Total On-line Access Data System (TOADS)

Phase II Final Report for the Period August 2002 – August 2004

September 2004

Authors:
Katherine L. Yuracko, Ph.D., YAHSGS LLC
Morey Parang, YAHSGS LLC
David C. Landguth, Oak Ridge National Laboratory
Robert Coleman, Oak Ridge National Laboratory

Prepared for the U.S. Department of Energy

Work Performed Under STTR Grant No. DE-FG02-01ER86133

Prepared by
YAHSGS LLC
P.O. Box 667
Richland, WA 99352

Total On-line Access Data System (TOADS)

Phase II Final Report for the Period August 2002 – August 2004

ABSTRACT

TOADS (Total On-line Access Data System) is a new generation of real-time monitoring and information management system developed to support unattended environmental monitoring and long-term stewardship of U.S. Department of Energy facilities and sites. TOADS enables project managers, regulators, and stakeholders to view environmental monitoring information in real-time over the Internet. Deployment of TOADS at government facilities and sites will reduce the cost of monitoring while increasing confidence and trust in cleanup and long term stewardship activities. TOADS:

- Reliably interfaces with and acquires data from a wide variety of external databases, remote systems, and sensors such as contaminant monitors, area monitors, atmospheric condition monitors, visual surveillance systems, intrusion devices, motion detectors, fire/heat detection devices, and gas/vapor detectors;
- Provides notification and triggers alarms as appropriate;
- Performs QA/QC on data inputs and logs the status of instruments/devices;
- Provides a fully functional data management system capable of storing, analyzing, and reporting on data;
- Provides an easy-to-use Internet-based user interface that provides visualization of the site, data, and events; and
- Enables the community to monitor local environmental conditions in real time.

During this Phase II STTR project, TOADS has been developed and successfully deployed for unattended facility, environmental, and radiological monitoring at a Department of Energy facility.

Table of Contents

List of Figures

1.0 Overview of TOADS

TOADS (Total On-line Access Data System) is a new generation of real-time remote monitoring and information management system to support unattended environmental monitoring and long-term stewardship of U.S. Department of Energy (DOE) facilities and sites. TOADS enables project managers, regulators, and stakeholders to view environmental monitoring information in real-time over the Internet. Deployment of TOADS at government facilities and sites will reduce the cost of monitoring while increasing confidence and trust in cleanup and long term stewardship activities. TOADS:

- Reliably interfaces with and acquires data from a wide variety of instruments, sensors and transducers, external databases, and other remote systems, such as contaminant monitors, area monitors, atmospheric condition monitors, visual surveillance systems, intrusion devices, motion detectors, fire/heat detection devices, and gas/vapor detectors;
- Provides notification and triggers alarms as appropriate;
- Performs QA/QC on data inputs and logs the status of instruments/devices;
- Provides a fully functional data management system capable of storing, analyzing, and reporting on data;
- Provides an easy-to-use Internet-based user interface that provides visualization of the site, data, and events; and
- Enables the community to monitor local environmental conditions in real time.

TOADS Architecture. The TOADS application is based on a client-server architecture developed under Unix. This approach allows multiple clients (such as sensors, users, and external systems) to access the core applications by making requests to TOADS over the network. The TOADS server queues and schedules the requests appropriately for a particular site based on the available hardware and software. Over time, as requirements grow and more resources are added, the TOADS server environment can be modified to meet these needs. This can be achieved by adding additional resources to the system or by distributing its workload. A conceptualized view of the TOADS system architecture and its major components is shown in Figure 1.

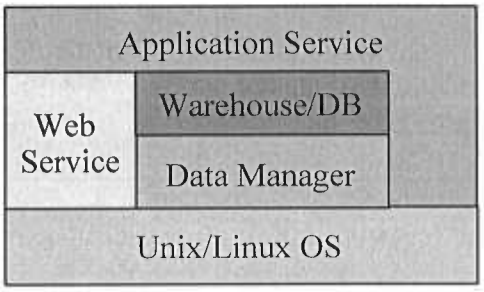

Figure 1. TOADS System Architecture.

The TOADS design strategy makes it both scalable and robust since it allows for physical separation of its system components allowing for distributed databases as well as distributed processing. For example, for a given site, the application server, the web server, and the TOADS warehouse may reside on separate platforms. Although TOADS' architecture is based on the traditional three-tier client/server/database approach, its core design incorporates and supports peer-to-peer capability between TOADS sites.

Data Management. A typical TOADS installation communicates with a number of critical sensors, facilitates remote access and services to other TOADS sites, as well as provides direct access to potentially a large group of users. TOADS' data management system uses a data warehousing approach using object databases and specialized datamarts. The core databases are implemented using the open-source software MySQL RDBMS. MySQL provides a robust SQL database server with fast, multi-threaded, and multi-user capabilities suitable for a production system. The TOADS database contains information about the TOADS site, configuration, sensors, sensor configuration, and sensor data (measurements). Each set of measurements is stored in a separate table specific to an instrument. In addition, the database stores meta data, including instruments, users, site data, as well as integrated information from external data sources such as preexisting databases. The TOADS datamarts store specialized bulk data such as streaming image/video, audio, engineering data (e.g., computational/analysis, CAD, etc.), inventory data, reports, and XML files. The interface to the database implements the flat/ring database table structure.

User Interface. TOADS provides a highly flexible and intuitive graphical user interface (GUI) that can be configured to meet virtually any remote facility/site management need. Depending on permission levels, users can access and control TOADS via Internet, Intranet, wireless, or dialup access and receive on-line surveillance data and customized reports tailored to their specific needs. The user interface is a web-based application that presents user-requested real-time information ranging from raw data and visual images to textual and graphical forms and analyses. The GUI application enables users to view: real-time measurement data in both raw and graphical form; real-time events such as alarms and alerts as well as their historical logs; map of the location site along with the deployed instruments/sensors with an easy to use point and click operation; and video images and sound tracks (as applicable) of any deployed audio/visual instruments for both present and past recordings. Figure 2 shows the TOADS top-level user interface "Monitor Window" invoked as a Java Applet. The user interface is written entirely in Java and works on virtually all major platforms. It has been tested on Unix/Linux and Windows® operating systems.

The TOADS client interface allows users to construct ad-hoc queries as well as reports and submit the request to the TOADS server for processing. The user constructed queries can include requests for both data and metadata. From the TOADS Monitor Window, the user can also invoke trend charts for important and critical sensors. These charts display the most recent data readings for the specific sensors in a real-time fashion. The trend charts may vary depending on the system configuration and may be customized to meet special needs. The charts contain hotspots where mouse movement on these locations will reveal additional information. Clicking the hotspots will pop-up new windows with detailed data-point values.

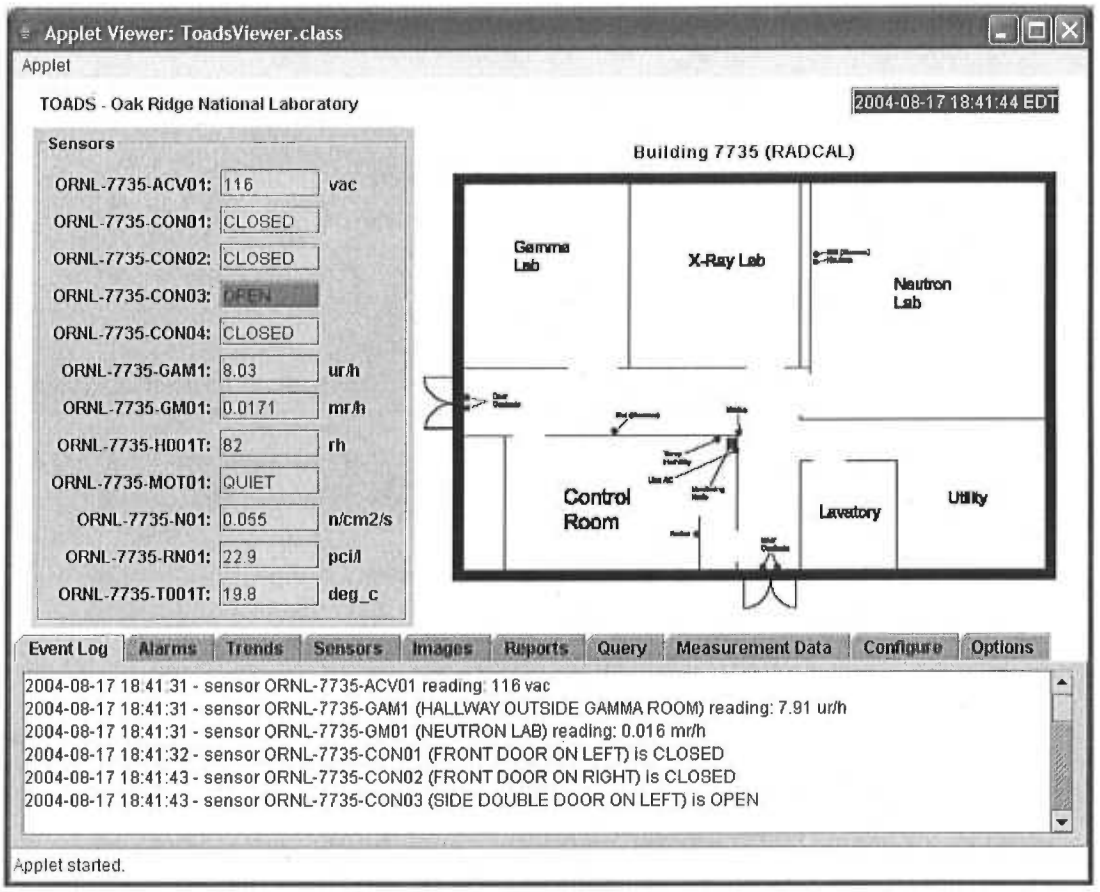

Figure 2. Top-level View of the TOADS User Interface.

Alerts and Alarms. TOADS generates and handles various alert and alarm events based on certain predefined conditions as well as dynamically configurable parameters. All events are logged in the database prior to dispatching. TOADS' event processing falls into three broad categories: system events, data events, and user generated events. System events are caused by hardware failures or malfunction, online/offline conditions, process failures, runtime errors, and system checkpoints. Data events are generated when measurement values exceed predefined high/low alarm set points, out-of-range thresholds, read errors, and other data validation errors. User events are generated for such things as login/logout activities and access violations. Alerts include such mundane conditions as *sensor online* and *sensor offline*. Alarms on the other hand include the all-important "sensor reading out-of-range" events or similar user/data specified conditions. When TOADS detects a sensor reading that exceeds some preset alarm value, the sensor is said to be in an alarm state and an alarm event is sent to the client(s). When an alert/alarm event is detected, they are logged in the database prior to dispatching the information, including forwarding the alarm/event to registered interested users. Depending on the configuration, acknowledgement of an event may be required, which is also logged in the database.

Remote Monitoring Node. The remote monitoring node consists of an embedded computer board with an array of inputs for monitoring most types of physical transducers which can be readily configured for data collection, conversion, storage and transmission to remote servers. Figure 3 shows the overall topology of a TOADS network, illustrating how multiple remote monitoring nodes can be deployed for a single central server.

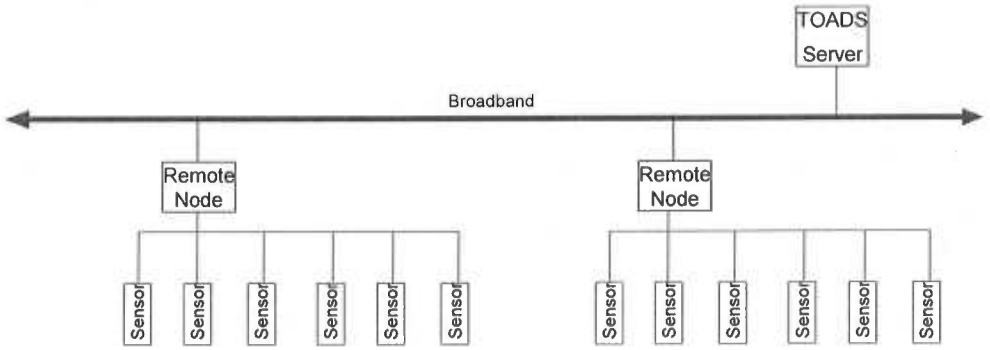

Figure 3. Overall Topology of a TOADS Remote Monitoring System.

The TOADS remote monitoring node consists of a controller, sensors, supporting electronics, signal conditioning and processing components, uninterruptible power supply, AC to DC conversion power supplies and associated support hardware and cabling needed to install the system. The node controller consists of an embedded processing unit with analog to digital, digital to analog, digital input/output as well as high speed pulse counting capability. The unit includes a variety of standard input/output mechanisms such as universal serial bus, 10BaseT Ethernet and RS-232 communications ports. Figure 4 provides a generalized schematic of the remote monitoring node.

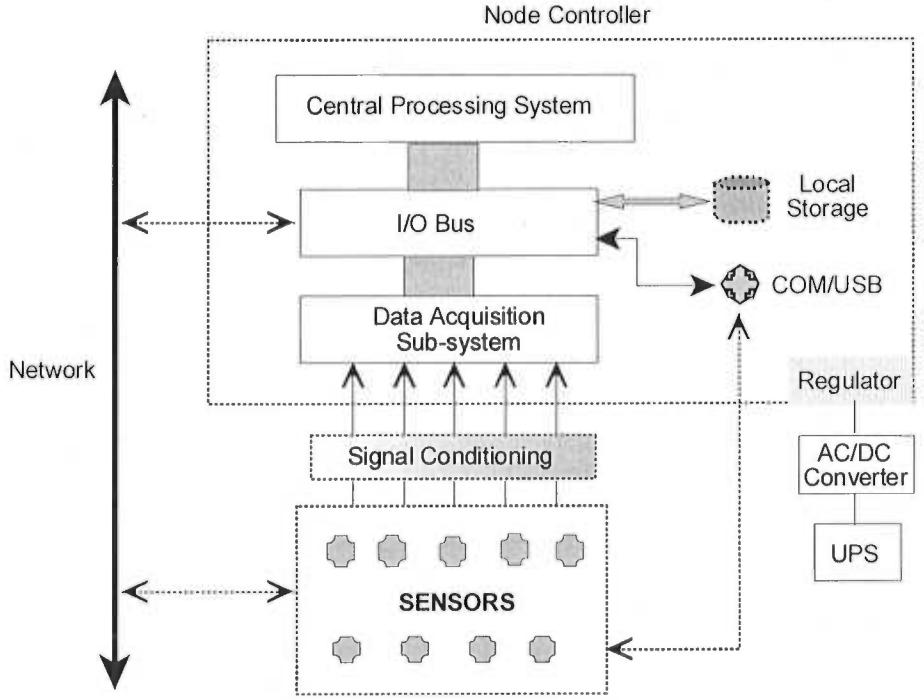

Figure 4. Schematic of a TOADS Remote Monitoring Node Including Primary Components.

The system's architecture is both versatile and extensible and is specifically designed for ease of addition of new sensors. Software drivers were written for each class of sensor employed in the demonstration system to enable the accurate collection of real-world measurements, convert signals to digital form, package the information into XML messages, and transmit the results to a central TOADS server. Although the Phase II development effort was primarily focused on providing interfaces for the sensors used in the demonstration project, the drivers were written so that they can be used with a variety of other sensor types as well.

Sensors. The remote monitoring node was designed for compatibility with most sensor types. A system architecture was designed that is compatible with a wide range of auxiliary hardware and that provides relatively easy development of sensor drivers. A variety of sensors were deployed to demonstrate the ability of the system to meet performance requirements. The sensors used included facility, environmental and radiological monitoring devices as summarized in Figure 5.

Sensor Class	Sensor Type	Quantity Measured	Reporting Units	Quantity Deployed
Facility	Door contact	Door status	Open/closed	4
	Motion sensor	Motion	Motion/quiet	1
	Line A/C condition	A/C voltage at facility	Volts AC	1
Environmental	Thermocouple	Temp	Degrees C	1
	Hygrometer	Relative Humidity	RH%	1
Radiological	NaI(Tl)	Gamma Exposure Rate	uR/h	1
	GM tube	Beta/Gamma Levels	mrem/h	1
	ZnS(Ag)	Fast neutron	$n_f/cm^2/s$	1
	Lucas cell	Radon	pCi/L	1

Figure 5. Twelve Sensors Deployed During the Phase II Demonstration Test.

Supporting Electronics. Some special electrical design and fabrication was also required in construction of the demonstration unit. This effort consisted of building a power supply center for the sensors, a line voltage monitoring circuit and low cost, high quality temperature and humidity sensors. Figure 6 is a schematic of the power supply and support circuitry layouts illustrating the manner in which they interconnect to the controller unit and sensors.

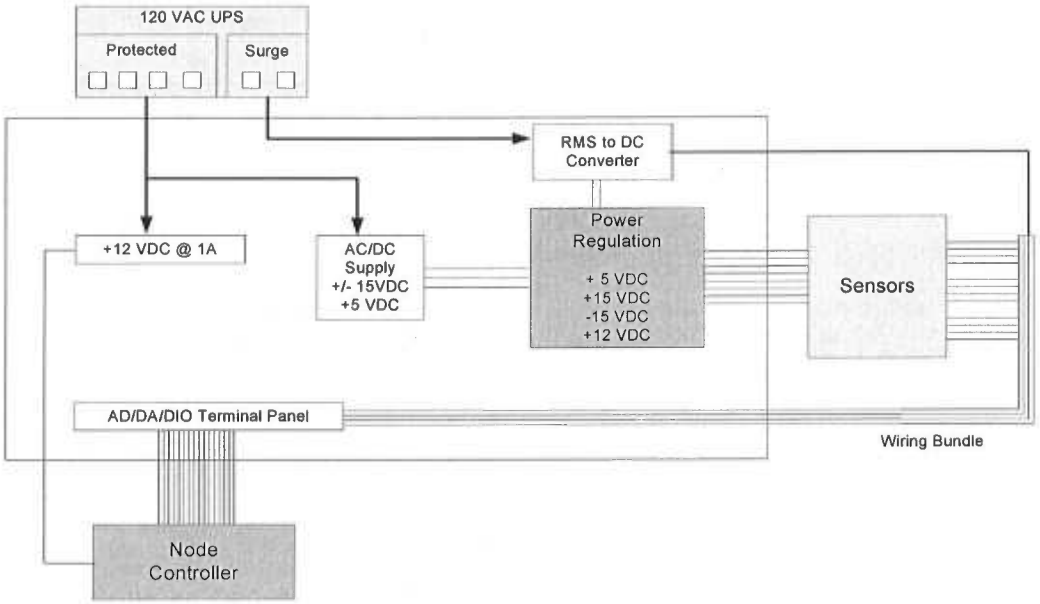

Figure 6. Power Supply, Regulation and Sensor Connections for the Demonstration Installation.

During this Phase II STTR project, the TOADS system was designed, constructed, installed, and tested to demonstrate automated remote monitoring at a DOE facility. TOADS has been demonstrated to successfully perform monitoring at the Oak Ridge National Laboratory (ORNL) DOSAR Calibration Laboratory with a variety of facility, environmental and radiation sensors providing continuous remote monitoring data to a central server. The monitoring data is transmitted to the TOADS server and is available to users at the TOADS website via the user interface described above. The demonstration system has been operating without failure since installation. During this Phase II demonstration, data was successfully collected and sent to the TOADS server 24 hours per day 7 days per week without any problems. The remainder of this report provides a detailed system description, and describes TOADS' installation and successful operation at ORNL.

2.0 Data Management

2.1 Database and Storage

A typical TOADS installation communicates with a number of critical sensors, facilitates remote access and services to other TOADS sites, as well as provides direct access to potentially a large group of users. The TOADS data management and storage implementation uses the open-source software MySQL RDBMS. MySQL provides a robust SQL database server with fast, multi-threaded, and multi-user capabilities suitable for a production system. The TOADS database contains information about the TOADS site, configuration, sensors, sensor configuration, and sensor data (measurements).

Additional properties and capabilities of the TOADS data management include:

- Information about each sensor (meta-data) and its relevant data is stored in a single database.
- The interface to the database provides convenient routines for simplifying access.
- Database access can be achieved through JDBC.
- The interface implements the flat/ring database table structure.

TOADS data management system uses a data warehousing approach using object databases and specialized datamarts. The core databases are implemented on the MySQL DBMS which stores the "measurement" data. Each set of "measurement" data is stored in a separate table specific to an instrument. Additionally, the database stores information on meta-data, instruments, users, site data, as well as integrated information from external data sources such as preexisting EM databases.

The TOADS datamarts store specialized bulk data (typically non-queried) such as streaming image/video, audio, engineering data (e.g., computational/analysis, CAD, etc.), inventory data, reports, and XML files.

2.2 Database Physical Structure

TOADS database physical structure is based on the following assumptions and requirements:

- The current stable version of MySQL is 4.0.15, is used in the deployment.
- MySQL's MyISAM tables is used, unless transaction-safe capability is required in which case InnoDB tables will be used.
- The database storage capacity is primarily determined by the sensor data tables. Assuming a 10-sensor TOADS site, each with interval of 1 minute data acquisition of approximately 512 bytes, held for a week: 10 * 7 * 1440 * 512 = 51,609,600 or approximately 52Mb of sensor data.

- Using ring tables for sensor data, the storage capacity is limitless.

2.3 Dynamic Tables

With the exception of SENSOR_<name>_DATA (and similar classes), the remaining classes in Figure 7 are "static" and are created at initialization (time t_0). The SENSOR_<name>_DATA and SENSOR_<name>_STATUS are dynamic classes (tables) representing a sensor defined during runtime. The columns of these tables define the sensor's data and status objects and the rows signify the collected information. These classes can be instantiated as either flat tables or a circular (ring) tables. The "wrapper" layer provides the needed facilitates and methods for creating/accessing the tables.

2.4 Database Logical Structure

The TOADS' data management component consists of a number of major classes. These classes and their broad relationships are shown in Figure 7. The class descriptions are listed in Figure 8.

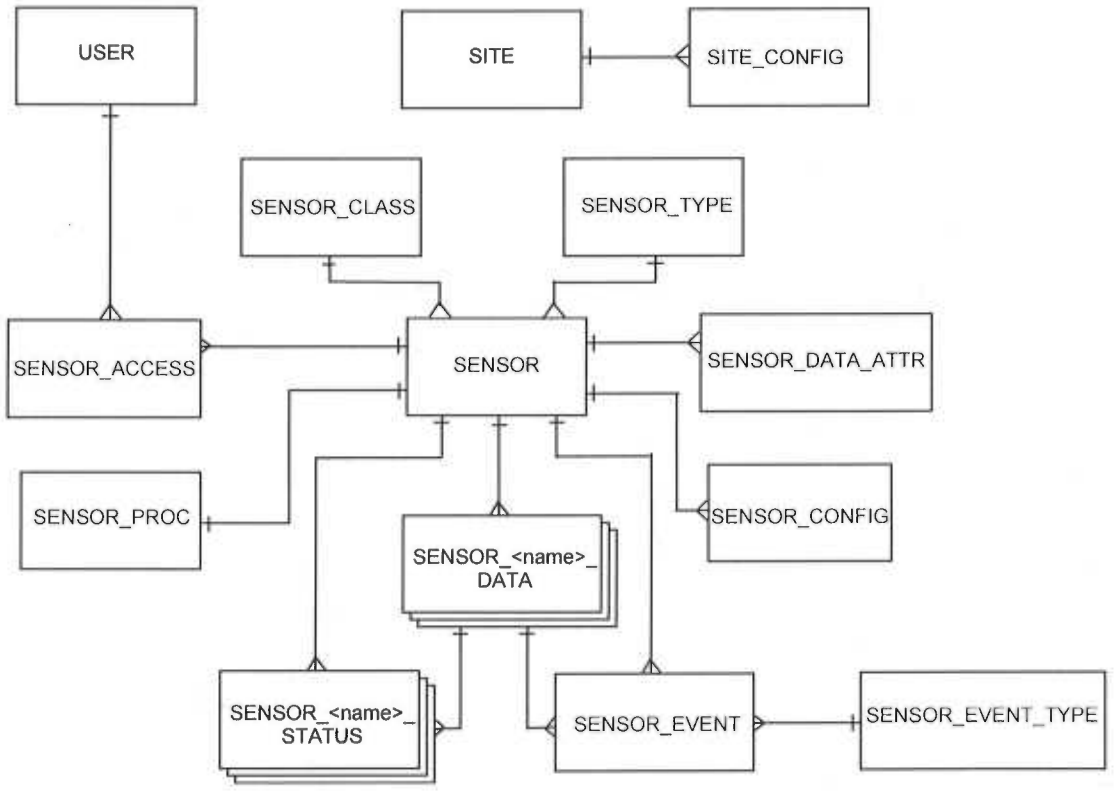

Figure 7. TOADS Database Entity Relationship Diagram. TOADS database schema approximately follows the Entity Relationship Diagram (ERD) shown here. For N:N relationships, intermediate "associative" tables are created.

SITE	Stores information on a TOADS site and its various properties.
SITE_CONFIG	Contains information on specific configuration of a TOADS site.
SITE_USER	Information on TOADS' users.
SENSOR	Information on TOADS' sensors (e.g., name, type, etc.).
SENSOR_CLASS	Stores information on classes of sensors.
SENSOR_TYPES	Contains information on types of sensors.
SENSOR_EVENTS	Stores information on all sensors' events.
SENSOR_EVENT_TYPES	Contains information on sensors' event types (e.g., "online", "offline", etc.).
SENSOR_DATA_ATTR	Stores information about each sensor's target measurement. The instances of this class, together with SENSOR, generate the class SENSOR_<name>_DATA.
SENSOR_ACCESS	Contains information on sensor access control.
SENSOR_PROC	Stores information on sensor processes (e.g., initialization, acquisition code, etc.).
SENSOR_CONFIG	Information on sensor configuration (e.g., acquisition type, on-demand or periodic, frequency, etc.)
SENSOR_<name>_DATA	Dynamic class generated from the combined instances of SENSOR + SENSOR_CONFIG + SENSOR_DATA_ATTR. This class stores measurement data and defines the characteristics of a "plug-n-play" sensor.
SENSOR_<name>_STATUS	Dynamic class generated from the combined instances of SENSOR + SENSOR_CONFIG + SENSOR_DATA_ATTR. This class stores sensor status information.

Figure 8. Class Descriptions.

The individual classes are further described below.

SITE Columns

The table SITE will have a single instance (row) storing basic information about the TOADS site. An implied relationship between this table and all the remaining tables is assumed (i.e., all tables belong to this site).

Name	Type	Null?	References	Description
id	u int4	N		The site id
name	varchar(32)	N		Name of the site
description	varchar(64)	Y		Description of the site
address	varchar(128)	Y		Site address
city	varchar(32)	Y		Site city
state	char(2)	Y		Site state
zip	varchar(10)	Y		Site zip code
latitude	double	Y		Site's GPS latitude
longitude	double	Y		Site's GPS longitude
altitude	double	Y		Site's altitude

SITE_CONFIG Columns

The table SITE_CONFIG stores information on site-specific configuration. Some features are reserved for future development.

Name	Type	Null?	References	Description
id	u int4	N		The configuration id
site_id	u int4	N	site.id	The site id
...	...	...		...

SITE_USER Columns

The table SITE_USER stores information on users of the TOADS site.

Name	Type	Null?	References	Description
id	u int4	N		User id
name	varchar(32)	N		User name
ip	varchar(32)	Y		User IP address

SENSOR_ACCESS Columns

The table SENSOR_ACCESS stores information on users access level for sensors.

Name	Type	Null?	References	Description
id	u int4	N		Access id
sensor_id	u int4	N	sensor.id	Sensor id
user_id	u int4	N	user.id	User id
read	boolean			Read access

write	boolean			Write access
update	boolean			Update access
delete	boolean			Delete access

SENSOR Columns

The table SENSOR catalogs basic information on the site's sensors.

Name	Type	Null?	References	Description
id	u int4	N		Sensor id; primary key; autoincrement
name	varchar(32)	N		Sensor name (same as transducer name); unique
gid	varchar(32)	N		Sensor global ID
location	varchar(64)	Y		Sensor location
description	varchar(64)	Y		Sensor description
site_id	u int2	Y	site.id	Sensor site id (see SITE)
type_id	u int2	Y	sensor_type.id	Sensor category type id (see SENSOR_TYPE)
class_id	u int2	Y	Sensor_class.id	Sensor class type id (see table SENSOR_CLASS)
manufact	varchar(32)	Y		Sensor manufacturer
model	varchar(32)	Y		Sensor model
sn	varchar(32)	Y		Sensor serial number
channels	u int2	Y		Number of data channels
who	varchar(32)	Y		Who created this sensor
ins_date	timestamp	Y		Sensor installation date

SENSOR_TYPE Columns

The cross-reference table SENSOR_TYPE stores information on sensor types.

Name	Type	Null?	References	Description
id	u int4	N		type id; auto-increment
name	varchar(32)	N		Type name (e.g., 'actuator')

SENSOR_CLASS Columns

The cross-reference table SENSOR_CLASS stores information on sensor classes.

Name	Type	Null?	References	Description
id	u int4	N		Class id; auto-increment
name	varchar(32)	N		Class name (e.g., 'radiation detector)

SENSOR_EVENT_TYPE Columns

The lookup table SENSOR_EVENT_TABLE stores information on sensor event types.

Name	Type	Null?	References	Description
id	u int4	N		Event id; auto-increment
name	Varchar(32)	N		Event name (e.g., 'alarm', 'up', 'down', etc.)

SENSOR_DATA_ATTR Columns

The table SENSOR_DATA_ATTR stores information about sensor data (channel measurement) columns. A single instance (row) of this table represents a single column in table SENSOR_<name>_DATA.

Name	Type	Null?	References	Description
sensor_id	u int4	N	sensor.id	The sensor id
data_no	u int2	N		The data column number.
data_name	varchar(32)	N		The data (measurement) name (e.g, 'dosage')
data_unit	varchar(64)	Y		Sensor data unit (e.g., 'mRads/hr')
data_type	varcahr(32)	N		Data type of this measurement (e.g., 'float')
nullable	boolean	Y		Whether null measurement is allowed
low_value	double	Y		Acceptable low value
high_value	double	Y		Acceptable high value
low_alarm	double	Y		Low value alarm setting
high_alarm	double	Y		High value alarm setting

SENSOR_CONFIG Columns

The table SENSOR_CONFIG stores information about sensor configuration.

Name	Type	Null?	References	Description
sensor_id	u int4	N	sensor.id	The sensor id
da_type	u int2	Y		data acquisition type (1=push, 2=pull)
samp_freq	float	Y		data sampling frequency (msec)
hi_value	float	Y		Measurement high value
low_value	float	Y		Measurement low value
hi_setpoint	float	Y		Alarm high setpoint
low_setpoint	float	Y		Alarm low setpoint

SENSOR_PROC Columns

The table SENSOR_PROC stores information on sensor processes.

Name	Type	Null?	References	Description
sensor_id	u int4	N	sensor.id	The sensor id
proc_name	varchar(32)	Y		Name of the process

proc_path	varchar(64)	Y		Directory path of the process

SENSOR_<name>_DATA Columns

This table contains measurement data for a sensor. This is a dynamic table whose columns are generated from the combined instances of SENSOR and SENSOR_DATA_ATTR. See for example, SENSOR_ ORNL-7735-ACV01_DATA.

Name	Type	Null ?	References	Description
id	u int4	N		The data instance id; auto-increment
sensor_id	u int4	N	sensor.id	The sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
…	…	…		…

SENSOR_<name>_STATUS Columns

This table contains status information on a sensor. This is a dynamic table whose name and columns are generated at runtime from the combined instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null ?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change data-time

SENSOR_ EVENT Columns

This table contains information on all sensors' event states.

Name	Type	Null?	References	Description
id	u int4	N		event id; autoincrement
sensor_id	u int4	N	sensor.id	The sensor id
data_id	u int4	Y	sensor_<name>_data.id	Data line causing the event
data_no	u int2	Y		Data number causing the event
type	u int2	N	sensor_event_type.id	event type (see SENSOR_EVENT_TYPE)
event_time	datatime	N		Alter datatime

SENSOR_ORNL-7735-ACV01_DATA Columns

The table SENSOR_ORNL-7735-ACV01_DATA contains measurement data for the ORNL-7735-ACV01 voltmeter. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
voltage_ac	float	Y		Voltage data

SENSOR_ORNL-7735-ACV01_STATUS Columns

This table stores status information on ORNL-7735-ACV01 sensor.

Name	Type	Null?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ORNL-7735-CON01_DATA Columns

The table SENSOR_ORNL-7735-CON01_DATA contains measurement data for the ORNL-7735-CON01 contact switch sensor. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
stat	text	Y		status of contact switch

SENSOR_ORNL-7735-CON01_STATUS Columns

This table stores status information on ORNL-7735-CON01 sensor.

Name	Type	Null?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ORNL-7735-CON02_DATA Columns

The table SENSOR_ORNL-7735-CON02_DATA contains measurement data for the ORNL-7735-CON02 contact switch sensor. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
stat	text	Y		status of contact switch

SENSOR_ORNL-7735-CON02_STATUS Columns

This table stores status information on ORNL-7735-CON02 sensor.

Name	Type	Null?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ORNL-7735-CON03_DATA Columns

The table SENSOR_ORNL-7735-CON03_DATA contains measurement data for the ORNL-7735-CON03 contact switch sensor. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude

longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
stat	text	Y		status of contact switch

SENSOR_ORNL-7735-CON03_STATUS Columns

This table stores status information on ORNL-7735-CON03 sensor.

Name	Type	Null ?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ ORNL-7735-CON04_DATA Columns

The table SENSOR_ORNL-7735-CON04_DATA contains measurement data for the ORNL-7735-CON04 contact switch sensor. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
stat	text	Y		status of contact switch

SENSOR_ORNL-7735-CON04_STATUS Columns

This table stores status information on ORNL-7735-CON04 sensor.

Name	Type	Null ?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ORNL-7735-GAM1_DATA Columns

The table SENSOR_ORNL-7735-GAM1_DATA contains measurement data for the ORNL-7735-GAM1 gamma exposure detector. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
exposure_rate	float	Y		Exposure rate in ur/h

SENSOR_ORNL-7735-GAM1_STATUS Columns

This table stores status information on ORNL-7735-GAM1 detector.

Name	Type	Null ?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ORNL-7735-GM01_DATA Columns

The table SENSOR_ORNL-7735-GM01_DATA contains measurement data for the ORNL-7735-GM01 gamma exposure detector. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
dose_rate	float	Y		Dose rate in mr/h

SENSOR_ORNL-7735-GM01_STATUS Columns

This table stores status information on ORNL-7735-GM01 detector.

Name	Type	Null ?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ORNL-7735-H001T_DATA Columns

The table SENSOR_ORNL-7735-H001T_DATA contains measurement data for the ORNL-7735-H001T hygrometer detector. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
rel_humidity	float	Y		Relative humidity in percent

SENSOR_ORNL-7735-H001T_STATUS Columns

This table stores status information on ORNL-7735-H001T detector.

Name	Type	Null ?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ORNL-7735-MOT01_DATA Columns

The table SENSOR_ORNL-7735-MOT01_DATA contains measurement data for the ORNL-7735-MOT01 motion sensor. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time

latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
stat	text	Y		Data status

SENSOR_ORNL-7735-MOT01_STATUS Columns

This table stores status information on ORNL-7735-MOT01 sensor.

Name	Type	Null ?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ORNL-7735-N01_DATA Columns

The table SENSOR_ORNL-7735-N01_DATA contains measurement data for the ORNL-7735-N01 ZNS sensor. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
neutron_fluence	float	Y		Neutron fluence in c/cm2/s

SENSOR_ORNL-7735-N01_STATUS Columns

This table stores status information on ORNL-7735-N01 sensor.

Name	Type	Null ?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ORNL-7735-RN01_DATA Columns

The table SENSOR_ORNL-7735-RN01_DATA contains measurement data for the ORNL-7735-RN01 Lucas cell detector. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
rn-222	float	Y		Rn-222 data in pci/l

SENSOR_ORNL-7735-RN01_STATUS Columns

This table stores status information on ORNL-7735-RN01 sensor.

Name	Type	Null ?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

SENSOR_ORNL-7735-T001T_DATA Columns

The table SENSOR_ORNL-7735-T001T_DATA contains measurement data for the ORNL-7735-T001T temperature sensor. This dynamic table is generated from the instances of SENSOR and SENSOR_DATA_ATTR.

Name	Type	Null?	References	Description
id	u int4	N		Data (measurement) id; auto-increment
sensor_id	u int4	N	sensor.id	Sensor id
time	datetime	N		Data measurement date/time
latitude	double	Y		Data's GPS latitude
longitude	double	Y		Data's GPS longitude
status	u int4	Y		Status of data
temperature	float	Y		Temperature in degree C

SENSOR_ORNL-7735-T001T_STATUS Columns

This table stores status information on ORNL-7735-T001T sensor.

Name	Type	Null ?	References	Description
id	u int4	N		status id; autoincrement
sensor_id	u int4	N	sensor.id	Sensor id
status	u int2	N		Sensor status 0=down;1=up.
status_time	datatime	N		Status-change datatime

3.0 TOADS Services

The TOADS Services are a collection of web-based application interfaces that reside in the TOADS server. They provide various services to the other producer and consumer members. The external members may be producer/consumer applications, other TOADS sites, external devices, as well as external data sources as illustrated in Figure 9.

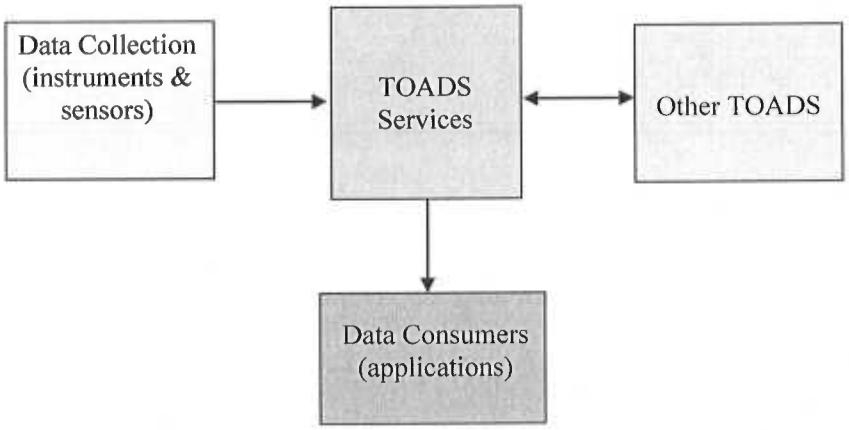

Figure 9. The Relationship between TOADS Services and Other TOADS Components.

The TOADS Services Layer is a middleware software that provides standards-based communications between the sensor(s), other TOADS sites, and data consumers. The services are implemented as Web services and utilize an XML schema and service specifications (future implementation may include GIS Consortium OGC standard). In addition to the services clients shown in Figure 9, external data sources and databases are supported by the Services middleware layer.

Note that the block diagram shown in Figure 9 is a logical description only. The TOADS Services Layer is not truly a separate component but, in practice, is implemented through Web service software running on the TOADS server as well as external sources (for example, external application server or other TOADS sites.)

3.1 Sensor Configuration and Initiation

The TOADS services can recognize a new sensor configuration and initiation using a *RegisterSensor* request sent by its data acquisition client. An example of a *RegisterSensor* XML packet is shown in Figure 10.

```
<?xml version="1.0"?>
<RegisterSensor><SENSORID>ORNL-7735-H001T</SENSORID>
<CLASS>ENVIRONMENTAL</CLASS>
<TYPE>HYGROMETER</TYPE>
<MISCLOCATION>CONTROL ROOM</MISCLOCATION>
<MANUFACTURER>HONEYWELL</MANUFACTURER>
<MODEL>IH-3610-2</MODEL>
<SERIALNUMBER>X10H001T</SERIALNUMBER>
<CALIBRATIONDATE>04/01/2004</CALIBRATIONDATE>
<SITEADDRESS1>BUILDING 7735 (RADCAL)</SITEADDRESS1>
<SITEADDRESS2>OAK RIDGE NATIONAL LABORATORY</SITEADDRESS2>
<SITEADDRESS3>BETHEL VALLEY ROAD</SITEADDRESS3>
<CITY>OAK RIDGE</CITY>
<STATE>TN</STATE>
<ZIP>37831</ZIP>
<DataWidth>1</DataWidth>
<DataLength>1</DataLength>
<DataFormat>float</DataFormat>
<DataTitles>RELHUMIDITY</DataTitles>
<DataUnits gooz="me">%RH</DataUnits>
</RegisterSensor>
<RegisterSensor>
```

Figure 10. RegisterSensor Request – An Example of RegisterSensor XML Packet.

Figure 11 illustrates the activities for sensor configuration and initiation at TOADS.

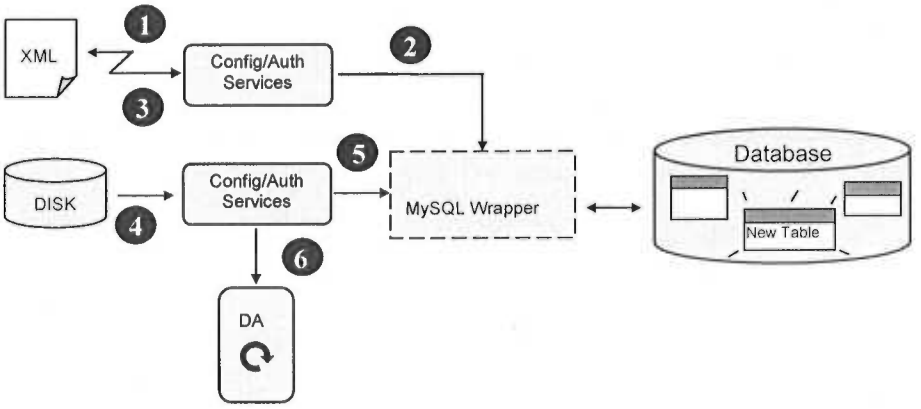

Figure 11. TOADS Sensor Initiation – TOADS Activities During Sensor Initiation and Configuration.

These activities are described as a series of 6 steps:

1. Request for a new sensor (or an existing sensor) configuration is sent by a TOADS' client.
2. The Config/Auth Service creates the new sensor table(s) (or updates the existing sensor tables) and updates the relevant met-data tables.
3. The request's success or failure is acknowledged.
4. The Config/Auth Service reads the necessary system parameters (e.g., directory path) and sensor meta-data.
5. The Config/Auth Service writes/updates the appropriate tables.
6. The Config/Auth Service validates the sensor's data acquisition.

3.2 On-Demand and Periodic Data Upload

The main data flow to TOADS is measurement data representing routine sensor/instrument readings that are sent to the TOADS server and archived in the database. Data upload is recognized by TOADS services when a *DataPacket* request is received from its data acquisition client. Figure 12 shows a sample *DataPacket* XML message.

```
<?xml version="1.0"?>
<DataPacket>
<SensorID>ORNL-7735-T001T</SensorID>
<Date>08/04/2004</Date>
<Time>16:55:13</Time>
<TEMPERATURE>2.00e+001</TEMPERATURE>
```

Figure 12. Data Packet Request – An Example of DataPacket XML Request.

Figure 13 illustrates the activities for on-demand or periodic data upload to TOADS by its data acquisition client.

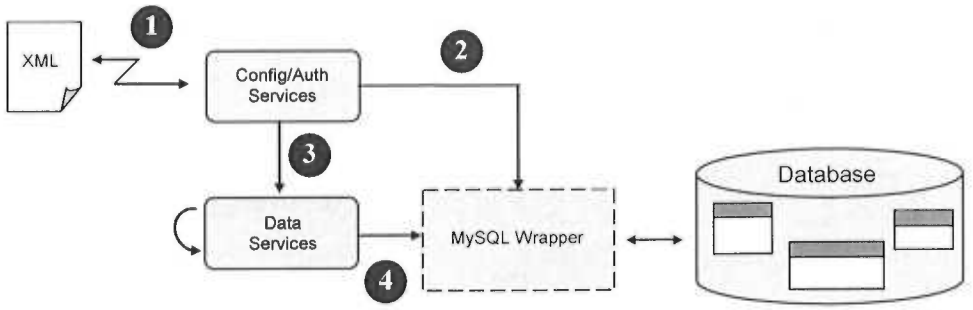

Figure 13. TOADS Data Upload – TOADS Activities During Sensor Data Upload.

The data upload activities are described as a series of 4 steps:

1. Request for on-demand or periodic upload for a sensor is sent by the TOADS client.
2. The Config/Auth Service validates the request and updates the appropriate tables (e.g., sensor fields, frequency).
3. The Config/Auth Service initiates the Data Service upload activity.
4. The Data Service registers its activity in the database and stores uploaded data.

3.3 Sensor Monitoring

The TOADS server provides specialized services for user interfaces and client application. Figure 14 illustrates the activities involved for direct monitoring of a TOADS site's sensor data initiated by an authorized remote user or client.

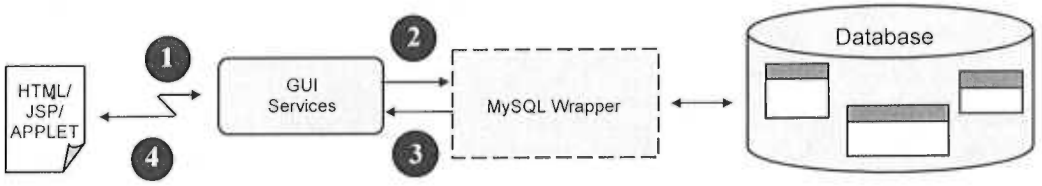

Figure 14. TOADS Monitoring – TOADS Activities During Client Interface.

The client interface activities are described as a series of 4 steps:

1. Request for sensor monitoring is sent by a client (HTML/JSP/Java Applet).

2. The GUI Service updates (registers) the appropriate database tables.
3. The GUI Service reads the necessary parameters from the database.
4. The GUI Service presents the sensor monitoring activity.

3.4 Event Handling

Events are conditions that require notification such as alarms and alerts. TOADS generates and handles various events based on certain predefined conditions as well as dynamically configurable parameters. All events are logged in the database prior to dispatching. In general, TOADS' event processing falls into three broad categories: **system** events, **data** events, and **user** generated events.

- System events are caused by hardware failures or malfunction, online/offline conditions, process failures, runtime errors, and system checkpoints.
- Data events are generated when measurement values exceed predefined high/low alarm set points, out-of-range thresholds, read errors, and other data validation errors.
- User events are generated for login/logout activities, access violations, etc.

4.0 Interface

Section 4.1 describes TOADS' user interface, and Section 4.2 provides details on the Data Acquisition Client.

4.1 User Interface

TOADS provides a highly flexible and intuitive graphical user interface (GUI) that can be configured to meet virtually any remote facility/site management need. Depending on permission levels, users can access and control TOADS via Internet, Intranet, wireless, or dialup access and receive on-line surveillance data and customized reports tailored to their specific needs. The user interface is a web-based application that presents user-requested real-time information ranging from raw data and visual images to textual and graphical forms and analyses. Figure 15 shows the TOADS' top-level user interface "Monitor Window" invoked as a Java Applet.

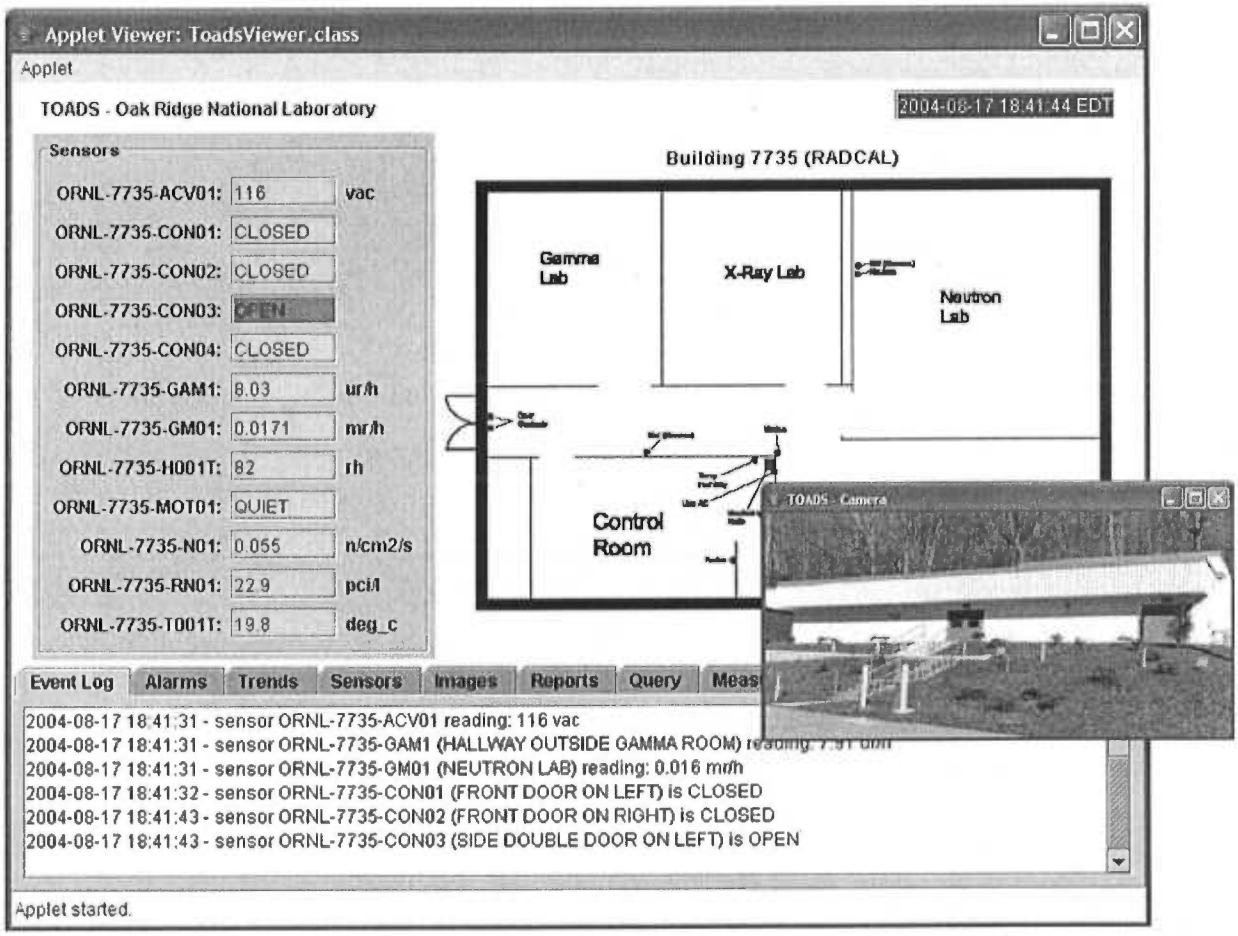

Figure 15. TOADS User Interface. The top-level view of the TOADS Java-based graphical user interface is shown. In this view, the site's sensors' status and readings are continuously monitored in near real-time. The sensor locations are mapped to the site's layout plan along with available surveillance images. Additional features provided at the lower part of the view (event logs shown above) assist in close and all-around monitoring of the site.

The user interface is written entirely in Java and works on virtually all major platforms. It has been tested on Unix/Linux and Windows® operating systems.

4.1.1 Alerts and Alarms

In TOADS, alert and alarm events are either system, data, or user generated conditions (see Section 3.4, Event Handling). Alerts include such mundane conditions as *sensor online* and *sensor offline*. Alarms on the other hand include the all-important "sensor reading out-of-range" events or similar user/data specified conditions. When TOADS detects a sensor reading that exceeds some preset alarm value, the sensor is said to be in an alarm state and an alarm event is sent to the client(s). When an alert/alarm event is detected, they are logged in the database prior to dispatching the information, including forwarding the alarm/event to registered interested users. An example of a sensor alert (open contact switch for sensor ORNL-7735-CON03) dispatched to a user is shown in Figure 16. Depending on the configuration, acknowledgement of an event may be required, which is also logged in the database.

Figure 16. TOADS Alert Screen – An Example of an Open-Door Sensor Alert.

4.1.2 Trend Charts

From the TOADS Monitor Window, the user can invoke trend charts for important and critical sensors. These charts display the most recent data readings for the specific sensors in a real-time fashion. The trend charts may vary depending on the system configuration and may be customized to meet special needs. The charts contain hotspots where mouse movement on these locations will reveal additional information. Clicking the hotspots will pop-up new windows with detailed data-point values. Figure 17 shows a sample trend chart for sensor "ORNL-7735-RN01".

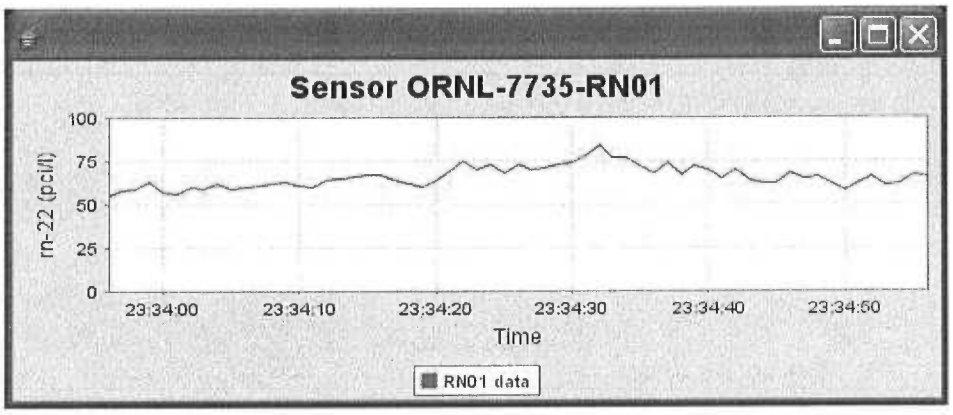

Figure 17. Trend Chart – A Sample Trend Chart for ORNL-7735-RN01 Sensor.

4.1.3 Ad Hoc Query

The TOADS client interface allows users to construct ad-hoc queries as well as reports and submit the request to the TOADS server for processing. The user constructed queries can include request for both data and meta-data information. An example of a database query for obtaining sensor information is shown in Figure 18.

ToadsQuery

```
select s.name,model,manufact,data_name,t.name,data_unit from sensor s,sensor_data_attr a,sensor_type t where s.id=sensor_id and t.id=s.type_id;
```

name	model	manufact	data_name	name	data_unit
ORNL-7735-ACV01		ORNL	voltage_ac	VOLTMETER	vac
ORNL-7735-CON01			status	CONTACT	
ORNL-7735-CON02			status	CONTACT	
ORNL-7735-CON03			status	CONTACT	
ORNL-7735-CON04			status	CONTACT	
ORNL-7735-GAM1	ANALYST, 489-55	BICRON, VICTOREEN	exposurerate	GAMMA EXPOSURE	ur/h
ORNL-7735-GM01	ANALYST, GMSW	BICRON	doserate	GAMMA EXPOSURE	mr/h
ORNL-7735-H001T	IH-3610-2	HONEYWELL	relhumidity	HYGROMETER	rh
ORNL-7735-MOT01			status	INFRARED MOTION D...	
ORNL-7735-N01	ANALYST, 2101	BICRON, ORNL	neutronfluence	ZNS	n/cm2/s
ORNL-7735-RN01	ANALYST, 300A	BICRON, PYLON	rn-222	LUCAS CELL	pci/l
ORNL-7735-T001T		ORNL	temperature	TEMPERATURE	deg_c

Figure 18. TOADS Ad Hoc Query – An Example of a TOADS Database Query for Information on All Sensors.

4.2 Data Acquisition Client

Similar to the TOADS user interface, the data acquisition application, **toadsclnt**, is a web-based application program that transmits data and meta-data information from a broad range of instruments and sensors to the TOADS server. The TOADS Data Acquisition Client is illustrated in Figure 19.

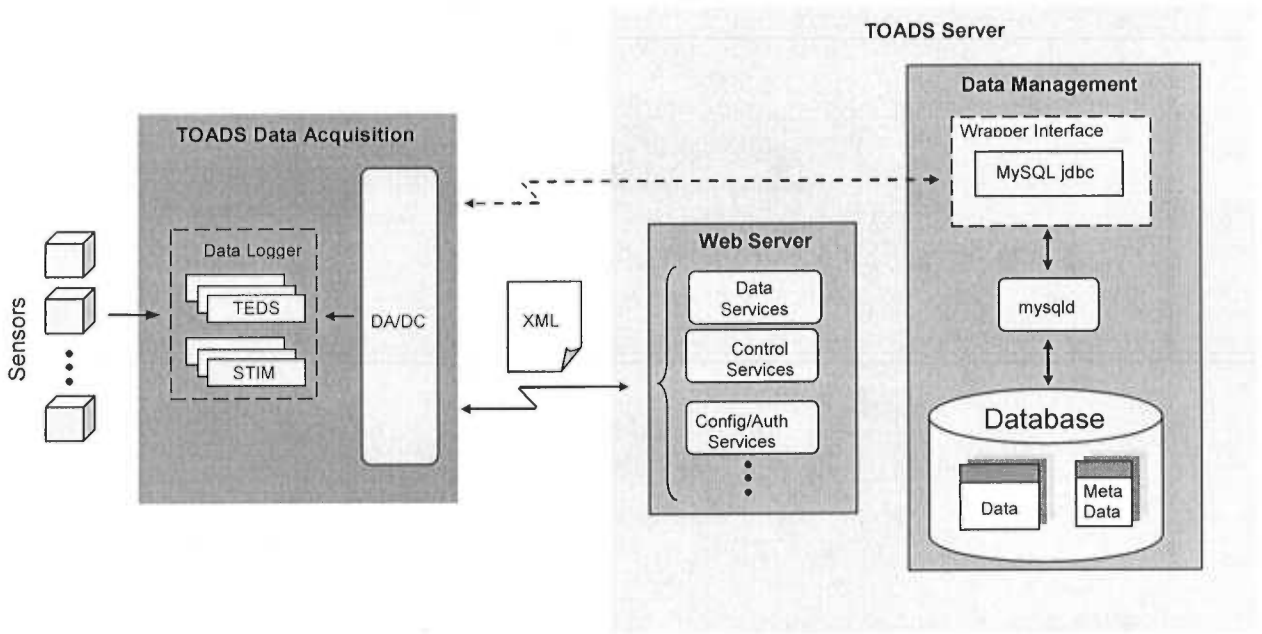

Figure 19. TOADS Data Acquisition Client – A web-based application client that transmits sensor data to the TOADS server.

The **toadsclnt** client program obtains both sensor meta-data and data from data logger(s), constructs the *RegisterSensor* and *DataPacket* XML messages, and transmits ("pushes") the packets using the HTTP protocol.

The **toadsclnt** application is written in C and runs in a variety of platforms. Its command syntax is:

toadsclnt [*options*]

where the argument *options* are:

-d *dir*	Target directory (default current directory)
-h	List help information
-p *number*	Port to connect (default 80)
-s *server*	TOADS host server (default 'toads.ornl.gov')
-t msec	Time delay between XML file transmit (default 1 sec)
-T msec	Time delay between directory lookup (default 5 secs)
-V	Run verbose (print status and informational messages)
-v	Show version

5.0 Remote Monitoring Node

The remote monitoring node consists of a controller, sensors, supporting electronics, signal conditioning and processing components, uninterruptible power supply (UPS), AC to DC conversion power supplies and associated support hardware and cabling needed to install the system. The node controller consists of an embedded processing unit with analog to digital (A/D), digital to analog (D/A), digital input/output (DIO) as well as high speed pulse counting capability. The unit includes a variety of standard input/output (I/O) mechanisms such as universal serial bus (USB), 10BaseT Ethernet and RS-232 communications ports.

A generalized schematic of a remote monitoring system or node is shown in Figure 20. The diagram shows the primary components incorporated into each monitoring node including the node controller, power supply, signal conditioning circuitry plus a variety of sensors for monitoring real-world parameters. The first operational node controller is shown in Figure 21. Figure 22 shows the unit with the top cover removed. Figure 23 displays an expanded schematic of the node controller including specific components used in the unit.

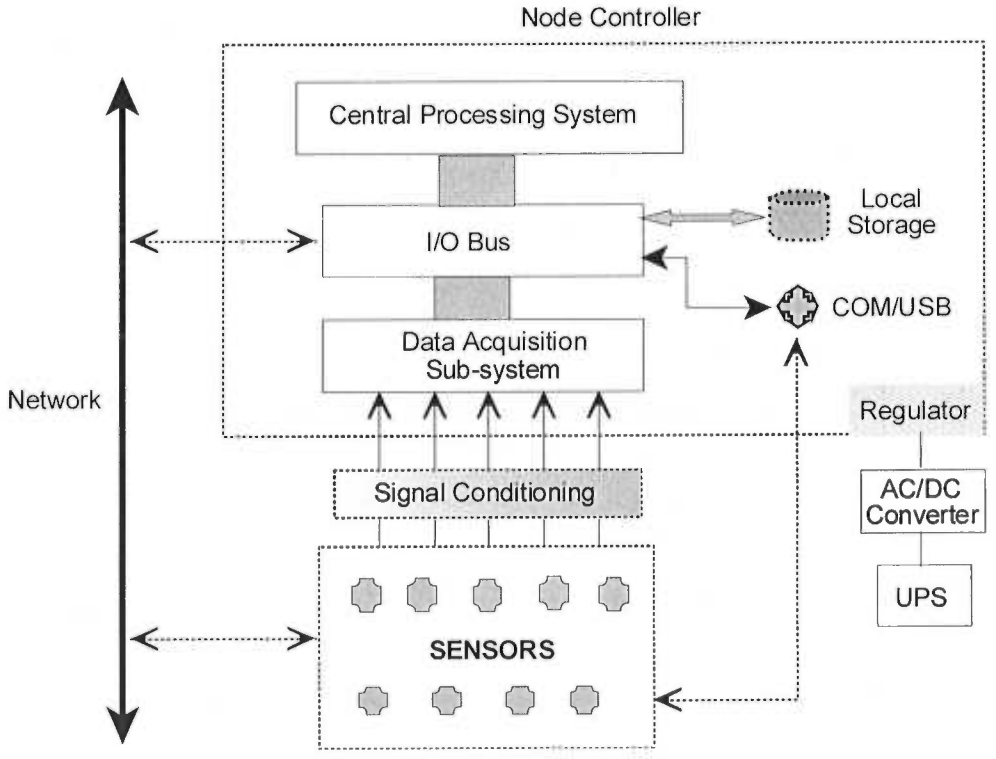

Figure 20. Generalized schematic of a TOADS remote monitoring node.

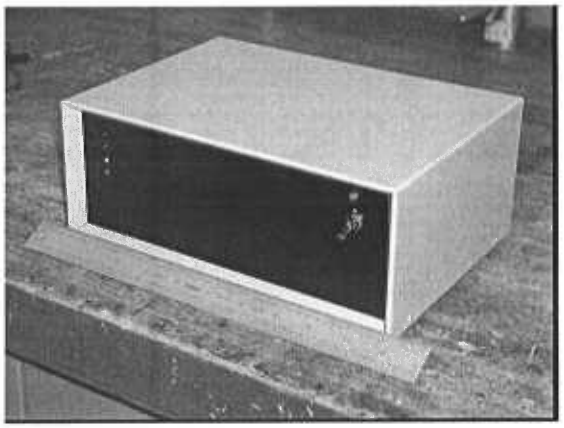

Figure 21. A remote monitoring node.

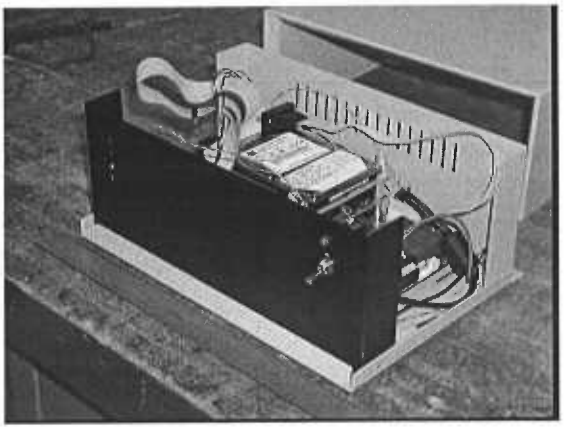

Figure 22. Node controller with the top cover removed.

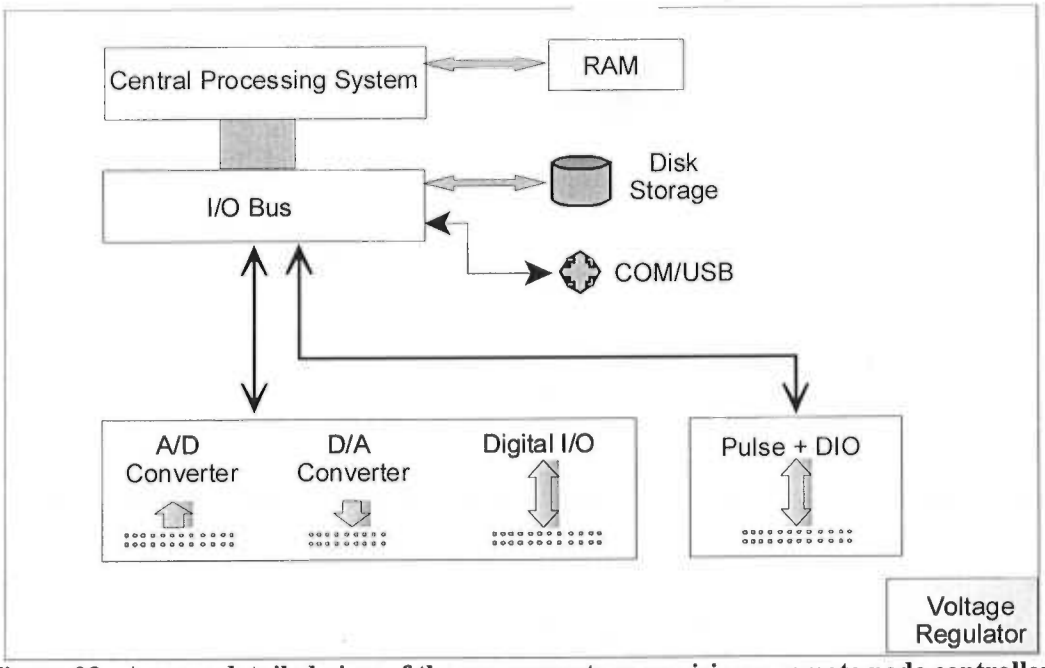

Figure 23. A more detailed view of the components comprising a remote node controller.

The controller is driven by an embedded EBX board with a high speed, fanless x86 compatible processor, 128 MB of random access memory, numerous IO ports, A/D converter, D/A converter and digital IO/control as well as a PC/104 ISA stack-through expansion bus. An additional ten channel high-speed counter/timer is installed on the bus and a large capacity IDE drive is incorporated for housing the operating system, drivers and local data storage. The system is built upon the Windows XP operating system; however, all code has been developed using C and C++ languages with common Win32 and Linux libraries to facilitate easy conversion to a Linux platform. The mature system will run equally well using either an embedded Linux or XP design.

The software architecture used within each field node is shown schematically in Figure 24. On activation the system performs the following actions.

1) Initialization of all local sensors configured for input.
2) Registration of sensors with the server.
3) System searches for the central TOADS server.
4) Once the TOADS server is found registration packets are transmitted containing detailed information about each sensor as-well-as indicating they are now "online".
5) Each driver begins independent operation monitoring its input and reporting the results.

Each process, or thread, writes measurement results to a local outbox at specified intervals while also writing summary results to a local data repository. A message transport process continually runs in the background watching the system outbox and, on seeing new messages, sends them to the central server.

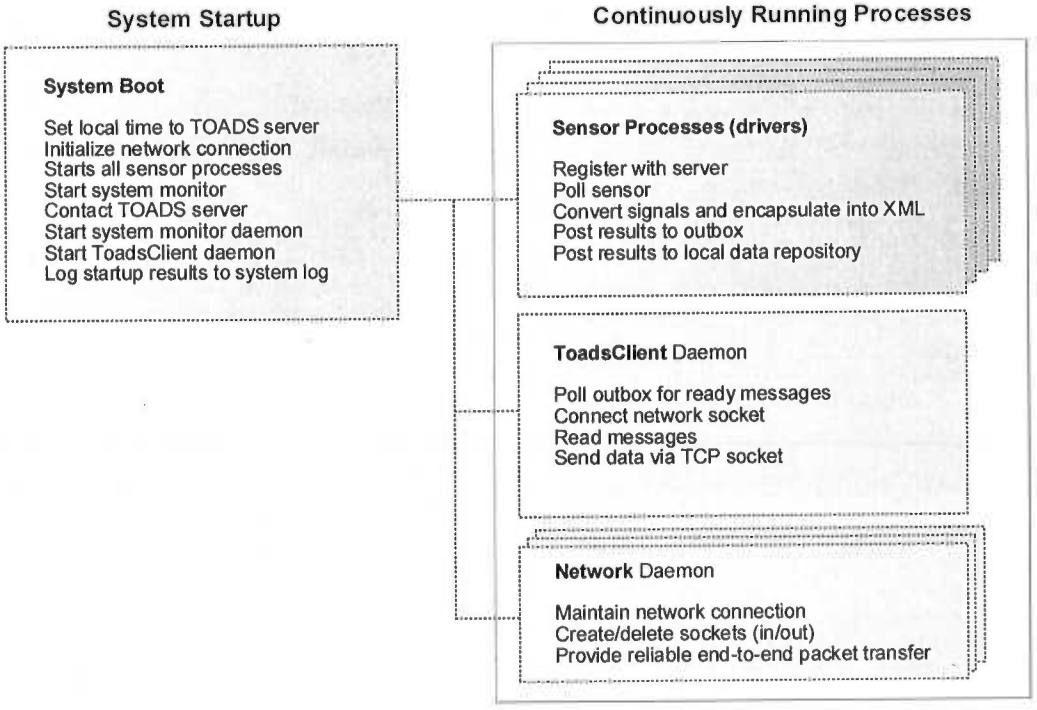

Figure 24. Software architecture used on remote monitoring nodes.

The system's architecture is both versatile and extensible. Since each driver operates independently, new sensor types can be added without any need to modify the core components. The small set of data packaging parameters means that sensor drivers can be developed without any knowledge of the code used for any other drivers being used. Each additional driver must only address where and how to post registration and data messages on the system.

5.1 Sensors

The remote monitoring node was designed for compatibility with most sensor types. A system architecture was designed that is compatible with a wide range of auxiliary hardware and that provides relatively easy development of sensor drivers. A variety of sensors were deployed to demonstrate the ability of the system to meet performance requirements. The sensors used included facility, environmental and radiological monitoring devices as summarized in Figure 25.

Sensor Class	Sensor Type	Quantity Measured	Reporting Units	Quantity Deployed
Facility	Door contact	Door status	Open/closed	4
	Motion sensor	Motion	Motion/quiet	1
	Line A/C condition	A/C voltage at facility	Volts AC	1
Environmental	Thermocouple	Temp	Degrees C	1
	Hygrometer	Relative Humidity	RH%	1
Radiological	NaI(Tl)	Gamma Exposure Rate	uR/h	1
	GM tube	Beta/Gamma Levels	mrem/h	1
	ZnS(Ag)	Fast neutron	$n_f/cm^2/s$	1
	Lucas cell	Radon	pCi/L	1
			Total Number of Sensors Deployed	12

Figure 25. Sensor types used for the Phase II TOADS demonstration.

The radiation sensors deployed in the demonstration system are pictured in Figure 26 through Figure 28. A more detailed description of each sensor is provided in Figure 29.

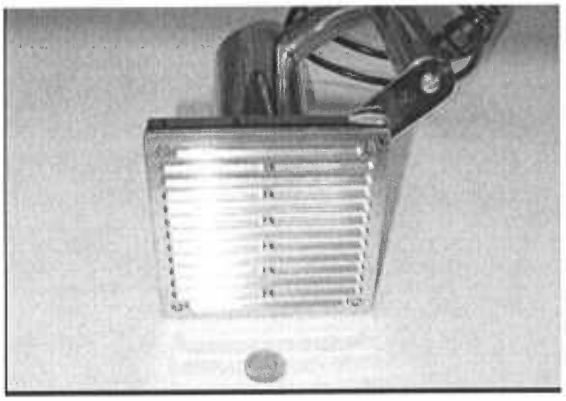

Figure 26. ZnS(Ag) sensor with a 100 cm^2 surface area.

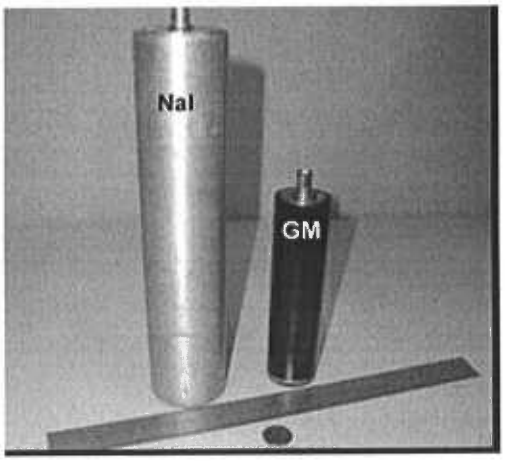

Figure 27. NaI(Tl) and Geiger-Mueller (GM) sensors.

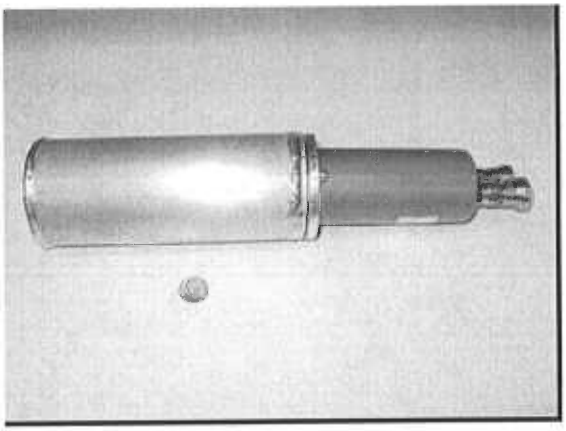

Figure 28. Lucas cell (Rn-222) sensor.

Sensor Type	Manufacturer	Model	Serial Number	TOADS ID Number	Measurement Type for Demonstration
Hygrometer	Honeywell	IH-3610-2	X10H001T	ORNL-7735-H001T	Relative Humidity
Thermometer	Analog Devices	AD637	X10T001T	ORNL-7735-T001T	Temperature (°C)
Door contact (magnetic)	Sentrol	2700	n/a	ORNL-7735-CON01 to ORNL-7735-CON04	Door contact
Motion Sensor (infrared)	Digital Security Controls	Bravo	n/a	ORNL-7735-MOT01	Motion sensor
NaI(Tl)	Victoreen	489-55	14110	ORNL-7735-GAM1	Gamma exposure rate
	Bicron	Analyst	B119H		
Geiger Mueller	Bicron	GMSW	A552U	ORNL-7735-GM01	Gamma exposure rate
	Bicron	Analyst	B762U		
Lucas cell	Pylon	300A	1472	ORNL-7735-RN01	Radon concentration in air
	Bicron	Analyst	A184X		
ZnS(Ag)	ORNL	2101	3047	ORNL-7735-N01	Fast neutron fluence rate
	Bicron	Analyst	B763H		
RMS to DC converter	ORNL	n/a	n/a	ORNL-7735-ACV01	Facility line voltage monitor

Figure 29. Sensors Deployed for Demonstration at the ORNL DOSAR Calibration Laboratory

5.2 Supporting Electronics

Some special electrical design and fabrication was required in construction of the demonstration unit. This effort consisted of building a power supply center for the sensors, a line voltage monitoring circuit and low cost, high quality temperature and humidity sensors.

Figure 30 is a schematic of the power supply and support circuitry layouts illustrating the manner in which they interconnect to the controller unit and sensors. An uninterruptible power supply was used to provide backup power and clean power to the remaining circuitry. A multi-output power supply, coupled with various voltage regulation components, was employed to provide power for all sensors as well as the monitoring node itself. An AC-RMS to DC voltage converter was built in order to provide a direct measure of the line voltage inside the facility. The circuit for this component is shown in Figure 32. The input voltage (AC) is converted to a linear output (DC) and subsequently fed into the ADC of the monitoring node to provide an accurate reading of the line voltage level.

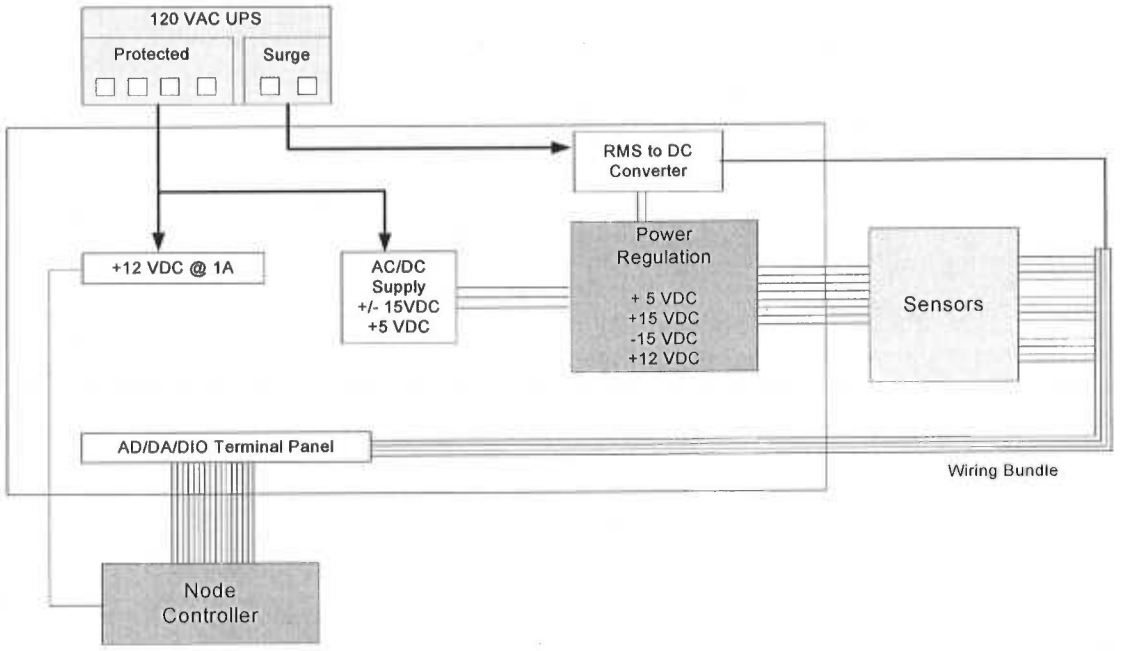

Figure 30. Power supply, regulation and sensor connections for the demonstration installation.

High quality temperature and humidity sensors can be relatively expensive with digitally controlled hygrometers running as high as $300 for a single unit. Consequently, chip components were selected to serve these functions based on cost, flexibility of application, and the fact that the controller unit is capable of providing direct ADC input from the sensors. The

temperature sensor selected was an Analog Devices AD22100 package. A Honeywell HIH-3610 was selected for the hygrometer function. Both devices are relatively inexpensive (less than $30 for the HIH-3610 and less than $10 for the AD22100) and are easily interfaced with the monitoring node input.

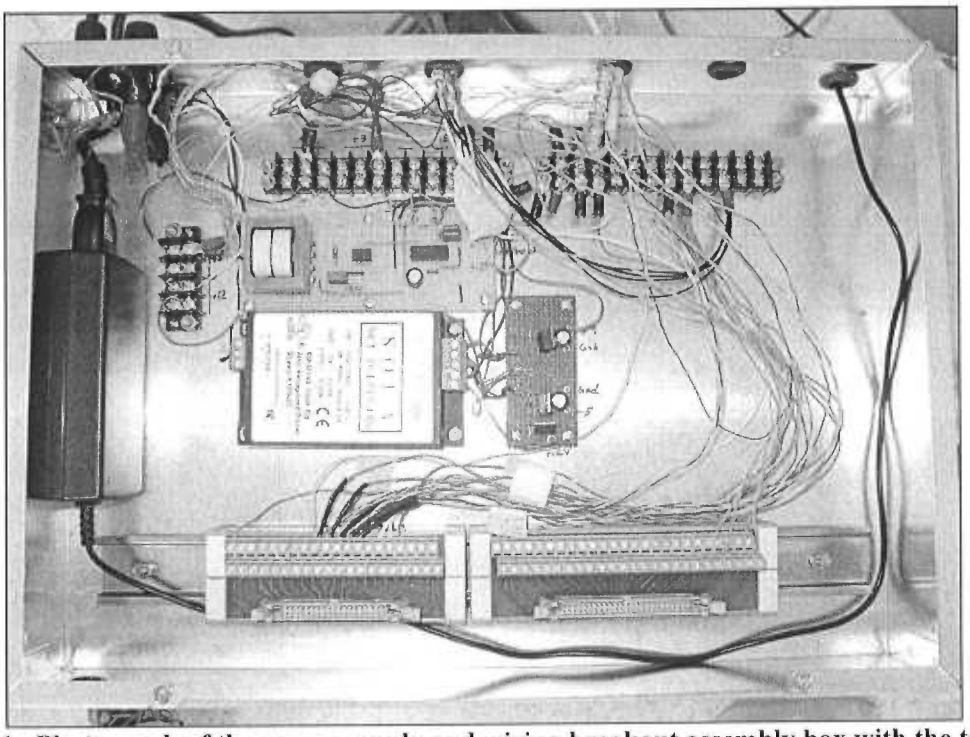

Figure 31. Photograph of the power supply and wiring breakout assembly box with the top cover removed.

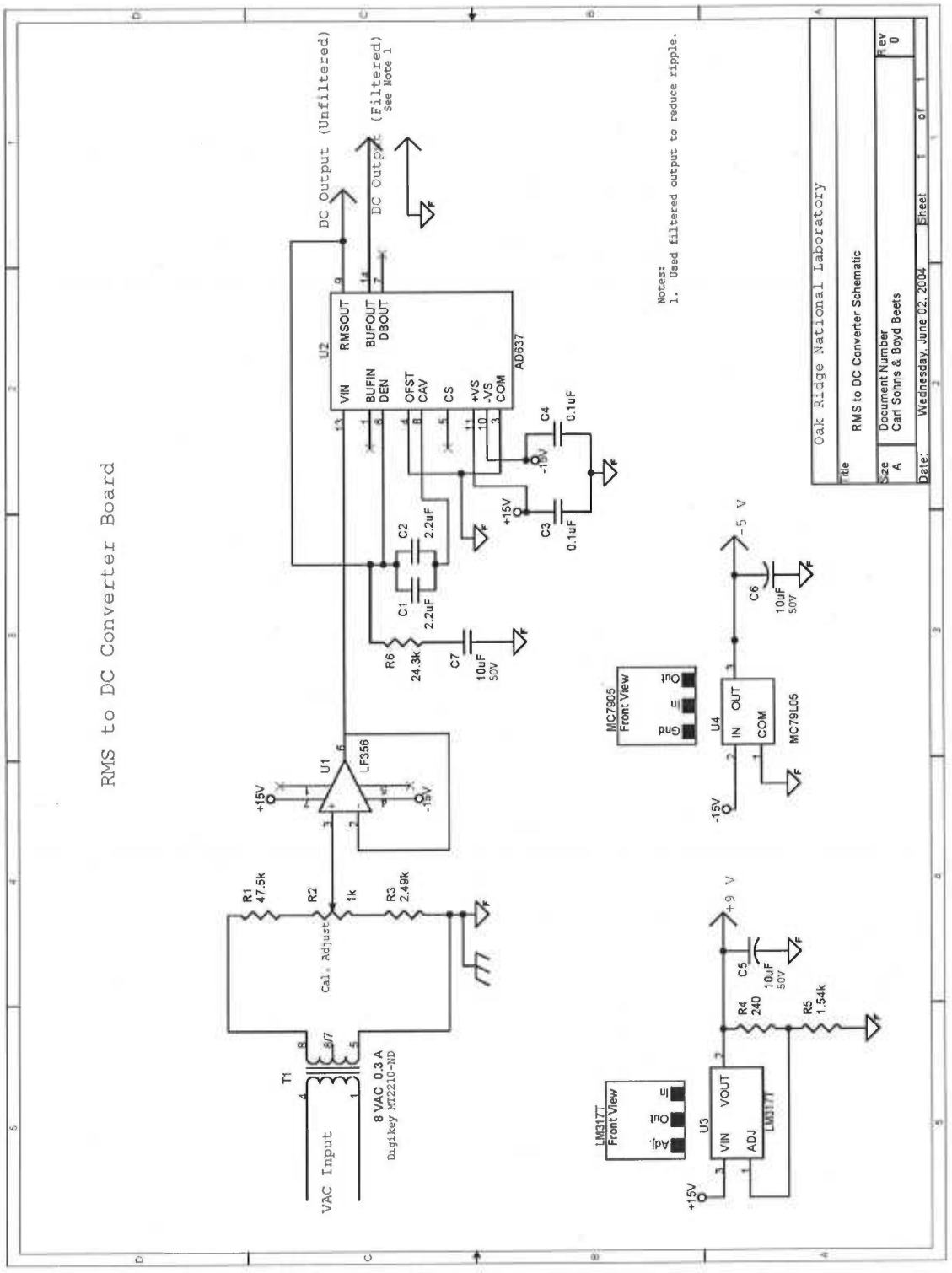

Figure 32. Line AC voltage circuit for monitoring facility voltage levels.

5.3 Software Drivers

Software drivers were written for each type of sensor used in the demonstration. All code was written using either C or C++ compilers to ensure ease of translation to other platforms. Each driver is loaded at system startup as an independent process and immediately registers its assigned device with the TOADS server by writing registration packets to the system outbox. The process then begins collecting data from its assigned device, converting the signal to the desired units of measure and then posting the results as XML data packets into the system outbox. A copy of the data is then written to the local storage system as a backup. It should be noted that, although the development effort was focused on providing interfaces for the sensors listed above, the drivers were actually written in such a way that they could be used with a variety of other sensor types if needed.

Each driver uses an initialization file for information about the sensor to which it is assigned. This file, along with a general system initialization file, provides the driver all that it needs to register, connect-to and monitor the sensor. An example of a sensor initialization file is shown in Figure 33. This example shows the types of information provided for the sensor driver including device type, physical connection parameters, update intervals, conversion equations, alarm parameters and units of measure. This information is assembled by the driver to form both the registration messages as well as the data packets themselves.

```
MasterIni= master.ini

// Address info for Quartz input
BaseAddress= 0x300h
IRQ = 7
InputChannel = 2

// Sensor Data
SensorID = ORNL-7735-GAM1
SensorDesc = NaI 1.25 x 1.5 inch
NickName = NaI-1    // for local diplay only
Class = Radiation
Type = Gamma Exposure
Manufacturer = Bicron, Victoreen
Model = Analyst, 489-55
SerialNumber = B119H, 1410
CalibrationDate = 04/05/2004
PhotoUUE = ORNL-7735-GAM1.uue    //uuencoded jpg file (sensor photo)

// supplemental sensor location info
MiscLocation = Hallway Outside Gamma Room
Latitude =
Longitude =
Elevation =   // (feet)

//DataFormat
DataTitles = ExposureRate  // (e.g., DoseRate)
DataUnits  = uR/h          // (e.g., mrem/hr)

// Data Conversion | result = (pulses/sec * m1) + c1
m1 = 0.15
c1 = 0

// Processing parameters
UpdateInterval = 4000 //read result (milliseconds)
DisplayInterval = 30  // output to screen (seconds)
SaveInterval = 3600  //average and save to Outbox (seconds)

// alarm High setpoints.  (these are "converted" values
// Note that BOTH setpoints must be configured if EITHER are to be used
AlarmSetpointHighEnable = 1             // 0 = No, 1 = Yes
AlarmSetpointHigh1 = 800
AlarmSetpointHigh1Interval = 600 // seconds
AlarmSetpointHigh2 = 2000
AlarmSetpointHigh2Interval = 120 // seconds

// alarm Low setpoints.
// Note that BOTH setpoints must be configured if EITHER are to be used
AlarmSetpointLowEnable = 1       // 0 = No, 1 = Yes
AlarmSetpointLow1 = 5
AlarmSetpointLow1Interval = 600         // seconds
AlarmSetpointLow2 = 2
AlarmSetpointLow2Interval = 300         // seconds
```

Figure 33. Example initialization file for a sensor driver.

The first step performed by each driver at system startup is the posting of a registration packet to the system outbox. This packet provides a detailed listing of information about the sensor including a unique identifier, physical description and location, the type of measurement, the reporting units, and data format. Some types of sensors do not need to register as much information as others so each message only includes data fields that are applicable to that particular sensor. A registration packet shown in Figure 34 illustrates the type of information that is submitted to the host server each time this particular sensor comes online.

```
<?xml version="1.0"?>
<RegisterSensor>
        <SensorID>7735-GDR01</SensorID>
        <Class>Radiation</Class>
        <Type>Gamma Exposure</Type>
        <MiscLocation>Neutron Lab</MiscLocation>
        <Manufacturer>Bicron</Manufacturer>
        <Model>Gmsw / Analyst</Model>
        <SerialNumber>X34567709 / A4567</SerialNumber>
        <CalibrationDate>04/05/2004</CalibrationDate>
        <Latitude>?</Latitude>
        <Longitude>?</Longitude>
        <Elevation>?</Elevation>
        <SiteAddress1>Building 7735 (Radcal)</SiteAddress1>
        <SiteAddress2>Oak Ridge National Laboratory</SiteAddress2>
        <SiteAddress3>Bethel Valley Road</SiteAddress3>
        <SiteAddress4>N/A</SiteAddress4>
        <City>Oak Ridge</City>
        <State>Tn</State>
        <Zip>37831</Zip>
        <PhotoUUE></PhotoUUE>
        <DataWidth>1</DataWidth>
        <DataLength>1</DataLength>
        <DataFormat>float</DataFormat>
        <DataTitles>DoseRate</DataTitles>
        <DataUnits>mR/h</DataUnits>
</RegisterSensor>
```

Figure 34. Example registration message from a sensor on a monitoring node.

Once registered, each driver begins to continuously monitor its assigned sensor and converting the raw input signals to units of measure. The final results are packaged into XML data packets and posted to the system outbox for subsequent transmission. An example of an actual data packet of information provided to the server is shown in Figure 35. A copy of the data is then written to a local data file as a permanent backup for all information sent to the host server.

```
<?xml version="1.0" ?>
<DataPacket>
        <SensorID>ORNL-7735-GAM1</SensorID>
        <Date>08/16/2004</Date>
        <Time>11:40:06</Time>
        <ExposureRate>8.64e+000</ExposureRate>
</DataPacket>
```

Figure 35. Example data packet from a gamma exposure rate sensor.

The system is also equipped with a system log to allow installers or administrators to diagnose issues that may arise during startup and operation. All error messages, initialization problems, etc. are written to this log file to allow retrospective analysis of system startup and operating performance. An example of actual entries made to a system log file during the initialization phase of a unit coming online is shown in Figure 36. This particular example displays the startup messages posted by sensors as they come online, register with the TOADS server, begin to monitor their inputs and report the results. This particular event log does not indicate any system errors or any issues out of the ordinary.

```
07/26/2004 17:29:40, ORNL-7735-T001T, Sensor starting
07/26/2004 17:29:40, ORNL-7735-T001T, Sensor registration message sent to OutBox
07/26/2004 17:29:40, ORNL-7735-T001T, Starting data acquistion.
07/26/2004 17:29:40, ORNL-7735-H001T, Sensor starting
07/26/2004 17:29:40, ORNL-7735-H001T, Sensor registration message sent to OutBox
07/26/2004 17:29:40, ORNL-7735-H001T, Starting data acquistion.
07/26/2004 17:29:41, ORNL-7735-ACV01, Sensor starting
07/26/2004 17:29:41, ORNL-7735-ACV01, Sensor registration message sent to OutBox
07/26/2004 17:29:41, ORNL-7735-ACV01, Starting data acquistion.
07/26/2004 17:29:41, ORNL-7735-GAM1, Sensor starting
07/26/2004 17:29:41, ORNL-7735-GAM1, Registration message sent to OutBox
07/26/2004 17:29:41, ORNL-7735-GAM1, Starting data acquistion.
07/26/2004 17:29:42, ORNL-7735-GM01, Sensor starting
07/26/2004 17:29:42, ORNL-7735-GM01, Registration message sent to OutBox
07/26/2004 17:29:42, ORNL-7735-GM01, Starting data acquistion.
07/26/2004 17:29:42, ORNL-7735-N01, Sensor starting
07/26/2004 17:29:42, ORNL-7735-N01, Registration message sent to OutBox
07/26/2004 17:29:42, ORNL-7735-N01, Starting data acquistion.
07/26/2004 17:29:43, ORNL-7735-RN01, Sensor starting
07/26/2004 17:29:43, ORNL-7735-RN01, Registration message sent to OutBox
07/26/2004 17:29:43, ORNL-7735-RN01, Starting data acquistion.
07/26/2004 17:29:43, ORNL-7735-CON01, Sensor starting
07/26/2004 17:29:43, ORNL-7735-CON01, Sensor registration message sent to OutBox
07/26/2004 17:29:43, ORNL-7735-CON01, Starting data acquistion.
07/26/2004 17:29:46, ORNL-7735-MOT01, Sensor starting
07/26/2004 17:29:46, ORNL-7735-MOT01, Sensor registration message sent to OutBox
07/26/2004 17:29:46, ORNL-7735-MOT01, Starting data acquistion.
... ... ...
```

Figure 36. Excerpt from the system log on the monitoring node controller.

The A/D and D/A circuitry on the unit can be calibrated remotely eliminating the need to visit the site of location. A calibration utility program was written during the development effort that will automatically calibrate all ranges on the unit and report the results to text file. An example of the results from an actual calibration test is shown in Figure 37.

```
04/21/04 13:36:02 AutoCal started
     Tolerance for D/A and A/D calibration is +/- 2 LSB.
     A/D converter is 16 bit. D/A converter is 12 bit.
          D/A 10V Bipolar: PASS: Offset= -0.05, Gain = 0.16 LSB
          D/A 10V Unipolar: PASS: Offset= -0.05, Gain = -0.01 LSB
          A/D 10V Bipolar: PASS: Offset= -0.12, Gain = -0.07 LSB
          A/D 5V Bipolar: PASS: Offset= -0.05, Gain = -0.11 LSB
          A/D 2.5V Bipolar: PASS: Offset= 0.06, Gain = 0.19 LSB
          A/D 1.25V Bipolar: PASS: Offset= 0.02, Gain = 0.10 LSB
          A/D 10V Unipolar: PASS: Offset= -0.02, Gain = -0.09 LSB
          A/D 5V Unipolar: PASS: Offset= -0.23, Gain = -0.31 LSB
          A/D 2.5V Unipolar: PASS: Offset= -0.14, Gain = -0.10 LSB
          A/D 1.25V Unipolar: PASS: Offset= -0.06, Gain = -0.28 LSB
       All channels passed.
```

Figure 37. Example of an automatic calibration of the D/A and A/D converters on a node controller.

5.4 Data Transmission

Communication with the central TOADS server is accomplished using TCP/IP connections. For the demonstration unit, the connection was made over a local Ethernet/T1 service line although the connection could be completed using many types of hardware protocols. A telephone modem, for example, can easily be incorporated into the unit as well as satellite or radio transmission hardware through the use of add-on components. Actual communication with a remote TOADS server was performed through TCP/IP sockets. This type of connection is an end-to-end managed, "connectionless" protocol where data is transferred via packaged data grams. This method is a very resilient mechanism and has become a de-facto standard used by many Internet applications. It provides a high degree of compatibility with existing network infrastructures and enables transparent communication capability between all computing systems that are commonly used today. Using this mechanism, remote nodes connected directly to the Internet or to a network having access privileges to the Internet will be capable of finding and communicating with a TOADS central server unit located at any other location on the planet.

6.0 Installation and Operation

The TOADS demonstration system was installed at the ORNL DOSAR Calibration Laboratory (Building 7735). To enable continuous communication and remote troubleshooting capabilities, a high speed broadband connection was extended to the facility from a neighboring building. This connection added a local 10Base-T network and was interconnected to a T1 connection at the neighboring facility. A photograph of the front exterior of the facility is shown in Figure 38.

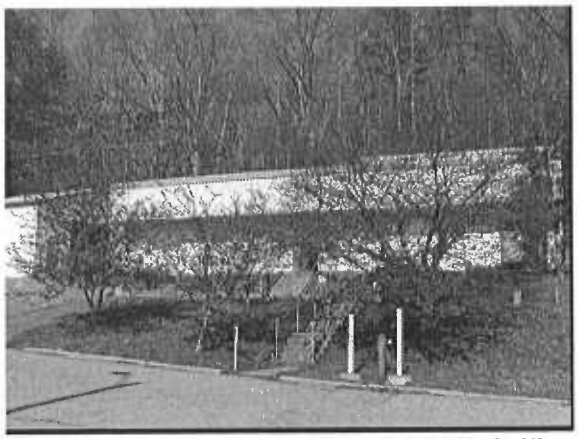

Figure 38. Front view of the DOSAR Calibration Facility (Bldg. 7735) at Oak Ridge National Laboratory.

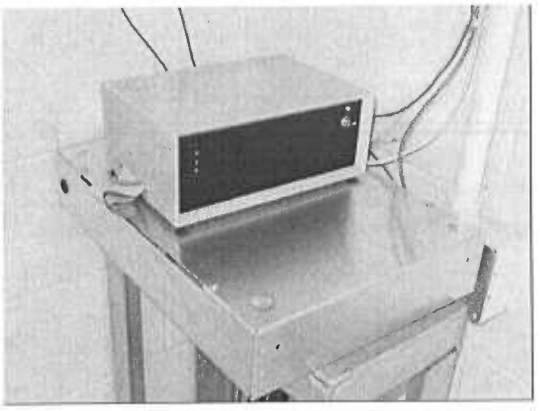

Figure 39. Remote monitoring node installed in the control room of the DOSAR Calibration Laboratory.

Signal cabling was installed between the control room and sensor installation points throughout the facility to provide a means of direct connection to each of the sensors. A layout of the interior of the building showing the install points for the remote monitoring node and all sensors for the demonstration system is shown in Figure 40. Figure 39 shows the complete demonstration monitoring node after installation sitting on top of the power supply and wiring breakout box. The UPS backup power is mounted near the floor beneath the controller but cannot be seen in the photo.

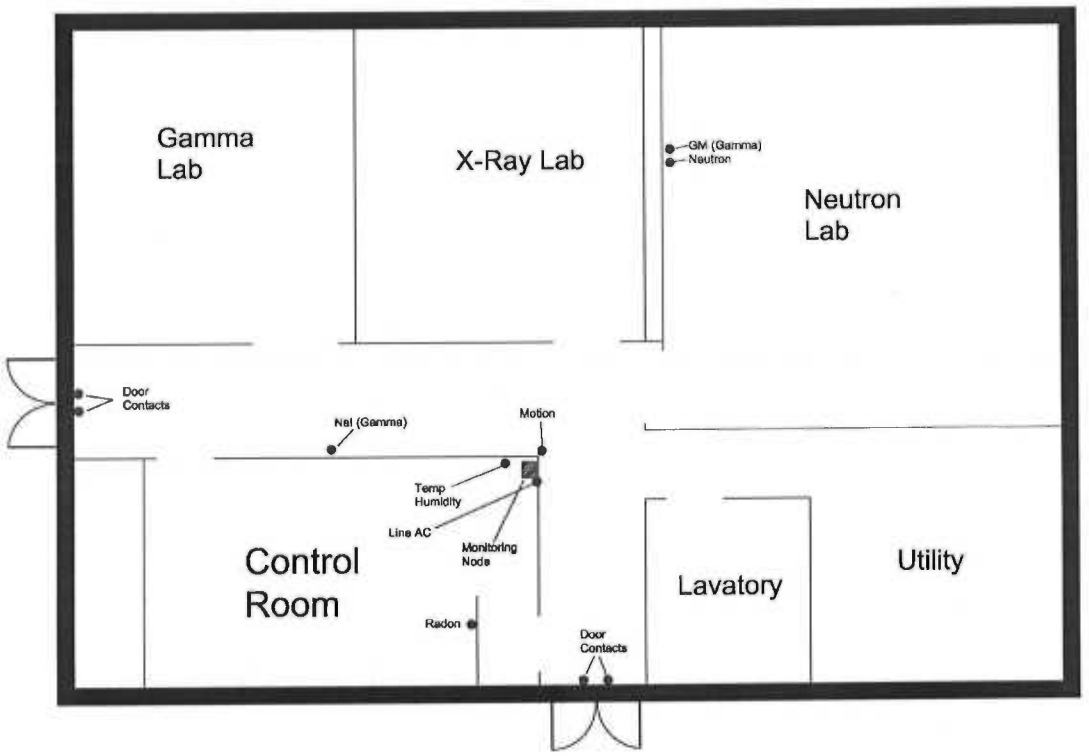

Figure 40. Facility layout for the DOSAR laboratory showing the install locations for all sensors.

The demonstration system has been operating without failure since installation. The data has been available on-line at the TOADS website, http://toads.ornl.gov/. A screenshot of the TOADS website is provided in Figure 41. Minor updates have been performed to tailor the operating parameters to better suit facility-specific issues. Data was collected, archived and sent to the TOADS server 24 hours per day 7 days per week without any problems.

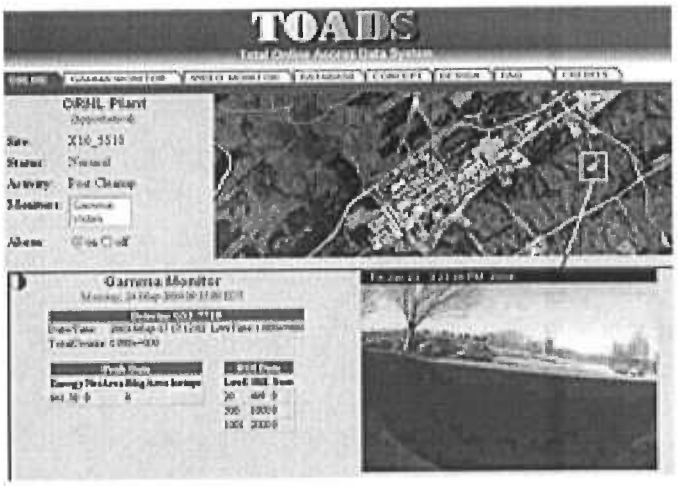

Figure 41 Screenshot of TOADS website, http://toads.ornl.gov/.

7.0 Summary and Conclusions

During this Phase II STTR project, TOADS has been developed and successfully demonstrated at a secure facility at ORNL, with the monitoring data available on-line at the TOADS website at http://toads.ornl.gov. From a systems perspective, TOADS consists of the following major services and internal processing:

- The TOADS server reliably interfaces with and acquires data from a wide variety of instruments, sensors and transducers, external databases, and other remote systems. A TOADS server is capable of interfacing with other remote TOADS servers for sharing data and processes.

- The TOADS data acquisition service automatically performs QA/QC on data inputs and logs the status of instruments/devices. In addition, it can notify and trigger alarms as well as remote electronic calibration and maintenance for target instruments.

- The web service provides graphical user interface capable of presenting information ranging from raw data to textual and graphical forms.

- The TOADS database management system is capable of storing and performing distributed operations and data sharing.

The remote monitoring node consists of specially designed and commercial-off-the-shelf components including a controller, sensors, supporting electronics, signal conditioning and processing components, uninterruptible power supply, AC to DC conversion power supplies and associated support hardware and cabling needed to install the system. The node controller consists of an embedded processing unit with analog to digital, digital to analog, digital input/output as well as high speed pulse counting capability. The unit includes a variety of standard input/output mechanisms such as universal serial bus, 10BaseT Ethernet and RS-232 communications ports.

The remote monitoring node was designed for compatibility with most sensor types. TOADS interacts with a wide variety of instruments and monitoring devices. These include single or multi-channel sensors, digital instruments/systems, analog/digital data loggers, contact switches, and audio/visual surveillance systems. In general, the classes of instruments recognized by TOADS fall under three categories.

- Devices which acquire finite units of measured data such as temperature, pressure, etc. These include a broad category of instruments such as single/multi channel sensors, monitors, data loggers, barcode readers, etc.

- Audio and Visual instruments such as TVCAMs, NETCAMS, and standalone microphones and cameras.

- Switches and contacts which indicate on/off conditions. These sensors are typically integrated within other devices (e.g., portal entries) or exist in conjunction with other sensors (e.g., proximity switches, level/leak detection sensors, motion detectors, etc.).

A system architecture was designed that is compatible with a wide range of auxiliary hardware and that provides relatively easy development of sensor drivers. The unit was tested at a secure facility at ORNL using a total of 12 facility, environmental, and radiological monitoring devices to demonstrate the ability of the system to meet performance requirements.

The following highlights the major functionality and capability provided by TOADS:

- **Monitors and Sensors** – TOADS can interface with a wide variety of sensors such as contaminant monitors, area monitors, atmospheric condition monitors, visual surveillance systems, intrusion devices, motion detectors, fire/heat detection devices, and gas/vapor detectors.

- **Remote Access and Services** – When required, users are allowed unmediated access to a TOADS server. A TOADS server provides access to sensor data and services that may actuate a sensor.

- **Scale** – A TOADS site may communicate with several tens of instruments and sensors. There may be up to several hundred TOADS sites in a state or regional area. A TOADS site may communicate with a large number of other TOADS sites.

- **Communication** – The physical link connecting TOADS with its sensors operate on a single mode of communication using the Internet protocol. Future TOADS implementation would support at least two modes of communication with its instruments. Additional modes could include wireless, satellite, or modem communication.

- **Events** – TOADS is capable of generating various events based on certain predefined conditions as well dynamically configurable parameters.