Experimental Controls System

James Nikkel[1]
Royal Holloway University of London

August 2, 2012

[1]james.nikkel@rhul.ac.uk

Contents

Chapter 1

Introduction

This document describes the design, installation, and use of the experimental slow control system. This systems is designed to monitor and control experimental telemetry for small to medium sized experiments with up to a few thousand sensors.

1.1 Architecture

The control systems centres around a single MySQL database that is populated by an array of independent processes that may or may not run on the same machine where the master database resides.

The master database is continuously replicated (copied) in real time to a variety of slave databases that exist on physically separate machines, some in remote locations. This provides a real-time backup of the master as well as reduces communication load on the master as general queries about detector conditions can be handled by the slaves rather than the master.

Out of bounds alarms (value and rate) are set via a web-based front end. The alarms work by monitoring the database for out of bounds conditions then setting a flag in the database indicating that there is an alarm, and adding an alert message to the log. A second process monitors the log and sends out messages to a pre-determined set of people if a new alert message is found. Alarms are sent via email, SMS, as well as a local auditory and visual siren. If an alarm is not acknowledged within a preset time limit, additional alerts are sent out, eventually adding additional people to the distribution list.

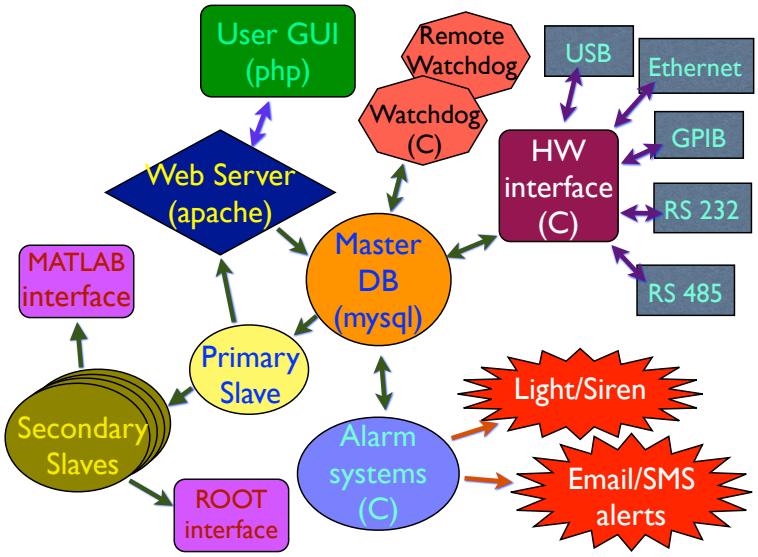

Figure 1.1: Schematic of the slow control system

All processes log themselves with the database and a pair of Watchdogs ensure that all required processes (and each other) are running. The second Watchdog resides on a remote machine to verify that the network is active. The remote Watchdog does not rely on communication with the database to send out alarms.

The primary user interface uses simple forms based on php and html. This means that the simplest web browser can have maximum control over the detector (including phones) without the need for client side plugins such as JAVA or AJAX. Any number of other front ends can be added with the only requirement that they be able to communicate with the MySQL server. Both rudimentary Matlab and ROOT interfaces have been developed by others.

Chapter 2

Installation

2.1 Hardware

The slow control system consists of one or more PCs running Linux. These can be pretty modest ; two cores, at least 2 GB of RAM, a couple of hard drives is not a bad idea. A machine with integrated video is a good idea as they normally use less power than a discrete video card. Get a good UPS.

One machine is designated the 'master'. It contains the master database, and all database writes are done there. One set of computers (which may contain the master) are used for the hardware interfaces, though each non-master machine should have its own watchdog system. Another set of machines (which may also contain the master) run the frontend web server(s), and contain a replicated copy (slave) of the master database. If there are a large number of slaves, it can be a good idea to distribute the load in a tree fashion, with a small number of primary slaves, then secondary slaves replicating them. Alternatively, for small control systems, everything (frontend and backend) can be done on a single machine.

There are a few consideration to make for the hardware interfaces.

If you need GPIB, get a National Instruments GPIB card, and install the driver that is included in the "extras" directory.

If you wish to use RS232 to USB converters, purchase a converter box with more ports than you need. Due to the way USB ports are enumerated, adding boxes in the future can be somewhat painful.

A much better option for serial devices is to use a serial device server. This is a small computer with one or two serial ports that effectively converts

serial to tcp. They can usually be configured for RS232, RS485 or RS488.

If you want webcams, try to buy one of the supported ones from this list: http://mxhaard.free.fr/spca5xx.html

If you have ethernet devices, get a second network card, or make sure you purchase a computer with 2 built in. Do not attach network capable instruments to the internet unless you really know what you are doing. Many run older versions of Windows that can not be patched and are a security risk. You'll also need an ethernet switch to connect the devices and computer together.

2.2 Pre-requisite Software

For operating system, any modern Linux is fine. OpenSUSE [1] is a good all around distribution with some nice configuration tools.

During installation, you need to install:

- MySql
- Apache (or any other webserver)
- php
- php-mysql
- phpMyAdmin (not necessary but convenient)
- C/C++ dev tools
- gspca (for cameras)
- subversion
- kernel source (for hardware that uses kernel modules)

After installation carefully go through the run levels to only run the services you need (webserver, mysql, ntp). Also run a systems update to make sure the system is up to date.

Systems are normally run headless and at runlevel 3. There isn't any point in having X running in the background if it can't or won't ever be called on.

[1] http://software.opensuse.org

2.3 Installing the 'Backend'

Now for the actual slow control software. There are two repositories to make the system easier to use and keep updated. All the documentation (including this one) is kept in the backend repository. The front end can be checked out directly into your web server directory and contains only php code. To check out the slow control backend, type:

`svn co svn://clean.physics.yale.edu/SC_backend`

go to:

`SC_backend/slow_control_code/lib`

and run make. Either add this directory to your library path, or make a link to /textttlibslow_control.so in a directory that is in your library path (`/usr/lib`, or `/usr/local/lib` for example). Then go to:

`SC_backend/slow_control_code`

and run make again. Copy `slow_control_db_conf.ini` from `SC_backend/Docs` to `/etc` and edit appropriately. The file looks like this:

```
[client]
host=localhost
database=control
password=control_user_password
user=control_user
port=3306
```

'host' is the location of the master database, which is 'localhost' if it resides on the same computer and a resolvable IP address otherwise. You will want to change the password to something else, but be aware that it is stored as plain text here. The port number for MySQL is 3306 by default, but you may use some other port on your system. The MySQL user name for the slow control software is 'control_user' but you can change that to something else here. Similarly, you can change the name of the database to something other than 'control'. If you wish to use a different `.ini` file name (for multiple installations on a single machine, for example), then you will have to change the definition of 'DEF_DB_CONF_FILE' in `SC_backend/slow_control_code/include/SC_db_interface.h`.

The system is not yet usable, you must first set up the master MySQL database.

2.4 Setting up MySQL

First make sure that you have a root password set for your MySQL installation. Log in as root and create your master database. As per the example above, I'll assume it is called 'control'. Import the SQL structure and data from `SC_backend/Docs/control_DB_base.sql` into this database.

See appendix B for a list of the tables that need to be defined in the control database. A copy of this database can be found in `SC_backend/Docs/`.

2.5 Installing the 'Frontend'

Go to your web server directory (with appropriate permissions):

```
svn co svn://clean.physics.yale.edu/SC_web
```

Set permissions so that the `jpgraph_cache` directory can be written to by whatever user your web server uses for php code. (a+rw is probably safe here, but only this directory).

Chapter 3

Usage

3.1 The 'Backend'

The 'Back-end' is made up of individual programs that are each dedicated to a small number of tasks. These programs are largely controlled through the web 'Front-end' (see section 3.2.6). The exception to this is the 'Watchdog' program which has to be started either manually or by the operating system upon boot-up. A script for starting the Watchdog is created by running the install script during installation (section **??**).

3.2 The 'Frontend'

3.2.1 Users and Privileges

All users of the Experimental Controls System have a unique user name and password. The log-in button is always in the upper right of all web pages. When a user logs onto the controls page, the connecting IP is captured, and user information is loaded from the database. One item from the database is a 'privilege list'. This list gives that user a set of privileges that allows him or her to control various aspects of the experiment. For example having the 'T-Control' privilege may mean that that user is allowed to change the set point of a temperature controller. Privileges may also be used in combination, so that an ion gauge may require both 'HV' and 'Pressure' privileges, for example.

The second access control is based on IP logging. Certain controls should

only be carried out from specific locations. For example, a high voltage supply may require that the user have 'HV' privileges and have a connecting IP that corresponds to the computer sub-net in the same room.

If a user does not have the correct access privileges, then not only may they not set that control, it does not even appear on the pages.

If a user does not log-in, then they are assigned guest privileges. A small subset of pages may be viewed, but no controls are accessible.

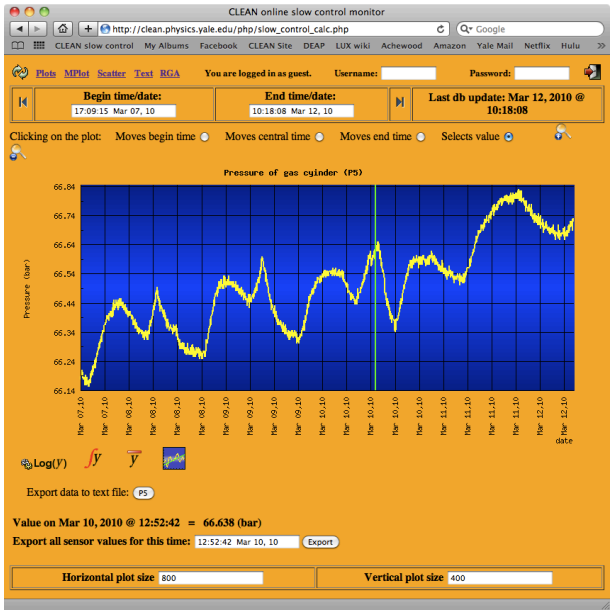

Figure 3.1: Screen capture of the 'Calcs' page showing a number of the standard interface elements.

3.2.2 Displays

There are a number of display pages that are useful for different situations and sensors. The 'Plots' page plots each sensor value over a fixed time interval on its own graph. The time interval can be edited manually, or a number of short-cuts can be entered into the 'Begin time' and 'End time' fields. (Show how -2h works here)

The 'MPlot' page plots up to 7 lines on the same graph, each with its own vertical axis. The 'Scatter' page plots one sensor value against another

over the given time interval. This can be useful for looking at pressure vs. volume or other phase information.

The 'Text' page is a simple table of the current values of the selected sensors. At the bottom of the page one can choose a number to increase the averaging that is done. This can be useful if you want to monitor a sensor that is noisy but read out quickly.

3.2.3 Controls

Figure 3.2: Screen capture of the 'Control' page showing the three types of controls.

If a sensor is designated as 'settable' in its configuration, then it will appear on this page. Settable sensors can be configured in one of two ways. If the set-point is a continuous variable, like a temperature or flow, then it will appear on the controls page as a text field, and only accept a float as an input. In this case, the units are set in the 'units' field on the configuration page.

If the control is discrete, then either buttons will appear on the page if there are fewer than five choices, or a pull down menu will be available if there are five or more choices. The configuration of the controls is described in section 3.2.6.

Buttons are controlled by clicking them once, pull downs are set by selecting the desired setting and clicking the 'Change' button. Numeric values are entered by typing, and hitting enter when complete.

3.2.4 HV Crate Controls

If a high voltage crate is to be used as part of the slow control, there is a specialized page that may be turned on by setting 'int1' of the 'have_crate'' entry in the 'globals' table to 1.

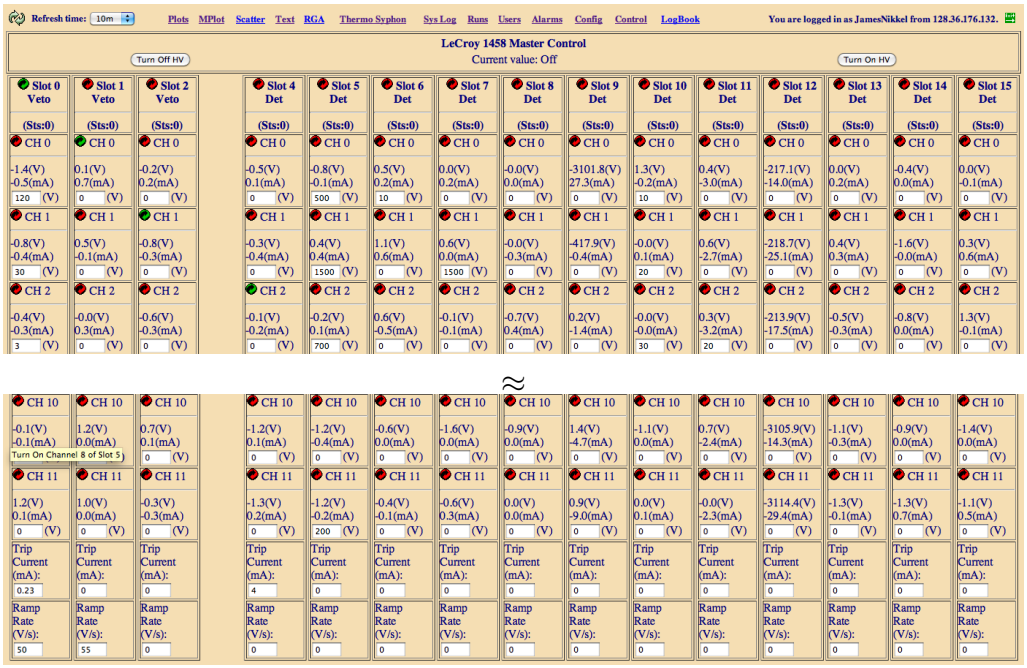

Figure 3.3: Screen capture of the 'HV Crate' page showing the display and control elements.

Figure 3.3 is a screen capture of the control page with the centre section cut out to save some space. At the top of the page below the header is the master On/Off control. The rest of the page is arrange the same way the cards are in the crate. The cell at the top of each column contains a single control for toggling that card on and off. Red indicates that it is off, green indicates that it is on. There is also a short designator for the slot, 'Veto, or 'Det (for Detector) in this case (this text is taken from the 'Subtype' in the Sensor Configuration). 'SSt:X indicates the status of that card. 'Sts:0 meansit is okay, non-zero values indicate some sort of problem or activity (such as ramping up a voltage).

Below the slot level cell are the individual channel control cells. Each channel can be toggled on or off by clicking the red (off→on) or green (on→off) button. If a channel has tripped off, the cell will turn to red and a reset button will appear. An alarm will be sent out, and the channel will be automatically set to Off. To reset the channel, fix the problem, and click

the ‘Reset button. The voltage will not come back on until you click the red (off→on) button on that channel.

Below that are the voltage and current outputs. Below that is the voltage set-point. Enter a new value and hit enter to commit it. If you change multiple fields then hit enter, only the last one will be changed.

At the bottom of each column are fields for the Current Trip value, and the Ramp Rate. These values are set for the entire slot (all 12 channels)

If a slot is not occupied or is otherwise incapacitated, then there will be a blank column. In this example, Slot 3 is empty. A slot can be disabled by clicking the ‘Hide button of the sensor in the Sensor Configuration corresponding to that slot.

3.2.5 Alarms

Alarms are an crucial part of the monitoring and control system. Each sensor entry may have up to 4 alarms set for it. A high and low value alarm, and a high and low rate alarm. The rate alarm becomes available if the “Show Rate” checkbox is selected in the sensor configuration for that entry (see Section 3.2.6).

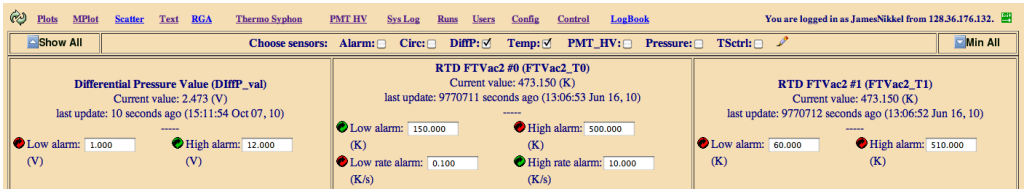

Figure 3.4: Screen capture of the ‘Alarms’ page showing the three sensors and their alarm settings.

Figure 3.4 shows the alarms page along with three different sensors and their alarm settings. The current value and the last update time are shown along with the alarm configuration. The sensor on the left has a high alarm set, so that if the value (2.47V as of 10 seconds ago) goes above 12, an alarm is sent. The middle entry has a low alarm set point, as well as a high rate set point. The sensor on the right has no alarms set.

There are two primary components to the alarm system. The first is the “Alarm Trip Daemon” that monitors the sensor values and alarm set points

and makes an entry into the systems log if a sensor goes out of bounds. The second is the "Alarm Alert Daemon" that monitors the systems log and sends out email and text messages to the "on-call" users to alert them to the problem. This daemon also sets a flag that there is an alarm set. This creates an "Acknowledge Current Alarm" button on the alarms page and also forces each page to play a siren sound. To stop the alarm, any regular user (i.e. not a guest account user) can click the "Acknowledge Current Alarm" button to reset the alarm flag and an entry with that username goes to the systems log. It is a good idea to fix the problem that caused the alarm to go off first to prevent it from sounding repeatedly.

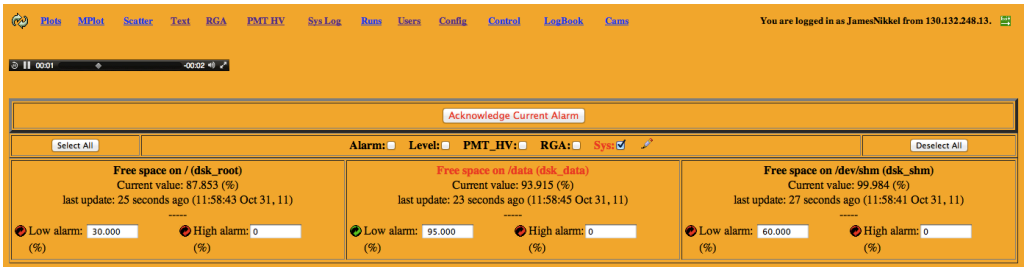

Figure 3.5: Screen capture of the 'Alarms' page with one sensor below the alarm set point. The black bar on the upper left is the media player that plays an alarm sound and below that is the alarm acknowledge button.

Figure 3.5 shows the alarms page along with

3.2.6 Configurations

Sensor Configuration

Here one can edit the settings that are associated with each individual sensor. All changes are saved to the systems log along with the user who made the change.

Instrument Configuration

Here one can edit and control the 'Instruments' that make up the control system. Each is a stand alone software program that may or may not communicate with one or more physical instrument. For example there may be

one entry for a temperature controller that has four thermometers and a heater, one program may communicate with six different ModBUS devices, and another may just perform some purely software task. The alarm system programs as well as the Watchdogs are all controlled and configured on this page.

Appendix A

Safety

A word should be said about personnel safety with regards to the Experimental Controls System. Since this system is open to a network and is necessarily very complicated, one should assume that it is possible for the system to become compromised, and that bugs are certain to be present. It can not be assumed that hardware attached to the experimental controls system can be made safe through its actions or in spite of them.

Computer systems that are used for life safety, in medical or flight applications, for example, go through rigorous testing both at the hardware and software level. No such rigour is possible for this controls system. To ensure safety of personnel, external fail-safe systems and interlocks are required when controlling hardware that may cause injury.

For example, this controls system may control a high voltage power supply. If the conductor connected to the supply is being installed or undergoing maintenance, then that supply must be powered off (from the mains, for example), or assumed to be live. It is not sufficient to assume that the control system has turned the voltage down to a safe level or that it will not turn it up to an unsafe level spontaneously.

Gas control systems are another case where unsafe situations may arise. While it may be convenient to have remote control operation of valves and flow rates available, that is not a substitute for reliable pressure relief valves and rupture devices.

The Experimental Controls System is a powerful tool for control and monitoring but it does not replace or mitigate any regular safety procedures or devices.

Appendix B

Control Database Structure

The following latex tables show the structure (but not data) of the control database. A full SQL copy, including some necessary data entries, can be found as 'control_DB_base.sql' in the Docs directory.

Table B.1: Structure of table daq_control

Column	Type	Null	Default
entry_id	int(11)	No	
lug_entry_id	int(11)	No	
utc	int(11)	No	
acquire_now	tinyint(1)	No	

Table B.2: Structure of table daq_control

Column	Type	Null	Default
entry_id	int(11)	No	
lug_entry_id	int(11)	No	
utc	int(11)	No	
acquire_now	tinyint(1)	No	

Table B.3: Structure of table daq_control_tmp

Column	Type	Null	Default
utc	int(11)	No	

Table B.3: Structure of table daq_control_tmp (continued)

Column	**Type**	**Null**	**Default**
user_name	text	No	
tag	text	No	
acquire_now	tinyint(1)	No	
xml_settings_file	text	No	
end_status	text	No	

Table B.4: Structure of table daq_control_tmp

Column	**Type**	**Null**	**Default**
utc	int(11)	No	
user_name	text	No	
tag	text	No	
acquire_now	tinyint(1)	No	
xml_settings_file	text	No	
end_status	text	No	

Table B.5: Structure of table daq_presets

Column	**Type**	**Null**	**Default**
entry_id	int(11)	No	
preset	mediumblob	No	

Table B.6: Structure of table daq_presets

Column	**Type**	**Null**	**Default**
entry_id	int(11)	No	
preset	mediumblob	No	

Table B.7: Structure of table dqm_channel_data

Column	Type	Null	Default
utc	int(11)	No	
runs	text	No	
prefixes	text	No	
dat_files	text	No	
avg_event_rate	float	No	
avg_livetime	float	No	
pulse_rates	text	No	
avg_pulse_lengths	text	No	
avg_baselines_mv	text	No	

Table B.8: Structure of table dqm_channel_data

Column	Type	Null	Default
utc	int(11)	No	
runs	text	No	
prefixes	text	No	
dat_files	text	No	
avg_event_rate	float	No	
avg_livetime	float	No	
pulse_rates	text	No	
avg_pulse_lengths	text	No	
avg_baselines_mv	text	No	

Table B.9: Structure of table dqm_control

Column	Type	Null	Default
utc_time	int(11)	No	

Table B.10: Structure of table dqm_control

Column	Type	Null	Default
utc_time	int(11)	No	

Table B.11: Structure of table dqm_event_data

Column	**Type**	**Null**	**Default**
utc_time	int(11)	No	
run	int(11)	No	
prefix	text	No	
event_rate	float	No	
livetime	float	No	
deadtime	float	No	

Table B.12: Structure of table dqm_event_data

Column	**Type**	**Null**	**Default**
utc_time	int(11)	No	
run	int(11)	No	
prefix	text	No	
event_rate	float	No	
livetime	float	No	
deadtime	float	No	

Table B.13: Structure of table dqm_event_display

Column	**Type**	**Null**	**Default**
utc	int(11)	No	
picture	longblob	No	

Table B.14: Structure of table dqm_event_display

Column	**Type**	**Null**	**Default**
utc	int(11)	No	
picture	longblob	No	

Table B.15: Structure of table globals

Column	**Type**	**Null**	**Default**
name	varchar(16)	No	
int1	int(11)	No	0
int2	int(11)	No	0
int3	int(11)	No	0
int4	int(11)	No	0
double1	double	No	0
double2	double	No	0
double3	double	No	0
double4	double	No	0
string1	varchar(16)	Yes	NULL
string2	varchar(16)	Yes	NULL
string3	varchar(16)	Yes	NULL
string4	varchar(16)	Yes	NULL

Table B.16: Structure of table globals

Column	**Type**	**Null**	**Default**
name	varchar(16)	No	
int1	int(11)	No	0
int2	int(11)	No	0
int3	int(11)	No	0
int4	int(11)	No	0
double1	double	No	0
double2	double	No	0
double3	double	No	0
double4	double	No	0
string1	varchar(16)	Yes	NULL
string2	varchar(16)	Yes	NULL
string3	varchar(16)	Yes	NULL
string4	varchar(16)	Yes	NULL

Table B.17: Structure of table lug_acq_sources

Column	Type	Null	Default
source_name	varchar(40)	No	

Table B.18: Structure of table lug_acq_sources

Column	Type	Null	Default
source_name	varchar(40)	No	

Table B.19: Structure of table lug_categories

Column	Type	Null	Default
category	varchar(40)	No	

Table B.20: Structure of table lug_categories

Column	Type	Null	Default
category	varchar(40)	No	

Table B.21: Structure of table lug_entries

Column	Type	Null	Default
entry_id	int(11)	No	
important_flag	tinyint(1)	Yes	0
action_user	varchar(40)	No	
action_time	datetime	No	
edit_user	varchar(40)	Yes	NULL
edit_time	datetime	Yes	NULL
run_number	int(11)	No	
category	varchar(40)	No	
subcategory	varchar(40)	No	
entry_description	longtext	No	
entry_image	mediumblob	Yes	NULL

Table B.21: Structure of table lug_entries (continued)

Column	Type	Null	Default
entry_image_thumb	blob	Yes	NULL
entry_file	mediumblob	Yes	NULL
filename	char(40)	Yes	NULL
source	varchar(40)	Yes	NULL
strikeme	tinyint(1)	Yes	0

Table B.22: Structure of table lug_entries

Column	Type	Null	Default
entry_id	int(11)	No	
important_flag	tinyint(1)	Yes	0
action_user	varchar(40)	No	
action_time	datetime	No	
edit_user	varchar(40)	Yes	NULL
edit_time	datetime	Yes	NULL
run_number	int(11)	No	
category	varchar(40)	No	
subcategory	varchar(40)	No	
entry_description	longtext	No	
entry_image	mediumblob	Yes	NULL
entry_image_thumb	blob	Yes	NULL
entry_file	mediumblob	Yes	NULL
filename	char(40)	Yes	NULL
source	varchar(40)	Yes	NULL
strikeme	tinyint(1)	Yes	0

Table B.23: Structure of table lug_gas_inventory_entries

Column	Type	Null	Default
entry_id	int(11)	No	
entry_date	datetime	No	
location_name	varchar(40)	No	
location_gas_type	varchar(10)	No	

Table B.23: Structure of table lug_gas_inventory_entries (continued)

Column	Type	Null	Default
gross_mass_kg	double(7,3)	Yes	NULL
net_gas_mass_kg	double(7,3)	Yes	NULL
comments	mediumtext	Yes	NULL
strikeme	tinyint(1)	Yes	0

Table B.24: Structure of table lug_gas_inventory_entries

Column	Type	Null	Default
entry_id	int(11)	No	
entry_date	datetime	No	
location_name	varchar(40)	No	
location_gas_type	varchar(10)	No	
gross_mass_kg	double(7,3)	Yes	NULL
net_gas_mass_kg	double(7,3)	Yes	NULL
comments	mediumtext	Yes	NULL
strikeme	tinyint(1)	Yes	0

Table B.25: Structure of table lug_gas_locations

Column	Type	Null	Default
location_id	int(11)	No	
location_name	varchar(20)	No	
tare_mass_kg	double(6,3)	Yes	NULL
location_description	mediumtext	Yes	NULL

Table B.26: Structure of table lug_gas_locations

Column	Type	Null	Default
location_id	int(11)	No	
location_name	varchar(20)	No	
tare_mass_kg	double(6,3)	Yes	NULL

Table B.26: Structure of table lug_gas_locations (continued)

Column	Type	Null	Default
location_description	mediumtext	Yes	NULL

Table B.27: Structure of table lug_gas_transfers

Column	Type	Null	Default
transfer_id	int(11)	No	
action_category	varchar(12)	Yes	Gas Transfer
transfer_user	varchar(40)	No	
transfer_date	datetime	No	
entry_user	varchar(40)	No	
entry_date	timestamp	No	CURRENT_TIMESTAMP
run_number	int(11)	No	
gas_type	varchar(10)	No	
transferred_from	varchar(40)	No	
transferred_to	varchar(40)	No	
transfer_total	double(7,3)	No	
transfer_total_units	varchar(10)	No	
strikeme	tinyint(1)	Yes	0

Table B.28: Structure of table lug_gas_transfers

Column	Type	Null	Default
transfer_id	int(11)	No	
action_category	varchar(12)	Yes	Gas Transfer
transfer_user	varchar(40)	No	
transfer_date	datetime	No	
entry_user	varchar(40)	No	
entry_date	timestamp	No	CURRENT_TIMESTAMP
run_number	int(11)	No	
gas_type	varchar(10)	No	
transferred_from	varchar(40)	No	
transferred_to	varchar(40)	No	

Table B.28: Structure of table lug_gas_transfers (continued)

Column	Type	Null	Default
transfer_total	double(7,3)	No	
transfer_total_units	varchar(10)	No	
strikeme	tinyint(1)	Yes	0

Table B.29: Structure of table lug_gas_type

Column	Type	Null	Default
type_id	int(11)	No	
type_name	varchar(10)	No	

Table B.30: Structure of table lug_gas_type

Column	Type	Null	Default
type_id	int(11)	No	
type_name	varchar(10)	No	

Table B.31: Structure of table lug_iqs

Column	Type	Null	Default
entry_id	int(11)	No	
submitted_by_user	varchar(40)	No	
entry_time	datetime	No	
last_edited_by_user	varchar(40)	No	
last_edited_time	timestamp	No	CURRENT_TIMESTAMP
iq_type	varchar(40)	No	
filename	char(40)	No	
source	varchar(40)	No	
values_xml	mediumtext	No	
comments	mediumtext	Yes	NULL
file_attach	mediumblob	Yes	NULL
image_attach	mediumblob	Yes	NULL

Table B.31: Structure of table lug_iqs (continued)

Column	Type	Null	Default
image_attach_thumb	blob	Yes	NULL
strikeme	tinyint(1)	Yes	0

Table B.32: Structure of table lug_iqs

Column	Type	Null	Default
entry_id	int(11)	No	
submitted_by_user	varchar(40)	No	
entry_time	datetime	No	
last_edited_by_user	varchar(40)	No	
last_edited_time	timestamp	No	CURRENT_TIMESTAMP
iq_type	varchar(40)	No	
filename	char(40)	No	
source	varchar(40)	No	
values_xml	mediumtext	No	
comments	mediumtext	Yes	NULL
file_attach	mediumblob	Yes	NULL
image_attach	mediumblob	Yes	NULL
image_attach_thumb	blob	Yes	NULL
strikeme	tinyint(1)	Yes	0

Table B.33: Structure of table lug_iqs_type

Column	Type	Null	Default
iq_name	varchar(40)	No	
description	text	Yes	NULL

Table B.34: Structure of table lug_iqs_type

Column	Type	Null	Default
iq_name	varchar(40)	No	
description	text	Yes	NULL

Table B.35: Structure of table lug_subcategories

Column	Type	Null	Default
category	varchar(40)	No	
subcategory	varchar(40)	No	

Table B.36: Structure of table lug_subcategories

Column	Type	Null	Default
category	varchar(40)	No	
subcategory	varchar(40)	No	

Table B.37: Structure of table msg_log

Column	Type	Null	Default
msg_id	int(11)	No	
time	int(11)	No	
ip_address	varchar(16)	Yes	NULL
subsys	varchar(16)	Yes	NULL
msgs	text	No	
type	varchar(16)	No	
is_error	tinyint(4)	No	0

Table B.38: Structure of table msg_log

Column	Type	Null	Default
msg_id	int(11)	No	
time	int(11)	No	
ip_address	varchar(16)	Yes	NULL
subsys	varchar(16)	Yes	NULL
msgs	text	No	
type	varchar(16)	No	
is_error	tinyint(4)	No	0

Table B.39: Structure of table msg_log_types

Column	Type	Null	Default
types	varchar(16)	No	

Table B.40: Structure of table msg_log_types

Column	Type	Null	Default
types	varchar(16)	No	

Table B.41: Structure of table runs

Column	Type	Null	Default
num	int(11)	No	
start_t	int(11)	No	
end_t	int(11)	No	
note	char(128)	No	

Table B.42: Structure of table runs

Column	Type	Null	Default
num	int(11)	No	
start_t	int(11)	No	
end_t	int(11)	No	
note	char(128)	No	

Table B.43: Structure of table sc_insts

Column	Type	Null	Default
name	varchar(16)	No	
description	varchar(64)	Yes	NULL
subsys	varchar(16)	No	
run	tinyint(1)	No	0
restart	tinyint(1)	No	0
WD_ctrl	tinyint(1)	No	1

Table B.43: Structure of table sc_insts (continued)

Column	Type	Null	Default
path	varchar(256)	No	rel_path
dev_type	varchar(16)	Yes	NULL
dev_address	varchar(24)	Yes	NULL
start_time	int(11)	No	-1
last_update_time	int(11)	No	-1
PID	int(11)	No	-1
user1	varchar(16)	Yes	NULL
user2	varchar(16)	Yes	NULL
parm1	double	No	0
parm2	double	No	0
notes	text	No	

Table B.44: Structure of table sc_insts

Column	Type	Null	Default
name	varchar(16)	No	
description	varchar(64)	Yes	NULL
subsys	varchar(16)	No	
run	tinyint(1)	No	0
restart	tinyint(1)	No	0
WD_ctrl	tinyint(1)	No	1
path	varchar(256)	No	rel_path
dev_type	varchar(16)	Yes	NULL
dev_address	varchar(24)	Yes	NULL
start_time	int(11)	No	-1
last_update_time	int(11)	No	-1
PID	int(11)	No	-1
user1	varchar(16)	Yes	NULL
user2	varchar(16)	Yes	NULL
parm1	double	No	0
parm2	double	No	0
notes	text	No	

Table B.45: Structure of table sc_sensors

Column	Type	Null	Default
name	varchar(16)	No	
description	varchar(64)	Yes	NULL
type	varchar(16)	No	unknown
subtype	varchar(16)	Yes	NULL
ctrl_priv	varchar(128)	No	full
num	int(11)	No	0
instrument	varchar(16)	No	inst_name
units	varchar(16)	No	units
discrete_vals	varchar(128)	Yes	NULL
al_set_val_low	double	No	0
al_set_val_high	double	No	0
al_arm_val_low	tinyint(1)	No	0
al_arm_val_high	tinyint(1)	No	0
al_set_rate_low	double	No	0
al_set_rate_high	double	No	0
al_arm_rate_low	tinyint(1)	No	0
al_arm_rate_high	tinyint(1)	No	0
alarm_tripped	tinyint(1)	No	0
grace	int(11)	No	0
last_trip	int(11)	No	-1
settable	tinyint(1)	No	0
show_rate	tinyint(1)	No	0
hide_sensor	tinyint(1)	No	0
update_period	int(11)	No	60
num_format	varchar(16)	Yes	NULL
user1	varchar(16)	Yes	NULL
user2	varchar(16)	Yes	NULL
user3	varchar(16)	Yes	NULL
user4	varchar(16)	Yes	NULL
parm1	double	No	0
parm2	double	No	0
parm3	double	No	0
parm4	double	No	0

Table B.45: Structure of table sc_sensors (continued)

Column	Type	Null	Default
notes	text	No	

Table B.46: Structure of table sc_sensors

Column	Type	Null	Default
name	varchar(16)	No	
description	varchar(64)	Yes	NULL
type	varchar(16)	No	unknown
subtype	varchar(16)	Yes	NULL
ctrl_priv	varchar(128)	No	full
num	int(11)	No	0
instrument	varchar(16)	No	inst_name
units	varchar(16)	No	units
discrete_vals	varchar(128)	Yes	NULL
al_set_val_low	double	No	0
al_set_val_high	double	No	0
al_arm_val_low	tinyint(1)	No	0
al_arm_val_high	tinyint(1)	No	0
al_set_rate_low	double	No	0
al_set_rate_high	double	No	0
al_arm_rate_low	tinyint(1)	No	0
al_arm_rate_high	tinyint(1)	No	0
alarm_tripped	tinyint(1)	No	0
grace	int(11)	No	0
last_trip	int(11)	No	-1
settable	tinyint(1)	No	0
show_rate	tinyint(1)	No	0
hide_sensor	tinyint(1)	No	0
update_period	int(11)	No	60
num_format	varchar(16)	Yes	NULL
user1	varchar(16)	Yes	NULL
user2	varchar(16)	Yes	NULL
user3	varchar(16)	Yes	NULL

Table B.46: Structure of table sc_sensors (continued)

Column	Type	Null	Default
user4	varchar(16)	Yes	NULL
parm1	double	No	0
parm2	double	No	0
parm3	double	No	0
parm4	double	No	0
notes	text	No	

Table B.47: Structure of table sc_sensor_types

Column	Type	Null	Default
num	int(11)	No	
name	varchar(16)	No	

Table B.48: Structure of table sc_sensor_types

Column	Type	Null	Default
num	int(11)	No	
name	varchar(16)	No	

Table B.49: Structure of table sc_sens_Alarm_Light

Column	Type	Null	Default
time	int(11)	No	
value	double	No	
rate	double	No	

Table B.50: Structure of table sc_sens_Alarm_Light

Column	Type	Null	Default
time	int(11)	No	
value	double	No	

Table B.50: Structure of table sc_sens_Alarm_Light (continued)

Column	Type	Null	Default
rate	double	No	

Table B.51: Structure of table sc_sens_Alarm_Siren

Column	Type	Null	Default
time	int(11)	No	
value	double	No	
rate	double	No	

Table B.52: Structure of table sc_sens_Alarm_Siren

Column	Type	Null	Default
time	int(11)	No	
value	double	No	
rate	double	No	

Table B.53: Structure of table sc_sens_dsk_data

Column	Type	Null	Default
time	int(11)	No	
value	double	No	
rate	double	No	

Table B.54: Structure of table sc_sens_dsk_data

Column	Type	Null	Default
time	int(11)	No	
value	double	No	
rate	double	No	

Table B.55: Structure of table sc_sens_dsk_root

Column	**Type**	**Null**	**Default**
time	int(11)	No	
value	double	No	
rate	double	No	

Table B.56: Structure of table sc_sens_dsk_root

Column	**Type**	**Null**	**Default**
time	int(11)	No	
value	double	No	
rate	double	No	

Table B.57: Structure of table sc_sens_dsk_shm

Column	**Type**	**Null**	**Default**
time	int(11)	No	
value	double	No	
rate	double	No	

Table B.58: Structure of table sc_sens_dsk_shm

Column	**Type**	**Null**	**Default**
time	int(11)	No	
value	double	No	
rate	double	No	

Table B.59: Structure of table users

Column	**Type**	**Null**	**Default**
user_name	varchar(32)	No	
password	char(32)	No	
full_name	varchar(32)	No	
affiliation	varchar(64)	No	

Table B.59: Structure of table users (continued)

Column	Type	Null	Default
email	varchar(32)	No	
sms	varchar(32)	No	
phone	varchar(16)	No	
on_call	tinyint(1)	No	
shift_status	varchar(16)	No	
privileges	text	No	

Table B.60: Structure of table users

Column	Type	Null	Default
user_name	varchar(32)	No	
password	char(32)	No	
full_name	varchar(32)	No	
affiliation	varchar(64)	No	
email	varchar(32)	No	
sms	varchar(32)	No	
phone	varchar(16)	No	
on_call	tinyint(1)	No	
shift_status	varchar(16)	No	
privileges	text	No	

Table B.61: Structure of table user_privileges

Column	Type	Null	Default
name	varchar(16)	No	
allowed_host	varchar(16)	No	all

Table B.62: Structure of table user_privileges

Column	Type	Null	Default
name	varchar(16)	No	
allowed_host	varchar(16)	No	all

Table B.63: Structure of table user_shift_status

Column	Type	Null	Default
status	varchar(16)	No	
can_manage	tinyint(1)	No	0

Table B.64: Structure of table user_shift_status

Column	Type	Null	Default
status	varchar(16)	No	
can_manage	tinyint(1)	No	0