional physically separate Gbit Ethernet network so that they can't interfere with data taking.

### *2.2. Computing & storage servers*

The cluster of the H.E.S.S. II Array consists of ten worker nodes and five storage servers. It is located in a climate controlled server room inside the control building next to the telescopes. Each storage server is equipped with a 12 TB RAID6 [13] with an additional hot spare disk using an XFS [14] file system. This ensures redundancy as well as enough IO bandwidth and disk space for data taking. The NFS [15] and GlusterFS [16] protocols are used for distributed file access through the worker nodes. Due to the low-bandwidth internet connection on-site, data have to be transported using magnetic tapes at the end of each Shift, currently LTO-4 tapes [17] are used. The storage capacity on-site is sufficient to hold all data until it has been received and verified in Europe which does not take longer than 3 months. Every month H.E.S.S. Phase I takes 420 GB of data and for H.E.S.S. Phase II this value is around 11 TB, while, the total available disk space on-site is about 60 TB.

Different hardware subsystems require special purpose machines, like e.g. custom boot servers for the camera and Central Trigger board electronics. These custom machines

have been developed together with their respective hardware components in the lab. On-site they have been turned into virtual machines which are hosted by one of the servers. These virtual machines are fully customized to their respective task, e.g. using 32 bit CPU architecture. Furthermore, it is possible to backup the machines and migrate them to another physical host in case of a hardware failure.

### *2.3. Cluster monitoring & maintainability*

The maintenance of the H.E.S.S. central DAQ computing cluster and the DAQ software itself is crucial for the operation of the experiment. To facilitate the maintenance and to reduce the complexity of the overall system, the cluster was designed to be as homogeneous as possible. This means that the different computing nodes and storage servers are based on the same hardware and run the same operating system allowing the same spare parts and same software to be used for all machines. To cope with power interruptions, which are quite common at the H.E.S.S. site (several times a week), a diesel generator in combination with online/double-conversion uninterruptible power supplies (UPSs) [18] are used to ensure a continuous power supply to the cluster. The diesel generators provide enough power for the telescopes and all electronic equipment on-site. However, since they need several seconds to kick in, the UPS are needed to ensure a constant power output to the computing cluster. If necessary, the DAQ cluster can be sustained by the UPS for up to 25 min. The status of the cluster is monitored constantly with several open source tools like *Ganglia* [19] and *Collectd* [20]. On top of that, email notifications as well as *SNMP* traps are used to get error notifications from the different hardware components.

### *2.4. Software maintenance & backups*

A uniform administration of all machines in the cluster is possible, using the same software packages. The *System Imager* tool [21] is used to create daily backups of the current operating system, including all relevant software to run the H.E.S.S. central DAQ. If the operation system is flawed after a software change, the System Imager allows the system to be rolled back to a previous, stable image with a single command. Dedicated backups of important files are performed on a daily basis. With increasing age of the backups, the frequency is reduced from daily, to weekly, to monthly snapshots. The databases (see section 3.6), that are being used throughout the DAQ, are also backed up on a daily basis.

## 3. DAQ software concept and implementation

### *3.1. Data format*

The transport and storage of objects needs a serialisation mechanism which is provided by the ROOT Data Analysis Framework [22] on which the H.E.S.S. raw data format is based.

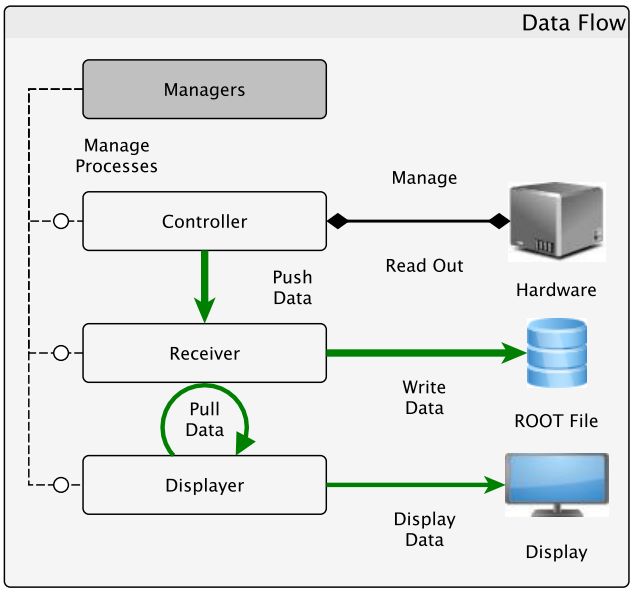


Figure 2: Data flow within the H.E.S.S. DAQ. Each piece of hardware is read out and managed by at least one DAQ *Controller* process. The data which is obtained from the hardware is sent using the *Push* mode to the corresponding receiver process. In *Push* mode data integrity is guaranteed. The received data is then stored in ROOT files using the common H.E.S.S. data format. For fast feedback to the Shift Crew, the data processed by a receiver can be pulled by a *Displayer* process and displayed on one of the available screens in the control room. The data can be sent at an arbitrary rate. Error handling and process synchronisation is taken care of by the *Manager* process.

To ensure a common interface and easy access to all the different raw data formats used in H.E.S.S., a common raw data base class is implemented in the software. This base class also takes care of correct time stamps for the different events that are recorded and saved to disk. This enforces a correct order in the loading and processing of events. Both the ROOT-based H.E.S.S. data format and the ROOT graphics and histogram classes are used online and off-line allowing a seamless integration of the high-level data analysis code into the DAQ software, see Section 4.2.

### *3.2. Data transport*

The communication between the different processes in the DAQ is built on the CORBA [23] distributed, object-oriented inter-process communication standard. A client process can call remote procedures on a server object using CORBA messages. The result of the function call (with possible read, read-write and write parameters) is propagated back to the client after the execution of the code on the server side. Due to the combined usage of CORBA and ROOT, several multi-threading issues in the ROOT Framework were discovered. To achieve the required thread safety, several patches have been contributed to ROOT, and are now part of all ROOT releases [24, 25].

CORBA allows the use of different programming languages and different operating systems on the server and

the client side. The H.E.S.S. DAQ uses the free (LGPL) omniORB [26] implementation, which comes with C++ and Python support. A central directory naming service provides object registration (using a hierarchical naming tree with processes being grouped by CORBA contexts) and allows any process to easily obtain connections to other remote processes.

The H.E.S.S. DAQ implements two different data transport mechanisms: pushing and pulling of data, see Figure 2. In *Push* mode the data are sent from one process to another, ensuring the data reception and integrity. The *Push* mode is used to store all measured data, including the scientific data from the Cherenkov cameras. In *Pull* mode, a client polls the server process for new data in periodic intervals. Like this, data can be dropped or requested at a lower rate, but the data integrity is still ensured. This transport mechanism is used by displaying processes where a sub-sample of the data is sufficient.

To store data on disks, multiple instances of a generic receiver process are used throughout the DAQ. These receiver processes are implemented in a generic way which allow the introduction of new data without the need to change existing code. The generic receiver makes use of the common base class of the H.E.S.S. raw data formats and can save any data using the ROOT serialisation mechanism. Moreover, basic data quality checks can be performed while the data is being received, e.g. monitoring if the time that passes between two events or the size of the event is in a certain range and has a certain mean value.

### *3.3. Node switching*

To be able to deal with high data rates so called *Node Switching* is used in the H.E.S.S. DAQ, this is a round-robin load balancing scheme [27]. All the telescopes in a given run send their data to one node for a given amount of time (usually 4 s). After that, the Central Trigger announces the next node that is free to receive data from all the cameras. In case of insufficient computing power the number of nodes can be adjusted manually. Using this scheme the DAQ can receive and process data at higher rates than required by the H.E.S.S. telescope array.

A specialisation of the generic receiver mentioned before is the *CameraReader*, which is the process responsible for receiving, buffering and processing the Cherenkov camera events. The received events are unpacked, converted into the H.E.S.S. raw data format, joined with the Central Trigger information and finally written to the storage servers. For very high data rates, the storage servers may not be able to store the events fast enough. In that case the nodes can be configured to use their local disks for the duration of the observation after which the data are merged and copied to the storage servers.

### *3.4. Controllers & state machine*

All hardware used by the DAQ is represented by a *Controller*, which is a software process running on a computing node on the DAQ cluster, acting as a uniform interface between the given hardware component and the DAQ itself. Simple hardware can be represented by a single *Controller* whereas more complex hardware like the Cherenkov cameras use multiple *Controllers*, e.g. *Camera HV Controller*, *Camera Trigger Controller*, *Camera Lid Controller*, ... A state machine maps the state of the hardware to the state of the corresponding *Controllers*. The mapping of *Controller* states to hardware states for some processes is shown in Table 1. If a *Controller* is in the *Safe* state, the corresponding hardware is either turned off or in a state of minimal activity. In preparation for data taking and in between runs all *Controllers* are in the *Ready* state, i.e. the hardware is turned on and slow control information is being recorded. As an intermediate step shortly before data taking, the *Configured* state indicates that the hardware has received all the necessary configuration parameters from its *Controller*. Finally, if a *Controller* is *Running* the corresponding hardware is read out and the data are processed by the DAQ and stored on disk.

State transitions are used to change from one state to another. A scheme of the state machine used by the H.E.S.S. DAQ can be seen in Figure 3. If a *Controller* cannot perform its state transition successfully (for example due to hardware problems) it will fall back to its previous state. A detailed description of the error handling is given in Section 4.4.

There is only one transition from one state to an adjacent state for a given direction which simplifies the handling of the state machine because loops or unreachable states cannot occur. Furthermore, a flat state machine allows processes to be sent to any given target state that need not be adjacent to the current state, i.e. from *Safe* to *Running* or vice versa. (The *Controller* will perform all necessary transitions to get from the *Safe* to the *Running* state automatically, i.e. the order of executed transitions would be *GettingReady*, *Configuring* and *Starting*.) Transitions towards the *Running* state are called "upwards", and those in the other direction are called "downwards". To be able to monitor the performance of the DAQ processes, time stamps are written into a MySQL database at the beginning of each transition, once all of the dependencies (see Figure 4) of a *Controller* have finished their transition and again at the end of each transition (see Section 4.3).

### *3.5. Managers & dependencies*

To synchronise the different processes during data taking *Managers* are used throughout the DAQ. Each *Manager* is also a *Controller*, i.e. a software process running on a computing node of the DAQ cluster, providing an extended interface to deal with collections of *Controllers*. The *Managers* are responsible for distributing the run configuration to their subordinated processes as well as managing their states. The state of the Manager is determined by the minimum state of the all managed processes, e.g. if the CT5/Tracking is *Running* (i.e. CT5 is tracking the observation target) but some camera process in CT5 is still *Starting*

| | Controller state | | | |
|---|---|---|---|---|
| Process | Safe | Ready | Configured | Running |
| Node Receiver | Idle | Buffers allocated | Camera Configuration Read | Receiving & Saving Data |
| Camera | Power Off | Power On, Cooling On | Drawers configured | Reading Data |
| Camera HV | Off, 0 V | On, 800 V | On, Full Voltage | On, Full Voltage |
| Camera Lid | Closed | Open | Open | Open |
| Camera Trigger | Off | Off | Input Configured | Trigger Active |
| Tracking | Parked In | Parked Out | On Target | On Target |

Table 1: Mapping of state machine states to real hardware states for some important software processes in the H.E.S.S. DAQ.

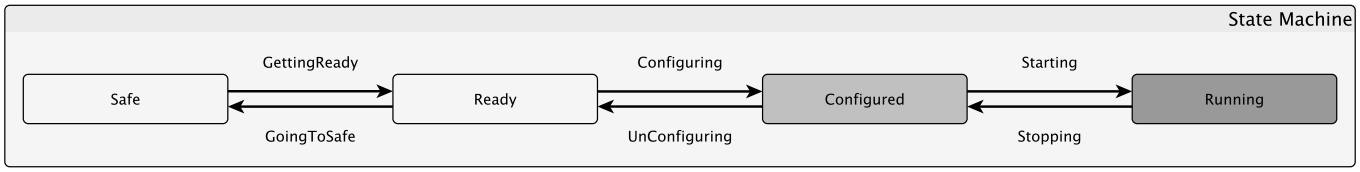


Figure 3: Visualisation of the state machine used in the H.E.S.S. DAQ. The boxes represent the states a *Controller* (and its respective device) can be in, while the arrows show the available transitions. This state machine is linear by design, which simplifies the synchronisation of multiple *Controllers*. As all the hardware *Controllers* are mapped to the same state machine, it becomes quite easy to determine the state of the whole array or any subset, e.g. when all processes of telescope CT1 are *Ready*, the whole telescope is *Ready* for observation.

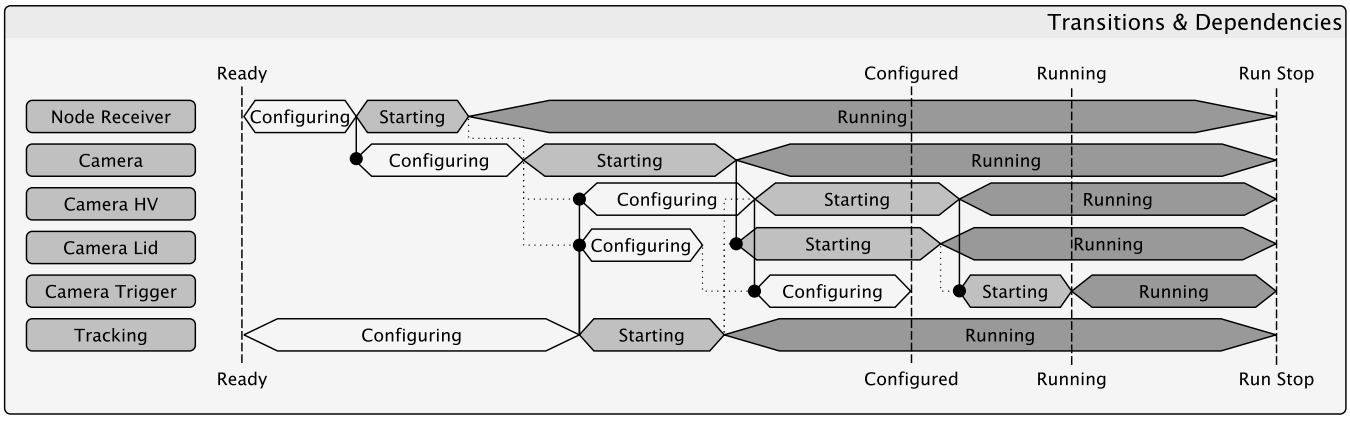


Figure 4: An example interaction of various important H.E.S.S. DAQ processes during the starting of an observation run. The change of the state of the corresponding *SubArray* is indicated by the dashed lines and ranges from *Ready*, over *Configured* to *Running*. The hexagonal boxes represent state transitions of the corresponding DAQ *Controllers*. The solid and dotted black lines with a filled circle at the end indicate dependencies between processes within the H.E.S.S. DAQ, i.e. the HV of the PMTs in the camera must not be turned on while the telescope is still moving and the camera trigger should not be activated before the camera HV is turned on. As a result a process only starts with its transition if all of its dependencies have successfully finished their own corresponding transition. The solid lines indicated the slowest dependency of a given process while the dotted lines represent dependencies that already reached the required state. The global state of the *SubArray* is determined by the slowest process, i.e. once the last process finished its transitions from *Ready* to *Configured* the whole *SubArray* is considered to be *Configured*.

(e.g. the high voltage is turned on) the CT5/Manager's state is still *Configured*. Furthermore, they are used for error handling which is discussed in detail in Section 4.4. The DAQ uses a hierarchical structure to control all the processes of a given run; with the *SubArray Manager* on top and *Managers* for every Context (see Section 3.2) in the order of the hierarchy of the contexts. The hierarchy of *Managers* and *Controllers* in the H.E.S.S. DAQ is shown in Figure 5.

The *Manager* of a given context gets the list of the processes needed for a given run from the database. There are three distinct types of processes that can be specified within the database. *Disturbing* processes are sent to a safe state at the beginning of the run, e.g. the camera lid is disturbing for a park-in run. Therefore, it is closed before the telescopes are parked in (i.e. moving the camera into the camera shelter and closing its roof). *Required* processes are necessary for a given run and a run will be stopped or will fail to start if a required process is not in the correct state. Finally, *Optional* processes are part of a

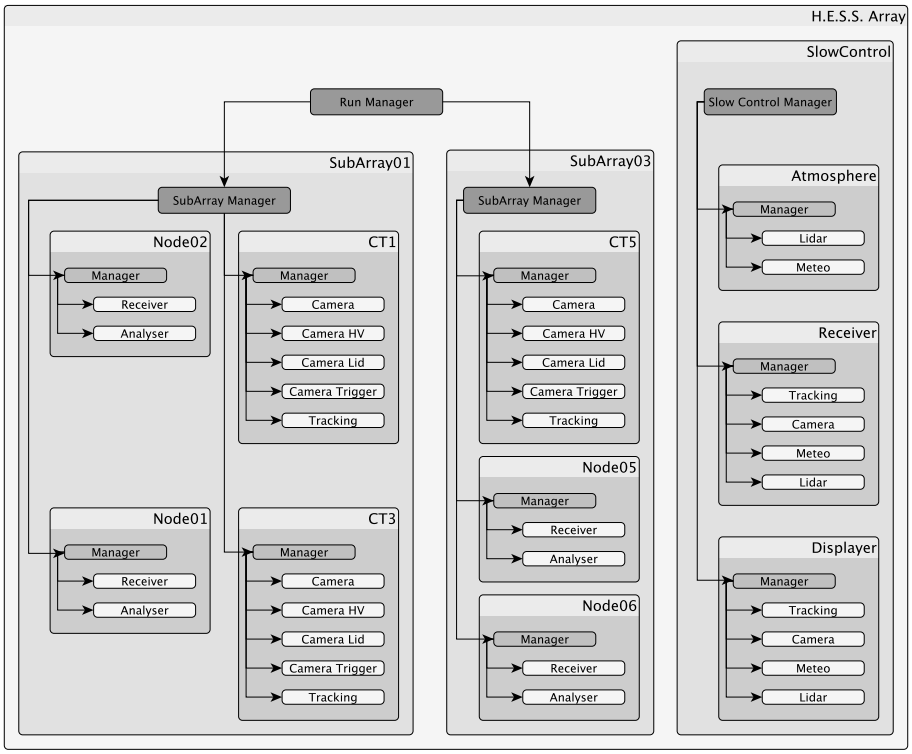


Figure 5: An example overview of the process hierarchy within the H.E.S.S. DAQ is shown. The *Run Manager* distributes the scheduled data taking runs over an arbitrary number of *SubArrays*. Each *SubArray* consists of a *SubArray Manager* process and several other sub contexts. These contexts normally are various Node and CT contexts representing the Cherenkov telescopes and the *CameraReaders*. Each *SubArray* can operate independently from the others and Nodes and CT contexts can be assigned in any combination to any *SubArray*. The so-called Slow Control context is responsible for atmospheric monitoring tasks as well as all processes that display data to the Shift Crew and is constantly in a *Running* state.

given run but an error or a misoperation of one of these processes does not influence the data taking, e.g. a crash of the process displaying monitoring data to the Shifters does not stop data taking. Once the problem with an optional process is fixed it is allowed to rejoin data taking.

In certain cases, a *Controller* has to be in a certain state before another *Controller* can begin to perform its transition to its target state. These dependencies are entered into the database and *Controllers* will get a list of all their dependencies for a given run type from their corresponding *Manager*. *Controllers* will wait for their dependent processes to finish their transitions before starting with their own, e.g. a camera trigger will wait for the camera HV to be turned on before configuring itself. Some example dependencies of processes within the DAQ can be seen in Figure 4. Dependencies are applied in reverse order for downwards transitions, this ensures that processes are shut down in the right order.

### *3.6. Configuration database*

In general, the whole configuration of the H.E.S.S. DAQ is contained in MySQL databases, including the processes and machines in use, as well as common environment variables and the configuration of the various hardware components. This allows a flexible configuration on-site and enables the setup of test and development environments in Europe where only certain pieces of hardware are available.

In addition to the on-site database there are duplicates in four different locations in Europe. These replicated databases are also used for off-line quality checks and analysis.

The H.E.S.S. DAQ uses a database abstraction layer, called *simpletable*, to access the database. It was designed and implemented as part of the H.E.S.S. DBTools [28] (in C/C++) and additionally implemented in Python for the Central DAQ. *simpletable* facilitates the grouping of multiple records in so-called *sets*. Such a set is for example a collection of calibration parameters for all pixels of a telescope. The library takes care of table locking and transactions, so that either a complete set is written/modified or nothing at all.

Another important aspect is that a permanent history of configurations, calibration parameters etc. is provided. It is possible to store and access multiple versions of a configuration set. Under normal circumstances only new sets of data are added, even when these are just minor modifications of older sets still residing in the database. This also makes it easy to create temporary configurations, e.g. deactivating a malfunctioning piece of hardware during the night and to roll back to an earlier version of the configuration when the error is resolved.

### *3.7. Logging and error propagation*

A dedicated logging framework has been implemented in the DAQ. Log files for every DAQ process are created on a daily basis and timestamps with microsecond precision are written with each message to file. These logs are easily accessible using the central DAQ GUI and are stored for several years on-site and occasionally copied to Europe.

There are six different message types available in the logging framework as shown in Figure 6. Possible actions on message reception can range from print-outs to the log

files, to pop-up messages with or without sound for the Shifters, to an automated security shutdown of the whole telescope array by the DAQ.

The log files are primarily used by the Shift Crew during error recovery to find the source of the underlying problem. Furthermore, the corresponding subsystem experts can use the log files at any time to search for problems.

If a severe error is detected by a hardware component, it has to be able to prevent any damage to itself and inform its *Manager* via its *Controller* about the error state after immediate danger is averted. Once the DAQ is informed about the malfunction a fully automated procedure takes over which, depending on the severity of the error, can bring the whole array to a safe and consistent state. This automatic procedure is designed to prevent possible damage to any other hardware equipment but especially to take care of human safety, i.e. to stop the telescope movement immediately, to shut down the Cherenkov Camera HV, etc. If an *Error* does occur the corresponding *SubArray* is brought to the *Ready* state as quickly as possible using so called "immediate transitions". They work exactly like normal transitions with the exception that all dependencies are ignored ensuring the arrival into a safe state as fast as possible. In case of a *Fatal* error message the *SubArray* is sent to the *Safe* state and all hardware belonging to the run is shut down. Once the DAQ is finished with its automatic response, the Shift Crew has to take over and identify and solve the problem to be able to continue with normal operation.

### *3.8. DAQ start & stop*

Another highly automated procedure is the start and shut down of the DAQ. Starting the DAQ can be done in less than 1 min and stopping in less than 0.5 min on average. The complete DAQ system can be started or stopped using a single button in the central GUI or running a single command on the shell in a matter of seconds. This starts the *Resource Handler*, which gets the list and configuration of all the processes that belong to the DAQ from the MySQL database and starts *Host Handlers* on every machine that has been configured to be part of the DAQ in the database. These handlers are responsible for starting, monitoring and, if necessary, restarting all the processes that run on their machine.

## 4. Array operation

The H.E.S.S. Array takes data in periods of 28 min, so called *runs*, during astronomical darkness only. There are *observation* runs which make up the bulk of the available dark time. Furthermore, dedicated *calibration* runs[3] are taken with the detector at regular intervals as well as other special purpose runs, e.g. system tests at the beginning of the night. All the different run types are specified in a database and run types can be added, removed or modified without the need to change any code. This includes any combination of hardware that has to take part in a run of a given type. This is also true for the detailed configuration parameters (*Run Parameters*) of the hardware used in a given run (which can be different for different run types).

Furthermore, the targets scheduled for observations during a given shift are contained in the same database and a dedicated tool, called the "AutoScheduler" [30] schedules all *observation* runs for a given night. The AutoScheduler takes into account various predetermined conditions, e.g. target priority, zenith angle, number of runs already taken on that target and available telescopes and uses an optimisation algorithm to prepare the schedule. This schedule is then written to the database and processed by the DAQ. The Shift Crew can adjust the schedule, e.g. by adding *calibration* runs manually, but is not allowed to change the observation schedule, unless there are exceptional circumstances, e.g. a ToO alert.

For further flexibility the DAQ can schedule runs for any combination of available telescopes. This includes multiple runs with different sets of telescopes, for example, an observation run using CT1, CT2 and CT4, a calibration run with CT3 and another observation run with CT5 on a different target. For that purpose, *SubArrays* are used to manage the participating hardware in a given run. In the example above, three *SubArrays* would have been used to take data with all five telescopes in three different runs at the same time.

The execution of the different scheduled runs is done by the *Run Manager*. It constantly monitors the available resources (e.g. telescopes and nodes) and is aware of the scheduled runs, their type and their requested hardware. It sequentially parses all scheduled runs and checks for free resources. As soon as all requirements are fulfilled (mainly there being enough free resources) the *Run Manager* allocates a *SubArray*, and sends the run configuration to the corresponding *SubArray Manager* (see Section 3.5) which will then proceed on its own, i.e. configure all the necessary hardware, start the data taking and, once the run is finished, un-configure the hardware again.

### *4.1. User interface*

All of the interaction between the Shift Crew—as well as other subsystem experts—and the DAQ is done in the central control room on-site. Several dedicated display machines are used to show monitoring information to the Shifters as well as to give fast feedback about the current status of the array. This includes information about the weather conditions outside (temperature, air pressure, wind speed and humidity), the camera monitoring (temperature, high voltage and currents), telescope pointing and motion as well as real time calibration and analysis data (e.g. shower images in the cameras and sky maps containing the source direction of all reconstructed gamma-like

[3] *Calibration* runs are needed to determine the conversion of the recorded electrical signal into a number of Cherenkov photons. A more detailed description of the H.E.S.S. calibration is given in [29].

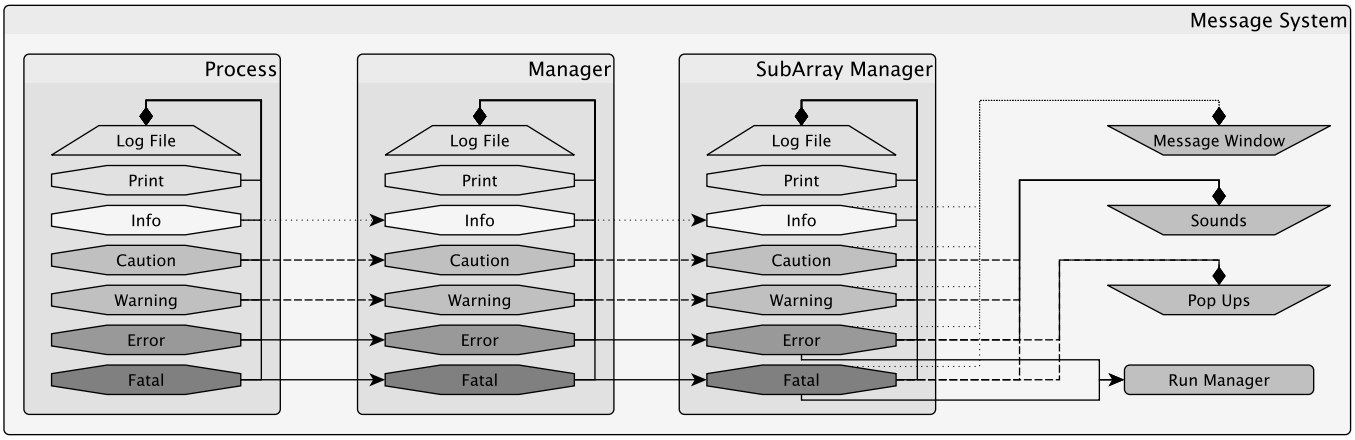


Figure 6: There are six distinct message types used in the H.E.S.S. DAQ system for inter-process communication and logging. A *Print* message is piped into the daily log file of the corresponding process and is not distributed any further. An *Info* message is send upwards in the hierarchy of the DAQ processes and also printed in the DAQ log message window which is displayed on one of the screens of the main DAQ control PC in the control room. *Caution* and *Warning* messages as well as *Error* and *Fatal* messages are also distributed upwards in the DAQ process hierarchy. Furthermore, all four of these message types also generate different sound notifications in the control room to get immediate shifter attention. *Error* and *Fatal* messages can also generate a pop-up window on the central screen of the main DAQ control PC. Moreover, the Run *Manager* is notified if an *Error* or *Fatal* message was issued and no further runs are scheduled, i.e. the *Run Manager* is doing a transition to the *Safe* state. Also the corresponding *SubArray* is send to the *Ready* state in case of an *Error* message and the *Safe* state in case of a *Fatal* message as a safety precaution.

events). A central graphical user interface (GUI) is used for interaction with the DAQ; a screenshot of the main part of the GUI is shown in Figure 7. It serves as a single point of contact between Shifters and the DAQ where all essential settings and configuration options can be changed. This allows Shifters to take over manual control in case of error recovery and special operations in an easy way. The central GUI, and most other GUIs in the H.E.S.S. DAQ, are implemented in the Python programming language using PyGTK [31]. Furthermore, all GUI processes implement the state *Controller* interface and are managed like all other processes in the DAQ, e.g. they receive *Run Parameters* and perform transitions like *Starting* and *Stopping*.

Building upon the common raw data format based on ROOT, a generic data *Displayer* has been developed (using C++). It uses the object introspection capabilities of the ROOT data analysis framework to gain access to any data member of any H.E.S.S. data storage format. Therefore it is able to plot any data that are recorded in the H.E.S.S. DAQ and can be configured solely using the MySQL database. Different specialisations of this *Displayer* can plot time lines, bar charts, wind roses, camera images, etc. An example of available displays that are shown in the control room can be seen in Figure 8.

Some basic data quality checks can be performed with the *Displayer* as well, e.g. range checks with warning sounds and pop-up messages. Normally, displays are updated at a rate of a few Hz providing fast feedback to the Shifters.

For complex hardware components, e.g. the Cherenkov cameras, expert GUI modes are also available. Those allow detailed control of the various hardware components and can be used by subsystem experts as well as experienced Shifters on-site for development, error handling and debugging.

### *4.2. Real-time pipeline*

A dedicated collection of Displayers is also used to show the plots of the "real-time pipeline" to the Shifters. It is a full analysis, based on the HAP TMVA analysis [32], of the data being taken running in real time. The only limitation is that a default camera calibration has to be used, e.g. the High Gain to Low Gain ratio, flatfield coefficients, single photo electron values as well as muon coefficients with the exception of the calculation of the pedestal position of the two gain channels of the photomultiplier [29, 33, 34]. For background determination either the Ring Background (the default; using several speed-ups concerning coordinate transformations), the Reflected Region or the Template Background method can be used [35]. Run results are stored on disk and used to perform a real-time analysis with data from consecutive runs on the same target. The output of the real-time pipeline are calibrated camera images with intensities in photo electrons as well as significance maps of the region of the sky that is currently being observed and $\theta^2$ plots around the target source position. If a significant detection of a source is made the Shift Crew is alerted using pop-up messages and advised to call for expert input to allow swift follow-up observations. To cope with the high-data rates the real-time pipeline is split into several different processes. Each *CameraReader* has a corresponding *Analyser* process which subscribes to the data stream that is processed by the *CameraReader*. The *Analyser* processes all events that are generated by the *CameraReader*. This includes event calibration and event

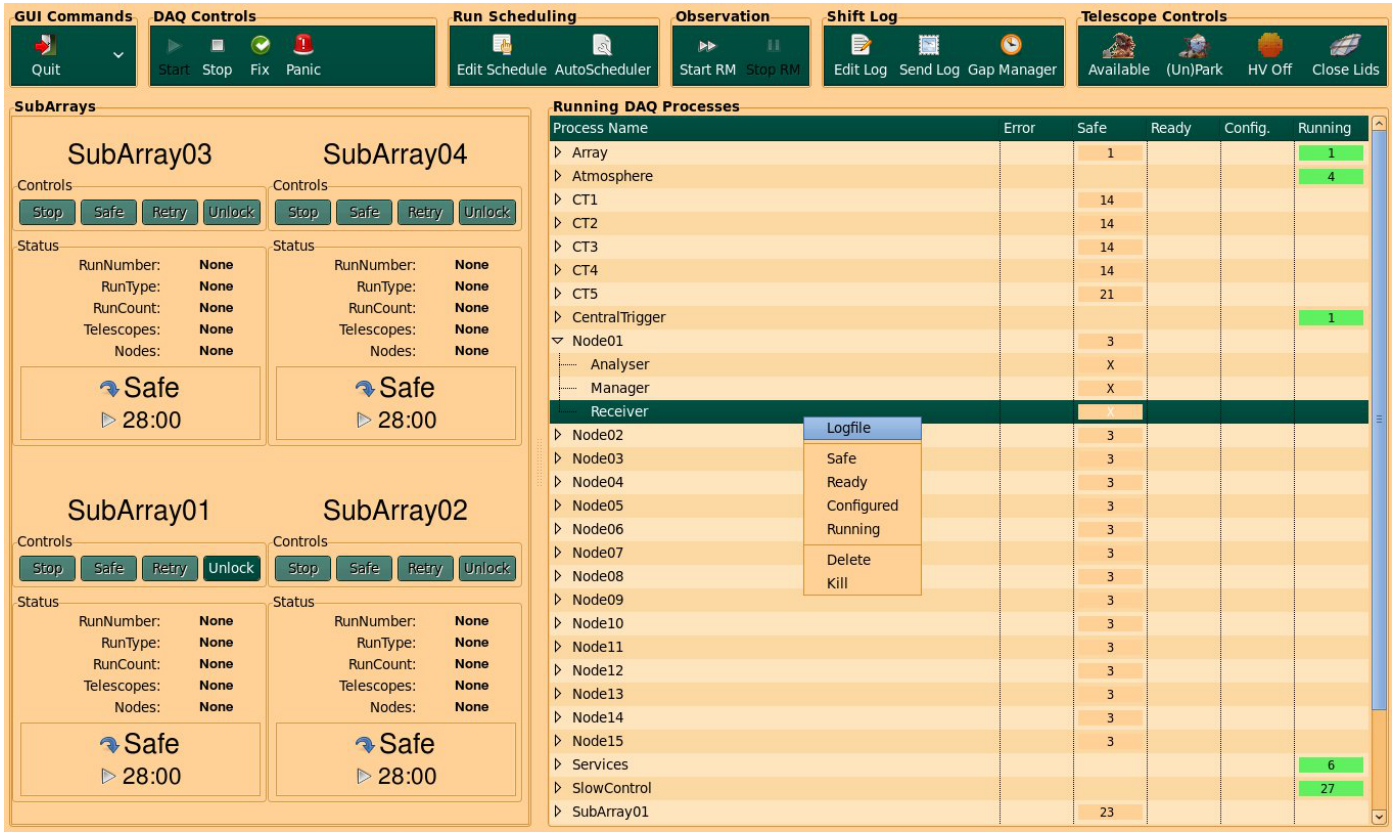


Figure 7: Main part of the central DAQ GUI used by the Shifters to interact with the H.E.S.S. II DAQ. It is separated in three distinct parts. The upper row consists of several groups of buttons that are needed during normal operation. The lower left part of the GUI is dedicated to giving detailed information about and control over the available *SubArrays*. This includes information about run duration, observation target, used telescopes, etc. The main part of the GUI is used to show all running DAQ *Controllers* to the Shifters. They are grouped in contexts and can be expanded to show all processes within this context. The amount of processes in a context as well as their current state are also shown. Detailed control over every single process (even multiple ones if more than one are selected) is possible using the right-click menu. In case of errors, the *Controllers* at fault are marked with an error flag (not shown).

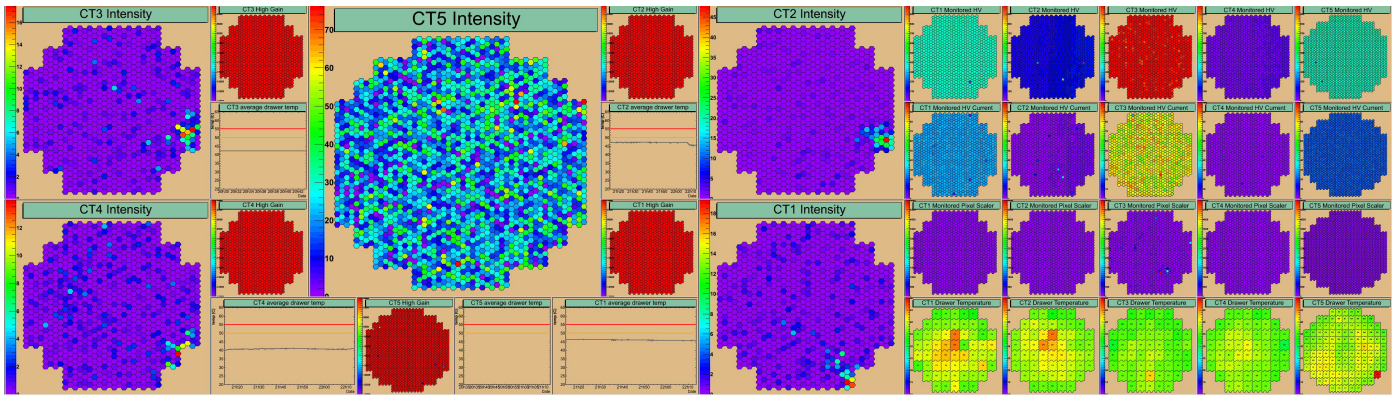


Figure 8: Example selection of slow-control displays for immediate feedback to the Shift Crew. The five largest camera displays show pixel intensities in photo-electron volts for all telescopes. Right next to the intensity plots the High Gain ADC count camera displays as well as the average camera drawer temperature over time histograms can be seen. On the right side of the screenshot the different rows of camera displays from the top to the bottom correspond to current pixel HVs, pixel currents, pixel scaler values and drawer temperatures (the pixel scaler value being the number of pixel triggers in a certain time period for a given pixel).

reconstruction as well as gamma-hadron separation. The processed data is collected by *AnalysisServer* processes; one for each *SubArray* in use. The input maps for the significance maps are filled at the *Analyser* and the final significance maps are created at the *AnalysisServer*. The latter is a time consuming processes (several minutes) and therefore happens in parallel to the input maps being received.

### 4.3. Monitoring and shift logs

To calculate the data taking efficiency of the H.E.S.S. telescope array the so called Transition Time Tools are used. It is a Python framework that analyses the gaps between the various runs and the timestamps, that were written to a database by the different DAQ *Controllers*. The DAQ monitors whether there are any gaps in data taking during dark time. If a gap is detected the Shifters are asked to give a reason for each gap between data taking runs. The answers the Shifters provide are stored in the same database. In combination, the information about the transitions, the gaps between runs and their reasons can be used to calculate the data taking efficiency of the H.E.S.S. Array as well as the percentage of lost observation time of the array due to DAQ problems. Furthermore, it is possible to benchmark different processes with microsecond resolution and identify bottlenecks in the data-taking procedure. A further source of information is the detailed shift log that is sent to a mailing list and that is stored on disk after every night of a shift to keep the collaboration up to date about the ongoing activities on-site.

### 4.4. Error handling

In case of a hardware failure, the DAQ automatically performs a safety shutdown, as described in section 3.7. Apart from this automatic procedure, manual overrides are available for every automatic configuration and automatic DAQ action. If a hardware device is not available or malfunctioning, its corresponding *Controller* can be replaced with a *noopController* which just implements the common interface of the state machine while not doing anything else. This procedure is wrapped in a Python script with the name of the *Controller* that should be replaced as a single command line parameter allowing the Shift Crew to easily continue observations without the failed hardware.

Along with the manual overrides, extensive documentation of all hardware components and the DAQ are available on-site. While the hardware manuals are mostly present in analogue form, the DAQ manual and the DAQ troubleshooting guide are located within a MediaWiki [36] on-site. Shifters are encouraged to contribute to the Wiki while they are on Shift with an emphasis on the Shifter's Notes, a summary of the current shift for the next Shift Crew.

At the beginning of each Shift the current Shift Crew is introduced to the handling of the array and to the emergency procedures by a local Shift Expert who spends the first ten days with the Shifters. After that time the Shifters are on their own. Should they be unable to solve a problem, subsystem experts on call are available to help resolve the problem.

### 4.5. Remote control

The H.E.S.S. site is located in a remote region with very limited internet connectivity. To make the remote maintenance easier and to minimise the number of maintenance trips from Europe, several remote maintenance tools are used. For instance, IPMI-cards [37] are installed on every machine in the DAQ cluster. This allows the machines to be power-cycled remotely as well as giving access to the BIOS and other configuration menus during the boot-up of the machines. Furthermore, VNC [38] servers are used to forward the graphical displays once the operating system has started. They can be started on every machine of the DAQ cluster and are running constantly on the machines in the control room. The VNC connections have proven to be an invaluable tool when it comes to remote assistance in case of problems during data taking as well as for remote maintenance of the cluster and related machines. The access to these VNC servers is restricted to DAQ experts to prevent tampering with the system by third parties.

The remote access to the network is possible with an OpenVPN [39] server running on the gateway machine of the DAQ cluster. With this it is possible to access all of the different networks on-site without compromising security or the separation between the data network and users network. To ensure a stable production environment, the Shift Crew as well as other members of the collaboration cannot change the software used for data taking. Only DAQ experts can make software changes and can, therefore, guarantee a properly working DAQ system. The Shifters are given restricted access to the cluster to minimise the possibility of human error.

### 4.6. Reaction to ToO alerts

The DAQ system can receive target of opportunity alerts from other experiments via the Gamma Ray Burst (GRB) Coordinates Network (GCN). A dedicated process, the *GCNAlerter*, has been developed by the H.E.S.S. Collaboration. It listens for messages from the GCN network, checks whether the coordinates are visible and takes further action. For H.E.S.S. Phase I the *GCNAlerter* informed the Shift Crew and prepared a script to alter the observation schedule, which had to be confirmed and executed by the human operators. The results of the Gamma Ray Burst observations with H.E.S.S. I are described in [40]. For the start of H.E.S.S. Phase II a revised target of opportunity alert scheme has been developed by the H.E.S.S. Collaboration [41] and is currently being implemented into the DAQ. If the *GCNAlerter* decides that an alarm justifies a prompt observation, the DAQ will react fully automatically and start data taking on the new target without the need for human intervention. For safety reasons the Shift Crew is not allowed to enter the array if the ToO alert system is active.

For these prompt observations the time span between receiving the alert and the beginning of the observation has to be minimised; transient events occur on time scales of a few seconds and several minutes. The bulk of the transition time between two runs is due to the slewing time of the telescopes to the new target. In normal operation, some additional time is used to switch off the high voltage while the telescopes are moving to prevent damage to the photo

multipliers of the Cherenkov cameras due to bright stars in the field of view.

In case of a GCN alert, the telescopes immediately start moving to the new target while the ongoing runs are stopped and the reconfiguration of all *Controllers* is done. Moreover, the high voltage of the camera is not turned off and the cameras are just sent to an internal paused state where they stop taking data but remain fully configured. The effect of these actions is that the transition time is just the slewing time of the telescopes which cannot be sped up any more (see [42]) and a short overhead for the unpausing of the camera and the reactivation of the camera pixels.

Moreover, dependencies which are optional processes are not waited upon so that they do not increase the duration of the transition. On top of that, only CT5, due to its higher movement speed, is required for the start of the data taking, CT1 to CT4 are configured to be optional processes and will join data taking once they are on target. The Cherenkov Camera Trigger and Central Trigger configuration are the same as during normal observations (CT5 monoscopic trigger are allowed all the time). The drive system of CT5 also has the option to use reverse tracking of ToO targets, i.e. driving beyond zenith, and the fine positioning of the telescope is done during data taking because the errors on the target position of a ToO alert are large.

## 5. Software management

### *5.1. Test-DAQ cluster & DAQ simulation*

To test the DAQ software without real hardware, a full DAQ simulation can be set up, i.e. raw camera data are sent from a camera emulation process to the node receivers and a Central Trigger emulation process sends trigger blocks to the corresponding receivers as well. This can be used to test receivers, the node switching as well as the real time analysis with real data as input. However, to process a full run of real data in an acceptable time frame, a computer cluster similar to the one in Namibia has to be used. The H.E.S.S. Test-DAQ-Cluster, a scaled down version of the DAQ cluster on-site (five nodes instead of ten, two storage servers instead of five, one switch instead of four), provides enough computing power for that purpose. Moreover, the same operating system and software is running on the Test-DAQ, making it an ideal test bench for software development. Its location in Europe also allows easy access in contrast to the cluster on-site.

### *5.2. Development tools*

During the development of the H.E.S.S. II DAQ software the Make-based build system [43] of the software was replaced by one using SCons [44]. Apart from a code cleaning during this transition, it is now possible to build the software using multiple jobs in parallel. The legacy build system was not able to build the software correctly using multiple jobs. It was simpler to re-implement the build system, instead of modifying the Make-based one[4]. Another benefit of SCons is the use of Python for its configuration script files which allows a quick start for beginners and facilitates maintenance of the build system. To further aid in development a Bugzilla Bug tracker [45] is used by the software developers of the H.E.S.S. Collaboration.

For development purposes and benchmarking so-called *Dummy Controllers* are available in C++ and Python. They provide the basic interface of the state machine and are used to mock missing hardware *Controllers* in a testing environment. The Dummy Controllers are also used to test the dependency and time-out handling within the DAQ as well as to test the automatic shutdown and mechanisms in case of errors. They mock long-running transitions or can throw exceptions during specified transitions, depending on the options that are passed as command line arguments.

To aid other subsystem experts in the development of their own DAQ *Controllers* for their hardware a dedicated virtual machine, the DAQ-VM, is available. Most of the time the subsystem hardware cannot be shipped to the Test-DAQ-Cluster and a software test environment has to be set up at the location of the hardware (for instance properly configured operating system, database and CORBA omniName server). The DAQ-VM was created using VMware Fusion 3 [46], the hardware requirements are a single-core 2 GHz processor, 1 GB memory and 20 GB disk space. Taking the hardware requirements into account, the DAQ-VM can be run on almost all currently available laptops. This allows non-DAQ-Experts to test new DAQ *Controllers* with their corresponding hardware under conditions which are as close as possible to those on-site without detailed knowledge about how to set up such a test environment.

Another helpful tool for non-DAQ-Experts is the detailed documentation available in an internal Wiki of the H.E.S.S. Collaboration. Together with basic example *Controllers* and the extensive how-to guides in the Wiki non-DAQ-Experts can quickly start to write and test new DAQ *Controllers*.

## 6. Performance

The H.E.S.S. DAQ system has been in use since the commissioning of the first H.E.S.S. telescope in 2003. Since then the DAQ system has evolved continuously to its current state as described in the previous sections. Over the 10 year period of operation corruption of data due to central DAQ malfunctions was extremely rare and is negligible. In preparation for H.E.S.S. Phase II the central DAQ software has been overhauled, including being ported from 32 bit to 64 bit, to make full use of the architecture of recent server

[4]The SCons built-in dependency management is more sophisticated and can be extended quite easily compared to Make's built-in dependency rules.

machines. The performance of the central DAQ software will be presented in the following, focusing on the system that was prepared for H.E.S.S. Phase II.

### *6.1. Stability*

The H.E.S.S. DAQ has been in operation for almost ten years and performed well throughout this time. In spite of frequently failing hard disks and the harsh environment for sensitive electronic equipment, the amount of data lost over time is negligible. The redundancy of the available hardware and software has played a critical role in this achievement, i.e. broken hard disks are replaced immediately by the RAID setup and other broken components can be replaced with spare parts from a common pool. At the same time it is possible to redistribute processes to other machines, because they have identical hard- and software environments. This is especially true for all services needed by the central DAQ and all DAQ *Controllers*, which can be started on any machine of the cluster (storage-server or farm node). This multi-level redundancy design both minimises the probability of losing data as well as the recovery time after a computer hardware failure.

The central DAQ overhead contributing to the transition time between two consecutive observation runs is negligible, i.e. the time needed by the hardware to be ready for the next run is several orders of magnitude bigger than the central DAQ overhead of the corresponding controllers. Moreover, the central DAQ does not increase the system deadtime. In the time period from 01.01.2009 to 31.12.2012 the H.E.S.S. Array was not operational due to central DAQ problems for 0.8 % of the available dark time. This demonstrates that problems with the central DAQ hardware and software during data taking can be quickly solved.

During data taking the H.E.S.S. II Array produces a total data rate of 46 $^{\mathrm{MB}}/_{\mathrm{s}}$ on average. This includes 1 % of slow control data and of log files which are stored every night for debugging purposes. The average CPU load of the cluster during data taking does not exceed 10 % (of five 2.5 GHz quad cores and ten 3 GHz octo cores), while the average memory usage is below 30 % (of a total of 200 GB). In a month worth of data-taking on average 11 TB of data are written to disk, which allows several months of observation data to be stored on disk in Namibia. Overall the performance of the central DAQ cluster is more than sufficient to handle the data rates of the H.E.S.S. II Array, leaving enough room for computationally intensive tasks, like e.g. preliminary analysis and elaborate data quality checks.

In addition to the slow-control data, real-time data quality checks and real-time analysis results provide fast feedback about the current status of the array and about the scientific quality of the running observation. Shifters have easy access to these data using the different screens in the control room, allowing immediate intervention in case of problems.

The operation of the H.E.S.S. Array by non-expert personnel is possible due to the detailed manuals and documentation about every hardware subsystem and the high level of automation within the DAQ. This automation and the open-source remote administration and monitoring tools greatly reduce the maintenance cost of the H.E.S.S. central DAQ software, which currently requires approximately a 0.5 FTE position in Europe.

### *6.2. Flexibility*

New hardware components can be added easily to the H.E.S.S. Array. Only the software for the *Controller* responsible for the new hardware has to be written and added to the DAQ. The rest of the system, i.e. data transport, storage and visualisation does not need to be modified. Furthermore, the inter-dependence of the new *Controller* with already existing ones have to be added to the database.

The central DAQ software used for Phase II of the H.E.S.S. telescope array evolved from the central DAQ software designed for Phase I of the experiment. One of the main differences between the new central DAQ software and the earlier implementation is compatibility with 64 bit architecture of recent CPUs, which required many minor patches. However, the principal design ideas of the H.E.S.S. central DAQ software remained the same. This includes the configuration of the DAQ itself and the controlled hardware from a central database, as well as the common data format for all monitoring and scientific data. For the latter only additional information had to be included, i.e. the-pixel wise timing information that became available with the newer electronics [11] of the CT5 camera. The fact that no redesign of the central DAQ software was necessary is due to the uniform interfaces for hard- and software: Ethernet standard for communication to all hardware components and the common interface of the software *Controllers* for all devices.

Another example of the capability of the H.E.S.S. DAQ to quickly adapt to new situations, is the fact that during the whole commissioning of the fifth telescope the live system was used to test and develop *Controllers* relevant for the new telescope while the array was taking data using the other four. This includes parallel and joined observations of the old and new telescope. Due to this ability to take data with different *SubArrays* the CT5 commissioning could be done mostly in parallel to data taking with CT1-4 which had only minor downtime.

The commissioning for H.E.S.S. Phase II of the central DAQ went smoothly. This was possible due to the extensive documentation and guides about DAQ *Controller* development, and the DAQ Virtual Machine (VM) which allowed non-DAQ-Experts to easily develop and test the *Controllers* for their own custom hardware. For example, the development of the Tracking Controller for CT5, the *Controller* for the calibration device of the new CT5 camera as well as the adjustment of the Central Trigger Controller has benefited. Moreover, the DAQ VM was used to test the central DAQ software with the camera test-benches (this includes old and new camera hardware,

i.e. test-benches mimicking CT1-4 as well as CT5) during the initial development phase of the DAQ for H.E.S.S. Phase II.

The main platform for the development of the H.E.S.S. Phase II central DAQ was the TestDAQ, including a full simulation of the data-taking process. In a simulated observation run, real raw data from previous runs taken with the H.E.S.S. DAQ are fed into the system, which allows the central DAQ to be tested under near real conditions, the main difference being that there is no actual hardware connected to the TestDAQ cluster and *Dummy Controllers* take the place of real ones. These *Dummy Controllers* can also be used to fake slow hardware and to test the error handling of the DAQ. The simulation runs can be used to test the real-time analysis, to test data quality checks and to do benchmarks with the hardware of the cluster as well as the DAQ software running on it to identify bottlenecks. Moreover, using the TestDAQ to test firmware and driver updates, as well as other hardware or software modification, before applying them to the live system helps to avoid complications on-site and contributes to the stability of the DAQ.

## 7. Conclusion

The H.E.S.S. central DAQ has been in operation for almost 10 years without any major problems since the inauguration of the first telescope in 2003. The central DAQ did only contribute to a loss of 0.8 % of the available dark time since 2009 proving its ability to quickly deal with hardware or software problems during data taking. Moreover, the amount of data that have been lost is negligible.

At this point we would like to share the most important lessons learned which could be interesting for other experiments like e.g. the upcoming CTA Project.

The ROOT-based data format proved to be useful as a common format for both on-line and off-line analysis software. However, the tight dependency on an external framework lead to various issues. The software had to be adapted to the three major releases of the ROOT-framework (v3, v4, v5) during the last years. This caused additional work in terms of software development (ensuring back-wards compatibility with existing raw data files and classes) and also in terms of system administration (maintaining different ROOT versions on different operating systems).

The distributed development of the central DAQ software and hardware *Controllers* in the H.E.S.S. Collaboration proved to be difficult due to the many different environments at the different member institutes and laboratories. To avoid some of the problems that usually arise during updates on site (i.e. introducing new functionality for *Controllers*, installing updates for the operating system or cluster hardware drivers) it proved very usefully to integrate the software and test the installation on the TestDAQ and the DAQ Virtual Machine, which helped to resolve many problems beforehand.

While technically the central DAQ software is organized in separate modules, there are a lot of historic interdependencies between the different modules and classes. As a result, it is very difficult to effectively test the different components of the DAQ software without integrating the full software distribution (both DAQ software, and off-line analysis packages). The DAQ Virtual Machine helped to test *Controllers* in a "standardized" environment. It provides a pre-installed reduced collection of H.E.S.S. DAQ software modules and their dependencies for easy development and testing. Thus it alleviated some of the mentioned problems. For future projects each software module should provide automated tests that can be run without the need to integrate other modules. This helps to ensure a modules functionality and also reduces the coupling between modules, making maintenance a lot easier.

The abstraction of internal hardware states to the simplified linear state machine proved to be very useful in terms of controlling the whole telescope array or various subsets. This is also true for the design decision to delegate the safety of each piece of hardware to its firmware, and only having one *Error* flag in the *Controller*, which tells the central DAQ whether a piece of hardware is working or not. With these simplifications it is easier to keep the whole DAQ in a consistent state and to use automatic procedures like the controlled emergency stopping of a run in case of hardware failures.

The virtualization of the legacy boot-servers (described in subsection 2.2) for the Cherenkov cameras and Central Trigger devices has proven to be very useful. It allowed to replace the outdated special purpose machines with new off-the-shelf hardware while keeping the specialized operating system with all its customizations without any additional reconfiguration or development. For future projects it is a good idea to decouple the configuration of the DAQ system from the physical set-up of the cluster computing hardware. This could be achieved by creating virtual machines for all critical tasks (e.g. database servers, boot servers, computing nodes, etc.) and distribute them dynamically on the available physical machines. This would also reduce the time to reconfigure the system after a physical server or computing node failure during data-taking.

Another lesson learned concerns the graphical user interfaces in the control room. The control and monitoring interfaces are very valuable to the Shifters and enable them to take charge of observations after a few nights of training. On the technical side it would have been better to decouple the control functionality from the actual display code. Quite some issues had to be resolved involving multi-threaded interprocess communication mixed with multi-threaded graphics display code. Our suggestion for future projects is to provide user interfaces using standard web interfaces. The main advantage is that the system becomes almost independent of the actual display machines. The web interfaces can be displayed on any current or future operating system or device as long it provides a web-browser. Moreover, it would simplify remote operation

and monitoring by providing remote access to the web interfaces allowing the same interactions with the system that would be possible on-site.

Due to the high flexibility and scalability of the design of the central DAQ, the inevitable changes to the central DAQ necessary due to upgrades of the array could all be resolved in an evolutionary fashion in contrast to a costly redesign. This is especially true in the light of the commissioning of the fifth telescope of H.E.S.S. Phase II.

## Acknowledgements

The authors would like to acknowledge the support of their host institutions. We want to thank the whole H.E.S.S. Collaboration for their support.

The support of the Namibian authorities and of the University of Namibia in facilitating the construction and operation of H.E.S.S. is gratefully acknowledged, as is the support by the German Ministry for Education and Research (BMBF), the Max Planck Society, the German Research Foundation (DFG), the French Ministry for Research, the CNRS-IN2P3 and the Astroparticle Interdisciplinary Programme of the CNRS, the U.K. Science and Technology Facilities Council (STFC), the IPNP of the Charles University, the Czech Science Foundation, the Polish Ministry of Science and Higher Education, the South African Department of Science and Technology and National Research Foundation, and by the University of Namibia. We appreciate the excellent work of the technical support staff in Berlin, Durham, Hamburg, Heidelberg, Palaiseau, Paris, Saclay, and in Namibia in the construction and operation of the equipment.

## References

[1] F. Aharonian, et al., Observations of the Crab nebula with HESS, Astronomy and Astrophysics 457 (2006) 899–915.

[2] J. Hinton, W. Hofmann, Teraelectronvolt astronomy, Annual Review of Astronomy and Astrophysics 47 (2009) 523–565.

[3] P. Vincent for the H.E.S.S. Collaboration, H.E.S.S. Phase II, in: Proc. 29 Int. Cosmic Ray Conference, Vol. 5, 2005, pp. 163–166.

[4] M. Punch for the H.E.S.S. Collaboration, H.E.S.S. II: Expansion of H.E.S.S. for higher sensitivity and lower energy, in: Towards a Network of Atmospheric Cherenkov Detector VII, 2005, pp. 379–391.

[5] Y. Becherini, M. Punch, H. E. S. S. Collaboration, Performance of HESS-II in multi-telescope mode with a multi-variate analysis, in: F. A. Aharonian, W. Hofmann, F. M. Rieger (Eds.), American Institute of Physics Conference Series, volume 1505 of *American Institute of Physics Conference Series*, 2012, pp. 741–744. doi:10.1063/1.4772366.

[6] M. Holler, A. Balzer, Y. Becherini, S. Klepser, T. Murach, M. de Naurois, R. Parsons, for the H. E. S. S. Collaboration, Status of the Monoscopic Analysis Chains for H.E.S.S. II, ArXiv e-prints (2013).

[7] C. Borgmeier for the H.E.S.S. Collaboration, The central data acquisition system of the H.E.S.S. telescope system, in: Proc. 28th Int. Cosmic Ray Conference, 2003, p. 2891.

[8] E. Hays, VERITAS Data Acquisition, in: International Cosmic Ray Conference, volume 3 of *International Cosmic Ray Conference*, 2008, pp. 1543–1546.

[9] D. Tescaro, J. Aleksic, M. Barcelo, M. Bitossi, J. Cortina, M. Fras, D. Hadasch, J. M. Illa, M. Martinez, D. Mazin, R. Paoletti, R. Pegna, for the MAGIC Collaboration, The readout system of the MAGIC-II Cherenkov Telescope, ArXiv e-prints (2009).

[10] M. Actis, G. Agnetta, F. Aharonian, A. Akhperjanian, J. Aleksić, E. Aliu, D. Allan, I. Allekotte, F. Antico, L. A. Antonelli, et al., Design concepts for the Cherenkov Telescope Array CTA: an advanced facility for ground-based high-energy gamma-ray astronomy, Experimental Astronomy 32 (2011) 193–316.

[11] J. Bolmont, et al., The camera of the fifth H.E.S.S. telescope. Part I: System description., 2013.

[12] S. Funk, G. Hermann, J. Hinton, D. Berge, K. Bernlöhr, W. Hofmann, P. Nayman, F. Toussenel, P. Vincent, The trigger system of the H.E.S.S. telescope array, Astroparticle Physics 22 (2004) 285–296.

[13] Wikipedia, Standard RAID levels — Wikipedia, The Free Encyclopedia, 2013. URL: http://en.wikipedia.org/w/index.php?title=Standard_RAID_levels&oldid=538005252, [Online; accessed 13-February-2013].

[14] R. Y. Wang, T. E. Anderson., xFS: A Wide Area Mass Storage File System., 1993.

[15] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, B. Lyon, Design and Implementation or the Sun Network Filesystem, 1985.

[16] GlusterFS, Clustered File Storage that can scale to petabytes, 2013. URL: http://www.gluster.org/, [Online; accessed 26-June-2013].

[17] Wikipedia, Linear Tape-Open — Wikipedia, The Free Encyclopedia, 2013. URL: http://en.wikipedia.org/w/index.php?title=Linear_Tape-Open&oldid=534100440, [Online; accessed 24-January-2013].

[18] Wikipedia, Uninterruptible power supply — Wikipedia, The Free Encyclopedia, 2012. URL: http://en.wikipedia.org/w/index.php?title=Uninterruptible_power_supply&oldid=528806006, [Online; accessed 16-January-2013].

[19] M. L. Massie, B. N. Chun, D. E. Culler, The Ganglia Distributed Monitoring System: Design, Implementation And Experience, Parallel Computing 30 (2003) 1.

[20] Collectd, collectd – The system statistics collection daemon, 2013. URL: http://collectd.org/, [Online; accessed 26-June-2013].

[21] B. E. Finley, VA systemimager, in: Proceedings of the 4th annual Linux Showcase & Conference - Volume 4, ALS'00, USENIX Association, Berkeley, CA, USA, 2000, pp. 11–11.

[22] R. Brun, F. Rademakers, ROOT - An Object Oriented Data Analysis Framework, in: AIHENP'96 Workshop, Lausane, volume 389, 1996, pp. 81–86.

[23] OMG, The Common Object Request Broker: Architecture and Specification, Technical Report 2.0, Object Management Group, 1995.

[24] The ROOT Team, ROOT version 3.02/06 Release Notes, 2001. URL: http://www.slac.stanford.edu/comp/unix/package/cernroot/30207/examples/Version302.news.html.

[25] C. Borgmeier, K. Mauritz, C. Stegmann, M. de Naurois, DAQ and Analysis at H.E.S.S., 2001. URL: http://www-root.fnal.gov/root2001/presentations/session4/rootAtHESS.pdf.

[26] S. lai Lo, S. Pope, The Implementation of a High Performance ORB over Multiple Network Transports, in: In Middleware 98: IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, 1998, pp. 157–172.

[27] Wikipedia, Round-robin scheduling — Wikipedia, The Free Encyclopedia, 2013. URL: http://en.wikipedia.org/w/index.php?title=Round-robin_scheduling&oldid=557149038, [Online; accessed 4-June-2013].

[28] K. Bernlöhr, DBTools - Database tools for H.E.S.S. v1.3.3, H.E.S.S., 2009. Internal manual.

[29] F. Aharonian, et al., Calibration of cameras of the H.E.S.S. detector, Astroparticle Physics 22 (2004) 109–125.

[30] K. Bernlöhr, The AutoScheduler - An Automated Target Scheduling Tool For H.E.S.S. v1.05, H.E.S.S., 2006. Internal manual.

[31] PyGTK, Python wrappers for the GTK+ graphical user interface library, 2013. URL: http://www.pygtk.org/, [Online; accessed 26-June-2013].

[32] S. Ohm, C. van Eldik, K. Egberts, $\gamma$/hadron separation in

very-high-energy $\gamma$-ray astronomy using a multivariate analysis method, Astroparticle Physics 31 (2009) 383–391.

[33] Sebastian, Funk, Online Analysis of Gamma-ray Sources with H.E.S.S., Diploma thesis, Humboldt University Berlin, 2005.

[34] A. Balzer, Systematic studies of the H.E.S.S. camera calibration, Diploma thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2010.

[35] D. Berge, S. Funk, J. Hinton, Background modelling in very-high-energy $\gamma$-ray astronomy, Astronomy and Astrophysics 466 (2007) 1219–1229.

[36] MediaWiki, MediaWiki — MediaWiki, The Free Wiki Engine, 2012. URL: `http://www.mediawiki.org/w/index.php?title=MediaWiki&oldid=545966`, [Online; accessed 24-January-2013].

[37] IPMI, The Intelligent Platform Management Interface (IPMI) is a standardised interface used to monitor the operation of computer systems, 2013. URL: `http://www.intel.com/design/servers/ipmi/`, [Online; accessed 26-June-2013].

[38] Wikipedia, Virtual Network Computing — Wikipedia, The Free Encyclopedia, 2013. URL: `http://en.wikipedia.org/w/index.php?title=Virtual_Network_Computing&oldid=532570491`, [Online; accessed 24-January-2013].

[39] OpenVPN, OpenVPN is an open source solution to establish virtual private networks (VPN), 2013. URL: `http://openvpn.net/index.php/open-source.html`, [Online; accessed 26-June-2013].

[40] F. Aharonian, et al., HESS observations of $\gamma$-ray bursts in 2003-2007, Astronomy and Astrophysics 495 (2009) 505–512.

[41] D. Lennarz, P. Chadwick, W. Domainko, R. D. Parsons, G. Rowell, P. H.T. Tam, for the H.E.S.S. collaboration, Searching for TeV emission from GRBs: the status of the H.E.S.S. GRB programme, in: Proceedings of the 7th Huntsville Gamma Ray Burst Symposium, Nashville, 2013.

[42] P. Hofverberg, R. Kankanyan, M. Panter, G. Hermann, W. Hofmann, C. Deil, F. A. Benkhali, H.E.S.S. Collaboration, Commissioning and initial performance of the H.E.S.S. II drive system, in: Proceedings of the 33rd International Cosmic Ray Conference (ICRC2013), Rio de Janeiro (Brazil), 2013. `arXiv:1307.4550`.

[43] Make, Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files, 2013. URL: `https://www.gnu.org/software/make/`, [Online; accessed 26-June-2013].

[44] SCons, SCons is an Open Source software construction tool—that is, a next-generation build tool, 2013. URL: `http://www.scons.org`, [Online; accessed 26-June-2013].

[45] Bugzilla, Bugzilla is server software designed to help you manage software development, 2013. URL: `http://www.bugzilla.org`, [Online; accessed 26-June-2013].

[46] V. Fusion, VMware Fusion is a software hypervisor developed by VMware for Macintosh computers with Intel processors, 2013. URL: `https://my.vmware.com/web/vmware/info/slug/desktop_end_user_computing/vmware_fusion/3_0`, [Online; accessed 26-June-2013].