Automated Sensor Web Adaptation to Sensor Networks

by

Gunita Saini

TR11-207, October 23, 2010

This is an unaltered version of the author's MCS thesis

Faculty of Computer Science
University of New Brunswick
Fredericton, N.B. E3B 5A3
Canada

Phone: (506) 453-4566
Fax: (506) 453-3566
E-mail: fcs@unb.ca
http://www.cs.unb.ca

Abstract

This thesis investigates the notion of a dynamic Sensor Web Language (SWL). As nodes appear and disappear in a Wireless Sensor Network (WSN), the protocols acting in the physical layer adapt dynamically to the new network structure. SWL is an extensible, object-oriented language and is especially useful for information transmission from the sensor nodes in the network to the web applications. A dynamic web architecture was extended from a mesh structure to provide automated web based response to changes in WSN.

Two new message types, RequestAnnounce and Announce are added to the SWL grammar to support this adaptation. All software layers, including sensor nodes, gateway, base station (including the database) and the web applications were updated. Two web applications were implemented to display dynamic adaptation on the web clearly.

A test network was established using 6 sensor nodes and 10 sensors (6 battery voltage sensors, 2 air temperature sensors and 2 solar radiation sensors), and 1 gateway. Each node was added one by one over 4 hours in the network, then removed one by one from the network over 4 hours. Testing indicates appearance of a node in the web application within seconds of being added to the WSN.

Acknowledgements

First and foremost with all my sincerity I would like to thank my supervisor, Dr. Bradford G. Nickerson for his kind support thoughout. Dr. Nickerson very patiently laid the foundation for a research mindset in me, and was always available for help and advice.

I would like to thank my sweet brother, Gomed Saini; without his emotional support I could have never completed my thesis. Thank go to my dear parents, Mr A.S. Saini and Mrs. Rita Saini, for always guiding me and aspiring me to give my best.

I would also like to thank John-Paul Arp, for giving me a head start by explaining the previous work, and his sweet help throughout. Also, my thanks goes to a previous research student Zhongwei Sun, for laying a strong foundation for SWL.

I would also like to thank the UNB Faculty of Computer Science for creating such a friendly atmosphere and for their generosity.

At last, I would like to thank my lovely friends, far or near who have always been there to cheer and encourage me.

Table of Contents

List of Tables

List of Figures

List of Symbols, Nomenclature or Abbreviations

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BS	Base Station
CSV	Comma Separated Version
FFD	Full Function Device
HTML	Hyper Text Markup Language
HTTP	HyperText Transfer Protocol
IEEE	Institute for Electrical and Electronic Engineers
IT	Information Technology
LLC	Logical Link Control
LR-WPAN	Low Rate-Wireless Personal Area Networks
MAC	Medium Access Control
nesC	Network Embedded Systems C
OGC	Open Geospatial Consortium, Inc.
PAN	Personal Area Network
PHY	Physical layer
PHP	Hypertext Preprocessor
RF	Radio Frequency
RFD	Reduced Function Device
RISC	Reduced Instruction Set Computer
SN	Sensor Node
SOS	Sensor Observation Service
SWE	Sensor Web Enablement
SWL	Sensor Web Language
UNB	University of New Brunswick

URL	Uniform Resource Locator
UTM	Universal Transverse Mercator
UUID	Universally Unique Identifier [32], a 16 byte character string
WSN	Wireless Sensor Network
W3C	World Wide Web Consortium
XML	EXtensible Markup Language

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) consist of spatially distributed autonomous sensors to cooperatively monitor the physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants. WSNs have found their way into a wide variety of applications and systems with vastly varying requirements and characteristics. Multi-disciplinary research areas have risen where close collaboration between users, application domain experts, hardware designers, and software developers is needed to implement efficient systems [26]. Examples of WSN applications include, environmental monitoring [19], area monitoring [22, 17], machine health monitoring [29], vehicle detection [28, 14], agriculture [31] and, green house monitoring [25].

The Sensor Web Language (SWL) [4, 23] is an extensible, object oriented language supporting robust message passing among various components including sensor nodes, gateway nodes, communication computers, LINUX server and web applications. SWL was first developed by the University of New Brunswick (UNB) for sensor web configuration and query processing in hierarchical sensor networks in 2004 [27]. SWL supports hierarchical routing using the same message structure through the sensor web. An SWL user can send a request for a sensor node reading from a web

application. The request is then received by the base station which, in turn, sends the request to the gateway. The gateway relays the request to the appropriate sensor node. The sensor node gets the reading from the appropriate sensor and sends the reading to the gateway. The gateway then relays the reading back to the base station which, in turn, sends the response to the web application. SWL provided an initial user interface to send a request to, and get a response from the field in near real time. An initial implementation of the SWL compiler made it easy to configure a consistent set of SWL instructions for five different software platforms instructions. Recent results [5] have added increased reliability to WSN messaging.

1.1 Problem Statement

WSNs are approximately divided into four tiers as illustrated in Figure 1.1. Tier 1 contains SWL sensor nodes. SWL sensor nodes can be a variety of motes, including Mica2, TelosB and MicaZ motes, with the optional MDA300 sensor board. Tier 2 contains SWL gateways. Gateways are sensor nodes with an embedded linux controller and cellular modem to support wireless communication with the internet. Tier 3 contains the base station. The base station has a server that listens and communicates with the sensor network (and clients), and stores all relevant information in a MySQL database. Tier 4 are the client applications. Clients are web applications enabling on-line communication with the WSN.

Energy is a very limited resource for sensor nodes [2] in WSNs and is not always renewable. As devices are commonly embedded into the environment, it is very challenging and sometimes even impossible to change batteries when nodes run out of energy. As a result, nodes may fail and not fulfill their intended purpose, long before the expected lifetime is reached. The network structure of WSNs can change irregularly due to low transmission power and due to limited battery power. There

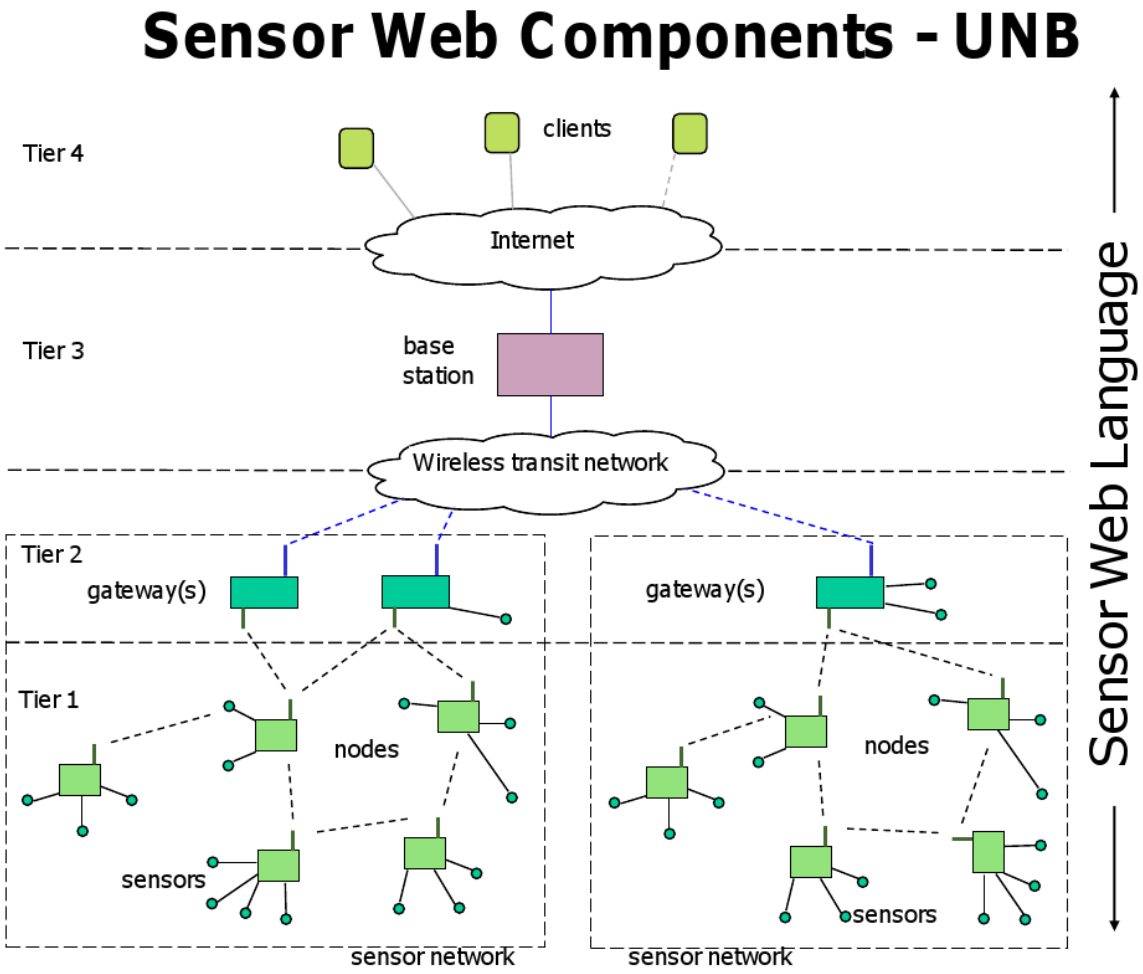


Figure 1.1: WSN architecture (from [4]).

are many protocols designed to conserve energy in a wireless sensor network. These protocols also dynamically adapt to the network structure of tiers 1 and 2 of WSNs. Even when nodes appear and disappear from the network, routing protocols make communication possible in tiers 1 and 2. The dynamic adaptation of WSNs does not extend easily to tiers 3 and 4. Thus, the changing network structure cannot be seen on the web, and hence online communication with a WSN from a web application is not necessarily current.

The goal of this thesis is to investigate new methods for automatically sharing the continually updated WSN structure from tiers 1 and 2 with tiers 3 and 4 of WSNs. In this way, sensor networks can display their dynamic structure easily to the web. This provides a foundation for improved real-time decision support systems making use of sensor networks.

1.2 Dynamic Protocols at the Physical Layer

Various protocols like IEEE 802.15.4 [16], 6LoWPAN [20] and TYMO in TinyOS, are designed to conserve energy in WSNs. They also dynamically adapt to the changing wireless network in the lower layers of a WSN.

1.2.1 IEEE 802.15.4

IEEE 802.15.4 standard is based on the open system interconnection (OSI) seven-layer model. A Low Rate-Wireless Personal Area Network (LR-WPAN) device is comprised of a Physical Layer (PHY), which contains a radio frequency (RF) transceiver along with its low-level communication mechanism, and a Medium Access Control (MAC) sublayer that provides access to the physical channel for all types of transfer. Figure 1.2 shows these blocks in a graphical representation. The upper layers consist of a network layer, which provides network configuration, manip-

ulation, and message routing, and an application layer, which provides the intended function of the device. An IEEE 802.2 Type 1 logical link control (LLC) can access the MAC sublayer through the service specific convergence sublayer (SSCS). The radio in the PHY layer operates at one or more of the following unlicensed bands: 868-868.6 MHz, 902-928 MHz or, 2400-2483.5 Mhz. Depending on the application environment IEEE 802.15.4 LR-WPAN may operate in either of two topologies: the star topology or the peer-to-peer topology. Wireless sensor networks use peer-to-peer network topology. In peer-to-peer topology each device is capable of communicating with any other device within its radio sphere of influence.

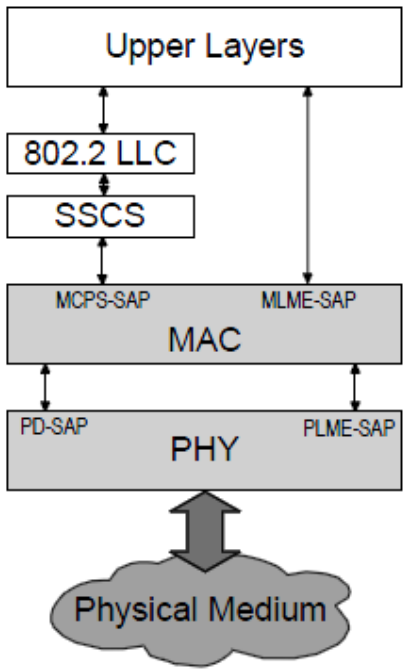


Figure 1.2: IEEE 802.15.4, LR-WPAN architecture (from [16]).

Two different device types can participate in an IEEE 802.15.4 network; a full function device (FFD) and a reduced function device (RFD). The FFD can operate in three modes as a personal area network (PAN) coordinator, a coordinator, or a device. An FFD can talk to RFDs or other FFDs, while an RFD can only talk to an FFD. An RFD is intended for applications that are extremely simple; they do not have the need to send large amounts of data and may only need to associate with

a single FFD at a time, Consequently, RFDs can be implemented using minimal resources and memory capacity.

An example of the use of peer-to-peer communications topology is the cluster tree. In a cluster tree network most devices are FFDs. An RFD connects to the cluster tree network as a leaf device at the end of the branch because RFDs do not allow other devices to associate. Any of the FFDs may act as a coordinator. Only one of these coordinators can be an overall PAN coordinator, selected using a contention resolution mechanism. The PAN coordinator may have greater computational resources than any other device(s) in the PAN. The PAN coordinator forms the first cluster by choosing an unused PAN identifier (64 bit address) and broadcasting beacon frames to neighboring devices. A candidate device receiving a beacon frame may request to join the network at the PAN coordinator. If the PAN coordinator permits the device to join, it adds the new device as a child device in the neighboring list. Then the newly joined device adds the PAN coordinator as its parent in its neighbor list and begins transmitting periodic beacons. Other candidate devices may then join a network at that device. A new device can join the network dynamically using the IEEE 802.15.4 protocol as defined.

Zigbee technology [15] is a low data rate, low power consumption, low cost, wireless networking protocol targeted towards automation and remote control applications. Zigbee is a commercial technology which builds on top of the IEEE 802.15.4 protocol stack. IEEE 802.15.4 focuses on the specification of the lower two layers (physical and data link layer). On the other hand, the Zigbee alliance aims to provide the upper layers of the protocol stack (from network to the application) for inter-operable data networking, security services and a range of wireless home and building control solutions. Zigbee compliant wireless devices are expected to transmit 10 to 75m, and operate in an unlicensed radio frequency world wide. The data rate is 250Kbps at 2.4GHz, 40Kbps at 915MHz, and 20Kbps at 868MHz.

1.3 Sensor Observation Service

Usage of sensor technology is growing rapidly. Devices are becoming smaller, less expensive and more reliable, more power efficient and more intelligent. Due to the variety of sensors available, a coherent infrastructure has become necessary to integrate heterogenous sensors in a platform independent and uniform way. A sensor web refers to web accessible sensor networks and archived sensor data that can be discovered and accessed using standard protocols and application program interfaces (APIs). The Open Geospatial Consortium, Inc. (OGC) initiative for building a framework of open standards for exploiting web-connected sensors and sensor systems of all types is called Sensor Web Enablement (SWE) [7].

SWE presents many opportunities for adding a real-time sensor dimension to the internet and to the web. SWE activities executed through OGC initiatives establishes the interfaces and protocols to enable sensor webs to access sensors of all types over the internet. The OGC initiatives have defined several foundation components needed for a sensor web. One such specific component called the Sensor Observation Service (SOS) [21] is of particular interest for connecting sensor networks to the web. The goal of SOS is to provide access to observations from sensors in a WSN in a standard way that is consistent for all sensor systems including remote, in-situ, fixed and mobile sensors. SOS leverages the Observations and Measurements (O&M) specification for modeling sensor observations and TransducerML and SensorML specifications for modeling sensors and sensor systems. SOS is an intermediary between a client and an observation repository (a database). Clients can also access SOS to obtain metadata information that describes associated sensors, platform, procedures and other descriptive data associated with the observations. This metadata is usually in an XML format.

1.4 Objectives

This thesis builds on previous work [4, 23, 24] to provide a dynamic nature to SWL enabling automatic updating of network structure to the web. To achieve this, the existing code for each layer of a WSN needs to be modified.

We need to reconfigure the base station such that it recognizes messages from new nodes and updates the database automatically. Changes need to be made in the existing web application software so that it automatically reconfigures itself when new nodes appear in the network. The OGC standard SOS can be used as an interface for adding real time sensor updates into the web. We also need to change the SWL compiler such that it supports automated WSN adaptation to the base station and the web applications.

More specifically, the objectives of the thesis includes the following:

1. To leverage sensor network adaptation capability into the suite of tools used to build sensor web applications.

2. To modify the base station and web server programs so that they automatically reconfigure themselves when new nodes are added.

3. To include context data (e.g. Google Maps) in the web application to make the network structure more understandable to the user.

4. To use the OGC standard SOS as an interface between the SWL server and SWL web application for building sensor web applications.

5. To add SWL compiler support for automatically generated sensor web service models.

Automatic updating of the WSN to the web provides a foundation for improved real-time decision support systems making use of sensor networks. In applications like

environmental monitoring, for example, a dying node can be detected and replaced to ensure a continuous stream of data.

1.5 Thesis Organization

This thesis is divided into 6 chapters. Chapter 2 gives an introduction to the previous work done, describing various existing components of SWL. Chapter 3 contains the changes made in 4 tiers of WSN code to achieve dynamic adaptation of the network. Chapter 4 describes the two web applications, Google Maps and SOS added into the SWL code. These implementations are added to make the web application more descriptive, platform independent and user friendly. Chapter 5 defines the simulations and experiments conducted to validate the work done. Chapter 6 concludes the work done along with a discussion on how to improve the current implementation in the future work. Appendices are also provided with the source code used in the simulations and experiments, and a more detailed explanation of the programs used in experimentation.

Chapter 2

Adaptive System Design

We implemented the adaptive SWL system building on the previous non-adaptive SWL models [4, 23, 24]. The adaptive code is an extension to the existing SWL mesh architecture code [27]. Chapter 2 first describes the existing SWL code. Section 2.2 introduces the adaptive design of SWL, briefly discussing changes in the SWL grammar and introducing the Announce message.

2.1 Existing Single Hop Structure

A sensor web using a mesh architecture for SWL system architecture is shown in Figure 2.1. A mesh sensor web consists of three major components: (a) a set of wireless sensor nodes with integrated computing, communication and power platforms (e.g. Mica2 platforms from Crossbow Technology Inc.), (b) a base station that provides Wide Area Network (WAN) and data logging (e.g. a Linux server with broadband network connection), and (c) one or more gateway(s) which are responsible for transmitting information between the sensor platforms and the base station (e.g. gateway(s) are connected to the base station through the base station interface unit MIB510CA, connected to the PC via RS232 or USB to a RS232 cable).

The SWL defines fourteen message types and formats of messages sent between the

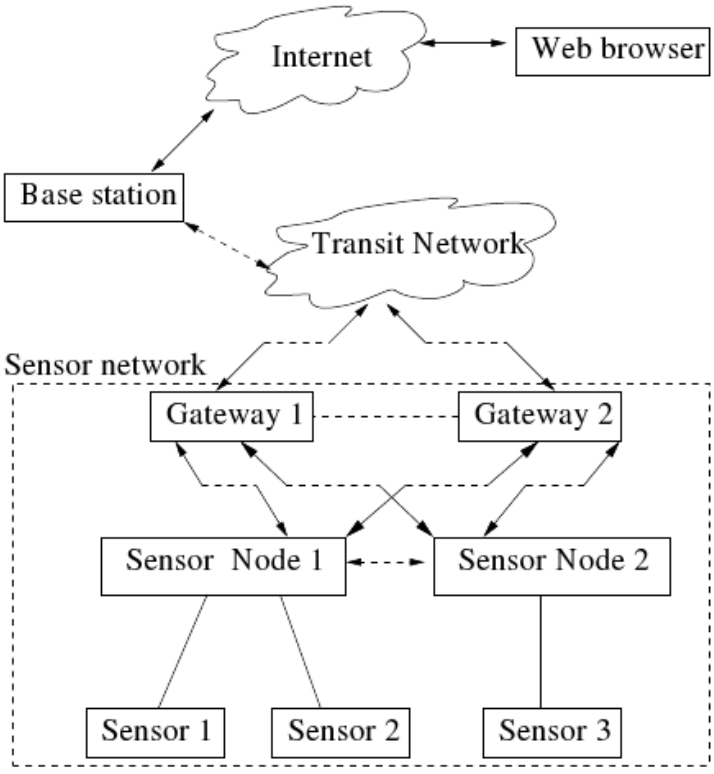


Figure 2.1: Mesh sensor web architecture (from [27]).

sensor nodes and the base station [27].

2.2 Adaptive Sensor Web Language Design

Every sensor node and sensor in a WSN is assigned a string called Universally Unique Identifier (UUID) to identify them uniquely throughout the WSN. A UUID [32] is a 16-byte (128-bit) number, consisting of 32 hexadecimal digits, displayed in five groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 digits and 4 hyphens). An example UUID is, `550e8400-e29b-41d4-a716-446655440000`. As discussed in Section 2.2.3, for a Mica2 sensor node the TOS message packet payload is 28 bytes out of which only 22 bytes can be used to send the message payload. Hence, in the SWL implementation the UUID is truncated from a 16-byte number to an 8-byte number (i.e 16 hexadecimal digits). The UUIDs are generated

using a program named UUID.java attached in Appendix B.1. This program uses the built in Java library java.util.UUID

Choosing to use 64 bits instead of 128 bits for the UUID increases the probability of generating two identical UUIDs. We estimate the probability of two UUIDs with 64 random bits having the same value after calculating 10,000 UUIDs is 3.3×10^{-10} [30, 33]. This is reasonable for the relatively small number of sensors and sensor nodes in a sensor network.

Each sensor node's alive state is recorded in a 1 byte integer called sessionID. Every sensor node in a WSN has a unique sessionID incremented whenever a node is restarted in the field.

2.2.1 RequestAnnounce and Announce Message Types

Two new message types 'RequestAnnounce' and 'Announce' are created. When a new node is turned on in the sensor network, the new node starts sending messages to the base station using the dynamic IEEE 802.15.4 protocol as explained in Section 1.2. The base station records all the sensor nodes it has seen so far using the node's unique senderID. When the base station receives a message from a new sensor node whose senderID is not in the base station's records, the base station identifies that node as a new node in the WSN and sends a RequestAnnounce message to that new node, requesting the new node to announce itself. The new node responds with an Announce message, announcing its presence in the network. The base station saves the values sent by the new node in the database and in its dynamic data structures. Figure 2.2 explains the above communication concept between a new sensor node and the base station in a time line diagram.

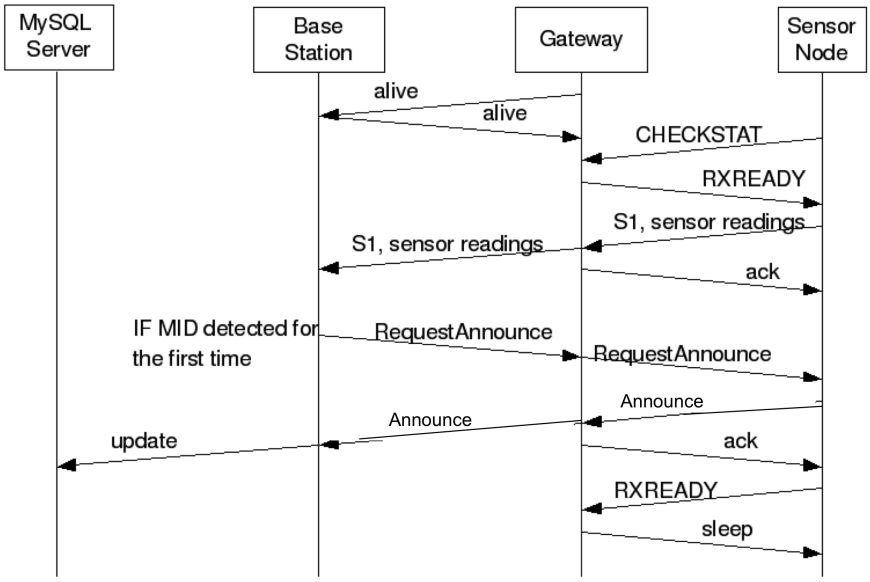


Figure 2.2: Time line showing the communication between the base station and a new sensor node S1 in WSN. MID is the sensor node's unique ID in the network.

2.2.2 SWL Adaptive Grammar

The updated SWL message grammar can be seen in Table 2.2. A new Request type RequestAnnounce is added in the grammar (see rule 15) to enable the RequestAnnounce message sent from the base station to the sensor nodes. A new Response type Announce is also added. Each Announce message consists of the nodeUUID, latitude, longitude, sessionID, number of sensors attached to the node and the sensorUUIDs. The +n sign in grammar rules 21, 49, 50, 51 from Table 2.2 indicate that precisely 'n' instances of the preceding token appear.

2.2.3 SWL Message Packet Structure

Sensor network platforms have their own communication stack and packet structure to enable efficient exchange of individual packets. Mica2 and Mica2Dot platforms employ a TOS message packet which has a 28 byte payload [18]. Every SWL message is split into packets which fit in the payload within a TinyOS packet [3]. A detailed design for SWL message packet structure is shown in Figure 2.3. The design of SWL

1	*Goal*	::=	*MainClass* [{ *ClassDeclaration* }+ | { *Action* }+] EOF
2	*MainClass*	::=	sensornet *Identifier* { *MainDecl* }*
3	*MainDecl*	::=	main (*Arg*) { { *VarDecl* }* { *Statement* }* }
4	*ClassDeclaration*	::=	*ClassSimpleDecl* | *ClassExtendsDecl*
5	*ClassSimpleDecl*	::=	class *Identifier* { { *VarDecl* }* { *Constructor* }* { *MethodDecl* }* }
6	*ClassExtendsDecl*	::=	class *Identifier* extends *Identifier* { { *Statement* }* { *VarDecl* }* { *Constructor* }* { *MethodDecl* }* }
7	*Constructor*	::=	*Identifier* ({ *Arg* }*) { { *SuperCon* }* { *Statement* }* }
8	*SuperCon*	::=	super ({ *Param* }*) ;
9	*Action*	::=	request { *SensorNetId* { *Request* }* } | response { *SensorNetId* { *Response* }* } | report { *SensorNetId* { *Response* }* } | alert { *SensorNetId* { *Response* }* } | changeI { *SensorNetId* { *Response* }* } | switchG { *SensorNetId* { *Response* }* }
10	*SensorNetId*	::=	sensornet (*Identifier*)
11	*Request*	::=	*RequestBasic* | *RequestArray* | *RequestConfig* | *RequestAnnounce*
12	*RequestBasic*	::=	{ *Identifier.IdRest* ([*Param*])};
13	*RequestArray*	::=	{ *ArrayRef.IdRest* ([*Param*])};
14	*RequestConfig*	::=	{ Configuration;}
15	*RequestAnnounce*	::=	{ *Identifier*.ga();}
16	*Response*	::=	*ResponseConstructor* | *ResponseArrayCon* | *ResponseArrayValue* | *ResponseConfig* | *Announce*
17	*ReponseConstructor*	::=	{ *Identifier.IdRest* ([*Param*]);}
18	*ResponseArrayCon*	::=	{ *ArrayRef.IdRest* ([*Param*]);}
19	*ResponseValue*	::=	{ *Identifier.IdRest* = *Exp*;}
20	*ResponseConfig*	::=	{ Configuration{{*varDecl*}*{*Exp*}*}}
21	*Announce*	::=	{ nid={*Hexdigit*}+16;lt={ N | S}*deglat,min,sec*;s={*IntegerType*};ln={ E | W }*deglon,min,sec*;n={*IntegerType*}(;SID={*Hexdigit*+})} | { *NodeConfig* }
22	*NodeConfig*	::=	{ Configuration{{*varDecl*}*{*Exp*}*}}
		:	
		:	
49	*deglat*	::=	{ *Digit*}+2
50	*deglon*	::=	{ *Digit*}+3
51	*min*	::=	{ *Digit*}+2
52	*sec*	::=	{ *Digit*}+5
53	*IntegerType*	::=	int
		:	
		:	
99	*Digit*	::=	{0 | ... | 9}
100	*HexDigit*	::=	{0 | ... | 9 | a | ... | f}

Table 2.2: A portion of Adaptive SWL Grammar, extended from [11, 27].

messages relies on the packet structure of the underlying wireless operating system packets. The address translation of packets is handled by the wireless operating system medium access control protocol [36].

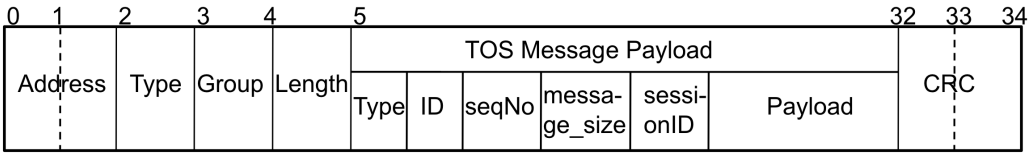


Figure 2.3: SWL message packet design extended from [3].

The current SWL message packet sent from the nodes to the base station is shown in Figure 2.4. Two fields `message_size` and `sessionID` are added. The `message_size` is a one byte integer consisting of the size of a message. If a message is sent in 6 parts then the `message_size` is 6. The `sessionID` is also a one byte integer, saved in the sensor node's flash memory. The `sessionID` tracks a nodes's activity in the network. Whenever the node is alive in the network, it has a `sessionID` which is sent to the base station though the SWL message. If the node gets restarted, the node's `sessionID` is incremented. As the `sessionID` is an 8-bit unsigned integer, we have $0 \leq$ `sessionID` ≤ 255.

```
typedef struct SWLMsg {
    uint8_t ackSeqNo;
    uint16_t senderID;
    uint8_t swlMsgType;
    uint8_t swlMsgID;
    uint16_t swlSeqNo;
    uint8_t message_size;
    uint8_t sessionID;
    uint8_t payload[22];
} SWLMsg;
```

Figure 2.4: Modified SWLMsg structure extended from [27].

Table 2.3 shows 14 existing SWL Message types. One more message type Announce was added with message type 15.

Table 2.3: SWL message types.

Message	SWL Msg. Type
ALIVE	0
REQUEST	1
RESPONSE	2
REPORT	3
SWITCHG	4
ALERT	5
CHANGEI	8
CHECKSTAT	9
RXREADY	10
TXDONE	11
SLEEP	12
WAIT	13
RESET	14
ANNOUNCE	15

The SWL message payload is 22 bytes long. The Announce message type sends the sensor node and sensor information to the base station. The Announce message requires six payload packets due to the limited 22 bytes for each one. Table 2.4 shows the Announce message sent from a sensor node with id 1 to the base station. The message_size is 6, message type is 15 and the sessionID of node 1 is 66.

Table 2.4: Example SWLMsg containing the Announce message split into six packets. Each packet fits in the payload within a TinyOS packet. nid=nodeUUID, lt=latitude, s=sessionID, ln=longitude, n=number of sensors attached to the node, SID=sensorUUID, 15 is the Announce message type.

Sender ID	SWL Msg. Type	Message ID	SWL Seq No.	session ID	Payload
1	15	1	0	66	nid=[16];
1	15	1	1	66	lt=[3],[2],[5];s=[3];
1	15	1	2	66	ln=[4],[2],[5];n=[2];
1	15	1	3	66	SID=[16];
1	15	1	4	66	SID=[16];
1	15	1	5	66	SID=[16];

2.2.4 RequestAnnounce Message

An example RequestAnnounce message sent from the base station to a node via the gateway is shown in Figure 2.5. s3 means a sensor node in the network with id=3 which is identified as a new node in the WSN by the base station. Hence the message is sent by the base station to sensor node 3 to announce itself via the gateway.

```
request{
s3.ga();
}
```

Figure 2.5: Example SWL RequestAnnounce message.

2.2.5 Announce Message

An example Announce message sent as a response to the Announce message is shown in Figure 2.6.

In Figure 2.6, `nid` is the nodeUUID, `lt` is Latitude of the node, `s` is the sessionID of that node, `ln` is the longitude, `n` is the number of sensors attached to the node and `SID` is the sensorUUID of a sensor attached to the node. The latitude and longitude of a node are written as `lt|ln` = hemisphere (`N` or `S` for latitude, `E` or `W` for longitude) `degrees`,`minutes`,`seconds`. Seconds are recorded using units of 1/10000 arc seconds.

```
response{
nid=46f52dc76c604231;
lt=N45,56,59045;s=097;
ln=W66,38,33978;n=03;
SID=c6cf046dc1fa4e63;
SID=cfdf449efd554fe0;
SID=391e816927484bcb;
}
```

Figure 2.6: Example Announce message as response to a RequestAnnounce message.

Chapter 3

SWL Adaptive System Architecture

As discussed in Chapter 1, a WSN is broadly divided into four layers: The physical layer consisting of the wireless sensor nodes, the wireless gateway, the base station and the web clients. To achieve dynamic adaptation of a sensor web to a sensor network, existing programs in all these layers need to be modified. This section begins with an overview of the SWL system architecture and then discusses each WSN layer individually. Modifications to achieve dynamic adaptation of sensor networks are described.

3.1 Software Architecture and Overview

The overall SWL software architecture is shown in Figure 3.1, extended from the previous SWL software architecture [11] by allowing the gateway node and two sensor nodes in the network to communicate through SWL messages. Web Server 1 consists of the SWL Database 1 defined in more detail in Section 3.5 and SWL Server (i.e the base station programs) defined in more detail in Section 3.3. Web Application 2 is defined in more detail in the next chapter.

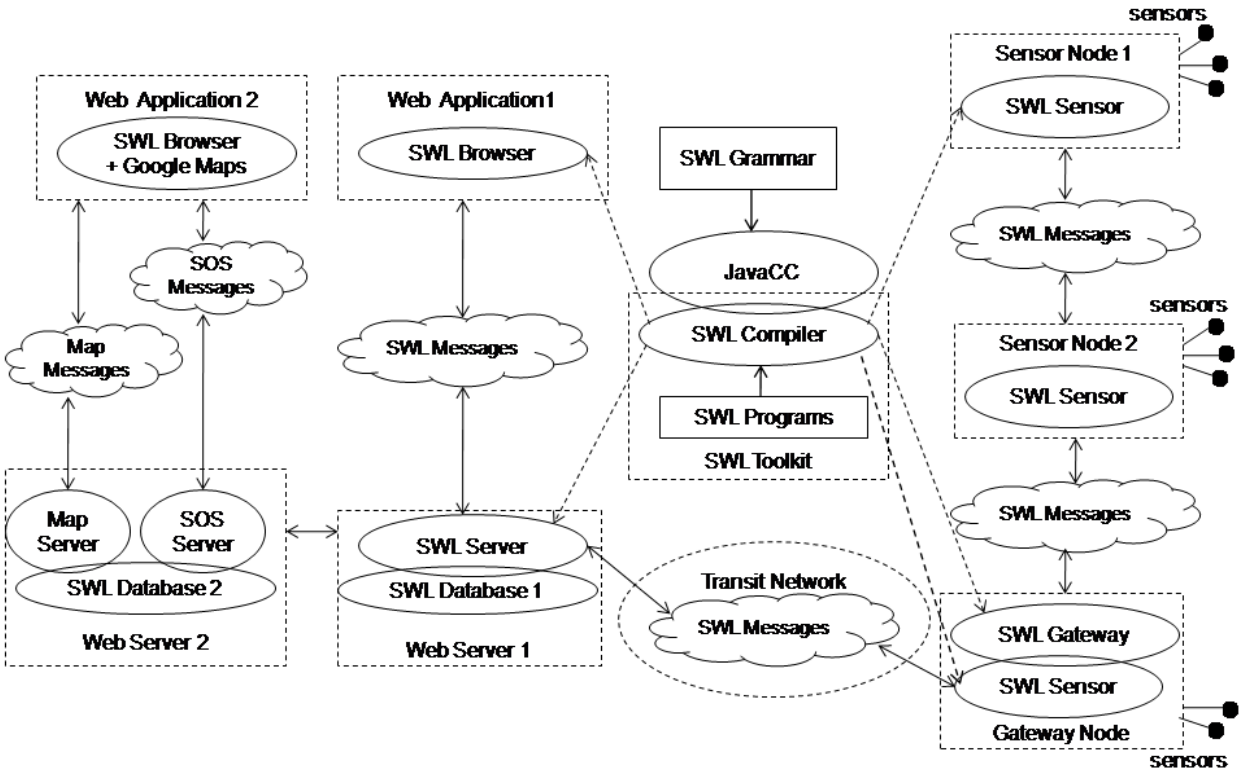


Figure 3.1: SWL system architecture (extended from [11]).

3.2 Sensor Node and Gateway

There are 2 sensor node programs called SWLMote.nc and SWLMoteM.nc and 2 header files, SWLMsg.h and config.h. SWLMote.nc is a top-level configuration file and the source file which the nesC compiler uses to generate an executable file. SWLMoteM.nc provides an implementation of the SWLMote application. SWLMsg.h defines the structure of an SWLMsg and the AckMsg. Figure 3.2 shows a diagrammatic view of the sensor node program compilation process. The sensor node program is installed and compiled on the sensor nodes using a Makefile. An example makefile is shown in Figure 3.3.

Figure 3.2 shows how the Config.h file is generated by the SWL compiler. The Config.h file is a unique file for each sensor node. It has the nodeUUID, latitude, longitude, number of sensors attached to the node and description of the sensors attached. The description contains the sensorUUIDs, sensor type, sensor channel,

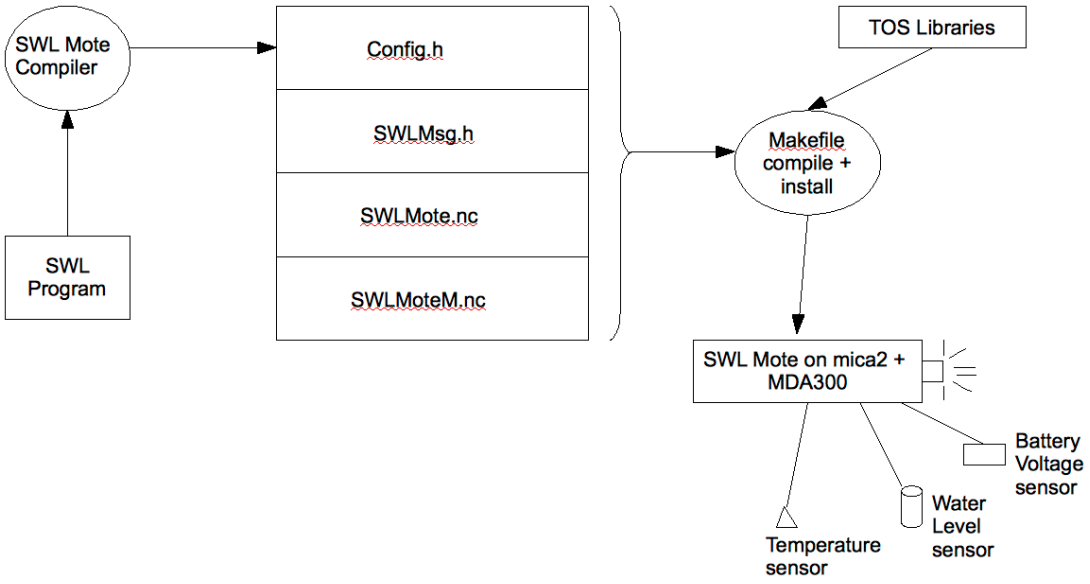


Figure 3.2: Illustration of a sensor node program compilation process.

```
COMPONENT=SWLMote
SENSORBOARD=mda300
XBOWROOT={TOSROOT}/contrib/xbow/tos
XBOWBETAROOT={TOSROOT}/contrib/xbow/beta/tos
PFLAGS = -I$(XBOWBETAROOT)/platform/mica2
PFLAGS += -I$(XBOWBETAROOT)/CC1000RadioPulse
PFLAGS += -I$(XBOWBETAROOT)/lib/ReliableRoute_Low_Power
$(SENSORBOARD)
PFLAGS += -I$ /opt/tinyos-1.x/contrib/xbow/beta/tos/sensorboards/mda300
PFLAGS += -I$(XBOWBETAROOT)/interfaces
PFLAGS += -I$(XBOWBETAROOT)/system
PFLAGS += -I$(XBOWBETAROOT)/lib
PFLAGS += -I$(TOSROOT)/tos/lib/Queue
PFLAGS += -I$(TOSROOT)/tos/lib/Broadcast
PFLAGS += -I$ /opt/tinyos-1.x/contrib/ucb/tos/lib/RandomMLCG
PROGRAMMER_EXTRA_FLAGS = -v=2
include $TOSROOT/contrib/xbow/apps/MakeXbowlocal
include $TOSROOT/apps/Makerules
```

Figure 3.3: Makefile for compiling and installing a sensor node program on the wireless nodes.

sensor bit and sensor parameter. An example Config.h file is shown in Figure 3.4 for a sensor node with three sensors (temperature, waterfall and battery) attached.

```
#define nodeUUID "46f52dc76c604231"
#define LAT_DEG "N45"
#define LAT_MIN "95"
#define LAT_SEC "599999"
#define LONG_DEG "W66"
#define LONG_MIN "63"
#define LONG_SEC "339128"
#define SENSOR1_UUID "c6cf046dc1fa4e63"
#define SENSOR2_UUID "cfdf449efd554fe0"
#define SENSOR3_UUID "391e816927484bcb"
#define NUM_SENSORS 3
enum {
REPORT_INT = 15, // Report Interval = 15 seconds
SENSOR1_TYPE = BATTERY,
SENSOR1_CHANNEL = 0,
SENSOR1_BIT = 0x0001, // A unique bit for each sensor
SENSOR1_INT = 20, // Sample Interval = 20 seconds
SENSOR1_PARAM = SAMPLER_DEFAULT,
SENSOR2_TYPE = ANALOG,
SENSOR2_CHANNEL = 0,
SENSOR2_BIT = 0x0002,
SENSOR2_INT = 900,
SENSOR2_PARAM = EXCITATION_25,
SENSOR3_TYPE = TEMPERATURE,
SENSOR3_CHANNEL = 0,
SENSOR3_BIT = 0x0004,
SENSOR3_INT = 900,
SENSOR3_PARAM = SAMPLER_DEFAULT,
};
```

Figure 3.4: Sample Config.h file.

An example SWLMsg.h file can be seen in Appendix A. As stated in Chapter 2 the Announce message type is defined as 15. We have added 2 more parameters: `sessionID` and `message_size` in the SWL message. Both are of integer type. If a message is sent in 6 parts then the `message_size` will be 6. We have also defined a struct SessionParam to keep track of the `sessionID`.

The SWLMoteM program takes the nodeUUID, latitude, longitude, number of sensors and sensorUUID from the Config.h file. When SWLMoteM receives a Reques-

tAnnounce message for a node, the node responds with an Announce message for that node. The Announce message is divided into a header consisting of three packets plus one additional packet for each sensor attached to the sensor node.

Each part of the Announce message is written into the SWL message payload and then sent to the gateway, which further relays the message to the base station. Each packet of the Announce message is sent only if an acknowledgement of the previous message is received by the SWLMoteM program.

3.2.1 Gateway Node

The gateway program is built on the previous Mesh SWL gateway node program. Before starting communication, the gateway sends an alive message to the base station to check if it is alive. If the gateway receives the alive message back, the gateway sends an RXREADY message as a response to the sensor node's CHECKSTAT message. Then the sensor nodes start communication by sending their sensor readings to the gateway. The gateway passes these sensor readings to the base station and sends an acknowledgement to the sensor nodes. The CHECKSTAT communication mechanism is shown in the Figures 3.5 and 3.6.

Figure 3.7 shows a diagrammatic view of the gateway node program structure. A makefile uses the gateway programs written in nesc and TOS libraries to install and compile the SWL gateway node program.

For the Announce message, we used the single hop gateway node program used by Zhongwei Sun [27] and written by John-Paul Arp. The only addition is that the gateway forwards any Announce message it receives from the sensor nodes to the base station.

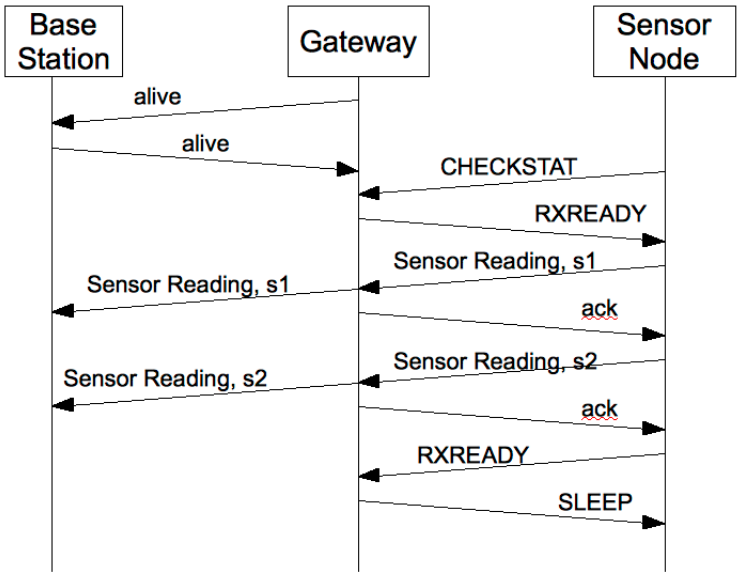


Figure 3.5: The gateway, sensor node and base station communication when a connection with the base station is generated.

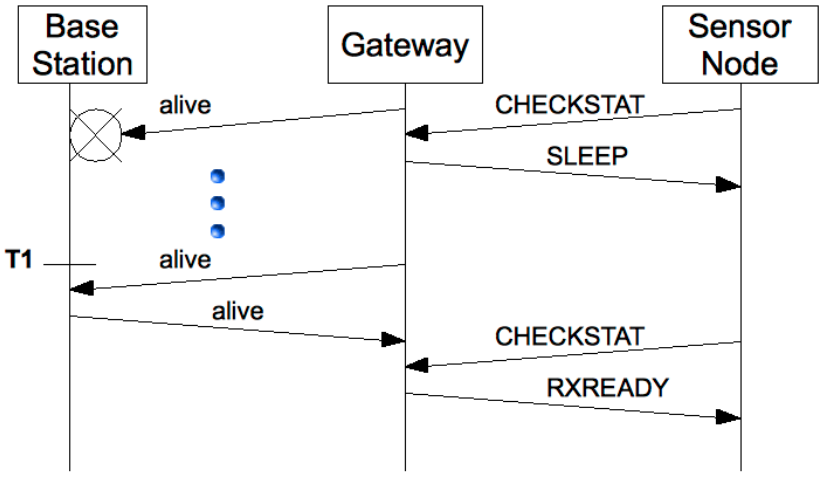


Figure 3.6: The gateway, sensor node and base station communication when the base station is not operational and then becomes operational at time T1.

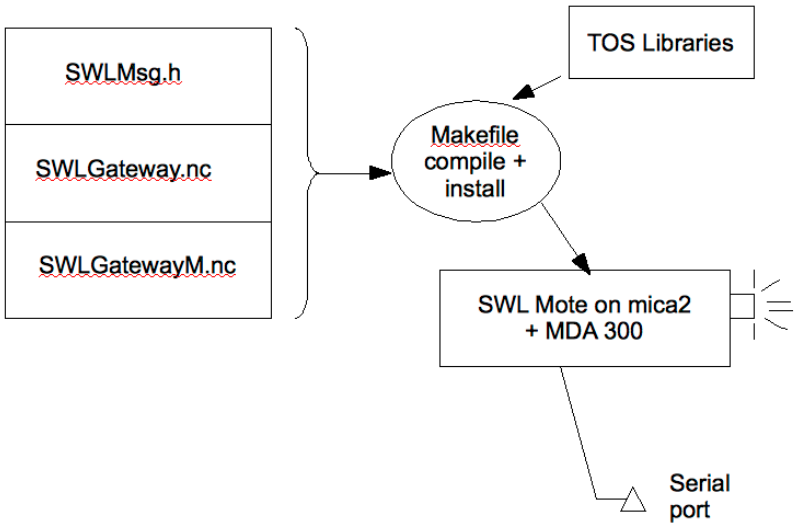


Figure 3.7: The gateway node program structure.

3.3 Base Station

The base station consists of programs SWLAnnounce.java, Moteindex.java, SWLResponseCollector.java, SWLAnnResponseCollector.java, AnnResponseMsg.java, and AnnounceBSClient.java.

SWLAnnounce.java is the central program, receiving Announce messages from the sensor nodes, updating values in the SWL database 1, and forwarding the updated network structure to the web application 1. SWLAnnounce.java is discussed in depth in the next section. SWLResponseCollector.java, SWLAnnResponseCollector.java and AnnResponseMsg.java are object classes used by SWLAnnounce.java. SWLResponseCollector.java and SWLAnnResponseCollector.java collect the whole response in an array while messages are received one by one by SWLAnnounce.java. SWLResponseCollector.java collects Response messages and SWLAnnResponseCollector.java collects Announce messages. The length of the Announce message is always three plus the number of sensors attached to a sensor node. Moteindex.java is also an object class called by the SWLAnnounce.java. Moteindex.java is also discussed in more detail in the next section. The AnnounceBSClient.java program is

a client socket program which passes new node information to the web application. The base station program structure is shown in Figure 3.8. mig (message interface generator) for nesc programming language, is a tool to generate code that processes TinyOS messages. Here mig generates SWLMsg.java and AckMsg.java (for base station use) from SWLMsg.h and AckMsg.h (TinyOS header files), respectively.

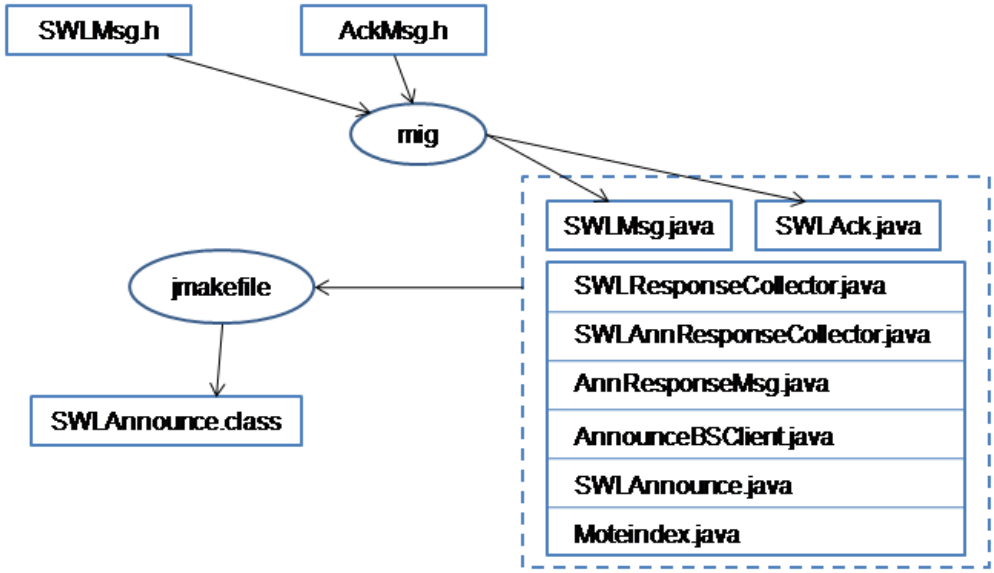


Figure 3.8: The base station programs.

3.3.1 SWLAnnounce

SWLAnnounce.java performs three operations: (a) Receive messages from the sensor nodes in the WSN and perform actions according to each message type received, (b) If the message type received is 15 (i.e. if message type is Announce) then update the database, and (c) forward the Announce message when received at the web application.

When communication is established between the sensor node, the gateway and the base station, and SWLAnnounce starts receiving messages, SWLAnnounce saves the session(s) of that node with a Sender ID in a hash table. If a new node enters the network, it has a new Sender ID not present in the hash table. SWLAnnounce

now knows that a new node has entered the network and needs to announce itself. SWLAnnounce sends a RequestAnnounce packet to the gateway. The gateway then forwards the RequestAnnounce to the sensor node. If a node is restarted, the Sender ID node will have a new session ID not present in the hash table. With this SWLAnnounce comes to know that a node is restarted and needs to announce itself. Moteindex.java is an empty class having only two member variables, id and sessionID. Moteindex.java is called by SWLAnnounce to save the latest session ID of each senderID into Moteindex.java's member variables. Moteindex.java is attached in Appendix B.3. For a clearer understanding, pseudocode for the node announce process is shown in Algorithm 1. The complete code is in the messageReceived() method of object class SWLAnnounce.java attached in Appendix B.1.

```
messageReceived(int to, SWLMsg m)
Data: Global Hash table Sessiontable, Moteindex mi
Result: If a new SN appears or a SN is restarted in WSN, then BS sends
        RequestAnnounce message to that SN
if Sessiontable.containsKey(m.senderID) then
  | mi ← Sessiontable.get(m.senderID)
end
if !(Sessiontable.containsKey(id)) ||
(Sessiontable.containsKey(m.senderID) && (mi.sessionID ≠
m.sessionID)) then
  | mi.id ← m.senderID
  | mi.sessionID ← m.sessionID
  | SWLAnnounce ann = new SWLAnnounce()
  | Compose RequestAnnounce message in ann
  | announce.sendMsg(ann)
end
```

Algorithm 1: Pseudo-code for the Announce mechanism in the messageRecieved() function of program SWLAnnounce.java.

SWLAnnResponseCollector.java is shown in Appendix B.2. SWLAnnResponseCollector is an object class used by SWLAnnounce to save the Announce messages in an array.

AnnounceBSClient.java is a client socket object class connecting to the web applica-

tion. This class is called by the SWLAnnounce program to create a connection with the web application and forward the new node information to the web application. The new node information includes the node UUID, ID, node name, manufacturer, description, model, latitude, longitude, height above sea and height above ground. AnnounceBSClient.java also forwards the attached sensor's UUID, ID, name, description, manufacturer, model and unit.

Lastly, SWLAnnounce connects to the database. After receiving an Announce message from a node, SWLAnnounce interacts with the database answering queries and storing data. The database operations done by SWLAnnounce using MySQL are discussed further in Section 3.5. Source code for SWLAnnounce.java is given in Appendix B.1. Figure 3.9 shows the SWLAnnounce program communication with the gateway, the sensor nodes and MySQL database.

3.4 Web Application 1

Web application 1 is the browser implementation defined in [27]. An example web application 1 screen generated by the SWL compiler [27] for two sensor nodes and a gateway in the network is shown in the Figure 3.10. A square button and two triangle buttons represent one gateway and two sensor nodes, respectively. Sensors are represented by the small buttons around the gateway and the sensor nodes. A button's name appears on the top part of the application when it is moused over.

A sensor web configuration for a simple sensor network is generated by the compiler. Figure 3.11 shows a sample sensor web configuration file for a sensor network with one gateway and one sensor node generated by the SWL compiler. As seen in Figure 3.11 the sensor network needs to be predefined for the web application before communication.

To solve this problem the configuration file is generated dynamically whenever the

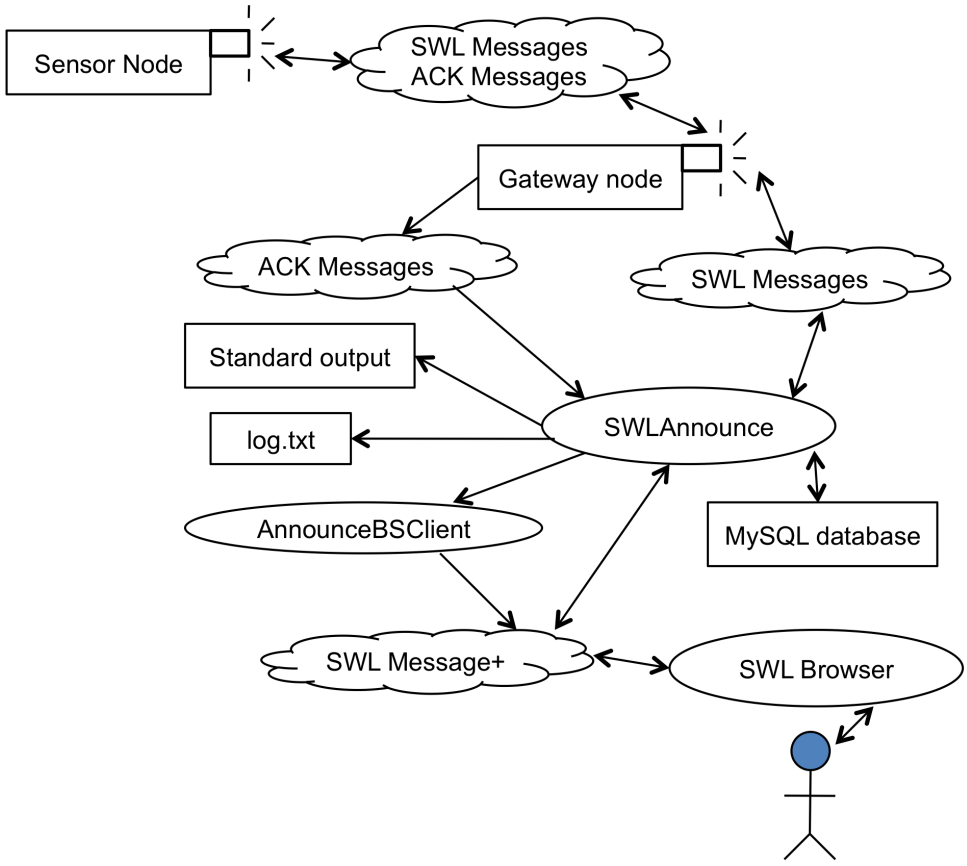


Figure 3.9: SWLAnnounce program communicating with the sensor node, gateway node and the web application.

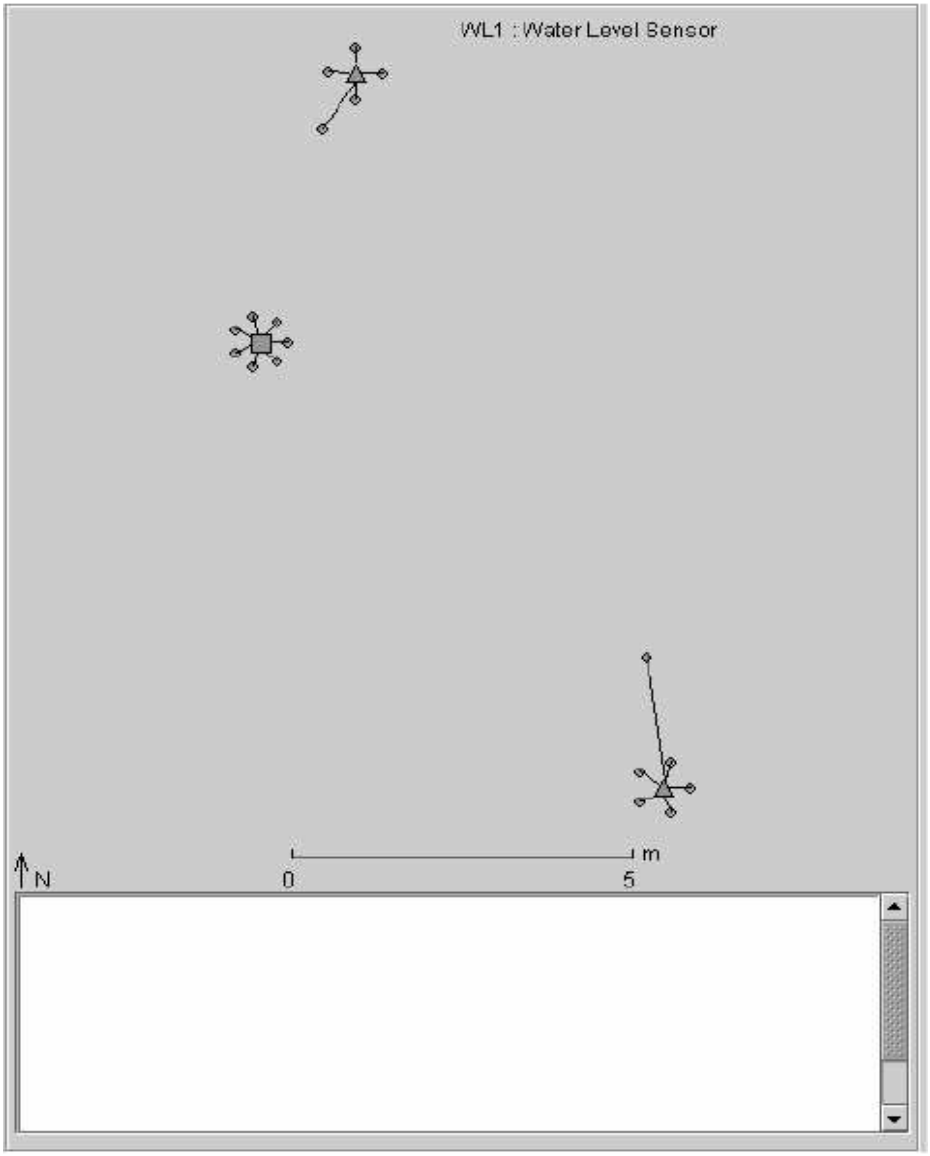


Figure 3.10: Sample SWL web application screen (taken from [27]).

```
sensornet UNBwoodlot response
{ SensornetId="UNBWoodlot";
  Configuration
  { Gateway G1;
    GatewayCD GCD1;
    GatewaySN GSN1;
    SN RSN1;
    Sensor AT1;
    Sensor WT1;
    Sensor SR1;
    Board b1;
    Board b2;
    Channel ch1;
    Channel ch2;
    G1.Latitude(45, 55, 27.0, "N");
    G1.Longitude(66, 40, 12.5, "W");
    GCD1.Latitude(45, 55, 27.0, "N");
    GCD1.Longitude(66, 40, 12.5, "W");
    GSN1.Latitude(45, 55, 27.0, "N");
    GSN1.Longitude(66, 40, 12.5, "W");
    RSN1.Latitude(45, 55, 27.0, "N");
    RSN1.Longitude(66, 40, 12.5, "W");
    RSN1.DataAcq(b1);
    AT1.Latitude(45, 55, 27.03, "N");
    AT1.Longitude(66, 40, 12.56, "W");
    SR1.Latitude(45, 55, 27.04, "N");
    SR1.Longitude(66, 40, 12.58, "W");
    b1.Latitude(45, 55, 27.03, "N");
    b1.Longtitude(66, 40, 12.56, "W");
    b1.SensorNumber(2);
    b1.Channel(ch1, AT1);
    b1.Channel(ch2, SR1);
    ch1.Latitude(45, 55, 27.03, "N");
    ch1.Longitude(66, 40, 12.56, "W");
    ch1.ChannelType(ANALOG);
    ch1.ChannelNumber(2);
    ch2.Latitude(45, 55, 27.04, "N");
    ch2.Longitude(66, 40, 12.58, "W");
    ch2.ChannelType(ANALOG);
    ch2.ChannelNumber(6);
    }
}
```

Figure 3.11: A sample sensor web configuration for a simple SWL network generated by the SWL compiler.

sensor network changes. This is done by creating dynamic data structures in the web application, as explained in more detail in the next section. Whenever the base station receives an Announce message, it forwards an SWL message+ to the web application. Figure 3.12 illustrates the communication between the base station and the web application by a time line diagram.

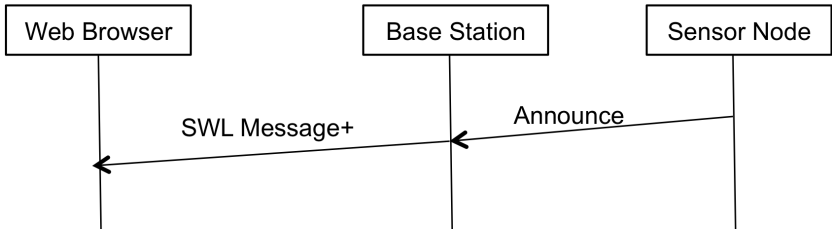


Figure 3.12: Time line showing the communication between the base station and the web application.

The SWL message+ consists of a node's UUID, ID, name, description, manufacturer, model, latitude, longitude, height above sea, height above ground and number of sensors attached. A complete description of all attached sensors is also sent, as explained in the Section 2.2.5.

The web application 1 code is written as a Java Applet and Servlet. Java Applet is the front-end language used to create dynamic web pages and Java Servlet is a back-end language used to communicate with the database.

3.4.1 Dynamic Web Application 1 Implementation

The web application consists of two applet programs: SWLBrowser.java and UTMAppletTest.java. The SWLBrowser program is a server socket applet program which creates connections with the AnnounceBSClient program on the base station. The SWLBrowser program creates a unique thread for each node via a ClientWorker.java object class program. Node information and information of all the sensors attached to that node are saved in dynamic hash tables maintained by the BuildNetwork.java program. From the values in the hash tables of the BuildNetwork.java program an

input file to UTMAppletTest.java program is created at run time. The SWLUpdateNetwork.java program creates this input file. This file consists of the complete network information. The UTMAppletTest.java program converts the latitude and longitude values of each node into a Universal Transverse Mercator (UTM) format, UTM is a map projection specifying locations on the surface of the earth in a 2-dimensional cartesian coordinate system [35]. UTMAppletTest.java also draws a scale bar on the bottom of the web application 1 screen.

Figure 3.13 illustrates the SWLBrowser program communication with the object classes ClientWorker, BuildNetwork, SWLUpdateNetwork, and UTMAppletTest applet program and with the base station AnnounceBSClient program. When a new node enters the WSN and announces itself to the base station, the base station forwards the new node information to the SWLBrowser listener program on the web by creating a socket with the AnnounceBSClient program on the base station. SWLBrowser is a listener program which creates a thread for the new node by calling the object class ClientWorker.java. The SWLBrowser program then updates the hash tables containing the sensor network values by calling the object class BuildNetwork.java. SWLBrowser then creates the input file for the UTMAppletTest program by calling another object class SWLUpdateNetwork.java. The sample SWLBrowser screen can be seen in Figure 3.14. The rectangle is the gateway node, the triangle is the new node RSN1 in the network (NE of the gateway), and the circles are the sensors.

Figure 3.15 is the SWLBrowser screen when a new node is added to the WSN. Note that the existing node RSN1 in the WSN has changed to a grey color, and the new node RSN3 is dark. Also, the position of RSN1 and the gateway has changed as the new node RSN3 (SE of the gateway) is added to the network. This is because the UTMAppletTest program readjusts the two nodes and the gateway on the web application 1 screen according to their latitude and longitude values.

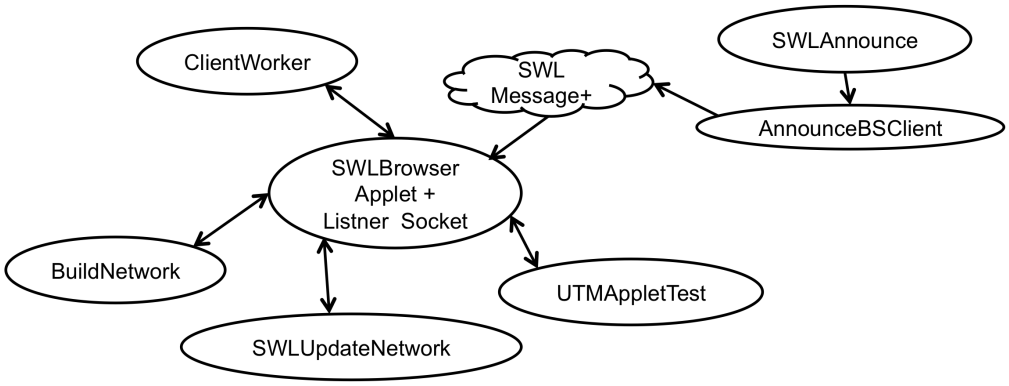


Figure 3.13: SWLBrowser program communicating with it's object classes, UTMAppletTest and the AnnounceBSClient base station program.

A color code is used to show the time duration for which a node is present in the network. The lighter the color, the older the node is in the network. Hence, when a new node pops on the screen it can be distinguished from other nodes on the screen by its brighter (darker) color. If the time that a sensor node is alive in the WSN is t, and $\alpha \in [\gamma, 1]$ is the node brightness on the web application, then α is computed as:

$$\alpha = max(\gamma, min(1, \frac{c}{t})) \tag{3.1}$$

where c is a constant controlling how quickly the node color fades. Note that a lower bound of $\alpha = \gamma$ ensures that long-lived nodes always remain on the screen.

3.5 SWL Database 1

A MySQL database was designed for recording new node information and sensor readings on the Linux base station. The initial relational schema designed for SWL is defined in [27]. Initially there were 11 tables in the Sensor Web Language database. The id field is the primary key for all the tables. One more table 'SESSION' is added and four tables, 'SENSOR', 'SENSORNODE', 'GATEWAY', and 'POSIT' are

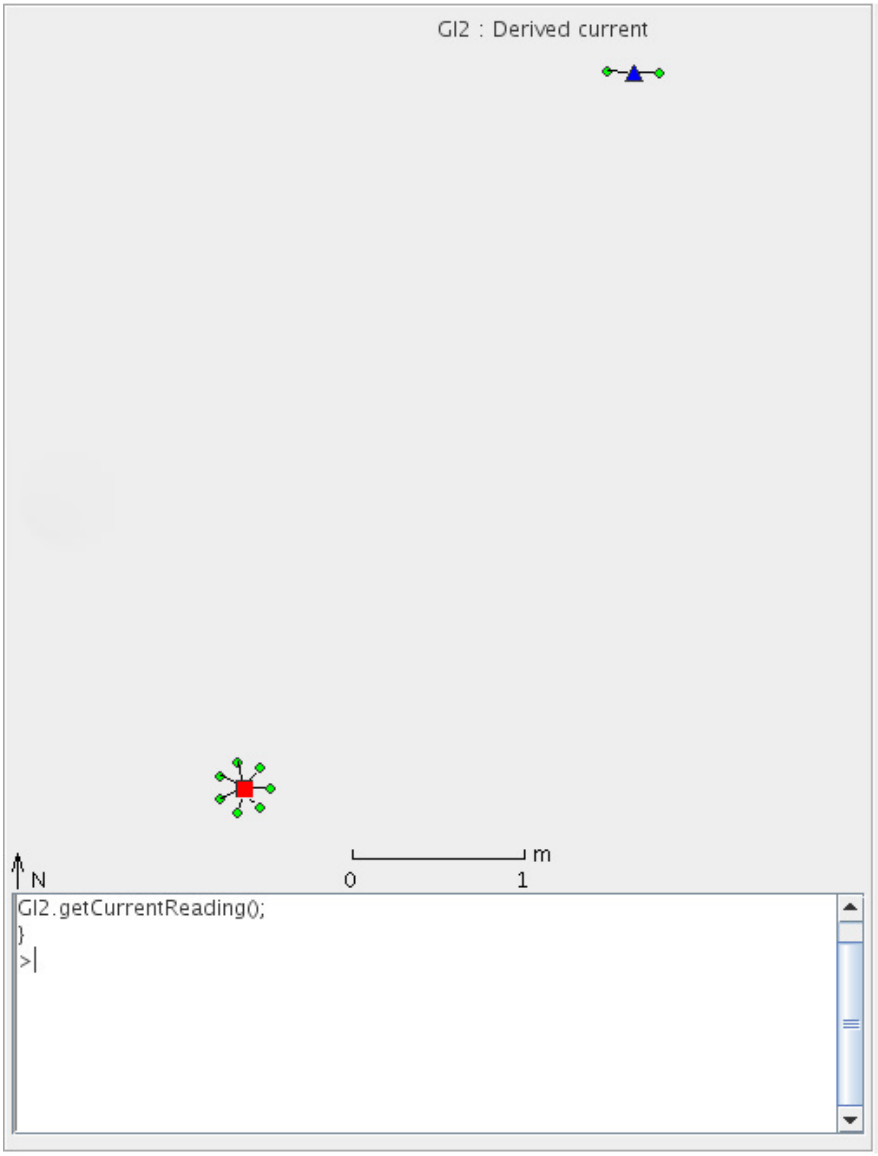


Figure 3.14: The web application 1 screen, with a gateway G, a sensor node RSN1, and sensors attached to both of them in the WSN.

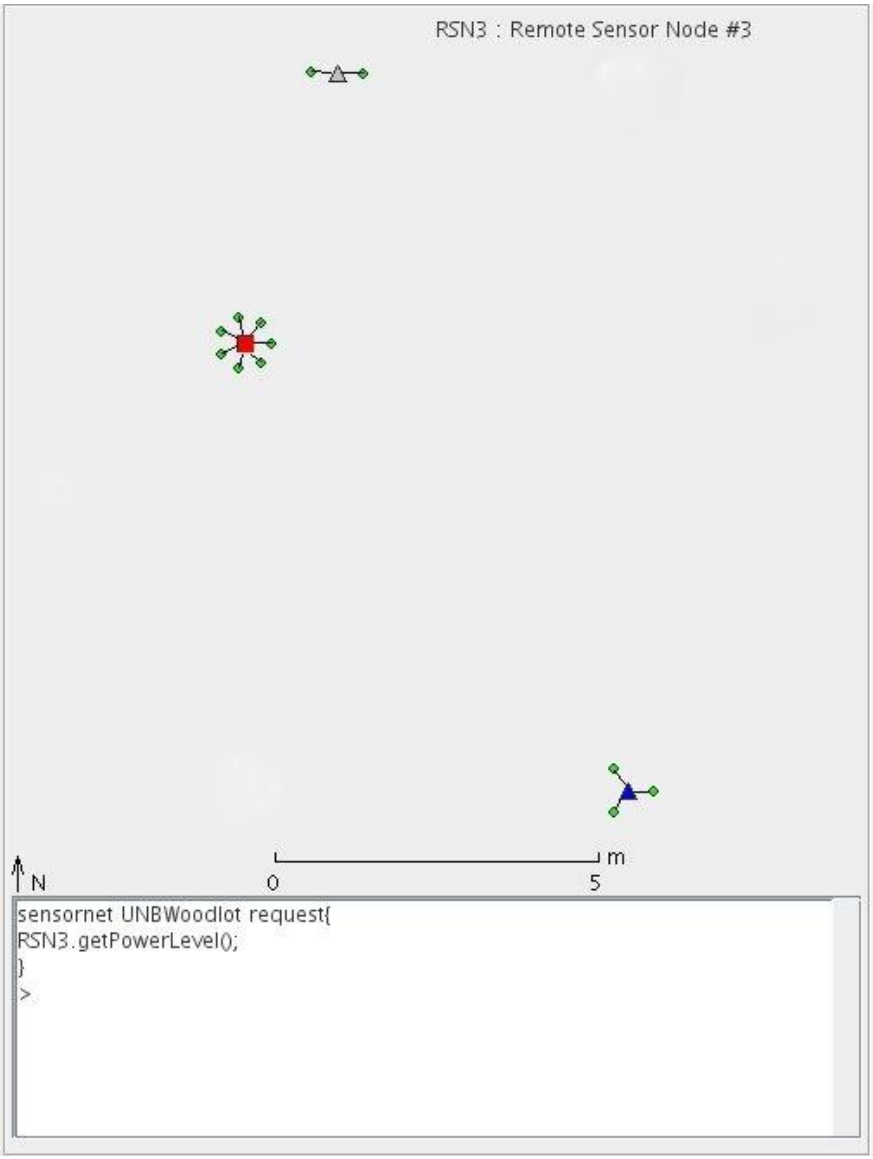


Figure 3.15: SWLBrowser when 1 more node RSN3 is added to the network, it can be seen that the previous node RSN1 in WSN is now grey in color.

modified. The updated relational database schema is shown in Figure 3.16. Each sensor node and sensor in the WSN can be uniquely identified by their UUID in the database. Every component in the WSN (e.g. sensor, sensornode and gateway) has a unique posit_id, which is the position in latitude and longitude of each component in the network. Fields in italics in Figure 3.16 are the new fields added to the database or the database tables to support the Announce operation.

1. SESSION

id	*SenderID*	*SessionID*	*datetime*

2. NETWORK

id	GCD_id	name	description

3. SENSOR

id	manufacturer	description	model	name	unit	attached_node	*posit_id*	*UUID*

4. SENSOR NODE

id	manufacturer	model	name	GCD_id	attached_GCD_id	Sensor_network_id	*posit_id*	*UUID*

5. GATEWAY

id	Sensor_network_id	Sensornode_id	type	GCC_id	*posit_id*

6. GATEWAYCD

id	manufacturer	model	name	Sensor_network_id	gateway_id

7. GCC

id	manufacturer	model	name	Sensor_network_id	gateway_id

8. CALIBRATION

id	start_date	end_date	derivation

9. COMP

id	comp_id	Observation_id

10. COMP_OBSERVATION

id	comp_value	calibration_id	timestamp

11. OBSERVATION

id	sensor_id	raw_value	derived_value	timestamp	calibration_id	no_of_readings

12. POSIT

id	component_id	latitude	longitude	height_above_sea	height_above_ground	start_date	end_date

Figure 3.16: Updated relational database schema designed for the SWL database (id is the primary key in each table), adapted from [27].

All SWL queries are run using MySQL embedded in SWLAnnounce.java, the base

station listener program. SWLAnnounce queries and updates four database tables. Algorithm 2 gives an overview of the database operations done by the SWLAnnounce.java program updating the SESSION, SENSORNODE, SENSOR and POSIT tables when a new node's information arrives at the base station.

As shown in Algorithm 2, SWLAnnounce does operations on four database tables. For the SESSION table, SWLAnnounce inserts the SessionID received by the base station in the SESSION table with the node_UUID and timestamp . SWLAnnounce checks if the node's UUID received by the base station is already present in the SENSORNODE table. If present, SWLAnnounce extracts the node's information from the table and if not present inserts node's UUID and node information into the table. SWLAnnounce also checks if all the sensor's UUIDs attached to the new node are present in the SENSOR table. If present, SWLAnnounce extracts the sensor information from the table otherwise SWLAnnounce inserts any new sensor UUIDs in the SENSOR table. In the POSIT table, SWLAnnounce checks if a new node's UUID is present, and if the end_date of the node is null. If so, SWLAnnounce then checks if the node's position is within a tolerance distance away from the previous location. If the node's position is changed (i.e. further than some tolerance distance away from the previous location), a new POSIT entry is created for the new location. If the node UUID is not at all present, then SWLAnnnounce creates a new node entry in the POSIT table.

```
databaseOperations(SenderID)
Data: Global AnnResponseMsg ARM
Result: New node information added to the SWL Database 1
// SESSION table
INSERT ARM.sessionID with the ARM.node_UUID and timestamp
// SENSORNODE table
if SenderID present then
    GET the node's name, manufacturer, description, model from
    SENSORNODE table
else
    INSERT the new ARM.node_UUID in the SENSORNODE table
    INSERT ARM.node_UUID, ARM.latitude, ARM.longitude and
    start_date = current_date, end_date = '0000-00-00' in POSIT table
end
// SENSOR table
forall the i=1 → ns do
    if sensor_UUID[i] present then
        GET the sensor's name, description, manufacturer, model and unit
        from SENSOR table
    else
        INSERT the ARM.sensor_UUID[i] in the SENSOR table
    end
end
// POSIT table
if node's dbend_date == '0000-00-00' then
    GET node's dblat and dblong from the database
    if |latitude − dblat| < ϵ && |longitude − dblong| < ϵ then
        start_date = current_date /* as a node is restarted, current
        date is it's new start date */
        GET the node's height_above_sea and height_above_ground
    else
        end_date= current_date
        INSERT node_UUID with new latitude and longitude and
        start_date = current_date in POSIT table
    end
end
```

Algorithm 2: Pseudo code describing database operations performed by program SWLAnnounce.java at the base station, this function is only called when an ANNOUNCE message is sent by a SN to the BS.

Chapter 4

Integration with Web Services

Web services are widely used in general purpose IT systems, such as business logic systems and databases. Web services provide a structured and interoperable mechanism for data acquisition, data storage, and data replication both within and outside of the sensor network, using W3C standards such as HTTP.

Sensor Observation Service (SOS) a part of Open Geospatial Consortium's (OGC) Sensor Web Enablement (SWE) group of specifications. SOS describes how applications and services will be able to access sensors of all types over the web. Specifically, SOS provides an API for managing deployed sensors and retrieving sensor data over the web.

Google Maps is a platform independent, web mapping service application that can be easily embedded in a browser using Google Map's API. The SOS API and the Google Maps API can be integrated to provide OGC standardized WSN information on a map. Showing WSNs on a Google map makes the implementation platform independent, so a WSN can be viewed in context on a map. The dynamic WSN structure can also be shown in context on the Google map.

This chapter first describes how Google Maps are integrated in the SWL web application 2, and then describes how the SOS standards are implemented to retrieve

information from WSNs with the map API controlling the scale, panning and background data.

4.1 Integration with Google Maps

Google Maps is a web mapping service application and technology provided by Google, that powers many map-based services. According to one of its creators (Lars Rasmussen), Google Maps is "a way of organizing the world's information geographically" [13, 34]. The Google Maps API requires JavaScript programming on the browser side, PHP programming on the server side and XML for data transfer between the browser and the server.

To incorporate Google Maps API in the existing adaptive web application code, the base station, database and web application programs were modified. The sensor node and the gateway programs remained the same. When a new node enters the WSN, it starts sending SWL messages to the base station. When the base station receives messages from a new node, it sends a RequestAnnounce message to that node; then that node sends an Announce message to the base station, in the same format as defined in Chapter 2. This messaging between the wireless sensor node and the base station is via the wireless gateway node attached to the base station through the serial port on the base station machine.

The modifications to the base station programs, the database and the web application programs are discussed in the following sections.

4.1.1 Base Station Version 2

Figure 4.1 shows the base station programs required for integration of the WSN structure with the Google Maps embedded web application. SWLAnnounce2.java is the central base station program. Similar to the SWLAnnounce.java program

discussed in Section 3.3.1, the SWLAnnounce2.java receives messages from wireless sensor nodes, and performs actions according to the SWL Message type. SWLAnnounce2.java differs in database operations from SWLAnnounce.java as explained in the following sections. SWLAnnounce2.java is not a client socket program, so AnnounceBSClient.java is not included in the base station programs.

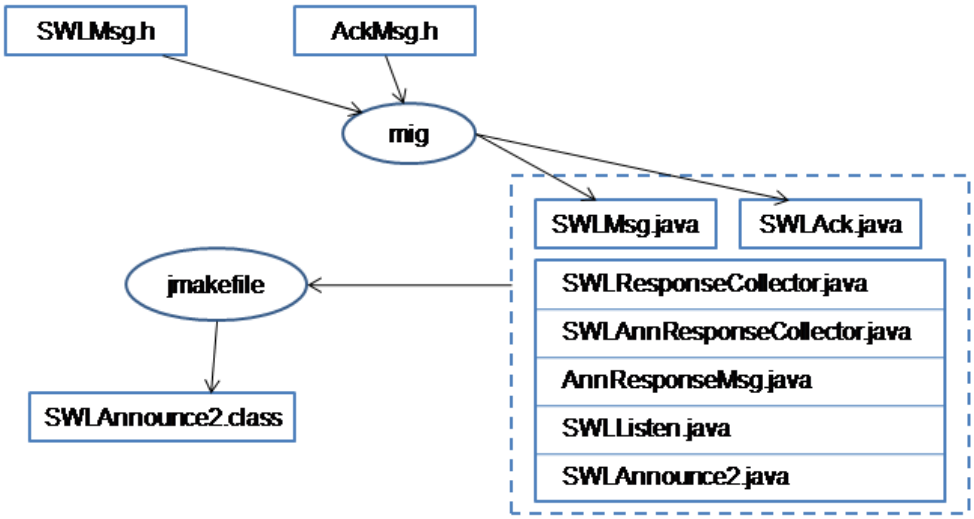


Figure 4.1: The Base station programs for integration of WSN with the Google Maps embedded web application.

4.1.2 SWL Database 2

Database operations play a central role for integrating Google Maps in the SWL web application. As in Chapter 3, the web application code communicates directly with the base station. When new node information arrives at the base station, it is passed onto the web application through a client-server socket connection between the two. In the web services implementation, the base station and the web application communicate though the database. Whenever the base station receives new node information, it saves this information in the database and the web application extracts the network information from the database in a polling process.

In the SWL database a new table 'SENSOR_SN' is added and the 'OBSERVATION'

and the 'SENSOR' tables are modified as shown in Figure 4.2. The SENSOR_SN table provides the attached relation between sensor nodes and sensors. Nodes and sensors are uniquely identified by their UUIDs. In the OBSERVATION table the data type of the sensor_id field is updated to char(16) to permit storage of the sensor UUID in it. In the SENSOR table, field attached_node_id is removed.

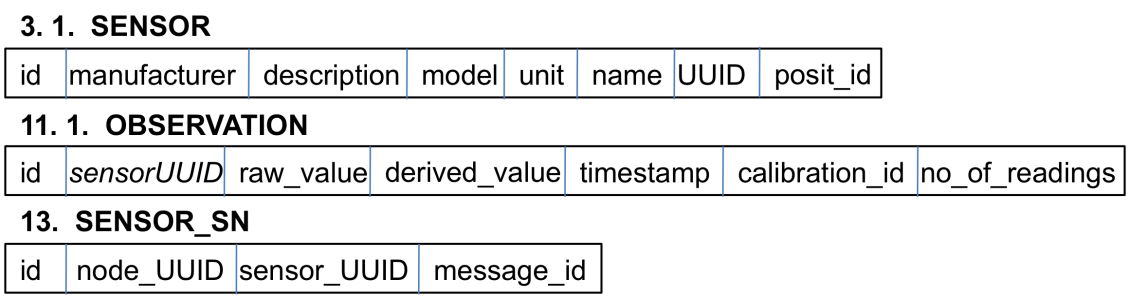


Figure 4.2: Modifications to the SWL database of Section 3.5.

Algorithms 3 and 4 describe the database operations performed by the SWLAnnounce2.java program.

messageReceived(int to, SWLMsg m)
Data: SWLListen Listen
Result: calibrated value recorded in database, Announce messages are processed

if *m.swlMsgType == 2 || m.swlMsgType == 3* **then** `/* 2 is RESPONSE and 3 is REPORT message types */`

    Listen.logSensorReport(m.senderID, m.swlSeqNo, m.payload)
    value = calibrate(m)
    INSERT m.Sensor_UUID, m.raw_value, value and timedate in the OBSERVATION table

else if *m.swlMsgType == 15* **then** `// 15 is ANNOUNCE message type`

    databaseOperations(m.senderID)

}

Algorithm 3: Pseudo code describing how database operations function is invoked in SWLAnnounce2.java program

As shown in Algorithms 3 and 4, SWLAnnounce2.java interacts with six database tables while receiving messages from the wireless sensor nodes, as discussed in more

databaseOperations(senderID)
Data: Global AnnResponseMsg ARM
Result: SWL Database 2 is updated
`// SESSION table`
INSERT ARM.sessionID, ARM.node_UUID and timestamp in SESSION table
`// SENSORNODE table`
if *SenderID is not present* **then**
  INSERT the ARM.node_UUID the SENSORNODE table & INSERT ARM.node_UUID, ARM.sensor_UUID[i=1→ns] in SENSOR_SN table.
end
`// SENSOR table`
forall the *sensors*[*i=1→ns*] **do**
  if *ARM.sensor_UUID*[*i*] *is not present* **then**
    INSERT ARM.sensor_UUID[i] in the SENSOR table & INSERT ARM.sensor_UUID[i] in the SENSOR_SN table.
  end
end
`// POSIT table`
if *node's dbend_datetime == '0000-00-00 00:00:00'* **then**
  GET the node_UUID's dblat and dblong from POSIT table **if** $|latitude - dblat| < \alpha AND |longitude - dblong| < \alpha$ **then**
    dbstart_datetime = current_datetime
  else
    INSERT ARM.node_UUID, dblat, dblong, start_datetime = current_datetime and dbend_datetime = '0000-00-00 00:00:00' in POSIT table
  end
else
  INSERT ARM.node_UUID, ARM.latitude, ARM.longitude, dbstart_datetime = current_datetime and dbend_datetime = '0000-00-00 00:00:00' in POSIT table
end

Algorithm 4: Pseudo code describing database update operations performed by program SWLAnnounce2.java for a received SWLAnnounce message.

detail below.

1. For the SESSION table, if the message type of the messages received by the base station is an ANNOUNCE message, the SessionID sender node is inserted in the SESSION table with the node_UUID and timestamp,

2. For the OBSERVATION table, if the message type of the messages received by the base station is either RESPONSE or REPORT, the readings sent by the wireless sensor node and the sender sensor UUID are processed. The raw readings are converted into derived values by applying relevant calibrations (calibrate(m) method of SWLListen), and then inserted into OBSERVATION table,

3. For the SENSORNODE table, the new node's UUID received by the base station is checked to see if it is already present in the SENSORNODE table. If not the node's UUID and node information are inserted into the table,

4. For the SENSOR table, the databaseOperations method checks if all the sensors UUID attached to the new node are present in the SENSOR table. If not present, the method inserts the new sensor UUID and the sensor information into the SENSOR table,

5. For the POSIT table, the node is checked to see if it has moved since the previous SWLAnnounce message received for this node. If the node has moved, the previous position is ended (and the time and date noted), and the new position is added to the POSIT table,

6. For the SENSOR_SN table, all the sensors are checked to make sure they are attached to their sensor node.

4.1.3 Web Application 2

The Web application code is written in JavaScript and HTML so that Google maps can be embedded in the browser. The database queries are done using server side PHP programming and XML files are sent as a response from the server to the browser. The Web application side programs and communication between them and the database is shown in Figure 4.3.

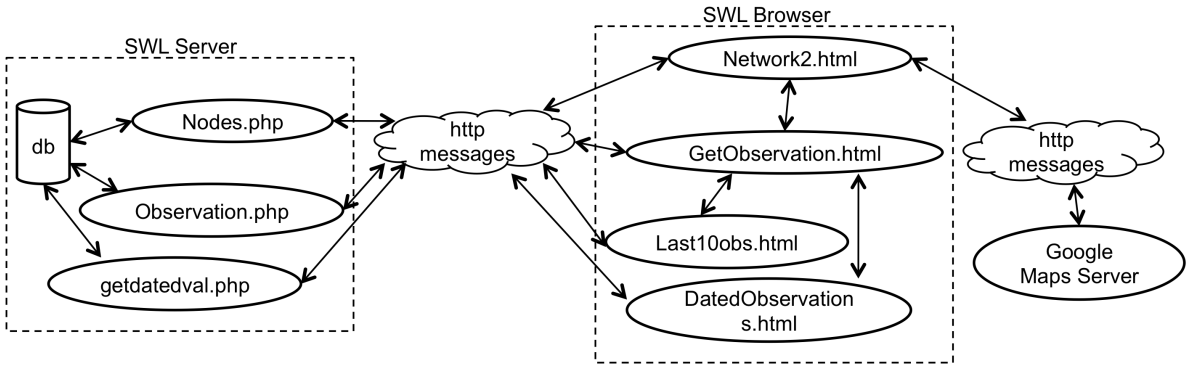


Figure 4.3: Communication between the web application programs (in dashed box) and the server side PHP programs and the database.

Program Network2.html embeds Google Maps in the web application. Network2.html invokes, the Nodes.php program which interfaces with the database and gets the WSN structure from the database. Nodes.php then passes this network information to Network2.html in XML format. On receiving a response from Nodes.php, Network2.html creates the GoogleMap, the nodes and the placeholders for the nodes in the map, and then displays them on the web application screen. This whole process is repeated automatically in a polling loop at fixed intervals (e.g. every 30 seconds). Continuous polling is done to keep the WSN structure updated on the web application.

Through the Network2.html program on the web application, the latest values sent by each sensor attached to a node can be viewed. A 'Show Observations' link is attached to each sensor for each node. On clicking the Show Observations URL, the Network2.html program opens another html page called GetObservation.html.

GetObservation.html calls the Observation.php program on page load. Observation.php extracts the last 10 observations of the sensor from the database and sends them back to the GetObservation.html program in XML format. The GetObservation.html program builds the page and displays the values in a table on the web application screen. The last 10 observations can also be seen in a comma separated text file format by clicking on the 'Export Observation' button on the "Show Observations" screen. Observations can also be viewed for a particular date range by choosing the start and the end day and then clicking on the 'Get Observations' button. The result is then displayed again in a comma separated text file format on another browser screen.

An example Network2.html screen is shown in Figure 4.4. In Figure 4.4, two nodes in the network can be seen. Clicking on one of the nodes provides a description of that node and the description of the sensors attached to the node can also be seen. Dynamic adaptation is also displayed by fading nodes with color strength β proportional to the time they have not been heard from, using the following equation:

$$\beta = max(\beta_{min}, \frac{T}{(T + max(0, t - T))}) \tag{4.1}$$

where T is the number of hours for which not hearing from a node is normal, and t is the number of hours since a message has been received from a node. Note that $(t\text{-}T) = 2T$ is the time at which the color strength will be 0.5. A threshold β_{min} is used to avoid disappearance of the node. If an old node becomes active (starts sending messages), then its color strength is reset to $\beta = 1$. The web application can thus show the dynamic network structure. If a new node enters the network, t will be small and β will be 1. A new node can also be distinguished from the existing nodes in the network.

Figure 4.5 displays the screen shown after clicking the Show Observations link for sensor name GV4. The last 10 observations of the battery voltage sensor are shown in

Figure 4.4: Sample Network2.html screen, displaying 2 nodes in the network in the bubble.

a table, Export Observation exports the last 10 readings in CSV format, Get Observation shows readings in between a time range in CSV format, SOS GetCapabilities metadata link displays the XML output sent from the server to the GetCapabilities request sent by the user, SOS DescribeSensor link displays SOS format metadata output describing the battery voltage sensor, SOS GetObservation metadata displays the observations in SOS format.

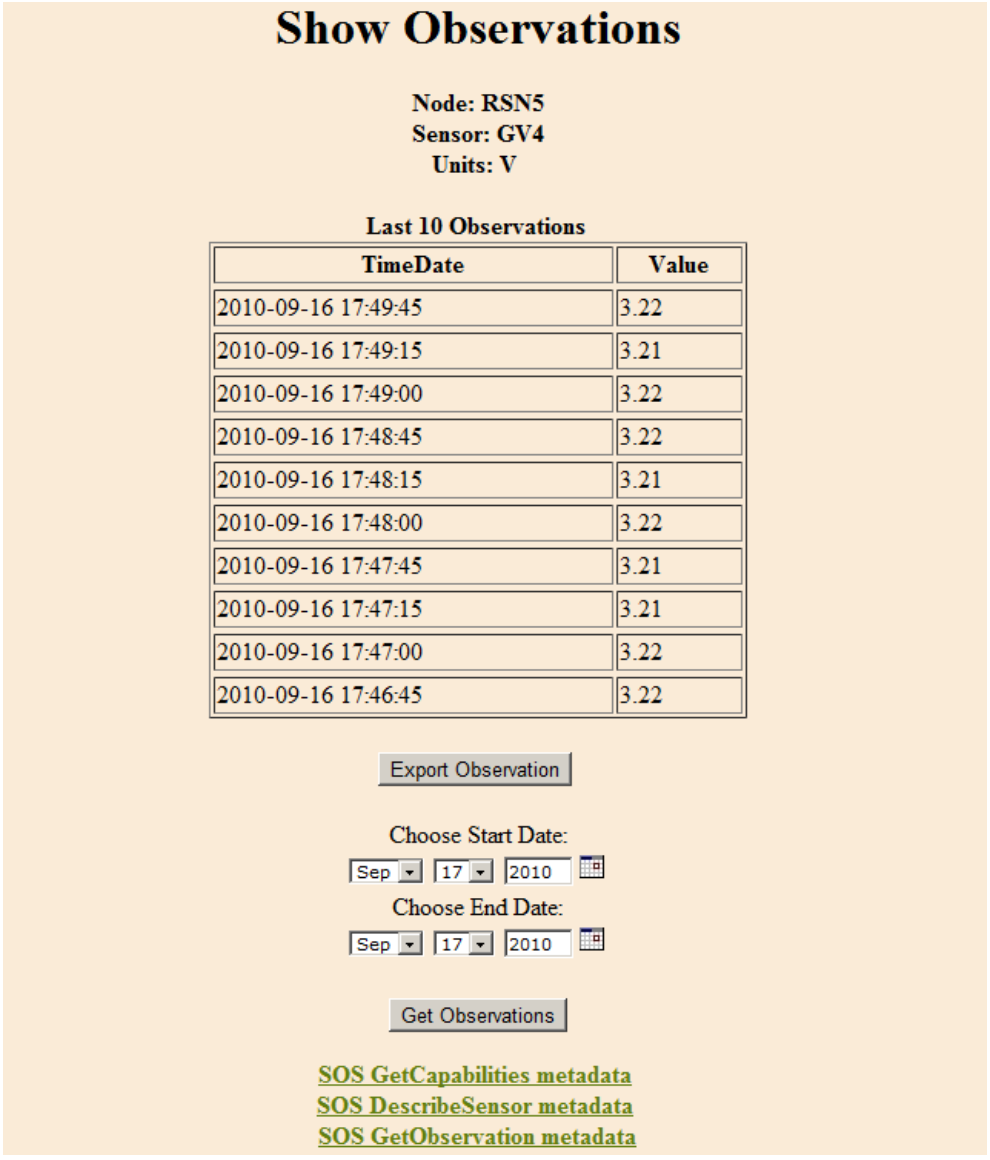


Figure 4.5: Sample Observation2.html screen, for sensor name GV4 attached to the node name RSN5.

4.2 Sensor Observation Service Integration

As described in Chapter 1, the goal of a Sensor Observation Service (SOS) is to provide access to observations from sensors and sensor systems in a standard way that is consistent for all sensor systems including remote, in-situ, fixed and mobile sensors. A Sensor Observation Service provides an API for managing deployed sensors, retrieving sensor data and sensor "observation" data. The SOS API, like the Google Maps API, has web application code written using JavaScript programming in html, server side PHP programming communicating with the database, and the server side sending data to the web application side in XML format.

SOS has three mandatory "core" operations: GetObservation, DescribeSensor, and GetCapabilities. The GetObservation operation provides access to sensor observations and measurement data via a spatio-temporal query that can be filtered by a phenomena. The DescribeSensor operation retrieves detailed information about the sensors and processes generating the measurements. The GetCapabilities operation provides the means to access SOS metadata. Several optional, non-mandatory operations have also been defined. There are two operations called RegisterSensor and InsertObservation, to support transactions, and six enhanced operations, including GetResult, GetFeatureOfInterest, GetFeatureOfInterestTime, DescribeFeatureOfInterest, DescribeObservationType, and DescribeResultModel.

Our focus is mainly on the core operations. A SOS is integrated with the above Google Maps code. Figure 4.6 illustrates the integration of SOS with the web application 2. As seen in Figure 4.6, the SOS messages are sent using http 'GET' and 'POST' requests. An SOS consumer can also send the SOS messages to SOS server to get SOS metadata. SOS metadata for GetObservation, DescribeSensor and GetCapabilities is also shown in the GetObservation.html page for every sensor. The SOS php file is called "server.php". Server.php communicates with the SWL database to form a SOS metadata file for all three operations. The metadata file

is in XML format. An SOS consumer can also use the SOS metadata file directly. The SOS messages sent from an SOS consumer or the web application is in a request form to the database. The format of these messages is defined by OGC [21] and is different for each type of SOS message.

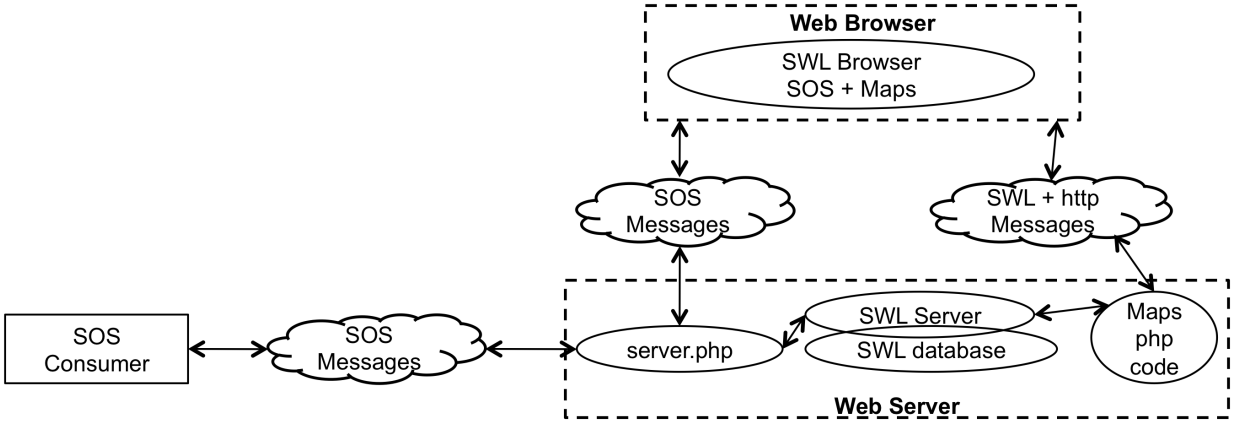


Figure 4.6: Integration of SOS with the existing Google Maps code and SWL database.

4.2.1 GetCapabilities Request

The GetCapabilities request allows the client to retrieve service metadata about a specific service instance. For an SOS service, it identifies such things as offerings and observed properties available, as well as information on sensors that are used.
Using a web application, parameters SERVICE=SOS&REQUEST=GetCapabilities are added to the end of the URL where SOS is hosted.
An example GetCapabilities request is shown in Figure 4.7.

```
http://ib214m09.cs.unb.ca/html/sos/server.php?service=SOS&request=GetCapabilities
```

Figure 4.7: A sample GetCapabilities request.

A part of the response sent by the SOS server for the GetCapabilities request is

shown in Figure 4.8 and rest is attached in Appendix E.2.

4.2.2 DescribeSensor Request

The DescribeSensor request gives a client the ability to retrieve the characteristics of a particular sensor and return the information in a SensorML XML document. In this implementation, the request is sent to the server and then a SensorML XML document is generated by the server as the response.

The following is the list of possible parameters for a DescribeSensor request [8]:

request : (Required) value must be"DescribeSensor",

service : (Required) value must be "SOS",

version : (Required) value must be "1.0.0", version specifies the SOS version according to which the server.php code is written,

procedure : (Required) this is the sensor UUID, to identify each sensor uniquely in the WSN,

outputformat : (Required) the format encoding to be returned by the server.

A sample DescribeSensorRequest for a sensor with id `391e816927484bcb` is shown in Figure 4.9:

The output for the above request is shown in Figure 4.10.

4.2.3 GetObservation Request

The GetObservation request is designed to query sensor systems to retrieve observation data in the form defined in the Observation and Measurement specification (O&M). Upon receiving a GetObservation request, a SOS shall either satisfy the request or return an exception report.

The following is a list of the possible parameters for a GetObservation request [8]:

```
-<Capabilities xsi:schemaLocation="http://www.opengis.net/sos/1.0
 http://schemas.opengis.net/sos/1.0.0/sosAll.xsd" version="1.0.0">
 -<ows:ServiceIdentification>
   <ows:Title>UNB FCS Test SOS</ows:Title>
    -<ows:Abstract>
      University of New Brunswick (UNB), Faculty of Computer Science (FCS)
    </ows:Abstract>
    -<ows:Keywords>
      <ows:Keyword>Room Temperature</ows:Keyword>
      <ows:Keyword>Voltage</ows:Keyword>
      <ows:Keyword>Solar</ows:Keyword>
    </ows:Keywords>
   <ows:ServiceType codeSpace="http://opengeospatial.net">OGC:SOS</ows:ServiceType>
   <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
   <ows:Fees>NONE</ows:Fees>
   <ows:AccessConstraints>NONE</ows:AccessConstraints>
   </ows:ServiceIdentification>
 -<ows:ServiceProvider>
 <ows:ProviderName>UNB FCS Test</ows:ProviderName>
 <ows:ProviderSite xlink:href="http://ib214m09.cs.unb.ca/html/sos/server.php?
                               service=SOS\&request=GetCapabilities"/>
 -<ows:ServiceContact>
   <ows:IndividualName>Gunita Saini</ows:IndividualName>
   <ows:PositionName>Student</ows:PositionName>
    -<ows:ContactInfo>
     -<ows:Phone>
       <ows:Voice>506-453-4566</ows:Voice>
       <ows:Facsimile>(506) 453-3566</ows:Facsimile>
     </ows:Phone>
     -<ows:Address>
       <ows:DeliveryPoint>540 Windsor Street</ows:DeliveryPoint>
       <ows:City>Fredericton</ows:City>
       <ows:AdministrativeArea>NB</ows:AdministrativeArea>
       <ows:PostalCode>E3B5A3</ows:PostalCode>
       <ows:Country>Canada</ows:Country>
       <ows:ElectronicMailAddress>www.unb.ca</ows:ElectronicMailAddress>
     </ows:Address>
    </ows:ContactInfo>
   </ows:ServiceContact>
 </ows:ServiceProvider>
```

Figure 4.8: A part of response to GetCapabilities request.

```
http://ib214m09.cs.unb.ca/html/sos/server.php?request=DescribeSensor&service=SOS
&version=1.0.0&outputformat=text/xml;subtype=sensorML/1.0.0&procedure=
urn:swl:station:wmo:391e816927484bcb:
```

Figure 4.9: A sample DescribeSensor request for a sensor with id 391e816927484bcb.

request : (Required) value must be "GetObservatio".

service : (Required) value must be "SOS".

version : (Required) value must be "1.0.0", version specifies the SOS version according to which the server.php code is written.

offering : (Required) The offering contains a unique sensor UUID.

observedProperty : (Required) The type of sensor, in our case it is either air temperature, voltage or solar radiation sensors.

responseformat : (Required) The format / encoding to be returned by the response. The format can be either 'text/xml', 'text/csv', 'text/tab-separated-values', 'kml format'.

eventTime : (Optional) Specifies the time period for which observations are requested.

procedure : (Optional) The procedure specifies the sensor system used. In this implementation, the procedure is equivalent to the sensor id (UUID) that will be returned from a DescribeSensor request.

An example GetObservation request for sensor `c6cf046dc1fa4e63` is shown in Figure 4.11.

The response is in text/xml format having voltage observations for sensor with id `c6cf046dc1fa4e63`. A portion of the sample response is shown in Figure 4.12 and the rest is attached in the Appendix E.3.

```
<sml:SensorML xsi:schemaLocation="http://www.opengis.net/sensorML/1.0.1
http://schemas.opengis.net/sensorML/1.0.1/sensorML.xsd" version="1.0.1">
    -<sml:member>
     -<sml:System gml:id="station-391e816927484bcb">
      -<sml:identification>
       -<sml:IdentifierList>
        -<sml:identifier name="StationId">
         -<sml:Term>
          <sml:value>urn:swl:station:wmo:391e816927484bcb:</sml:value>
         </sml:Term>
        </sml:identifier>
       </sml:IdentifierList>
      </sml:identification>
     -<sml:contact xlink:role="urn:ogc:def:classifiers:OGC:contactType:operator">
      -<sml:ResponsibleParty>
        <sml:organizationName>Crossbow Tech Inc.</sml:organizationName>
       </sml:ResponsibleParty>
      </sml:contact>
     -<sml:positions>
      -<sml:PositionList>
       -<sml:position name="stationPosition">
        -<swe:Position referenceFrame="urn:ogc:crs:EPSG:4326">
         -<swe:location>
          -<swe:Vector gml:id="STATION_LOCATION"
           definition="urn:ogc:def:property:OGC:location">
           -<swe:coordinate name="latitude">
            -<swe:Quantity axisID="Y">
              <swe:uom code="deg"/>
              <swe:value>45.949722</swe:value>
             </swe:Quantity>
           </swe:coordinate>
          -<swe:coordinate name="longitude">
           -<swe:Quantity axisID="X">
             <swe:uom code="deg"/>
             <swe:value>-66.642778</swe:value>
           </swe:Quantity>
          </swe:coordinate>
         </swe:Vector>
        </swe:location>
       </swe:Position>
      </sml:position>
     </sml:PositionList>
    </sml:positions>
   </sml:System>
  </sml:member>
 </sml:SensorML>
```

Figure 4.10: Example Response to SOS describeSensor request, for sensor with UUID 391e816927484bcb.

```
http://ib214m09.cs.unb.ca/html/sos/server.php?request=GetObservation&service=SOS
&responseformat=text/xml;schema=swl/0.6.1&observedproperty=Voltage
&offering=urn:swl:station:wmo:c6cf046dc1fa4e63:
```

Figure 4.11: An example GetObservation request for sensor `c6cf046dc1fa4e63`.

```
-<om:CompositeObservation xsi:schemaLocation="http://www.opengis.net/om/1.0
 http://www.csc.noaa.gov/swl/schema/swl-DIF/swl/0.6.1/schemas/
 swlObservationSpecializations.xsd"
 gml:id="VoltagePointCollectionTimeSeriesObservation">
  -<gml:description>
    Voltage observations at a point at a series of times
  </gml:description>
  -<gml:name>
    Voltage observations at station ID 391E816927484BCB
  </gml:name>
  -<gml:boundedBy>
    -<gml:Envelope srsName="urn:ogc:def:crs:epsg::4326">
        <gml:lowerCorner>45.949722 -66.642778</gml:lowerCorner>
        <gml:upperCorner>45.949722 -66.642778</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  -<om:samplingTime>
    -<gml:TimePeriod gml:id="ST">
        <gml:beginPosition>2010-06-24T13:25:24Z</gml:beginPosition>
        <gml:endPosition>0000-00-00T00:00:00Z</gml:endPosition>
    </gml:TimePeriod>
  </om:samplingTime>
  -<om:procedure>
   -<om:Process>
    -<swl:CompositeContext gml:id="SensorMetadata"
        recDef="http://www.csc.noaa.gov/swl/schema/swl-DIF/swl/
               0.6.1/recordDefinitions/
        PointSensorMetadataRecordDefinition.xml">
            -<gml:valueComponents>
                <swl:Count name="NumberOfStations">1</swl:Count>
                  -<swl:ContextArray gml:id="StationArray">
                    -<gml:valueComponents>
                      -<swl:CompositeContext gml:id="Station1Info">
                        -<gml:valueComponents>
                            <swl:StationName>Voltage</swl:StationName>
                            <swl:Organization>78e89380bffe11de</swl:Organization>
                            <swl:StationId>urn:swl:station:wmo:391e816927484bcb:
                            </swl:StationId>
                            -<gml:Point gml:id="Station1LatLon">
                              <gml:pos>45.949722 -66.642778</gml:pos>
                            </gml:Point>
                            <swl:VerticalDatum>urn:ogc:def:datum:epsg::5113
                            </swl:VerticalDatum>
                            <swl:VerticalPosition xsi:nil="true" nilReason="missing"/>
                            <swl:Context name="TotalWaterDepth" uom="m">116
                            </swl:Context>
                            <swl:Count name="Station1NumberOfSensors">1
                            </swl:Count>
```

Figure 4.12: Part of a response to a GetObservation request.

Chapter 5

Experimental Evaluation

To evaluate the SWL Announce system, we designed and implemented an experiment. This experiment checked the dynamic appearance of the sensor nodes in the web application 2 when a new node was added in the WSN. The active and inactive sensor nodes in the network were displayed in the web application using a color coding scheme as defined in Section 5.2.1.

5.1 Experimental Equipment

Three environmental sensors, seven sensor nodes, six data acquisition boards, one sensor programming board and a base station computer were used in this experiment.

5.1.1 Sensors

The air temperature sensor used in the experiment was a 10 kΩ Negative Coefficient (NC) temperature sensor called a thermistor. Thermistors are widely used for environmental monitoring due to their high measuring precision (e.g. 0.1 °C for accurate reference voltage). This sensor takes in an external excitation voltage of 2.5 Volts and outputs voltage values that have been calibrated to represent temperature values to a tenth of a degree. Figure 5.1 shows a picture of the air temperature sensor.

Table 5.1: Equipment used for evaluating SWL Adapative experiment, Q is the required quantity of the equipment.

Type	Manufacturer	Model	Units	Range	Q
Temperature Sensor	BC Components	232264055103	°C	-40 to 125	2
Solar Radiation Sensor	Hamamatsu Corp.	Si Photodiode S1133	W/m^2	0 to 1000	2
Voltage Sensor	Crossbow Technology, Inc.	ADC	V	0-16	6
Sensor Node	Crossbow Technology	Mica2 MPR400 (916 MHz)	N/A	N/A	7
Data Acquisition Board	Crossbow Technology, Inc.	MDA300CA	N/A	N/A	6
Sensor Program -ming Board	Crossbow Technology, Inc.	MIB510 (RS-232)	N/A	N/A	1
Base Station	IBM	Pentium 4 (Ubuntu, version 8.04)	N/A	N/A	1

Figure 5.2 shows a circuit diagram of the air temperature sensor.

The solar radiation (SR) sensors measure the photons received from a light source. They mainly include quantum sensors for measuring Photosynthetically Active Radiation (PAR) in the 400 to 700 nm waveband. This is radiation close to what the human eye sees. Solar radiation sensors can be connected to the board using a shunt resistance. Figure 5.3 shows a picture of the solar radiation sensor and Figure 5.4 shows a circuit diagram of a general solar radiation sensor.

5.1.2 Mica2 Sensor Node

Mica2 mote is a wireless sensor platform manufactured by Crossbow Technology Inc. It consists of three key parts: the radio frequency module, the processing module, and the sensing module. Mica2 uses the CC1000 single-chip RF transceiver (ChipCon model). A CC1000 chip on Mica2 supports 868/916MHz radio channels. Mica2 has a built-in battery voltage channel, which can be used for both battery monitoring and voltage referencing. A 51-pin expansion connector in Mica2 allows

Figure 5.1: Air temperature sensor with thermistor embedded in expoxy and attached to a shielded cable.

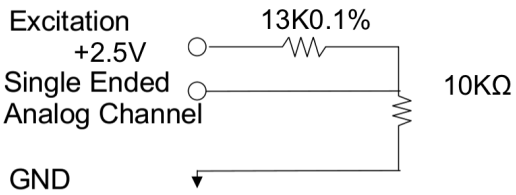


Figure 5.2: Air temperature sensor circuit diagram (from CENS [12]).

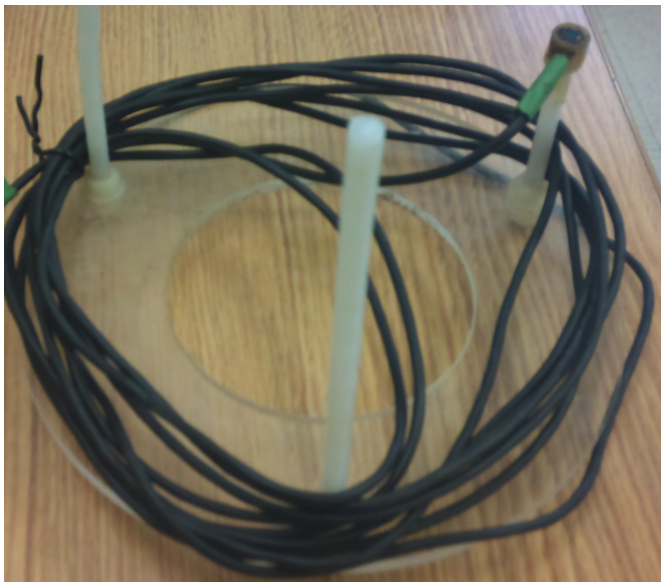


Figure 5.3: Solar Radiation sensor encased in expoxy and mounted on top of a circular rod.

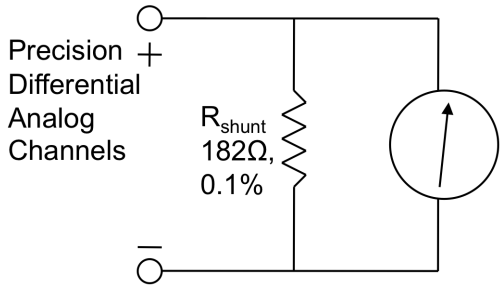


Figure 5.4: Solar Radiation sensor circuit diagram (from CENS [12]).

light, temperature, pressure, acceleration, and other external sensors to be connected to it [6]. Figure 5.5 shows a picture of the Mica2 sensor mote.

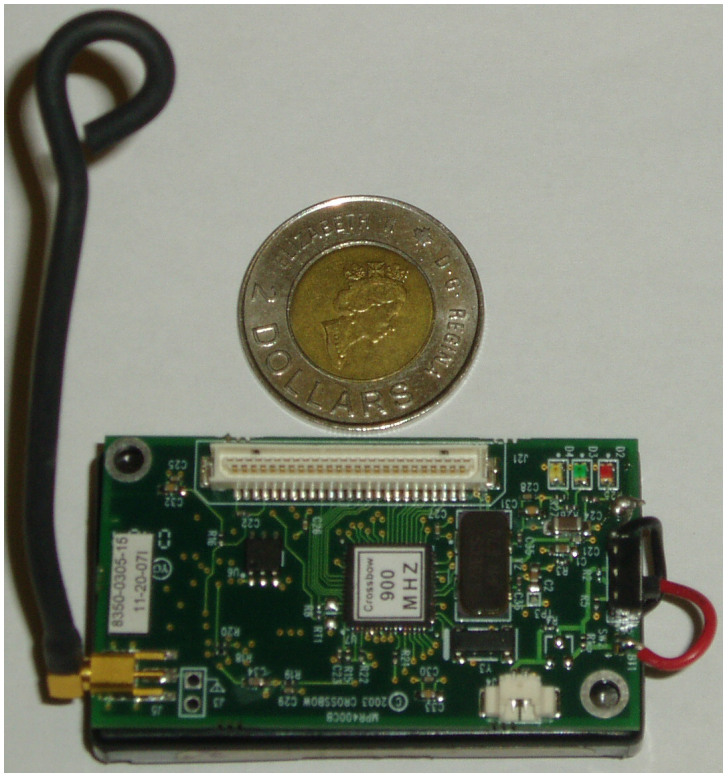


Figure 5.5: Mica2 sensor node.

The processing module of Mica2 uses the Atmel AVR ATMEGA128L architecture, which is a Reduced Instruction Set Computer (RISC) with an 8 bit ALU/data-path, 128K bytes of flash code memory (ROM), and 8K bytes of SRAM data memory (RAM). The current draw is 8 mA in active mode, and less than 15 μA in sleep mode. Figure 5.6 shows a block diagram of the AVR architecture. The AVR architecture uses a Harvard architecture. The program and data use separate memory modules and buses. The processor can fetch instructions directly from the flash memory without first loading them into RAM.

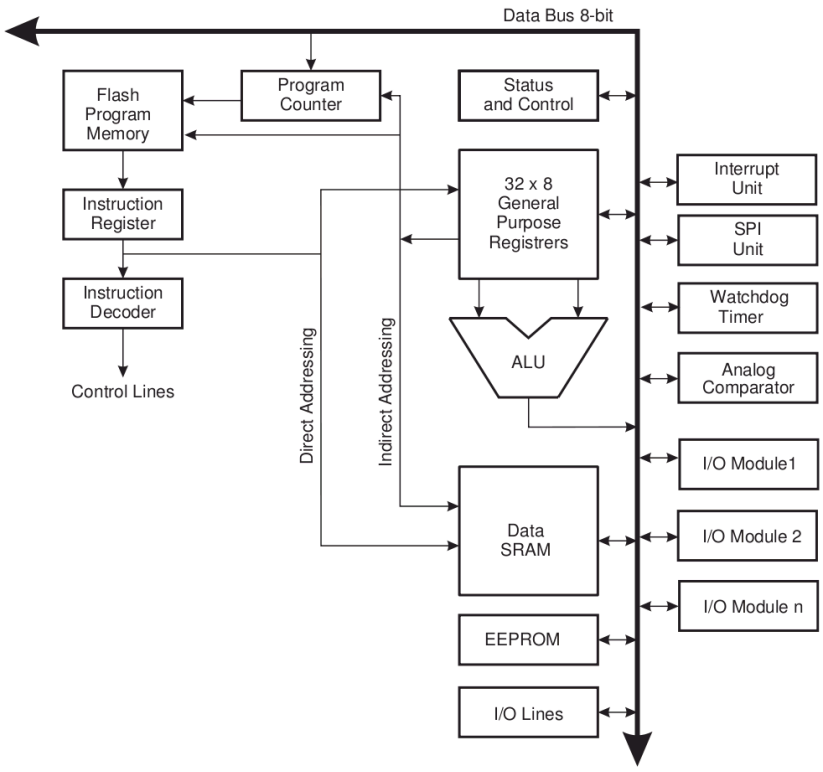


Figure 5.6: Block diagram of the AVR architecture (from [9]).

5.1.3 MDA300 Data Acquisition Board

MDA300CA is designed as a general measurement platform for the MICAz and MICA2. The top view of MDA300CA board is shown in Figure 5.7. The MDA300CA data acquisition board can host a Mica2 mote through its 51-pin connector, and interface with external sensors through its digital and analog channels. Analog sensors can be attached to different channels based on the expected precision and dynamic range. Digital sensors can be attached to the provided digital or counter channels. Figure 5.8 shows a sample lab experiment setup containing 5 Mica2s installed on 5MDA300 boards with attached sensors and 1 gateway node.

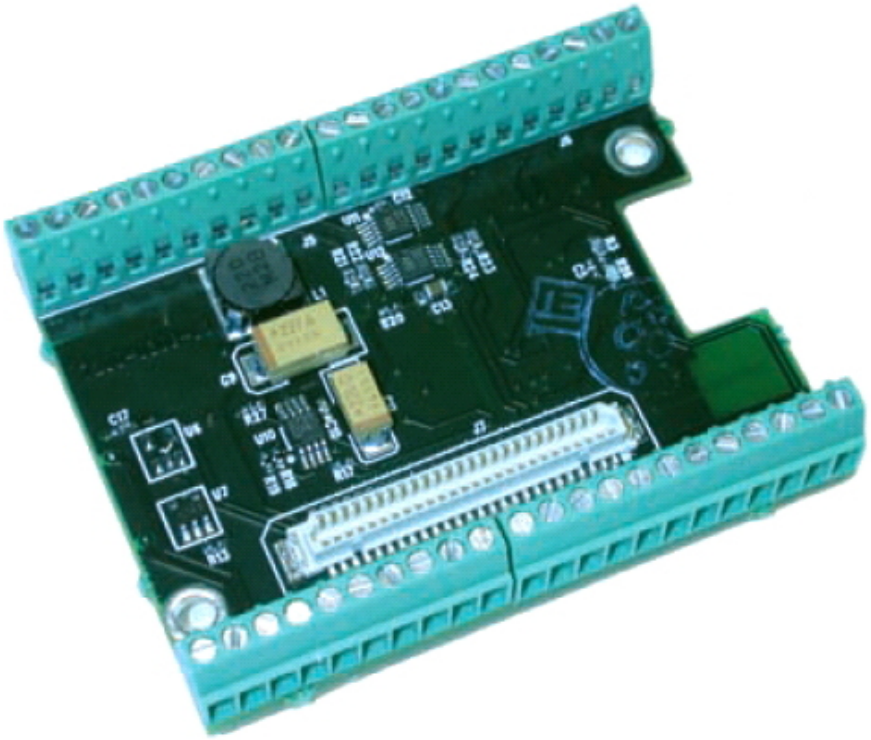


Figure 5.7: Top view of MDA300CA data acquisition board. This is the side a Mica2 mote would be attached to (from [10]).

Figure 5.9 shows how to connect the air temperature sensor and the solar radiation sensor on MDA300 data acquisition board. The air temperature sensor is connected to a 2.5V voltage excitation channel and a single-ended analog channel (e.g. channel

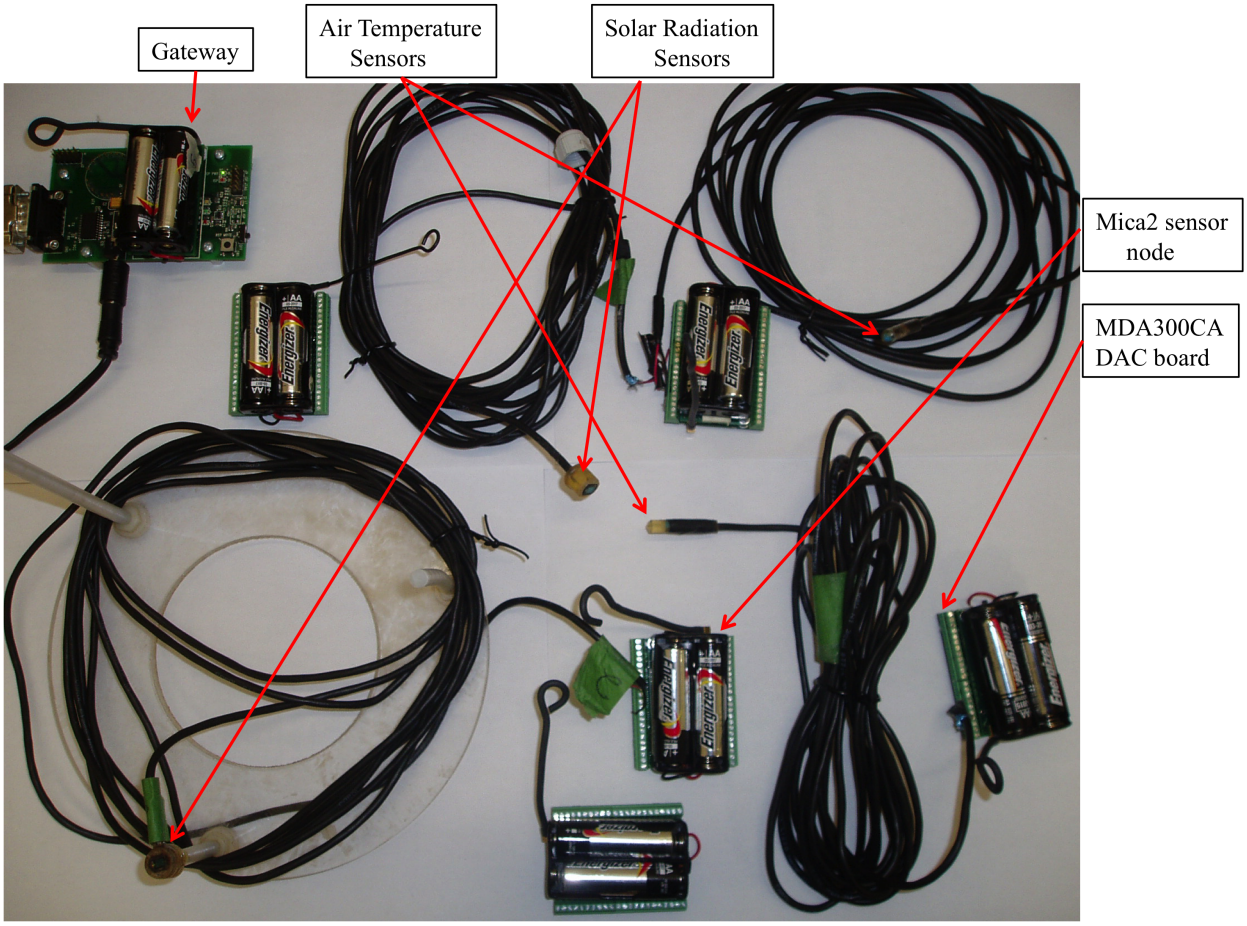


Figure 5.8: A sample lab experiment setup showing 5 Mica2s installed on 5 MDA300 boards with attached sensors and 1 gateway node.

1), which has reference voltage of 2.5V.

The solar radiation sensor is connected to analog channel 4. Unlike an air temperature sensor, the solar radiation sensor does not need an external excitation voltage.

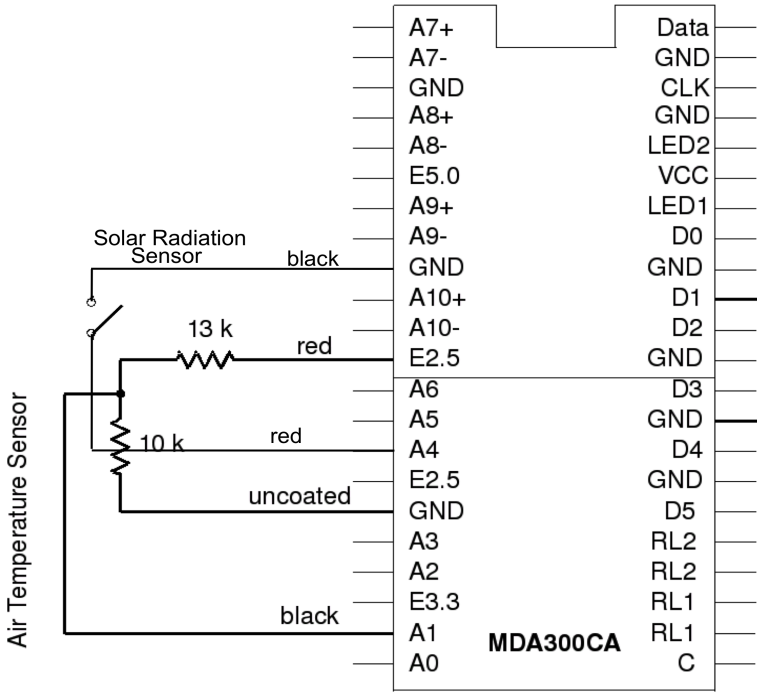


Figure 5.9: MDA300CA schematic diagram for air temperature sensor and solar radiation sensor connections.

5.2 Experimental Design

A lab experiment was conducted in the IB214 lab in the Information Technology Center in the University of New Brunswick. The experiment results are displayed on web application 2 for a clearer view of the simulated changing network structure. A Pentium IV 3.0 GHz IBM machine, IB214M09 was used as a base station. It is preinstalled and configured to handle multiple Java Servlet requests with the Tomcat server version 5 and a relational database MySQL version 3.23.58. PHP version 6 is also installed for development.

An experiment was setup consisting of six mica2 sensor nodes, one gateway node, six MDA300CA boards, two air temperature sensors and two solar radiation sensors. Each sensor node gets power from two AA batteries. Figure 5.10 shows the experimental sensor network design in the IB214 lab.

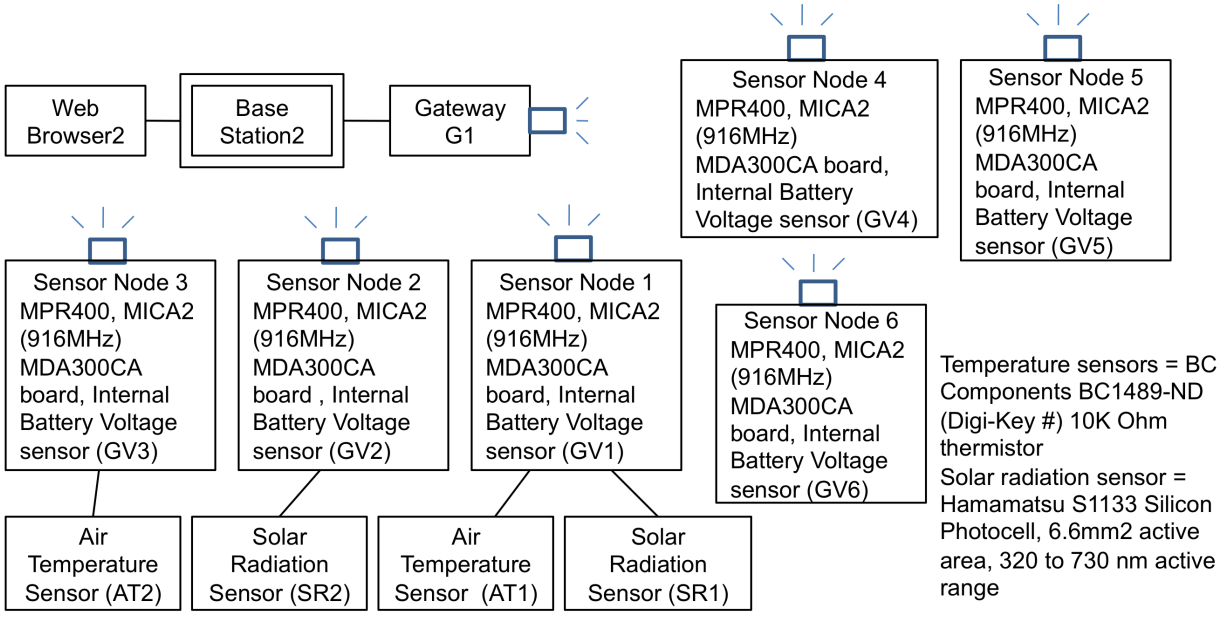


Figure 5.10: Experimental sensor network design in the IB214 lab.

The testing was done in three steps:

1. We introduce one sensor node per four hours over a one day period into the network.

2. On successful completion of Step 1, we remove one sensor node per four hours in a one day period to check the change in WSN structure on the web application 2.

3. On successful completion of Step 2, the dynamic network adaptation was checked by adding and removing sensor nodes from the network randomly.

5.2.1 Color Codes

As already defined in equation (4.1), let β_{min}=0.1, and T (number of hours for which not hearing from a node is normal) be set to 4 hours.

If t (number of hours since a message has been received from a node) ≤ 4 then $\beta = 1$; if t is between [4, 40] then β is in range [1, 0.1], and if t ≥ 40 then $\beta = 0.1$.

If β is 1 then the color of the node will be red, if β in the range [1, 0.1] then the color of the node is light orange. If $\beta = 0.1$ (i.e. β_{min}), then the color of the node is faded brown.

5.3 Experimental Results

Figure 5.11 shows 5 nodes, nodes added one by one per 4 hours in the WSN. Whenever a new node is added in the network, it automatically appears on the web application 2. Newer nodes in the network are brighter in color.

In SWL, a node is never removed from the network. If a node stops communicating with the adjacent node or the gateway, then it is considered to be inactive and its color intensity on the web application decreases over time. Figure 5.12 shows the WSN running for 2 days. All three nodes have not sent messages for more than five hours, hence their color intensity is reduced to a light orange color.

Figure 5.13 shows the network when nodes are removed one by one per four hours from the WSN. As these nodes are removed they stop sending messages to the base station and after 40 hours the nodes are considered as inactive. Inactive nodes are displayed in faded brown.

Figure 5.14 shows a sample Observation2.html screen for the sensor named AT2 attached to node named RSN3. The screen shows the Air temperature readings taken by the air temperature sensor attached to node RSN3, in the Faculty of Computer Science lab IB214.

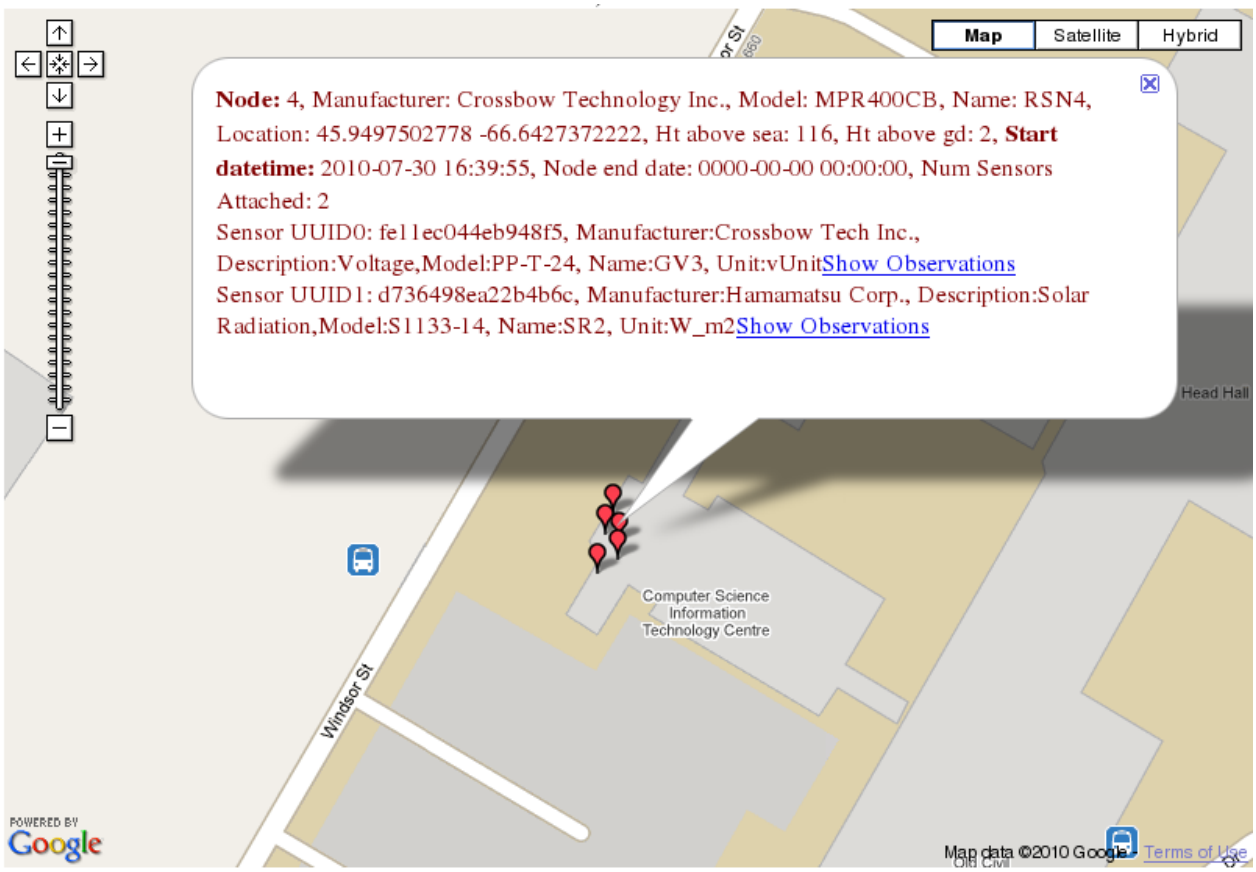


Figure 5.11: Sample Network2.html screen, displaying 5 active nodes in the network as red bubbles.

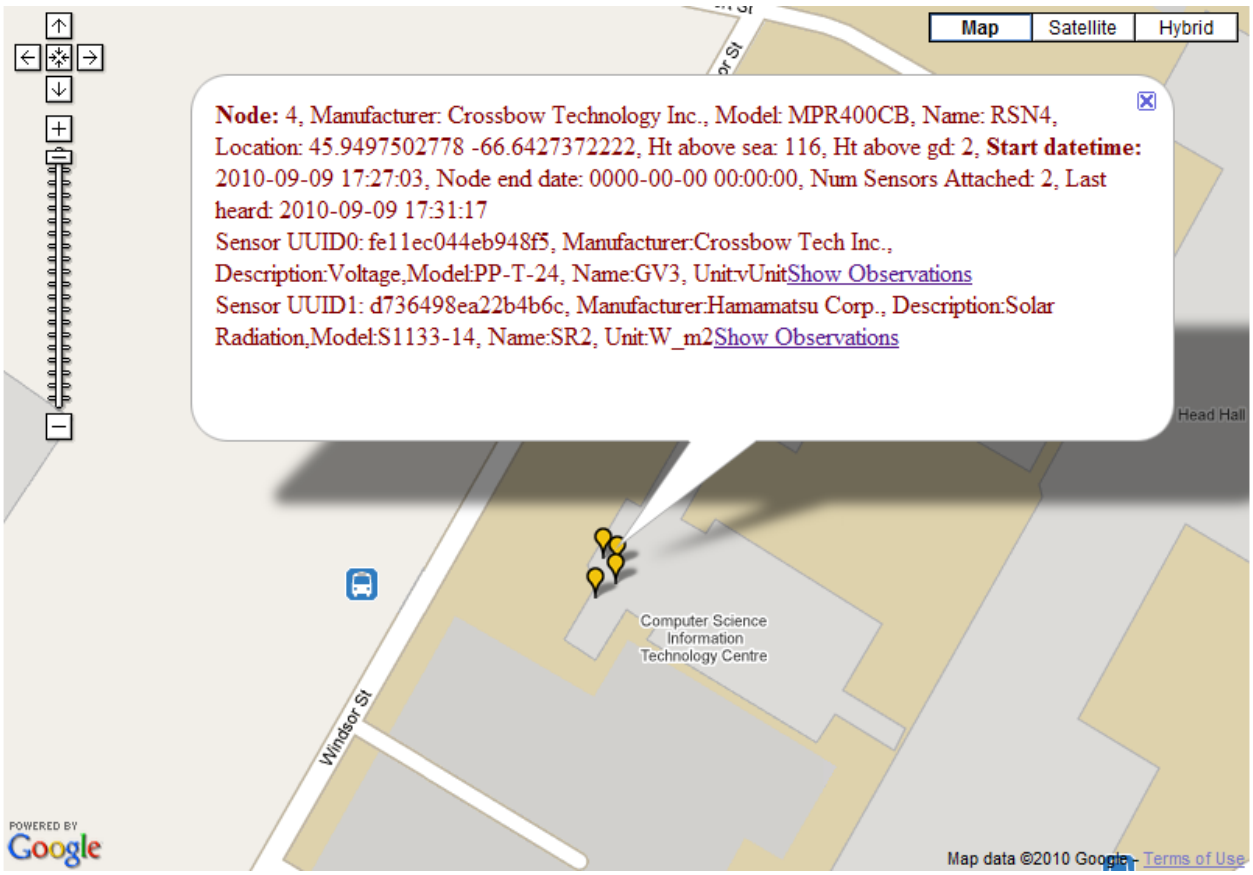


Figure 5.12: Sample Network2.html screen, displaying nodes which have not communicated with the base station for more than five hours as orange bubbles.

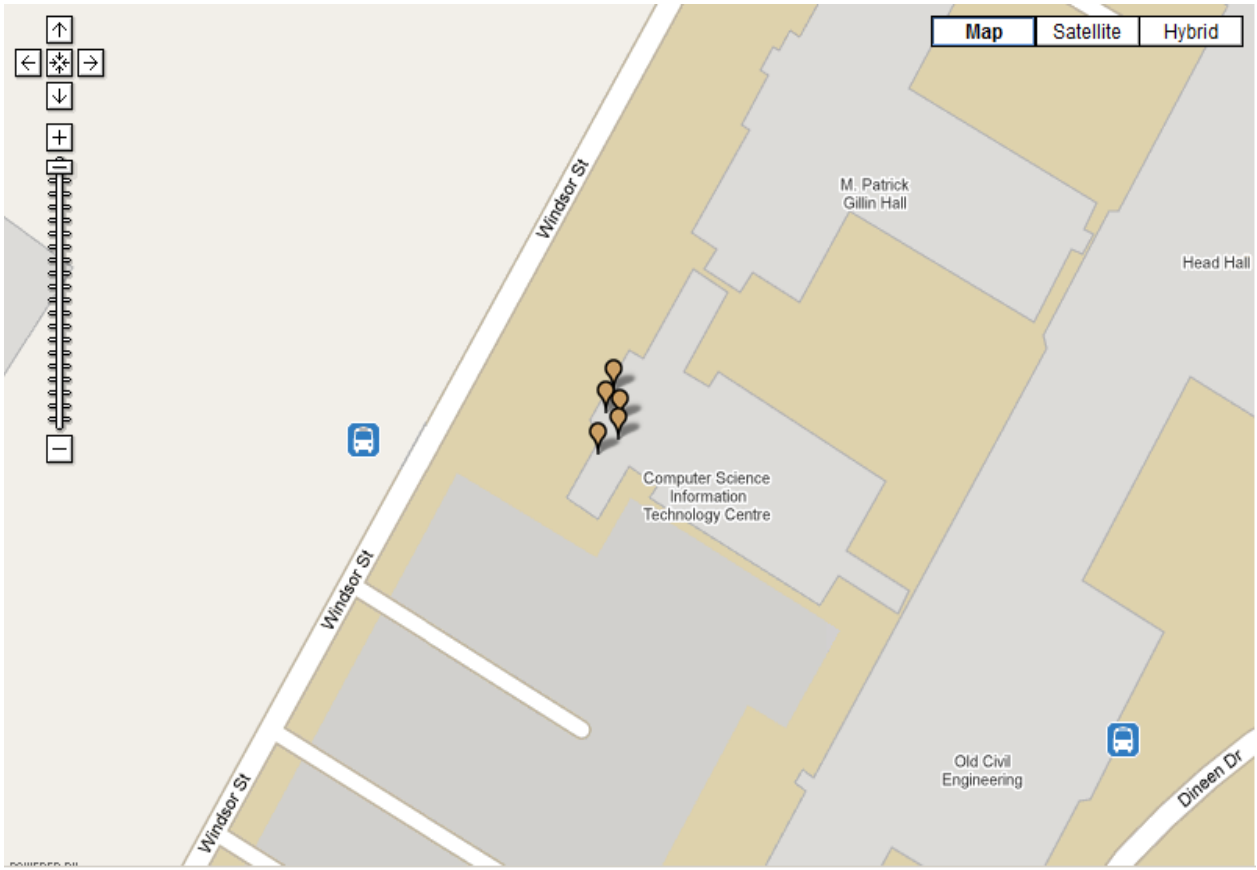


Figure 5.13: Sample Network2.html screen, displaying inactive nodes in the network as faded brown bubbles.

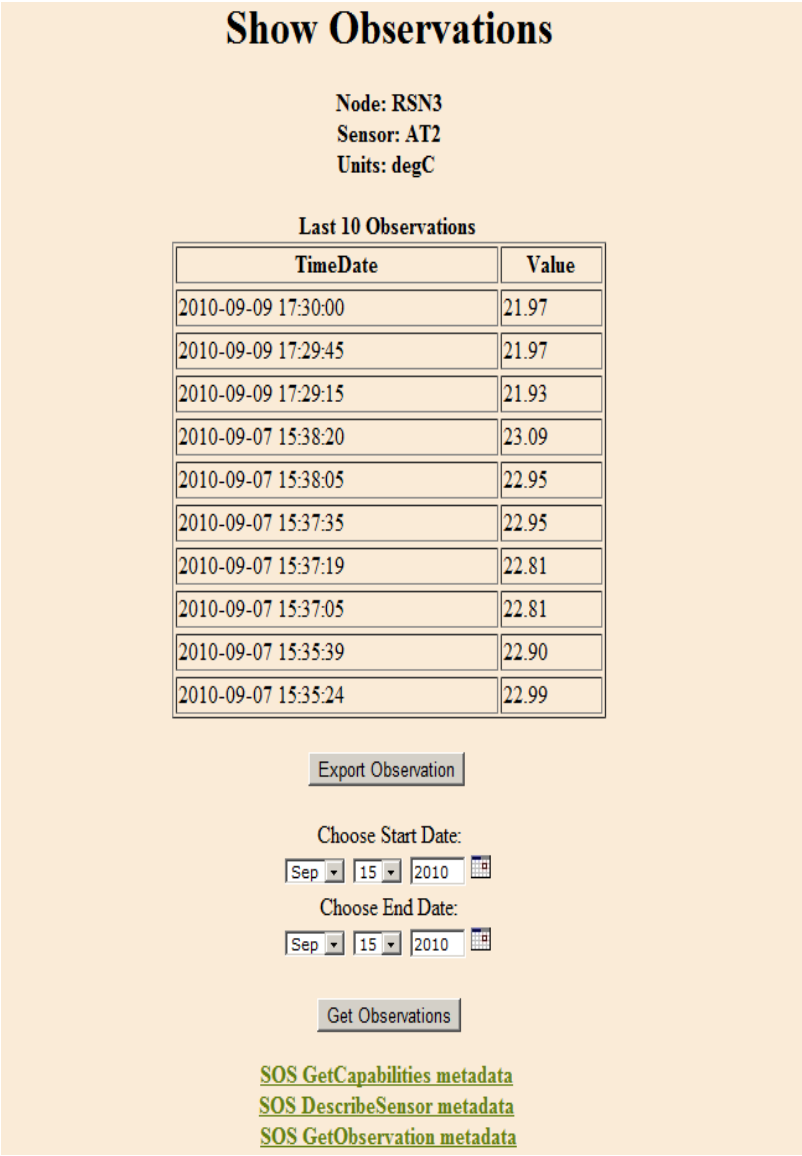
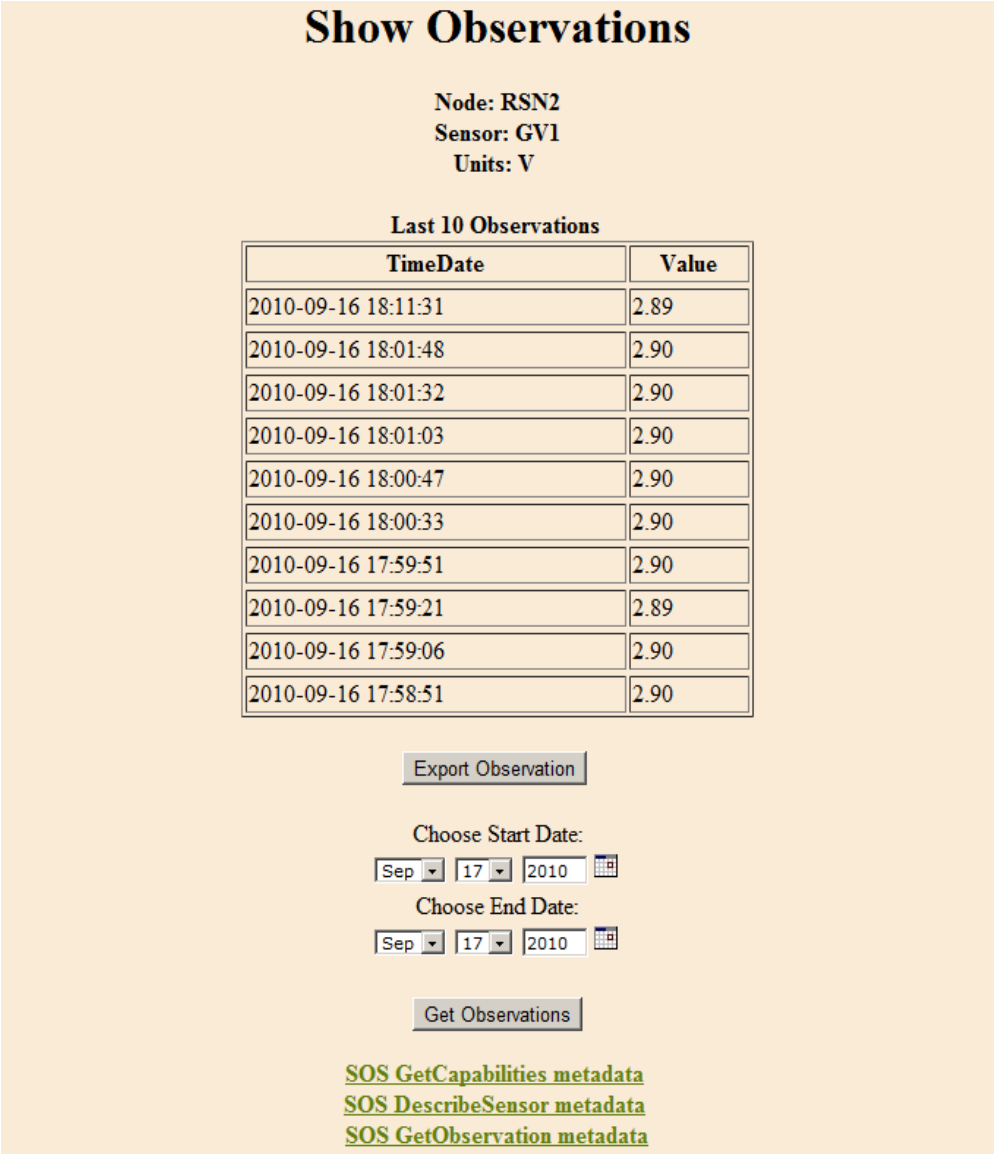


Figure 5.14: Sample Observation2.html screen, for sensor name AT2 attached to the node name RSN3.

Figure 5.15 shows a sample Observation2.html screen for sensor named GV1 attached to node named RSN2. The screen shows the battery voltage readings taken by the internal battery voltage sensor attached to Radio Sensor Node 2, in the Faculty of Computer Science lab IB214.



Show Observations

Node: RSN2
Sensor: GV1
Units: V

Last 10 Observations

TimeDate	Value
2010-09-16 18:11:31	2.89
2010-09-16 18:01:48	2.90
2010-09-16 18:01:32	2.90
2010-09-16 18:01:03	2.90
2010-09-16 18:00:47	2.90
2010-09-16 18:00:33	2.90
2010-09-16 17:59:51	2.90
2010-09-16 17:59:21	2.89
2010-09-16 17:59:06	2.90
2010-09-16 17:58:51	2.90

Export Observation

Choose Start Date:
Sep 17 2010
Choose End Date:
Sep 17 2010

Get Observations

SOS GetCapabilities metadata
SOS DescribeSensor metadata
SOS GetObservation metadata

Figure 5.15: Sample Observation2.html screen, for sensor name GV1 attached to the node name RSN2.

Figure 5.16 shows a sample Observation2.html screen for sensor named SR1 attached to node named RSN2. The screen shows the solar radiation readings taken by the solar radiation sensor attached to RSN2, in the Faculty of Computer Science lab

IB214. The difference in the values is because the observations were taken by turning the lights "on" and "off" in the lab repeatedly.

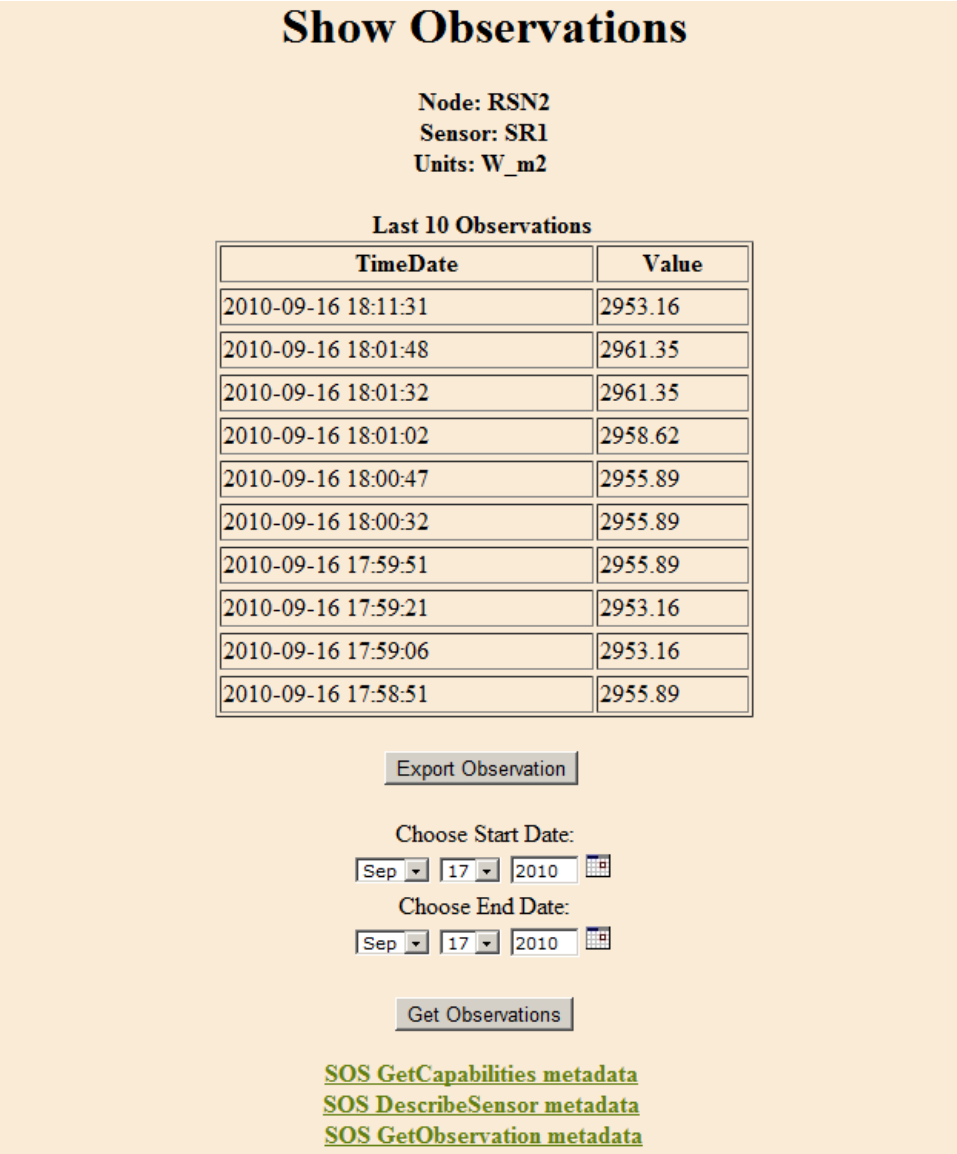


Figure 5.16: Sample Observation2.html screen, for sensor name SR1 attached to the node name RSN2.

5.3.1 Propagation Time

Propagation time is the time taken from when a node is started in the WSN to the time it appears on the web application. When a new node appears in the WSN,

its presence is detected first at the base station, and then propagated to the web application. In the above experimental setup, the time when a node is started, received at the base station and appears on the web application are recorded. If a node is started at time t_0, it announces at the base station at time t_1 and appears on the web application at time t_2. The propagation time $t_p = t_2 - t_0$.

To determine propagation time experimentally we set the polling interval Δ_s to 30 seconds. A polling interval less than 1 second was tried, but the web application became unstable as it was not able to handle the high incoming and outgoing traffic of messages at such a short interval of time. For the experiments, the system time in milliseconds is taken when a node is started, when it starts communicating, when it announces on the base station and when it pops up on the broswer. All the processes (except those running on the nodes and the gateway) are run on the same workstation.

Table 5.2: The average node startup time and propagation time at the base station of four sensor nodes in the WSN, n=10.

Sender ID	Node Name	Average node startup time (s)	Average time t_1-t_0 (s)	Average time t_2-t_1 (s)
2	RSN2	33.165	1.327	13.432
3	RSN3	31.574	0.6761	12.291
4	RSN4	35.221	1.523	13.340
5	RSN5	34.376	0.911	11.976

In Table 5.2 the node startup time and the propagation time at the base station are averaged over 10 values for each node. The average node startup time is 33.584s with a standard deviation of 1.584s, the average time taken by 4 sensor nodes to appear on the base station is 1.11s seconds with a standard deviation of 0.385s and the average time taken by 4 sensor nodes to pop up on the web application after announcing at the base station is 12.76s with a standard deviation of 0.735s. Hence, the average propagation time t_p for a sensor node to appear on the web application

is 1.11 + 12.76 = 13.87 seconds.

5.3.2 System Latency

System latency is a measure of time delay experienced in dynamic SWL due to other external factors like the polling interval and time taken by a sensor node to start communicating with the gateway. Figure 5.17 shows a time line diagram for when a node is started until it is seen on the web application.

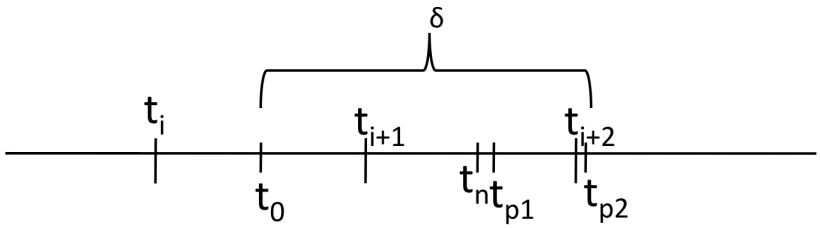


Figure 5.17: Time line diagram showing a node started at time t_0 and announcing at the web application at time t_{p2}.

In Figure 5.17, t_i is the start time of a polling interval, $\Delta_s = t_{i+1} - t_i$ is a polling interval, $t_i + \Delta_s/2$ is the expected node startup time t_0 assuming uniform random distribution of turning nodes on, t_{p1} is the time epoch at which the node's Announce message is received at the base station, t_{p2} is the time epoch at which the node's Announce message is received at the web application and t_n is the time epoch when node has completed startup. The average system latency $\bar{\delta}$ is defined as:

$$\bar{\delta} = \frac{\sum_{i=1}^{m} (t_{p2} - t_0)_i}{m} \tag{5.1}$$

where m is the number of nodes measured for. Defining

$\Delta_n = t_n - t_0$, $\Delta_{p1} = t_{p1} - t_n$,$\Delta_{p2} = t_{p2} - t_{p1}$, then $\delta = \Delta_n + \Delta_{p1} + \Delta_{p2} = 33.58 + 1.11 + 12.76 = 47.45$ seconds, where the polling interval is set to 30 seconds, and m=4.

The average adaptive SWL system latency, as measured experimentally, is approximately 48 seconds. Note that the average system latency does not depend on Δ_s as the node startup time Δ_n exceeds the polling interval. The Announce message defining a new node is only "seen" by the web application when it queries the base station in the polling loop. Appendix D.3.1, method load() has the complete polling loop code. Due to the polling interval Δ_s, a node that starts at $t_0 \in [t_i, t_{i+1}]$ will not be seen until after the t_{i+2} time epoch as $\Delta_n > \Delta_s$. The Δ_{p2} average delay of 12.76s is due to the base station waiting for the web application 2 to poll.

Chapter 6

Conclusions and Future Work

6.1 Summary

The ability to dynamically adapt a changing sensor network to the web has been added to Sensor Web Language (SWL). Two new message types in SWL were added to support this adaptation. The SWL grammar, the base station programs and the SWL database were updated. Two web applications were implemented to display dynamic adaptation on the web clearly. As new nodes enter the WSN, they dynamically appear on the web application using the dynamic base station and web server programs. This provides a foundation for improved real-time decision support systems making use of sensor networks.

Testing results show that: (a) dynamic WSN changes can be propagated to the web using 2 techniques (SWL alone or SWL + SOS), (b) SWL integrates well with a Sensor Observation Service (SOS), and (c) by polling the web application 2 at 30 second intervals, a new node's average propagation time was determined to be 13.87 seconds to appear on the web application and average system latency was estimated to be 47.45 seconds. To check the robustness of the dynamic SWL architecture, a sensor network with six sensor nodes and 10 sensors was built and tested, adding

and removing each node in the network one by one or simultaneously.

6.2 Future Work

The present adaptive sensor web implementation is a proof-of-concept. The following possible things can be done to make the current implementation more dynamic and more robust.

(a) Check the current ANNOUNCE mechanism on bigger WSNs, with multiple gateways and multiple sensor nodes under them.

(b) Extend the current implementation to run on TinyOS 2.x. This would allow support for multi-hop sensor networks.

(c) A new programming concept named HTML5 was recently introduced to permit more versatile web-based communication [1]. The implementation of HTML5 will remove the need for a continuous polling loop, and allows so-called web socket communication.

(d) The design of the Google Map web page and the SOS web page can be improved further to make them look more attractive. For example, in the 'Get Observation' page a user could have the option of viewing the output in text, CSV, XML or KML formats.

References

[1] *HTML 5: A vocabulary and associated APIs for HTML and XHTML*, Tech. report, W3C, August 2009, `http://www.w3.org/TR/2009/WD-html5-20090825/`.

[2] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella, *Energy conservation in wireless sensor networks: A survey*, Ad Hoc Networks **7** (2009), no. 3, 537–568.

[3] J. P. Arp, B. G. Nickerson, J. Lu, and Z. W. Sun, *Sensor web language communications protocol*, Tech. Report TR04-167, UNB Faculty of Computer Science, Fredericton, N.B., May 16, 2004.

[4] John-Paul Arp and Bradford G. Nickerson, a user friendly toolkit for building robust environmental sensor networks, Fifth Annual Conference on Communication Networks and Services Research (CNSR 2007), 14-17 May 2006, Fredericton, New Brunswick, Canada, 2007, pp. 76–84.

[5] _____, end-to-end acknowledgement for data collection in wireless sensor networks, 8th Annual Conference on Communication Networks and Services Research, CNSR 2010, 11-14 May 2010, Montreal, Canada, IEEE Computer Society, 2010, pp. 93–101.

[6] Prasanna Ballal, introduction to crossbow mica2 sensors, `http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`, 2003.

[7] Mike Botts, George Percivall, Carl Reed, and John Davidson, *Ogc sensor web enablement: Overview and high level architecture*, GeoSensor Networks (Silvia Nittel, Alexandros Labrinidis, and Anthony Stefanidis, eds.), Lecture Notes in Computer Science, vol. 4540, Springer Berlin / Heidelberg, 2008, 10.1007/978-3-540-79996-2_10, pp. 175–190.

[8] National Data Buoy Center, ioos sensor observation service (sos), `http://sdf.ndbc.noaa.gov/sos/`, July 2010.

[9] Atmel Corporation., atmega128, `http://www.atmel.com/atmel/acrobat/doc2467.pdf`, 2008.

[10] Crossbow, mts/mda sensor board users manual, http://www.investigacion.frc.utn.edu.ar/sensores/Equipamiento/Wireless/MTS-MDA_Series_Users_Manual.pdf, June 2007.

[11] Ke Deng, *Improving responsiveness of the sensor webs*, Master's thesis, The University of New Brunswick, Fredericton, NB, Canada, November 2008.

[12] Center for Embedded Networked Sensing (CENS), mica2 data acquisition board, http://arri.uta.edu/acs/ee5369/ee5369$%$20lectures/Introduction$%$20to$%$20Crossbow$%$20Mica2$%$20Sensors.pdf.

[13] Google, the google maps javascript api v3 - basics, http://code.google.com/apis/maps/documentation/javascript/basics.html, 2010.

[14] Lin Gu, Dong Jia, Pascal Vicaire, Ting Yan, Liqian Luo, Ajay Tirumala, Qing Cao, Tian He, John A. Stankovic, Tarek F. Abdelzaher, and Bruce H. Krogh, lightweight detection and classification for wireless sensor networks in realistic environments, SenSys, 2005, pp. 205–217.

[15] Mikko Kohvakka, Mauri Kuorilehto, Marko Hännikäinen, and Timo D. Hämäläinen, performance analysis of ieee 802.15.4 and zigbee for large-scale wireless sensor network applications, PE-WASUN '06: Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks (New York, NY, USA), ACM, 2006, pp. 48–57.

[16] Younggoo Kwon and Yohan Chae, traffic adaptive ieee 802.15.4 mac for wireless sensor networks, EUC, 2006, pp. 864–873.

[17] Theofanis P. Lambrou and Christos G. Panayiotou, *Collaborative area monitoring using wireless sensor networks with stationary and mobile nodes*, EURASIP J. Adv. Signal Process **2009** (2009), 1–16.

[18] N. Lee, P. Levis, and J. Hill, *Mica high speed radio stack*, Tech. Report UCB/ERL M02/34, EECS Department, University of California, Berkeley, 2002.

[19] Alan M. Mainwaring, David E. Culler, Joseph Polastre, Robert Szewczyk, and John Anderson, wireless sensor networks for habitat monitoring, WSNA, 2002, pp. 88–97.

[20] Geoff Mulligan, the 6lowpan architecture, EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors (New York, NY, USA), ACM, 2007, pp. 78–82.

[21] A. Na, M. Priest, H. Niedzwiadek, and J. Davidson, *Ogc implementation specific 06-009r6: Opengis sensor observation service (sos).*, Tech. report, OGC 06-009r6, October, 2007.

[22] Eduardo Freire Nakamura, Antonio Alfredo Ferreira Loureiro, and Alejandro César Frery, *Information fusion for wireless sensor networks: Methods, models, and classifications*, ACM Comput. Surv. **39** (2007), no. 3.

[23] Bradford G. Nickerson and Jing Lu, a language for wireless sensor webs, CNSR, 2004, pp. 293–300.

[24] Bradford G. Nickerson, Zhongwei Sun, and John-Paul Arp, a sensor web language for mesh architectures, CNSR, 2005, pp. 269–274.

[25] Andrzej Pawlowski, Jose Luis Guzman, Francisco Rodrguez, Manuel Berenguel, Jos Snchez, and Sebastin Dormido, *Simulation of greenhouse climate monitoring and control with wireless sensor network and event-based control*, Sensors **9** (2009), no. 1, 232–252.

[26] Kay Rmer and Friedemann Mattern, *The design space of wireless sensor networks*, IEEE Wireless Communications **11** (2004), no. 6, 54–61.

[27] Z. Song, *Mesh architecture for environmental sensor webs*, Master's thesis, The University of New Brunswick, Fredericton, NB, Canada, September 2005.

[28] Yang Sun, intelligent wireless sensor network based vehicle detection and classification, Ph.D. thesis, University, MS, USA, 2007, Adviser-Daigle, John N.

[29] Ankit Tiwari, Prasanna Ballal, and Frank L. Lewis, *Energy-efficient wireless sensor network design and implementation for condition-based maintenance*, TOSN **3** (2007), no. 1, 1.

[30] Wapedia, *Wiki: Universally unique identifier*, `http://wapedia.mobi/en/Universally_Unique_Identifier`, September 2010.

[31] Tim Wark, Peter I. Corke, Pavan Sikka, Lasse Klingbeil, Ying Guo, Christopher Crossman, Philip Valencia, Dave Swain, and Greg Bishop-Hurley, *Transforming agriculture through pervasive wireless sensor networks*, IEEE Pervasive Computing **6** (2007), no. 2, 50–57.

[32] Wikipedia, universally unique identifier, `http://en.wikipedia.org/wiki/Universally_Unique_Identifier`, September 2009.

[33] ———, *Birthday problem*, `http://en.wikipedia.org/wiki/Birthday_paradox#Probability_table`, September 2010.

[34] ———, google maps, `http://en.wikipedia.org/wiki/Google_Maps`, July 2010.

[35] ———, universal transverse mercator coordinate system, `http://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system`, July 2010.

[36] Wei Ye, John Heidemann, and Deborah Estrin, an energy-efficient mac protocol for wireless sensor networks, Proceedings of the IEEE Infocom (New York, NY, USA), USC/Information Sciences Institute, IEEE, June 2002, pp. 1567–1576.

Appendix A

Sensor Node code and the Gateway code

This chapter contains the changes made in the sensor node and the gateway node code to make a new node in the WSN announce itself.

A.1 SWLMoteM.nc

As described in Chapter 3, SWLMoteM.nc program provides implementation of the SWLMote application. Here the SWLMoteM.nc program is described in more details.
Whenever a sensor node is started, a 1 byte integer sessionID is written in the node's flash memory. The following code fragment shows how sessionID is written to and read from the node's flash memory.

Listing A.1: SWLMoteM.nc - Write SessionID in the node's flash memory

```
 task void readtask()
{  call EEPROMRead.read(0, fromSessionLogArray);
   return;
}

task void writetask()
{

   toSessionLogArray[0] = r_sessionid;
   call EEPROMWrite.startWrite();
   call EEPROMWrite.write(0, toSessionLogArray);
   return;
}

task void sendtask()
{  toRadioMsg->addr=gatewayaddr;
   setAsANNOUNCEMsg(toRadioMsg);
   toRadioSWLMsg->sessionid = r_sessionid;
   call SendSWLMsg.send(0, sizeof(struct SWLMsg), toRadioMsg);
   return;
}
```

```
   event result_t EEPROMRead.readDone(uint8_t *buf, result_t success
      )
   {
      r_sessionid = fromSessionLogArray[0] + 1;
      call Leds.greenOn();
      post writetask();
      return SUCCESS;
   }

   event result_t EEPROMWrite.writeDone(uint8_t *buf)
   {
      if (call EEPROMWrite.endWrite() == FAIL)
      return SUCCESS;
   }

   event result_t EEPROMWrite.endWriteDone(result_t success)
   {  call Leds.yellowOn();
      //post sendtask();
      toRadioSWLMsg->sessionid = r_sessionid;
      EEPROM_state = IDLE;
      return SUCCESS;
   }
```

When a new node enters the network, and starts sending messages to the base station, then the base station sends a RequestAnnounce message to that node. The following program shows what the node program does when it receives the RequestAnnounce message from the base station.

Listing A.2: SWLMoteM.nc - RequestAnnounce

```
 task void ProcessMessageTask()
   {
   int i = 0;
   int id;
   SWLMsg *  cmd;
   int sid = 0;
   cmd = (struct SWLMsg *) msg->data;
   cmd->senderID = TOS_LOCAL_ADDRESS;

switch (cmd->swlMsgType)
   {
      .
      .
      .
      case REQUEST:
      call SleepTimer.stop();
      switch (cmd->payload[i])
      {

                  .
               .
               .
               case 'g': //get value
               i++;    // skip the g
```

```
                switch (cmd->payload[i])
                {

                   case 'a': //announce the sensor node

                         i++;
                   call Leds.yellowOn();

                   writeToPayload(msg, 0);

                   //calls function to write the Announce message in
                      the payload
                            break;
                   .
                   .
                   .
                      }
                 break;
                 default:
                 sendResponse();
                 break;
                .
                .
                .
      }
      break;
      default:
      sendResponse();
      break;
    }
   }
   return;
   }
```

The Announce message is written on the payload in the following code, as payload is 22 bytes long, the Announce message is divided in 3 + number of sensors attached to the node, parts. When the Sensor node receives RequestAnnounce, it replies by sending first part of the Announce message, with Sequence Number = 1.

Listing A.3: SWLMoteM.nc - writeToPayload()

```
inline void writeToPayload(TOS_MsgPtr pmsg, int ch)
{
   SWLMsg * cmd;
   char session[3];
   cmd = (struct SWLMsg *) pmsg->data;
   switch(ch)
   {
     case 0: //node uuid
          memcpy(&AnnounceArray, "nid="nodeUUID";", sizeof(
             AnnounceArray));
          memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
             payload));
          toRadioSWLMsg->swlMsgType = ANNOUNCE;
```

```
        toRadioSWLMsg->swlMsgID = reportIndex;
         toRadioSWLMsg->swlSeqNo = ch+1;
         toRadioSWLMsg->sessionid = r_sessionid;
        memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
           toRadioSWLMsg->payload));
         toRadioMsg->addr=gatewayaddr;
         call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
            toRadioMsg);
         return;
         break;
    case 1: //session id and latitude of the node
         memcpy(&AnnounceArray, "lt="LAT_DEG","LAT_MIN","LAT_SEC";s
            =", 18);
         AnnounceArray[18] = (r_sessionid - r_sessionid%100)/100
            +48;
         AnnounceArray[19] = (r_sessionid%100 - r_sessionid%10)/10
            +48;
         AnnounceArray[20] = r_sessionid%10 +48;
         AnnounceArray[21] = ';';
         memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
            payload));
         toRadioSWLMsg->swlMsgType = ANNOUNCE;
        toRadioSWLMsg->swlMsgID = reportIndex;
         toRadioSWLMsg->swlSeqNo = ch+1;
         toRadioSWLMsg->sessionid = r_sessionid;
        memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
           toRadioSWLMsg->payload));
         toRadioMsg->addr=gatewayaddr;
         call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
            toRadioMsg);
         return;
         break;
    case 2: //number of sensors and the longitude of the node
         memcpy(&AnnounceArray, "lon="LONG_DEG","LONG_MIN","
            LONG_SEC";n=", 19);
         AnnounceArray[19] = (NUM_SENSORS - NUM_SENSORS%10)/10 +48;
         AnnounceArray[20] = NUM_SENSORS%10 +48;
         AnnounceArray[21] = ';';
         memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
            payload));
         toRadioSWLMsg->swlMsgType = ANNOUNCE;
        toRadioSWLMsg->swlMsgID = reportIndex;
         toRadioSWLMsg->swlSeqNo = ch+1;
         toRadioSWLMsg->sessionid = r_sessionid;
        memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
           toRadioSWLMsg->payload));
         toRadioMsg->addr=gatewayaddr;
         call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
            toRadioMsg);
         return;
         break;
     case 3: //sensor1 uuid
         memcpy(&AnnounceArray, "SID="SENSOR1_UUID";", sizeof(
            AnnounceArray));
```

```
        memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
            payload));
        toRadioSWLMsg->swlMsgType = ANNOUNCE;
       toRadioSWLMsg->swlMsgID = reportIndex;
        toRadioSWLMsg->swlSeqNo = ch+1;
        toRadioSWLMsg->sessionid = r_sessionid;
       memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
          toRadioSWLMsg->payload));
        toRadioMsg->addr=gatewayaddr;
        call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
            toRadioMsg);
        return;
        break;
     case 4: //sensor2 uuid
        memcpy(&AnnounceArray, "SID="SENSOR2_UUID";", sizeof(
            AnnounceArray));
        memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
            payload));
        toRadioSWLMsg->swlMsgType = ANNOUNCE;
       toRadioSWLMsg->swlMsgID = reportIndex;
        toRadioSWLMsg->swlSeqNo = ch+1;
        toRadioSWLMsg->sessionid = r_sessionid;
       memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
          toRadioSWLMsg->payload));
        toRadioMsg->addr=gatewayaddr;
        call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
            toRadioMsg);
        return;
        break;
     case 5: //sensor3 uuid
        memcpy(&AnnounceArray, "SID="SENSOR3_UUID";", sizeof(
            AnnounceArray));
        memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
            payload));
        toRadioSWLMsg->swlMsgType = ANNOUNCE;
       toRadioSWLMsg->swlMsgID = reportIndex;
        toRadioSWLMsg->swlSeqNo = ch+1;
        toRadioSWLMsg->sessionid = r_sessionid;
       memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
          toRadioSWLMsg->payload));
        toRadioMsg->addr=gatewayaddr;
        call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
            toRadioMsg);
        return;
        break;
    case 6: //sensor5 uuid
        memcpy(&AnnounceArray, "SID="SENSOR4_UUID";", sizeof(
            AnnounceArray));
        memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
            payload));
        toRadioSWLMsg->swlMsgType = ANNOUNCE;
       toRadioSWLMsg->swlMsgID = reportIndex;
        toRadioSWLMsg->swlSeqNo = ch+1;
        toRadioSWLMsg->sessionid = r_sessionid;
```

```
        memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
            toRadioSWLMsg->payload));
         toRadioMsg->addr=gatewayaddr;
         call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
            toRadioMsg);
         return;
         break;
      case 7: //sensor6 uuid
         memcpy(&AnnounceArray, "SID="SENSOR5_UUID";", sizeof(
            AnnounceArray));
         memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
            payload));
         toRadioSWLMsg->swlMsgType = ANNOUNCE;
        toRadioSWLMsg->swlMsgID = reportIndex;
         toRadioSWLMsg->swlSeqNo = ch+1;
         toRadioSWLMsg->sessionid = r_sessionid;
        memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
            toRadioSWLMsg->payload));
         toRadioMsg->addr=gatewayaddr;
         call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
            toRadioMsg);
         return;
         break;
      case 8: //sensor7 uuid
         memcpy(&AnnounceArray, "SID="SENSOR6_UUID";", sizeof(
            AnnounceArray));
         memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
            payload));
         toRadioSWLMsg->swlMsgType = ANNOUNCE;
        toRadioSWLMsg->swlMsgID = reportIndex;
         toRadioSWLMsg->swlSeqNo = ch+1;
         toRadioSWLMsg->sessionid = r_sessionid;
        memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
            toRadioSWLMsg->payload));
         toRadioMsg->addr=gatewayaddr;
         call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
            toRadioMsg);
         return;
         break;
       case 9: //sensor8 uuid
         memcpy(&AnnounceArray, "SID="SENSOR7_UUID";", sizeof(
            AnnounceArray));
         memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
            payload));
         toRadioSWLMsg->swlMsgType = ANNOUNCE;
        toRadioSWLMsg->swlMsgID = reportIndex;
         toRadioSWLMsg->swlSeqNo = ch+1;
         toRadioSWLMsg->sessionid = r_sessionid;
        memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
            toRadioSWLMsg->payload));
         toRadioMsg->addr=gatewayaddr;
         call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
            toRadioMsg);
         return;
```

```
      break;
    case 10: //sensor9 uuid
      memcpy(&AnnounceArray, "SID="SENSOR8_UUID";", sizeof(
          AnnounceArray));
      memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
          payload));
      toRadioSWLMsg->swlMsgType = ANNOUNCE;
     toRadioSWLMsg->swlMsgID = reportIndex;
      toRadioSWLMsg->swlSeqNo = ch+1;
      toRadioSWLMsg->sessionid = r_sessionid;
     memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
        toRadioSWLMsg->payload));
      toRadioMsg->addr=gatewayaddr;
      call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
          toRadioMsg);
      return;
      break;
     case 11: //sensor10 uuid
      memcpy(&AnnounceArray, "SID="SENSOR9_UUID";", sizeof(
          AnnounceArray));
      memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
          payload));
      toRadioSWLMsg->swlMsgType = ANNOUNCE;
     toRadioSWLMsg->swlMsgID = reportIndex;
      toRadioSWLMsg->swlSeqNo = ch+1;
      toRadioSWLMsg->sessionid = r_sessionid;
     memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
        toRadioSWLMsg->payload));
      toRadioMsg->addr=gatewayaddr;
      call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
          toRadioMsg);
      return;
      break;
     case 12: //sensor10 uuid
      memcpy(&AnnounceArray, "SID="SENSOR10_UUID";", sizeof(
          AnnounceArray));
      memcpy(&(cmd->payload), &AnnounceArray, sizeof(cmd->
          payload));
      toRadioSWLMsg->swlMsgType = ANNOUNCE;
     toRadioSWLMsg->swlMsgID = reportIndex;
      toRadioSWLMsg->swlSeqNo = ch+1;
      toRadioSWLMsg->sessionid = r_sessionid;
     memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
        toRadioSWLMsg->payload));
      toRadioMsg->addr=gatewayaddr;
      call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
          toRadioMsg);
      return;
      break;
     default: memcpy(cmd->payload, "error1                ",
        sizeof(cmd->payload));
      toRadioSWLMsg->swlMsgType = ANNOUNCE;
     toRadioSWLMsg->swlMsgID = reportIndex;
      toRadioSWLMsg->swlSeqNo = ch+1;
```

```
          toRadioSWLMsg->sessionid = r_sessionid;
         memcpy (&(toRadioSWLMsg->payload), &(cmd->payload), sizeof(
            toRadioSWLMsg->payload));
          toRadioMsg->addr=gatewayaddr;
          call SendSWLMsg.send(gatewayaddr, sizeof(struct SWLMsg),
             toRadioMsg);
          return;
   }
}
```

On receiving the acknowledgement from the base station that the base station received the first part of the Announce message, the Sensor node sends the next part of message, now Sequence Number = 2. In the same way all the parts of the Announce message are send to the base station. The following code fragment shows the Acknowledgement section of the SWLMoteM.nc program.

Listing A.4: SWLMoteM.nc - Acknowledgement

```
event TOS_MsgPtr ReceiveACKMsg.receive(TOS_MsgPtr pmsg)
{
    ACKMsg * rack;
    rack = (struct ACKMsg *) pmsg->data;

    if (pmsg->addr != TOS_LOCAL_ADDRESS) return pmsg;
    if (rack->receiverAddress != toRadioMsg->addr) return pmsg;
    call ACKTimer.stop();
    noAcks = 0;

    switch (rack->swlMsgType)
    {
   .
   .
   .
   case ANNOUNCE : if(count < (toRadioSWLMsg->message_size-1))

         {
   count++;
             writeToPayload(pmsg, count);

         }
         else{

             call SleepTimer.start(TIMER_ONE_SHOT, 2024UL);
              }
        default : break;
    }
    return pmsg;
}
```

A.2 SWLMote.nc

As described in Chapter 3, SWLMote.nc is a top-level configuration file and the source file which the nesC compiler uses to generate an executable file.

Listing A.5: SWLMote.nc

```
/** Copyright (c) 2005 University of New Brunswick.  All
 *  rights reserved.  Redistribution and use in source and
 *  binary forms, with or without modification, are permitted
 *  provided that the conditions explained at
 *  http://www.cs.unb.ca/research-groups/geoidesw/swl
 *  are met.  This follows the Apache open source license
 *  model.
 */

/**@SWLMote.nc
 * @SWLMote.nc: SWLMoteTemp/SWLMote.nc is a basic component that
    uses four Tempurature sensors,
 *    as well as the Battary Voltage.
 *    The implementation can be found in SWLMoteTemp/SWLMoteM.nc
 *
 * @author John-Paul Arp.
 * @version 1.1.
 * @date August 1 2007.
 */

includes SWLMsg;
includes sensorboard;
includes config;

configuration SWLMote { }
implementation {
  components Main,
         SWLMoteM,
         HPLPowerManagementM,
         ByteEEPROM,
         EEPROM,
         RandomMLCG,
         SysTimeC,
            TimerC,
         LogicalTime,
         QueuedSend,
         GENERICCOMMPROMISCUOUS as Comm,
         CC1000RadioC,
         SWLSamplerC,
            LedsC,
         ResetC;
         //, DelugeC;

   Main.StdControl -> TimerC.StdControl;
   Main.StdControl -> SWLMoteM.StdControl;


   SWLMoteM.Reset -> ResetC;

   // Wiring PowerManagment Components
   SWLMoteM.PowerManagement -> HPLPowerManagementM;
   SWLMoteM.PowerMgmtEnable -> HPLPowerManagementM.Enable;
```

```
    // Wiring Logging Components
    SWLMoteM.AllocationReq1 -> ByteEEPROM.AllocationReq[TEST_ID1];
    SWLMoteM.WriteData1 -> ByteEEPROM.WriteData[TEST_ID1];
    SWLMoteM.ReadData1 -> ByteEEPROM.ReadData[TEST_ID1];
    SWLMoteM.ByteEEPROMStdControl -> ByteEEPROM;

         SWLMoteM.EEPROMControl -> EEPROM;
    SWLMoteM.EEPROMRead -> EEPROM;
    SWLMoteM.EEPROMWrite -> EEPROM.EEPROMWrite[42];

    // Wiring Radio and Communications Components
    SWLMoteM.RadioControl -> CC1000RadioC.StdControl;
    SWLMoteM.QueueControl -> QueuedSend;
    SWLMoteM.CommControl -> Comm;

    SWLMoteM.SendSWLMsg -> QueuedSend.SendMsg[AM_SWLMSG];
    SWLMoteM.ReceiveSWLMsg -> Comm.ReceiveMsg[AM_SWLMSG];

    //SWLMoteM.SendACKMsg -> QueuedSend.SendMsg[AM_ACKMSG];
    SWLMoteM.SendACKMsg -> Comm.SendMsg[AM_ACKMSG];
    SWLMoteM.ReceiveACKMsg -> Comm.ReceiveMsg[AM_ACKMSG];


    // Wiring Timing Communications Components
    SWLMoteM.ACKTimer -> TimerC.Timer[unique("Timer")];
    SWLMoteM.RXTimer -> TimerC.Timer[unique("Timer")];
    SWLMoteM.SleepTimer -> TimerC.Timer[unique("Timer")];
    SWLMoteM.SampleTimer -> TimerC.Timer[unique("Timer")];
    SWLMoteM.ReportTimer -> TimerC.Timer[unique("Timer")];
    SWLMoteM.ClockTimer -> TimerC.Timer[unique("Timer")];
    SWLMoteM.Time -> LogicalTime.Time;
    SWLMoteM.TimeSet -> LogicalTime.TimeSet;
    SWLMoteM.TimeUtil -> LogicalTime.TimeUtil;

    // Wiring Sampling Components
    SWLMoteM.SamplerControl -> SWLSamplerC.SamplerControl;
    SWLMoteM.Sample -> SWLSamplerC.Sample;

    // Wiring the Leds
    SWLMoteM.Leds -> LedsC;

    SWLMoteM.Random-> RandomMLCG.Random32;

    SWLMoteM.SysTime-> SysTimeC;
}
```

A.3 SWLMsg.h

This is SWL message header file, defining a structure of SWL message. Two new fields are added in the SWLMsg, sessionid and message_size. Also, ANNOUNCE message type 15 is added in the existing messages.

Listing A.6: SWLMsg.h

```
#include "TosTime.h"
#define BATTERY_PORT 7

enum {
 ALIVE        = 0,
 REQUEST      = 1,
 RESPONSE     = 2,
 REPORT    = 3,
 SWITCHG      = 4,
 ALERT        = 5,
 CHANGEI      = 8,
 CHECKSTAT    = 9,
 RXREADY   = 10,
 TXDONE    = 11,
 SLEEP        = 12,
 WAIT         = 13,
 RESET        = 14,
 ANNOUNCE         =  15,
// Add TSYNC message for helping time sync
};

enum {
 UP = 1,
 DOWN = 0,
};

enum {
 TEST_ID0 = unique("ByteEEPROM"),
 TEST_ID1 = unique("ByteEEPROM"),
 TEST_ID2 = unique("ByteEEPROM"),
 TEST_ID3 = unique("ByteEEPROM"),
 TEST_ID4 = unique("ByteEEPROM")
};

enum {
 AM_SWLMSG = 8,
 AM_ACKMSG = 9,
};

typedef struct SWLMsg {
    uint8_t ackSeqNo;
    uint16_t senderID; // 0x007E = TOS_UART_ADDR = Base Station
    uint8_t swlMsgType;
    uint8_t swlMsgID;
    uint16_t swlSeqNo;
    uint8_t message_size;
    uint8_t sessionid;
    uint8_t payload[22];
} SWLMsg;

typedef struct ACKMsg {
      uint8_t swlMsgType;
      uint8_t ackSeqNo;
```

```
      uint16_t receiverAddress;
} ACKMsg;

typedef struct SessionParam {
      uint8_t sessionid;
}SessionParam;
```

A.4 SWLGatewayM.nc

The existing gateway code is extended to recognize Announce message. Following fragment shows the changes done in SWLGatewayM.nc program.

Listing A.7: SWLGatewayM.nc - Announce

```
task void ProcessMessageTask()
{
   struct SWLMsg * cmd;
   int i = 0;
   int id;
   cmd = (struct SWLMsg *) msg->data;
   .
    .
    .
   if ( ((cmd->swlMsgType == REPORT) || (cmd->swlMsgType == ANNOUNCE
      )) &&
      (msg->addr == TOS_LOCAL_ADDRESS  || msg->addr == TOS_UART_ADDR
         )   )
   {  msg->addr = TOS_UART_ADDR;
      call SendSWLMsg.send(TOS_UART_ADDR, sizeof(struct SWLMsg), msg
         );
      call ACKTimer.stop();
      if (COMMBUSY == TRUE)
      {  call CommBusyTimer.stop();
         call CommBusyTimer.start(TIMER_ONE_SHOT, 4098);
      }
      msgPending = FALSE;
      return;
   }
}
```

Appendix B

Adaptive base station1

B.1 UUID.java

UUID.java generates UUIDs for sensor components. A 32 byte hexadecimal character generated by the internal java library is truncated to a 16 byte character.

Listing B.1: UUID.java

```
import java.util.UUID;

public class UUID{
   public void genUID(){
      UUID a = UUID.randomUUID();
      long lsb = a.getLeastSignificantBits();
      System.out.println(lsb);
   }

   public static void main(String args[]){
      UUID t1 = new UUID();
      t1.genUID();
   }
}
```

B.2 SWLAnnounce1.java

Whenever a new node enters the WSN, it starts sending SWL messages to the base station. The base station saves sessionID of all the nodes in the network in a hashtable with the SenderID as the key of the table. The name of the hashtable used in the following program is Sessiontable. The base station checks the senderID and the sessionID of the message received with the ones in the Sessiontable. If SenderID not present or if for that SenderID the sessionID is different then the base station sends the RequestAnnounce message to that sensor node.

Listing B.2: SWLAnnounce1.java - RequestAnnounce

```
public class SWLAnnounce1 implements MessageListener
{
  ...
  Hashtable<Integer, Moteindex> Sessiontable;
```

```
...
public SWLAnnounce1()
{...
 Sessiontable = new Hashtable<Integer, Moteindex>();
 ...
 }

 public void messageReceived(int to, Message m)
 {
   //Receiving messages from sensor nodes
   Calendar newCal = Calendar.getInstance();
   String time = newCal.get(Calendar.HOUR_OF_DAY) + ":" +
          newCal.get(Calendar.MINUTE) + ":" +
          newCal.get(Calendar.SECOND);

   if (m instanceof SWLMsg)
   {
    SWLMsg msg = (SWLMsg)m;
    int id = (int)msg.get_senderID();
    int sess_id = (int)msg.get_sessionid();
    int seqNo = (int)msg.get_swlSeqNo();
    String payload = msg.getString_payload();
    int nodeid;

    if (Sessiontable.containsKey(id)) //check if the Sessiontable
       contains the session id for that SenderID
    {
       Moteindex mi = Sessiontable.get(id); //get the session id
       if (mi.sessionid != sess_id) //if the session id in the
          Sessiontable is not equal to the incoming session id
       {
          //The node has been restarted, needs to ANNOUNCE
          mi.id=id; //new Sender ID is saved in the MoteIndex
             object class
          mi.sessionid=sess_id; //new session id is saved in the
             MoteIndex object class
          SWLAnnounce1 announce = new SWLAnnounce1();
          swlARC = new SWLAnnResponseCollector(13);
          try
             {
                // Create a new file output stream
                out = new FileOutputStream("TestAnnounce2.txt");
                // Connect print stream to the output stream
                p = new PrintStream( out );
             p.println ("request{");
                p.println ("s"+id+".ga();");
             p.println ("}");
                p.close();
              }
              catch (Exception e)
              {
                System.err.println ("Error writing to file");
                }
             try
```

```
            { announce.sendMsg("TestAnnounce2.txt");
            }
            catch(Exception e){
              e.printStackTrace();
            }
         }
      }
      else
      {  System.out.println("Announce"); //A new node in the network
         Moteindex mi = new Moteindex();
         mi.id=id;
         mi.sessionid=sess_id;
         Sessiontable.put(id, mi);
         SWLAnnounce1 announce1 = new SWLAnnounce1();
         swlARC = new SWLAnnResponseCollector(13);
         try
            {
               // Create a new file output stream
               out = new FileOutputStream("TestAnnounce2.txt");
               // Connect print stream to the output stream
               p = new PrintStream( out );
           p.println ("request{");
               p.println ("s"+id+".ga();");
           p.println ("}");
               p.close();
          }
          catch (Exception e)
          {
               System.err.println ("Error writing to file");
           }
           try
         {   announce1.sendMsg("TestAnnounce2.txt");
         }
         catch(Exception e){
            e.printStackTrace();
         }
      }
  ...
```

MoteIndex is an empty object class consisting of 2 variables, id and session_id.

Listing B.3: SWLAnnounce1.java - MoteIndex

```
import java.util.*;

public class moteindex
{
public int id;
public int session_id;
};
```

The SWLAnnounce1.java perform different functions according to the type of the message received. If the message type received is alive, then the SWLAnnounce1 sends back the Alive message, confirming to the base station that it is ready to receive messages. If the message type received is Response, then the SWLAnnounce1

collects the response (if sent in parts) and then prints it on the console. If the message type received is Announce, then SWLAnnounce1 collects the response, prints on the console and then calls the databaseOperations function.

Listing B.4: SWLAnnounce1.java - Action performed according to the message type received.

```
public void messageReceived(int to, Message m)
{ ...
      if(msg.get_swlMsgType() == 0) //ALIVE Message
    {
          try
          {  msg.set_senderID(0x007E);
             moteif.send(id, msg);
          }
          catch(Exception e)
          {  e.printStackTrace();
          }
    }
      ...
    if (msg.get_swlMsgType() == 2) //Response message
    {
          short incoming[] = msg.get_payload();
          char incChar[] = new char[incoming.length];

          for (int i = 0; i < incoming.length; i++)
          {  incChar[i] = (char)incoming[i];
          }

          String input = new String(incChar);
          swlRC.addLine((int)msg.get_swlSeqNo(), input);

          if(swlRC.ready())
          {  System.out.println("response ready:");
             System.out.println(swlRC.toString());
             //System.exit(0);
          }
    }
          if(msg.get_swlMsgType() == 15) //Announce message
    {
       short incoming[] = msg.get_payload();
       char incChar[] = new char[incoming.length];

       for (int i = 0; i < incoming.length; i++)
       {  incChar[i] = (char)incoming[i];
       }

       String input = new String(incChar);
       swlARC.addLine((int)msg.get_swlSeqNo(), input);

       if(swlARC.ready())
       {  System.out.println("response ready:");
          System.out.println(swlARC.toString());
          ARM.parseAnnResponse(id, swlARC.totalLines, swlARC.
             response); //The response from the sensor node is
```

```
            collected
         databaseOperations(id); //update database function is
            called
         //System.exit(0);
      }
   }
}
```

The following code shows the database operations performed by SWLAnnounce1.java.

Listing B.5: SWLAnnounce1.java - database operations performed

```
public void databaseOperations(int SenderID)
{
 int dnID = 0;
 int dnGCDID = 0;
 int dnnetID = 0;
 int ncount = 0;
 String dnmanufacturer = new String();
 String dnmodel = new String();
 String dnname = new String();
 String dntype = new String();
 String dnpositID = new String();
 int dnhtabsea = 0;
 int dnhtabgd = 0;
 String dnUUID="";
 int stot_rec = 0;
 int ntot_rec = 0;
 int postot_rec = 0;
 int j=0;
 String NULL_DAT = "0000-00-00";
 String node_lat="";
 String node_lon="";
 int dsID[] = new int[ARM.nsensors];
 String dsmanufacturer[] = new String[ARM.nsensors];
 String dsdescrip[] = new String[ARM.nsensors];
 String dsmodel[] = new String[ARM.nsensors];
 String dsname[] = new String[ARM.nsensors];
 String dsunit[] = new String[ARM.nsensors];
 String dsattnodeid[] = new String[ARM.nsensors];
 String dsUUID[] = new String[ARM.nsensors];
 String dspositID[] = new String[ARM.nsensors];

 //Query for Sensor Node table
 try
 {
    rs = stmt.executeQuery("SELECT DISTINCT UUID from SENSORNODE
       where id="+SenderID);
    while(rs.next()) {
    dnUUID = rs.getString("UUID");
    }
    if(ARM.nUUID.equals(dnUUID))
    {
    rs = stmt.executeQuery("SELECT * from SENSORNODE where UUID
```

```
    ="+"'"+ARM.nUUID+"'");
 while(rs.next()) {
    dnID = rs.getInt("id");
    dnmanufacturer = rs.getString("manufacturer");
    dnmodel = rs.getString("model");
    dnname = rs.getString("name");
    dntype = rs.getString("type");
    dnGCDID = rs.getInt("attached_GCD_id");
    dnnetID = rs.getInt("sensor_network_id");
    dnpositID = rs.getString("posit_id");
    dnUUID = rs.getString("UUID");
 }
 }
 else
 {
 rs = stmt.executeQuery("SELECT count(*) as num_record from
    SENSOR");
 while(rs.next()) {
    ntot_rec = rs.getInt("num_record");
 }
 ncount = ntot_rec +1;
 stmt.executeUpdate("INSERT INTO SENSORNODE(ID,"
                +"UUID)"
             +"VALUES"+"("+ncount+","
                +"'"+ARM.nUUID+"')");
 System.out.println("Writing to database complete!!!");
 }



//Query for Sensor table
      int l=0;
 rs = stmt.executeQuery("SELECT DISTINCT UUID from SENSOR");
 while(j<ARM.nsensors)
 {
 while(rs.next()) {
    dsUUID[j] = rs.getString("UUID");
    break;
 }
    j++;
 }

 for(int i=0; i<ARM.nsensors;i++)
 {
   if(ARM.sUUID[i].equals(dsUUID[i]))
   {
 rs = stmt.executeQuery("SELECT * from SENSOR where UUID="+"'"+
    ARM.sUUID[i]+"'");
 while(rs.next()) {
   dsID[i] = rs.getInt("id");
   dsmanufacturer[i] = rs.getString("manufacturer");
   dsdescrip[i] = rs.getString("description");
   dsmodel[i] = rs.getString("model");
   dsname[i] = rs.getString("name");
```

```
  dsunit[i] = rs.getString("unit");
  dsattnodeid[i] = rs.getString("attached_Node_id");
  dsUUID[i] = rs.getString("UUID");
  dspositID[i] = rs.getString("posit_id");
  //System.out.println(dsID[i]+" "+dsmanufacturer[i]+" "+
     dsdescrip[i]+" "+dsmodel[i]+" "+dsname[i]+" "+dsunit[i
     ]+"
    "+dsattnodeid[i]+" "+dsUUID[i]);
  //UPDATE SERVLET
     }
  }
 else
  {
l=l+1;
  }

  /*if(l==0)
  {
rs = stmt.executeQuery("SELECT count(*) as num_record from
    SENSOR");
while(rs.next()) {
   stot_rec = rs.getInt("num_record");
}
int scount = stot_rec +1;
stmt.executeUpdate("INSERT INTO SENSOR(ID,"
                +"UUID)"
             +"VALUES"+"("+scount+","
                +"'"+ARM.sUUID[i]+"')");
System.out.println("Writing to database complete!!!");
//UPDATE SERVLET
  }*/
}

 //Query for POSIT table
rs = stmt.executeQuery("SELECT latitude, longitude from POSIT
    where UUID='"+ARM.nUUID+"' AND end_datetime =
 '0000-00-00 00:00:00'");
while(rs.next()) {
   node_lat = rs.getString("latitude");
   node_lon = rs.getString("longitude");
}
// System.out.println(ARM.lttot+","+node_lat+","+ARM.lntot
   +","+node_lon);
if(!(ARM.lttot.equals(node_lat) && ARM.lntot.equals(node_lon))
   )
{
rs = stmt.executeQuery("SELECT count(*) as num_record from
   POSIT");
while(rs.next())
{
   postot_rec = rs.getInt("num_record");
}
int poscount = postot_rec +1;
/*stmt.executeUpdate("update POSIT SET end_date =CURDATE()
```

```
        where UUID='"+ARM.nUUID+"'");
    stmt.executeUpdate("INSERT INTO POSIT(ID,"
                    +"UUID,"
                    +"LATITUDE,"
                    +"LONGITUDE,"
                    +"START_DATE)"
                + "VALUES" +"("+poscount+","
                    +"'"+ARM.nUUID+"',"
                    +"'"+ARM.lttot+"',"
                    +"'"+ARM.lntot+"',"
                    +"CURDATE()"+")");
    stmt.executeUpdate("INSERT INTO SENSORNODE(ID,"
                     +"MANUFACTURER,"
                     +"MODEL,"
                     +"NAME,"
                     +"TYPE,"
                     +"ATTACHED_GCD_ID,"
                     +"SENSOR_NETWORK_ID,"
                     +"POSIT_ID,"
                     +"UUID)"
                    + "VALUES" +"("+ ncount+1+","
                               +"'"+ dnmanufacturer+"',"
                     +"'"+ dnmodel+"',"
                     +"'"+dnname+"',"
                     +"'"+dntype+"',"
                     +dnGCDID+","
                     +dnnetID+","
                     +poscount+","
                     +"'"+ARM.nUUID+"')");*/
    System.out.println("Writing to the database complete!!!!!!");
    }
    else
    {
    rs = stmt.executeQuery("SELECT  HEIGHT_ABOVE_SEA,
       HEIGHT_ABOVE_GROUND from POSIT where UUID="+"'"+ARM.nUUID
       +"' AND
      END_datetime = '0000-00-00 00:00:00'");
    while(rs.next()) {       dnhtabsea = rs.getInt("
       HEIGHT_ABOVE_SEA");
      dnhtabgd = rs.getInt("HEIGHT_ABOVE_GROUND");
      }
    }
 }  catch(Exception e)   {     e.printStackTrace();    }
  AS.connect(ARM, dnID, dnmanufacturer, dnmodel, dnhtabsea,
     dnhtabgd, dsID,
   dsdescrip, dsmodel, dsmanufacturer, dsunit);  //make client
      connection to the web browser1
}
```

B.3 SWLAnnResponseCollector.java

An object class for collecting the Announce message sent in parts in an array and then return that array to SWLAnnounce1.java.

Listing B.6: SWLSensorNetwork.swl.

```
import java.lang.*;
import java.util.*;
import java.util.*;
import java.io.*;

public class SWLAnnResponseCollector
{
    String response[];
    boolean collected[];
    public int totalLines;
    public int j=0, count=0;

    public SWLAnnResponseCollector () {}

    public SWLAnnResponseCollector (int n)
    {
        totalLines = n+2;
        response = new String[totalLines];
        collected = new boolean[totalLines];

        response[0]= "response{";
        collected[0] = true;

        for (int i = 1; i < totalLines-2; i++)
        {    response[i] = "";
             collected[i] = false;
        }

        response[totalLines-1] = "}";
        collected[totalLines-1] = true;
    }

    public void reset(int n)
    {
        totalLines = n+2;
        response = new String[totalLines];
        collected = new boolean[totalLines];
    }

    public void addLine(int lineNo, String line)
    {
        response[lineNo] = line;
        collected[lineNo] = true;

        if (collected[3] == true)
        {
           String id = response[3];
           int num_sensor = 0;
```

```
            StringTokenizer st = new StringTokenizer(id, ";");
            String long1 = st.nextToken(); if(!st.hasMoreTokens())
               return;
            String sensor = st.nextToken();
            num_sensor = Integer.parseInt(sensor.substring(2));
            int tot_size;

            tot_size = 3 + num_sensor;
            totalLines = tot_size+2;
      response[totalLines-1] = "}";
      collected[totalLines-1] = true;
        }
    }

    public boolean ready()
    {
          boolean result = true;
          for (int i = 0; i < totalLines; i++)
          {    if (collected[i] == false)
               {    result = false;
               }
          }
          return result;
     }

     public String toString()
     {    String str = "";
          for (int i = 0; i < totalLines; i++)
          {    str = str + response[i] + "\n";
          }
          return str;
     }
}
```

B.4 AnnResponseMsg.java

An object class for parsing and saving the Announce message sent from the sensor node in their respective variables to assist database operations.

Listing B.7: SWLSensorNetwork.swl.

```
import java.lang.*;
import java.util.*;
import java.util.*;
import java.io.*;

public class AnnResponseMsg
{
  public String nUUID;
  public int ltdeg;
  public int ltmin;
  public int ltsec;
  public double lt;
```

```
  String ltdir;
  public String lttot;
  public int lndeg;
  public int lnmin;
  public int lnsec;
  String lndir;
  public double ln;
  public String lntot;
  public int nsensors;
  public int session_id;
  public String sUUID[];

  public AnnResponseMsg() {}

  public void parseAnnResponse(int ref,int totallines,String[]
     response)
  {
     int i=0;
     int num = totallines - 5;
     sUUID = new String[num];
     nUUID = response[1].substring(4,20);
     ltdir = response[2].substring(3,4);
     ltdeg = Integer.parseInt(response[2].substring(4,6));
     ltmin = Integer.parseInt(response[2].substring(7,9));
     ltsec = Integer.parseInt(response[2].substring(10,15));
     lt = (double)ltsec/1000;
     lttot = (ltdeg+","+ltmin+","+ltsec+","+ltdir);
     session_id = Integer.parseInt(response[2].substring(18,21));
     lndir = response[3].substring(4,5);
     lndeg = Integer.parseInt(response[3].substring(5,7));
     lnmin = Integer.parseInt(response[3].substring(8,10));
     lnsec = Integer.parseInt(response[3].substring(11,16));
     ln = (double)lnsec/1000;
     lntot = (lndeg+","+lnmin+","+lnsec+","+lndir);
     nsensors = Integer.parseInt(response[3].substring(19,21));
     //System.out.println(nUUID+" "+ltdeg+" "+ltmin+" "+ltsec10000+"
          "+lt+" "+lndeg+" "+lnmin+" "+lnsec10000+" "+ln+" "+
        nsensors);

     for(int j=4; j< totallines-1;j++)
     {
      sUUID[i] = response[j].substring(4,20);
        i++;
     }
  }
}
```

B.5 AnnounceBSClient.java

Client socket connecting to the web application 1 server socket. This object class is called by the SWLAnnounce1 to forward the Announce message to the web application 1.

Listing B.8: SWLSensorNetwork.swl.

```
import java.net.*;
import java.io.*;

public class AnnounceBSClient {
    public int i;
    String nm, des, snm;

    public AnnounceBSClient()
    {}

    public void connect(AnnResponseMsg ARM, int nID, String nman,
       String nmode, int ht_abs, int ht_abg, int[] sID, String[]
       sdescrip, String[] smode, String[] smanf, String[] sunit)
   {

        Socket announceSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;

        try {
            announceSocket = new Socket("ib214m09.cs.unb.ca", 4444);
            out = new PrintWriter(announceSocket.getOutputStream(),
               true);
            in = new BufferedReader(new InputStreamReader(
                                        announceSocket.
                                           getInputStream()));


        } catch (UnknownHostException e) {
            System.err.println("Don't know about host: ib214m09.cs.
               unb.ca");
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for "
                               + "the connection to: ib214m09.cs.unb
                                  .ca");
            System.exit(1);
        }

   try
   {
       BufferedReader stdIn = new BufferedReader(new
          InputStreamReader(System.in));
            String fromServer;
            String fromUser;
       //fromUser = "Connected, Send Values!!";

          //out.println(fromUser);
           nm="RSN"+nID;
           des="Remote Sensor Node #"+nID;

            while ((fromServer = in.readLine()) != null) {
      out.println(ARM.nUUID);
```

```
      out.println(nID);
      out.println(nm);
      out.println(des);
      out.println(nman);
      out.println(nmode);
      out.println(ARM.ltdeg);
      out.println(ARM.ltmin);
      out.println(ARM.lt);
      out.println(ARM.ltdir);
      out.println(ARM.lndeg);
      out.println(ARM.lnmin);
      out.println(ARM.ln);
      out.println(ARM.lndir);
      out.println(ht_abs);
      out.println(ht_abg);
      out.println(ARM.nsensors);

      for(i=0; i< (ARM.nsensors); i++)
      {
          snm="SR"+sID[i];
          out.println(ARM.sUUID[i]);
          out.println(sID[i]);
          out.println(snm);
          out.println(sdescrip[i]);
          out.println(smode[i]);
          out.println(smanf[i]);
          out.println(sunit[i]);
      }
                out.println("Bye.");
       }
       out.close();
       in.close();
       stdIn.close();
       announceSocket.close();
   }
   catch (UnknownHostException e) {
                System.err.println("Trying to connect to unknown
                   host: " + e);
        } catch (IOException e) {
                System.err.println("IOException:  " + e);
        }


    }
}
```

Appendix C

Web Application 1

C.1 SWLAnnounce.jsp

Listing C.1: SWLAnnounce.jsp

```
<html>
<head><title> SWL Browser </title></head>

<body>

<center>
<!--<script type="text/javascript">
document.write("<h4>"+Sensor Web Language (SWL) UNBWoodlot Sensornet
   "+"<\h4>\n");
</script> -->
<h4>Sensor Web Language (SWL) UNBWoodlot Sensornet</h4>
<h6>
UNB <a href="http://www.cs.unb.ca">Computer Science</a>Research
   Project GEOIDE <a href="http://www.cs.unb.ca/research-groups/
   geoidesw/"> MNG#BER</a>

</h6>
<jsp:plugin type="applet" code="SWLBrowser.class" codebase="applet"
   jreversion="1.4"  width="510" height="680">

</jsp:plugin>

</center>
</body>

</html>
```

C.2 SWLBrowser.java

Listing C.2: SWLBrowser.java

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import javax.swing.border.*;
import javax.swing.*;
import java.net.*;
import java.io.*;
import java.util.*;
//import baseswl.*;

public class SWLBrowser extends JApplet implements ActionListener,
                    MouseListener, KeyListener, Runnable
{ URL url = null;
  String host;
  String protocol;
  String urlFile;
  int port;
  URLConnection con = null;
  JTextArea output = new JTextArea("",10,40);//text area displaying
     results
  JTextField tf1=new JTextField(30);//test field displaying
     component names
  long startTime = 0;
  long endTime = 0;
  long lastTime = 0;
  int sncount=0;//# of sensor nodes
  int xspace=UTMAppletTest.xspace;//xspace beside plottable area 5
  int yspace=UTMAppletTest.yspace;//yspace beside plottable area 20
  int bracespacey=40;
  int bracespacex=10;
  int width=UTMAppletTest.plotx-xspace*2;//width of panel of
     plttable area  502
  int height=UTMAppletTest.ploty-yspace*2;//height of panel of
     plttable area 472
  int textbrace=10;
  int textwidth=width-textbrace;
  int textheight=140;//height of text area
  int space=10;//space left under text area
  int titlex=250, titley=5;//x y coordinate of the starting point of
      title
  int titlew=250, titleh=20;//width and height for component name
     display
  int bound=40;//equals the bound of triangle in UTMAppletTest
  ServerSocket server=null;
  //Socket client = null;
  BuildNetwork BN = new BuildNetwork();
  SWLUpdateNetwork un = new SWLUpdateNetwork();
  //public JPanel jp;
  //private PanelView pv;
  int r=0;

  JPanel browser=new JPanel();
  JPanel jp=new JPanel();
  //public SWLBrowser() {
```

```
  //     new Thread(this).start();
//}

public void paint(Graphics g) {
     super.paint(g);
     g.clearRect(xspace, 0, width, height+yspace*2);
     this.getContentPane().paintAll(g);
     //g.setColor(Color.black);
     int i=0;
 // drawing the nodes
     while(i<UTMAppletTest.xp1.size())
          {g.drawLine((int)(Double.parseDouble((UTMAppletTest.xp1.
             get(i)).
                  toString())),
       (int)(Double.parseDouble(((UTMAppletTest.yp1.get(i)).
                   toString()))),
       (int)(Double.parseDouble((UTMAppletTest.xp2.get(i)).
                  toString())),
       (int)(Double.parseDouble((UTMAppletTest.yp2.get(i)).
                  toString())));
          i++;
          }
    //draw scaled bar
     double max=0;
     int unit=0;
     double factor=UTMAppletTest.factor;
     int labelheight=5;

     max=UTMAppletTest.maxX-UTMAppletTest.minX;//maximum meter on
        x direction
     //get the length of the scale bar
     if(max<5) unit=1;
     else if(max>=5&&max<10) unit=5;
     else if(max>=10&&max<20) unit=10;
     else if(max>=20&&max<50) unit=20;
     else if(max>=50&&max<100) unit=50;
     else if(max>=100&&max<200) unit=100;
     else if(max>=200&&max<500) unit=200;
     else unit=500;

     int startx=(int)((UTMAppletTest.plotx-unit*factor)/2);
     int endx=(int)(UTMAppletTest.plotx-(UTMAppletTest.plotx-unit*
        factor)/2);
   g.drawLine(startx,height+yspace, endx,height+yspace);
   g.drawLine(startx,height+yspace,startx,height+yspace-
      labelheight);
   g.drawLine(endx,height+yspace, endx, height+yspace-labelheight)
      ;
   g.drawString("m",endx+5,height+yspace*9/8);
        g.drawString("0", startx-5, height+yspace*2-3);
   g.drawString(""+unit, endx-5, height+yspace*2-3);

   //draw north arrow
   g.drawLine(bracespacex+3,height+yspace*2-3,
```

```
          bracespacex+3, height+yspace*2-3-20);
    g.drawLine(bracespacex+3, height+yspace*2-3-20,
          bracespacex+3-3,height+yspace*2-3-10);
    g.drawLine(bracespacex+3, height+yspace*2-3-20,
          bracespacex+3+3,height+yspace*2-3-10);
    g.drawString("N", bracespacex+3+8,height+yspace*2-3);
  }//end of paint()

  public void init(){
    //applet initialization
    protocol = "http";
    host = "ib214m09.cs.unb.ca";
    port = 8080;
    urlFile = "/SWLServer/SWLServlet";

    new Thread(this).start();
    drawNetwork();
  }//end of init()

  private void servletCalling(int ind,String st){
    String head = "sensornet UNBWoodlot request{";
    String tail = "}";
    InputStream is = null;
    BufferedReader result = null;
    /*    //request[15] = "GSN.getStatus();";
    request[15] = "GSN.getPowerLevel();";
    request[16] = "configuration;";
    request[17] = "GCD.getDescription;";*/
    String action = "";
    if(ind!=-1){
      action = head+st+tail;
    }
    else
      action = head +""+tail;
    String response = "";
    String input = "";
    long rtt = 0;
    try{
      urlFile = "/SWLAnnounce/SWLServlet";
      urlFile = urlFile +"?action="+action;
      urlFile = urlFile.replaceAll(" ","%20");
      url = new URL(protocol,host,port,urlFile);
      //System.out.println("urlFile" + urlFile);
      con = url.openConnection();
      con.setDoInput(true);
      con.setDoOutput(true);
      con.setUseCaches(false);
      is = con.getInputStream();
      result  = new BufferedReader(new InputStreamReader(is));
      while((input = result.readLine())!=null){
   response += input;
      }
      endTime = getTime();
      rtt = endTime - startTime;
```

```
      is.close();
    }catch (Exception e)
    {e.printStackTrace();}
    outputResult(response);
    System.out.println("Round Trip Time for this Request is: " + rtt
        + " milliseconds");
 }

 private void outputResult(String res){
  int ind = res.lastIndexOf("{");
  res = res.substring(0,ind+1)+"\n" +res.substring(ind+1,res.length
     ()).trim();
  String[] resArr = res.split(";");
  output.setText("");
  for(int i=0; i<resArr.length; i++){
   if(i<resArr.length-1){
     output.append(resArr[i].trim()+";\n");
   }
   else
     output.append(resArr[i].trim()+"\n");
  }
  output.append(">");
  output.requestFocus();
 }

//actions for buttons
 public void actionPerformed(ActionEvent e){
   for(int i=0;i<UTMAppletTest.buttons.length;i++)
    if(e.getSource()==UTMAppletTest.buttons[i])
    {String thisname=(UTMAppletTest.nodetiptext.get(i)).toString();
      servletCalling(i,thisname+".getPowerLevel();");
    }
   for(int j=0;j<UTMAppletTest.overlapsensNum;j++)
    if(e.getSource()==UTMAppletTest.olsensbuttons[j])
     {String thisname=UTMAppletTest.overlapsensorTable[j].getName()
         ;
      servletCalling(j+UTMAppletTest.buttons.length,
           thisname+".getCurrentReading();");
    }
   for(int j=0;j<UTMAppletTest.nonoverlapsensNum;j++)
    if(e.getSource()==UTMAppletTest.nolsensbuttons[j])
     {String thisname=UTMAppletTest.nonoverlapsensorTable[j].
        getName();
       servletCalling(j+UTMAppletTest.buttons.length+
            UTMAppletTest.overlapsensNum,".getCurrentReading();");
    }
 }

  public void mouseEntered(MouseEvent e) {
   for(int i=0;i<UTMAppletTest.buttons.length;i++)
        if(e.getSource()==UTMAppletTest.buttons[i])
          {String thisname=(UTMAppletTest.nodetiptext.get(i)).
             toString();
```

```
       String descript=(UTMAppletTest.nodedescript.get(i)).toString
          ();
            tf1.setText(thisname+" : "+descript);
            }

     for(int j=0;j<UTMAppletTest.overlapsensNum;j++)
     if(e.getSource()==UTMAppletTest.olsensbuttons[j])
      {String thisname=UTMAppletTest.overlapsensorTable[j].getName()
         ;
       String descript=UTMAppletTest.overlapsensorTable[j].
          getDescription();
       tf1.setText(thisname+" : "+descript);
     }

     for(int j=0;j<UTMAppletTest.nonoverlapsensNum;j++)
     if(e.getSource()==UTMAppletTest.nolsensbuttons[j])
      {String thisname=UTMAppletTest.nonoverlapsensorTable[j].
         getName();
       String descript=UTMAppletTest.nonoverlapsensorTable[j].
          getDescription();
       tf1.setText(thisname+" : "+descript);
     }
    }
   public void mousePressed(MouseEvent e) { }

   public void mouseReleased(MouseEvent e) { }

   public void mouseExited(MouseEvent e) {
    for(int i=0;i<UTMAppletTest.buttons.length;i++)
         if(e.getSource()==UTMAppletTest.buttons[i])
           {tf1.setText("");}

     for(int j=0;j<UTMAppletTest.overlapsensNum;j++)
     if(e.getSource()==UTMAppletTest.olsensbuttons[j])
      { tf1.setText(""); }

     for(int j=0;j<UTMAppletTest.nonoverlapsensNum;j++)
     if(e.getSource()==UTMAppletTest.nolsensbuttons[j])
      {tf1.setText(""); }
}
   public void mouseClicked(MouseEvent e) {}


public void processingRequest(){
     String st1 = output.getText();
     int index = st1.indexOf(">")+1;
     String st2 = st1.substring(index).trim();
     System.out.println(st2);
     servletCalling(-1,st2);

}
public void keyPressed(KeyEvent e){
   if(e.getKeyCode()==KeyEvent.VK_ENTER){
          processingRequest();
```

```
          startTime = getTime();
        }
}
public void keyReleased(KeyEvent e){}
public void keyTyped(KeyEvent e){}
 // returns the current time in MilliSeconds
 public long getTime(){
    Calendar rightNow = Calendar.getInstance();
    java.util.Date d = rightNow.getTime();
    long ld = rightNow.getTimeInMillis();
    return ld;
}
public void drawNetwork()
{
    //Graphics g = myframe.getGraphics();
    //this.getContentPane().paintAll(g);
      //pv = new PanelView(this);
     //pv.clear();
    //Thread t1= new Thread(jPanel1);
   //t1.start();
    UTMAppletTest.initialize();

    getContentPane().setVisible(true);
    getContentPane().setLayout(null);
    getContentPane().setSize(UTMAppletTest.plotx,
              UTMAppletTest.ploty+textheight+space);

    //Jpanel jp=new JPanel();
    jp.setLayout(null);
    jp.setLocation(xspace,yspace);
    jp.setSize(width-bracespacex,height);
    //jp.setBackground(Color.WHITE);
    lastTime = getTime();
    System.out.println(lastTime);
    un.getNetwork(BN, lastTime);
    UTMAppletTest.drawConfig(un.G, un.GCD, un.GSN, un.RSN, xspace,
       yspace, width,
              height-bracespacey,jp);
    sncount = UTMAppletTest.sncount;
    //text area initialization
    output.setVisible(true);
    output.setEditable(true);
    output.setLocation(xspace,yspace*2+height);
    output.setSize(textwidth,textheight);
    output.setBorder(new BevelBorder(BevelBorder.LOWERED));
    output.setLineWrap(true);
    //JScrollPane initialization: containing text area
    JScrollPane scrollPane = new JScrollPane(output);
    scrollPane.setPreferredSize(new Dimension(textwidth, textheight)
       );
    scrollPane.setVisible(true);
    JPanel p=new JPanel();
    p.setLayout(new BorderLayout());
    p.add(scrollPane);
```

```
    p.setLocation(xspace,yspace*2+height);
    p.setSize(textwidth,textheight);
    //JPanel browser initialization: containing all components
    //JPanel browser=new JPanel();
    browser.setLocation(xspace,0);
    browser.setSize(width, height+yspace*2+textheight+space);
    browser.setLayout(null);
    browser.setBorder(BorderFactory.createEtchedBorder(EtchedBorder.
       LOWERED));
    //text field initialization: displaying all component names
    tf1.setLocation(titlex,titley);
    tf1.setBorder(null);
    tf1.setBackground((browser.getBackground()));
    tf1.setSize(titlew,titleh);
    browser.add(p);
    browser.add(jp);
    //browser.add(pv);
    browser.add(tf1);
    //browser.add(output);
    //browser.add(scrollPane);
     //add browser to the applet
    getContentPane().add(browser);


    //add listeners and tooltiptext to buttons
    int k;
    for(k=0;k<UTMAppletTest.buttons.length;k++)
    {UTMAppletTest.buttons[k].addActionListener(this);
     UTMAppletTest.buttons[k].addMouseListener(this);
     }
    for(k=0;k<UTMAppletTest.overlapsensNum;k++)
    {UTMAppletTest.olsensbuttons[k].addActionListener(this);
     UTMAppletTest.olsensbuttons[k].addMouseListener(this);
     }
    for(k=0;k<UTMAppletTest.nonoverlapsensNum;k++)
    {UTMAppletTest.nolsensbuttons[k].addActionListener(this);
     UTMAppletTest.nolsensbuttons[k].addMouseListener(this);
     }

}

public void run()
{
try{
    server = new ServerSocket(4444);
  } catch (IOException e) {
    System.out.println("Could not listen on port 4444");
    System.exit(-1);
  }
  System.out.println("Listening");
  //drawNetwork();
  while(true){
    ClientWorker w;
    try{
```

```
      w = new ClientWorker(server.accept(), this);
      Thread t = new Thread(w);
      t.start();
    } catch (IOException e) {
      System.out.println("Accept failed: 4444");
      System.exit(-1);
    }
  }
}

public void insert(Vector<String> Input)
{
   Graphics g=getGraphics();
   String nID = "";
      int n;
      int l=0;
        if(Input.size()!=0)
   {
          nID = Input.get(0);
          //System.out.println(nID);
          n=BN.NodenUUID.size();
          //System.out.println(n);

          if(n==0)
          {
      BN.nst_time.put(nID, getTime());
      System.out.println(BN.nst_time.get(nID));
      BN.addvalues(Input);
      jp.removeAll();
      jp.updateUI();
       //browser.clear();
      //this.repaint();
      super.paint(g);
      //SWLBrowser.this.repaint();
            drawNetwork();
          }
          else{
            for(int j=0; j< n; j++)
            {       System.out.println(BN.NodenUUID.get(j));
               if(nID.equals(BN.NodenUUID.get(j)))
               {
               l++;
               }
            }
      System.out.println(l);
         if(l==0)
         {
          BN.nst_time.put(nID, getTime());
          BN.addvalues(Input);
          jp.removeAll();
          jp.updateUI();
          super.paint(g);
          //this.repaint();
            //pv.clear();
```

```
            drawNetwork();
         }
          }
   }
}

protected void finalize(){
//Objects created in run method are finalized when
//program terminates and thread exits
     try{
        server.close();
    } catch (IOException e) {
        System.out.println("Could not close socket");
        System.exit(-1);
    }
  }

}
```

C.3 ClientWorker.java

Listing C.3: ClientWorker.java

```
import java.applet.*;
import java.awt.*;
import java.io.*;
import java.util.*;
import java.awt.event.*;
import java.net.*;

class ClientWorker implements Runnable {
  private Socket client;
  SWLBrowser applet;
  Vector<String> Input = new Vector<String>();

//Constructor
  ClientWorker(Socket client, SWLBrowser b) {
    this.client = client;
    //System.out.println("-----------------------"+client);
    applet = b;
  }

  public void run(){
      long time = System.currentTimeMillis();
      BufferedReader in = null;
   PrintWriter out = null;
   int i=0;

        try
   {
            out = new PrintWriter(client.getOutputStream(), true);
```

```
            in = new BufferedReader(new InputStreamReader(client.
               getInputStream()));
       String inputLine, outputLine;
       outputLine = "Connected, Send values!!";
       out.println(outputLine);

       while ((inputLine = in.readLine()) != null)
       {
         Input.add(i, inputLine);
      //System.out.println(inputLine);
         i++;
           if (inputLine.equals("Bye."))
               {   break; }
       }
          out.close();
          in.close();
          client.close();
   }
   catch (Exception e) {
              System.out.println("in or out failed");
       System.out.println("Read failed");
             System.exit(-1);
        }

        applet.insert(Input);
    }
}
```

C.4 BuildNetwork.java

Listing C.4: BuildNetwork.java

```
import java.lang.*;
import java.util.*;
import java.io.*;

public class BuildNetwork
{
  public String nodeID=null;
  public Integer numsensor=0;
  Key K;
  //Vector Input = new Vector();
  //String nsens_UUID[];
  //SWLUpdateNetwork un=new SWLUpdateNetwork();
  public Hashtable<String, Integer> NodeID = new Hashtable<String,
     Integer>();
  public Hashtable<String, String> Nodename = new Hashtable<String,
     String>();
  public Hashtable<String, String> Nodedescription = new Hashtable<
     String, String>();
  public Hashtable<String, String> Nodemanufacturer = new Hashtable<
     String, String>();
```

```
public Hashtable<String, String> Nodemodel = new Hashtable<String,
    String>();
public Hashtable<String, Integer> Nodeltdeg = new Hashtable<String
    , Integer>();
public Hashtable<String, Integer> Nodeltmin = new Hashtable<String
    , Integer>();
public Hashtable<String, Double> Nodelt = new Hashtable<String,
    Double>();
public Hashtable<String, String> Nodeltdir = new Hashtable<String,
    String>();
public Hashtable<String, Integer> Nodelngdeg = new Hashtable<
    String, Integer>();
public Hashtable<String, Integer> Nodelngmin = new Hashtable<
    String, Integer>();
public Hashtable<String, Double> Nodelng = new Hashtable<String,
    Double>();
public Hashtable<String, String> Nodelngdir = new Hashtable<String
    , String>();
public Hashtable<String, Integer> Nodeht_absea = new Hashtable<
    String, Integer>();
public Hashtable<String, Integer> Nodeht_abgd = new Hashtable<
    String, Integer>();
public Hashtable<String, Integer> Nodensensor = new Hashtable<
    String, Integer>();
public Hashtable<Integer, String> NodenUUID = new Hashtable<
    Integer, String>();
public Hashtable <String, String> Nodesensor_UUID = new Hashtable<
    String, String>();
public Hashtable<String, Integer> SID = new Hashtable<String,
    Integer>();
public Hashtable<String, String> sname = new Hashtable<String,
    String>();
public Hashtable<String, String> sdescription = new Hashtable<
    String, String>();
public Hashtable<String, String> smodel = new Hashtable<String,
    String>();
public Hashtable<String, String> smanufacturer = new Hashtable<
    String, String>();
public Hashtable<String, String> sunit = new Hashtable<String,
    String>();
public Hashtable<String, Integer> sltdeg = new Hashtable<String,
    Integer>();
public Hashtable<String, Integer> sltmin = new Hashtable<String,
    Integer>();
public Hashtable<String, Double> slt = new Hashtable<String,
    Double>();
public Hashtable<String, String> sltdir = new Hashtable<String,
    String>();
public Hashtable<String, Integer> slngdeg = new Hashtable<String,
    Integer>();
public Hashtable<String, Integer> slngmin = new Hashtable<String,
    Integer>();
public Hashtable<String, Double> slng = new Hashtable<String,
    Double>();
```

```
public Hashtable<String, String> slngdir = new Hashtable<String,
    String>();
public Hashtable<String, Integer> sht_absea = new Hashtable<String
    , Integer>();
public Hashtable<String, Integer> sht_abgd = new Hashtable<String,
     Integer>();
public Hashtable<String, Long> nst_time = new Hashtable<String,
    Long>();

public BuildNetwork()
{
 K = new Key();

}

public void addvalues(Vector<String> Input)
{
   int j=0;
   nodeID = Input.get(0);
   numsensor = Integer.parseInt(Input.get(16).substring(0,1)) ;
   String sensID[] = new String[numsensor];
   //nsens_UUID = new String[numsensor];
   //System.out.println(nodeID+","+numsensor);
    //System.out.println(K.i+"------"+nodeID);
 NodeID.put(nodeID, Integer.parseInt(Input.get(1).substring(0,1)))
    ;
 Nodename.put(nodeID, Input.get(2));
 Nodedescription.put(nodeID, Input.get(3));
 Nodemanufacturer.put(nodeID, Input.get(4));
 Nodemodel.put(nodeID, Input.get(5));
 Nodeltdeg.put(nodeID, Integer.parseInt(Input.get(6).substring
    (0,2)));
 Nodeltmin.put(nodeID, Integer.parseInt(Input.get(7).substring
    (0,2)));
 Nodelt.put(nodeID, Double.valueOf(Input.get(8).trim()).
    doubleValue());
 Nodeltdir.put(nodeID, Input.get(9));
 Nodelngdeg.put(nodeID, Integer.parseInt(Input.get(10).substring
    (0,2)));
 Nodelngmin.put(nodeID, Integer.parseInt(Input.get(11).substring
    (0,2)));
 Nodelng.put(nodeID, Double.valueOf(Input.get(12).trim()).
    doubleValue());
 Nodelngdir.put(nodeID, Input.get(13));
 Nodeht_absea.put(nodeID, Integer.parseInt(Input.get(14).substring
    (0,3)));
 Nodeht_abgd.put(nodeID, Integer.parseInt(Input.get(15).substring
    (0,1)));
 Nodensensor.put(nodeID, Integer.parseInt(Input.get(16).substring
    (0,1)));
 NodenUUID.put(K.i, nodeID);
 K.i = K.i+1;

   for(int i=0; i < (Input.size()-18); i=i+7)
```

```
    {
   double latsec, longsec;
   latsec = (Double.valueOf(Input.get(8).trim()).doubleValue());
   longsec = (Double.valueOf(Input.get(12).trim()).doubleValue());
   sensID[j] = Input.get(17+i);
   Nodesensor_UUID.put(sensID[j],nodeID);
   SID.put(sensID[j], Integer.parseInt(Input.get(18+i).substring
      (0,1)));
   sname.put(sensID[j], Input.get(19+i));
   sdescription.put(sensID[j], Input.get(20+i));
   smodel.put(sensID[j], Input.get(21+i));
   smanufacturer.put(sensID[j], Input.get(22+i));
   sunit.put(sensID[j], Input.get(23+i));
   sltdeg.put(sensID[j], Integer.parseInt(Input.get(6).substring
      (0,2)));
   sltmin.put(sensID[j], Integer.parseInt(Input.get(7).substring
      (0,2)));
   slt.put(sensID[j], latsec);
   sltdir.put(sensID[j], Input.get(9));
   slngdeg.put(sensID[j], Integer.parseInt(Input.get(10).substring
      (0,2)));
   slngmin.put(sensID[j], Integer.parseInt(Input.get(11).substring
      (0,2)));
   slng.put(sensID[j], longsec);
   slngdir.put(sensID[j], Input.get(13));
   sht_absea.put(sensID[j], Integer.parseInt(Input.get(14).substring
      (0,3)));
   sht_abgd.put(sensID[j], Integer.parseInt(Input.get(15).substring
      (0,1)));
   j++;
     }

  }
}
```

C.5 SWLUpdateNetwork.java

Listing C.5: SWLUpdateNetwork.java

```
/* SWL Network, updated dynamically with the change in network
*/
import java.lang.*;
import java.util.*;
import java.util.*;
import java.io.*;
import baseswl.*;

public class SWLUpdateNetwork
{
  int tot_node;
  int tot_sensor;
  int num_sensor;
```

```
int[] sensor;
long[] totTime;
String nUID = "";
String sUID="";
String sens_UUID[];
String nsens_UUID[][];
/* Object declarations and instance variables*/
String name; //name of the sensorNet
Gateway[] G; // gateway
GatewayCD[] GCD;// gateway Comm. device
SN[] GSN; // gateway sensor nodes attached to the gateway
SensorNode[] RSN; //list of remote sensor nodes
DACBoard b1;
DACBoard b2;
DACBoard b3;
Channel[] ch1;
Channel[] ch2;
Channel[] ch3;
Sensor[] sens;// list of sensors attached to sensor nodes
degC degcUnit;
cm cmUnit;
mm mmUnit;
ms_cm ms_cmUnit;
mV mvUnit;
V vUnit;
A aUnit;
W_m2 W_m2Unit;
Percent percentUnit;
TempObservation tempOb;
SrObservation srOb;
WlObservation wlOb;
OrpObservation orpOb;
EcObservation ecOb;
RhObservation rhOb;
CompObservation compOb;
VoltsObservation voltOb;
//int p=0;

public SWLUpdateNetwork(){}

public void getNetwork(BuildNetwork BN, long lTime)
{
    /* Initializes all the objects */
    b1=new DACBoard();
    b2=new DACBoard();
    b3=new DACBoard();
    G=new Gateway[1];
    GCD=new GatewayCD[1];
    GSN=new SN[1];
    ch1= new Channel[6];
    ch2= new Channel[4];
    ch3= new Channel[5];
    ch1[0]=new Channel("ANALOG",2);
    ch1[1]=new Channel("ANALOG",3);
```

```
ch1[2]=new Channel("ANALOG",4);
ch1[3]=new Channel("ANALOG",5);
ch1[4]=new Channel("ANALOG",6);
ch1[5]=new Channel("ANALOG",1);
ch2[0]=new Channel("ANALOG",2);
ch2[1]=new Channel("ANALOG",3);
ch2[2]=new Channel("ANALOG",5);
ch2[3]=new Channel("ANALOG",1);
ch3[0]=new Channel("ANALOG",1);
ch3[1]=new Channel("ANALOG",2);
ch3[2]=new Channel("ANALOG",3);
ch3[3]=new Channel("ANALOG",4);
ch3[4]=new Channel("ANALOG",5);
b1.setChannels(ch1);
b2.setChannels(ch2);
b3.setChannels(ch3);

degcUnit = new degC();
cmUnit = new cm();
mmUnit = new mm();
ms_cmUnit = new ms_cm();
mvUnit = new mV();
vUnit = new V();
aUnit = new A();
W_m2Unit = new W_m2();
percentUnit = new Percent();
tempOb = new TempObservation();
srOb = new SrObservation();
wlOb = new WlObservation();
orpOb = new OrpObservation();
ecOb = new EcObservation();
rhOb = new RhObservation();
compOb = new CompObservation();
voltOb = new VoltsObservation();
tot_node = BN.NodeID.size();
RSN = new SensorNode[tot_node];
tot_sensor = BN.SID.size();
sens=new Sensor[tot_sensor+7];
sens_UUID = new String[tot_sensor];
totTime = new long[tot_node];
int a=0;
//System.out.println(tot_sensor+","+tot_node);
for (Enumeration e = BN.SID.keys(); e.hasMoreElements() ;)
{
    sens_UUID[a] = (String)e.nextElement();
a++;
}

for(int l=0; l< tot_sensor; l++)
{
sens[l] = new Sensor();
     sens[l].setName(BN.sname.get(sens_UUID[l]));
sens[l].setDescription(BN.sdescription.get(sens_UUID[l]));
sens[l].setModel(BN.smodel.get(sens_UUID[l]));
```

```
   sens[l].setManufacturer(BN.smanufacturer.get(sens_UUID[l]));
      sens[l].setObservation(srOb);
   sens[l].setLat(BN.sltdeg.get(sens_UUID[l]), BN.sltmin.get(
      sens_UUID[l]), BN.slt.get(sens_UUID[l]), BN.sltdir.get(
      sens_UUID[l]));
   sens[l].setLong(BN.slngdeg.get(sens_UUID[l]), BN.slngmin.get(
      sens_UUID[l]), BN.slng.get(sens_UUID[l]),BN.slngdir.get(
      sens_UUID[l]));
   sens[l].setHAboveSea(BN.sht_absea.get(sens_UUID[l]));
   sens[l].setHAboveGround(BN.sht_abgd.get(sens_UUID[l]));
   sens[l].setUnit(W_m2Unit);
   sens[l].setLowBound(320);
   sens[l].setHighBound(1000);
   sens[l].setChannel(ch1[l]);
   }
   /*for(int l=0;l< tot_sensor; l++)
   {
System.out.println(BN.sname.get(sens_UUID[l])+","+BN.sdescription
   .get(sens_UUID[l])+","+BN.smodel.get(sens_UUID[l])+","+BN.
   smanufacturer.get(sens_UUID[l]));
   }*/
   for(int i=0; i< tot_node; i++)
   {
nUID = BN.NodenUUID.get(i);
//System.out.println("          "+nUID);
String nodeID;
int l=0;
     num_sensor = BN.Nodensensor.get(nUID);
     nsens_UUID = new String[tot_node][num_sensor];
System.out.println("-------------"+lTime+"---------------"+BN.
   nst_time.get(nUID));
totTime[i] = lTime - BN.nst_time.get(nUID);
if(totTime[i]==0)
{
   totTime[i] = 1;
}
RSN[i]= new SensorNode(0,num_sensor);
RSN[i].setName(BN.Nodename.get(nUID));
RSN[i].setDescription(BN.Nodedescription.get(nUID));
RSN[i].setManufacturer(BN.Nodemanufacturer.get(nUID));
RSN[i].setModel(BN.Nodemodel.get(nUID));
RSN[i].setLat(BN.Nodeltdeg.get(nUID), BN.Nodeltmin.get(nUID), BN.
   Nodelt.get(nUID), BN.Nodeltdir.get(nUID));
RSN[i].setLong(BN.Nodelngdeg.get(nUID), BN.Nodelngmin.get(nUID),
   BN.Nodelng.get(nUID), BN.Nodelngdir.get(nUID));
RSN[i].setHAboveSea(BN.Nodeht_absea.get(nUID));
RSN[i].setHAboveGround(BN.Nodeht_abgd.get(nUID));
RSN[i].setCounterSensor(num_sensor);
RSN[i].setttime(totTime[i]);

for(int k=0; k< tot_sensor; k++)
{
    nodeID = BN.Nodesensor_UUID.get(sens_UUID[k]);
    if (nodeID == nUID)
```

```
        {
       RSN[i].setSensor(l,sens[k]);
       l=l+1;
        }
   }
       //p=p+1;
       }
        for(int p=0; p< tot_node; p++)
   {
     //System.out.println(p+"---"+RSN[p].getPosition().getLatitude()
        +"---");
     //System.out.println(p+"---"+RSN[p].getPosition().getLongitude
        ()+"---");
       System.out.println(p+"---"+RSN[p].getttime()+"---");
   }

       sens[tot_sensor] = new Sensor();
       sens[tot_sensor].setName("GT1");
       sens[tot_sensor].setDescription("Gateway temperature sensor");
       sens[tot_sensor].setModel("PP-T-24");
       sens[tot_sensor].setManufacturer("Omega Technologies Company")
          ;
       sens[tot_sensor].setObservation(tempOb);
       sens[tot_sensor].setLat(45, 55, 26.0, "N");
       sens[tot_sensor].setLong(66, 40, 12.0, "W");
       sens[tot_sensor].setHAboveSea(116.0);
       sens[tot_sensor].setHAboveGround(2.0);
       sens[tot_sensor].setUnit(degcUnit);
       sens[tot_sensor].setLowBound(-60);
       sens[tot_sensor].setHighBound(105);
       sens[tot_sensor].setChannel(ch3[0]);

       sens[tot_sensor+1] = new Sensor();
       sens[tot_sensor+1].setName("GV1");
       sens[tot_sensor+1].setDescription("Voltage sensor");
       sens[tot_sensor+1].setModel("PP-T-24");
       sens[tot_sensor+1].setManufacturer("Crossbow Technology Inc.")
          ;
       sens[tot_sensor+1].setObservation(voltOb);
       sens[tot_sensor+1].setLat(45, 55, 26.0, "N");
       sens[tot_sensor+1].setLong(66, 40, 12.0, "W");
       sens[tot_sensor+1].setHAboveSea(116.0);
       sens[tot_sensor+1].setHAboveGround(2.0);
       sens[tot_sensor+1].setUnit(vUnit);
       sens[tot_sensor+1].setLowBound(0);
       sens[tot_sensor+1].setHighBound(3.3);
       sens[tot_sensor+1].setChannel(ch3[1]);

       sens[tot_sensor+2] = new Sensor();
       sens[tot_sensor+2].setName("GV2");
       sens[tot_sensor+2].setDescription("Voltage sensor");
       sens[tot_sensor+2].setModel("PP-T-24");
       sens[tot_sensor+2].setManufacturer("Crossbow Technology Inc.")
          ;
```

```
        sens[tot_sensor+2].setObservation(voltOb);
        sens[tot_sensor+2].setLat(45, 55, 26.0, "N");
        sens[tot_sensor+2].setLong(66, 40, 12.0, "W");
        sens[tot_sensor+2].setHAboveSea(116.0);
        sens[tot_sensor+2].setHAboveGround(2.0);
        sens[tot_sensor+2].setUnit(vUnit);
        sens[tot_sensor+2].setLowBound(0);
        sens[tot_sensor+2].setHighBound(3.3);
        sens[tot_sensor+2].setChannel(ch3[2]);

        sens[tot_sensor+3] = new Sensor();
        sens[tot_sensor+3].setName("GV3");
        sens[tot_sensor+3].setDescription("Voltage sensor");
        sens[tot_sensor+3].setModel("PP-T-24");
        sens[tot_sensor+3].setManufacturer("Crossbow Technology Inc.")
           ;
        sens[tot_sensor+3].setObservation(voltOb);
        sens[tot_sensor+3].setLat(45, 55, 26.0, "N");
        sens[tot_sensor+3].setLong(66, 40, 12.0, "W");
        sens[tot_sensor+3].setHAboveSea(116.0);
        sens[tot_sensor+3].setHAboveGround(2.0);
        sens[tot_sensor+3].setUnit(vUnit);
        sens[tot_sensor+3].setLowBound(0);
        sens[tot_sensor+3].setHighBound(3.3);
        sens[tot_sensor+3].setChannel(ch3[3]);

        sens[tot_sensor+4] = new Sensor();
        sens[tot_sensor+4].setName("GV4");
        sens[tot_sensor+4].setDescription("Voltage sensor");
        sens[tot_sensor+4].setModel("PP-T-24");
        sens[tot_sensor+4].setManufacturer("Crossbow Technology Inc.")
           ;
        sens[tot_sensor+4].setObservation(voltOb);
        sens[tot_sensor+4].setLat(45, 55, 26.0, "N");
        sens[tot_sensor+4].setLong(66, 40, 12.0, "W");
        sens[tot_sensor+4].setHAboveSea(116.0);
        sens[tot_sensor+4].setHAboveGround(2.0);
        sens[tot_sensor+4].setUnit(vUnit);
        sens[tot_sensor+4].setLowBound(0);
        sens[tot_sensor+4].setHighBound(3.3);
        sens[tot_sensor+4].setChannel(ch3[4]);

        sens[tot_sensor+5] = new Sensor();
        sens[tot_sensor+5].setName("GI1");
        sens[tot_sensor+5].setDescription("Derived current");
        sens[tot_sensor+5].setModel("PP-T-24");
        sens[tot_sensor+5].setManufacturer("Crossbow Technology Inc.")
           ;
        sens[tot_sensor+5].setObservation(compOb);
        sens[tot_sensor+5].setLat(45, 55, 26.0, "N");
        sens[tot_sensor+5].setLong(66, 40, 12.0, "W");
        sens[tot_sensor+5].setHAboveSea(116.0);
        sens[tot_sensor+5].setHAboveGround(2.0);
        sens[tot_sensor+5].setUnit(aUnit);
```

```
 sens[tot_sensor+6] = new Sensor();
 sens[tot_sensor+6].setName("GI2");
 sens[tot_sensor+6].setDescription("Derived current");
 sens[tot_sensor+6].setModel("PP-T-24");
 sens[tot_sensor+6].setManufacturer("Crossbow Technology Inc.")
    ;
 sens[tot_sensor+6].setObservation(compOb);
 sens[tot_sensor+6].setLat(45, 55, 26.0, "N");
 sens[tot_sensor+6].setLong(66, 40, 12.0, "W");
 sens[tot_sensor+6].setHAboveSea(116.0);
 sens[tot_sensor+6].setHAboveGround(2.0);
 sens[tot_sensor+6].setUnit(aUnit);


/* Sets properties of the gateway sensor node */
 GSN[0]=new SN(2,7);
 GSN[0].setName("GSN1");
 GSN[0].setDescription("Gateway Sensor node");
 GSN[0].setManufacturer("Crossbow Technology Inc.");
 GSN[0].setModel("MPR400CB");
 GSN[0].setLat(45, 55, 26.0, "N");
 GSN[0].setLong(66, 40, 12.0, "W");
 GSN[0].setHAboveSea(116.0);
 GSN[0].setHAboveGround(2.0);
 GSN[0].setCounterRSN(tot_node);
 for(int i=0;i<tot_node;i++)
 {
    GSN[0].setRSN(0,RSN[i]);
 }

 GSN[0].setCounterSensor(7);
 GSN[0].setSensor(0,sens[tot_sensor]);
 GSN[0].setSensor(1,sens[tot_sensor+1]);
 GSN[0].setSensor(2,sens[tot_sensor+2]);
 GSN[0].setSensor(3,sens[tot_sensor+3]);
 GSN[0].setSensor(4,sens[tot_sensor+4]);
 GSN[0].setSensor(5,sens[tot_sensor+5]);
 GSN[0].setSensor(6,sens[tot_sensor+6]);

 GCD[0] = new GatewayCD();
 GCD[0].setName("GCD1");
 GCD[0].setDescription("Gateway Comm device");
 GCD[0].setManufacturer("Nokia Corp.");
 GCD[0].setModel("6310i");
 GCD[0].setHAboveSea(116.0);
 GCD[0].setHAboveGround(2.0);
 GCD[0].setLat(45, 55, 26.0, "N");
 GCD[0].setLong(66, 40, 12.0, "W");

/* Sets properties of the gateway */
 G[0] = new Gateway();
 G[0].setName("G1");
 G[0].setDescription("Gateway");
```

```
      G[0].setGCD(GCD[0]);
      G[0].setGSN(GSN[0]);
      G[0].setHAboveSea(116.0);
      G[0].setHAboveGround(2.0);
      G[0].setLat(45, 55, 26.0, "N");
      G[0].setLong(66, 40, 12.0, "W");
  }
}
```

Appendix D

Web Application 2 - Google Maps

D.1 Base Station2

D.1.1 SWLAnnounce2.java

In SWLAnnounce2.java the database operations and operations performed when message type received is report are different from the SWLAnnounce1.java. In the following code fragment the databaseOperations function and the messageReceived function is attached.

Listing D.1: SWLAnnounce2.java - Database Operations

```
public void messageReceived(int to, Message m)
  { ...
   if (msg.get_swlMsgType() == 3) //report message
   {
      //cal = Calendar.getInstance();
      Listen.logSensorReport(id, seqNo, payload);
   }
   ...
  }
   ...
  public void databaseOperations(int SenderID)
  {
   int dnID = 0;
   int dnGCDID = 0;
   int dnnetID = 0;
   int ncount = 0;
   int nc=0;
   int dnhtabsea = 0;
   int dnhtabgd = 0;
   String dnUUID[]= new String[10];
   String rsnID="";
   int stot_rec = 0;
   int ntot_rec = 0;
   int postot_rec = 0;
   int j=0;
   String NULL_DAT = "0000-00-00";
   String node_lat="";
   String node_lon="";
```

```
int dsID[] = new int[ARM.nsensors];
String dsUUID[] = new String[ARM.nsensors];
String dspositID[] = new String[ARM.nsensors];
int test=0;
int t=0;
//Query for Sensor Node table
try
{
   rs = stmt.executeQuery("SELECT count(*) as tot_node from
      SENSORNODE");
   while(rs.next()) {
   nc = rs.getInt("tot_node");
   }
if(nc>0)
{
   rs = stmt.executeQuery("SELECT DISTINCT UUID from SENSORNODE")
      ;
   while(t<=nc)
   {
     while(rs.next()) {
   dnUUID[t] = rs.getString("UUID");
   break;
     }
     t++;
   }
  for(int p=0;p<=t;p++)
  {
    if((ARM.nUUID.equals(dnUUID[p])))
    {   test=test+1; }
  }
  if(test== 0)
  {
   ncount = nc +1;
   rsnID = "RSN"+SenderID;
   stmt.executeUpdate("INSERT INTO SENSORNODE(id,"
                  +"manufacturer,"
                  +"model,"
                  +"name,"
                  +"UUID)"
                +"VALUES"+"("+SenderID+","
                       +"'Crossbow Technology Inc.'"+","
                       +"'MPR400CB'"+","
                       +"'"+rsnID+"',"
                       +"'"+ARM.nUUID+"')");
   System.out.println("Writing to database complete!!!");
   }
 }else {
      ncount = nc +1;
   rsnID = "RSN"+SenderID;
   stmt.executeUpdate("INSERT INTO SENSORNODE(id,"
                  +"manufacturer,"
                  +"model,"
                  +"name,"
                  +"UUID)"
```

```
                +"VALUES"+"("+SenderID+","
                        +"'Crossbow Technology Inc.'"+","
                        +"'MPR400CB'"+","
                        +"'"+rsnID+"',"
                        +"'"+ARM.nUUID+"')");
}
  //Query for Sensor table
        int l=0;
   rs = stmt.executeQuery("SELECT DISTINCT UUID from SENSOR");
   while(j<ARM.nsensors)
   {
   while(rs.next()) {
      dsUUID[j] = rs.getString("UUID");
      break;
   }
      j++;
   }
   for(int i=0; i<ARM.nsensors;i++)
   {
     for(int k=0; k< ARM.nsensors; k++)
     {
      if(!(ARM.sUUID[i].equals(dsUUID[k])))
      {
      l=l+1;
      }
     }
     if(l<0)
     {
   rs = stmt.executeQuery("SELECT count(*) as num_record from
       SENSOR");
   while(rs.next()) {
      stot_rec = rs.getInt("num_record");
   }
   int scount = stot_rec +1;
   stmt.executeUpdate("INSERT INTO SENSOR(ID,"
                  +"manufacturer,"
                  +"description,"
                  +"model,"
                  +"name,"
                  +"unit,"
                  +"UUID)"
                +"VALUES"+"("+scount+","+"Omega Technologies Comp
                    ."+","+"Analog Sensor"+","
                    +"WL300"+","+"Analog"+","+"W_m2Unit"+","+"'"+
                       ARM.sUUID[i]+"')");
   System.out.println("Writing to database complete!!!");
     }
   }
    //Query for POSIT table
   rs = stmt.executeQuery("SELECT latitude, longitude from POSIT
       where UUID='"+ARM.nUUID+"'");
   while(rs.next()) {
      node_lat = rs.getString("latitude");
      node_lon = rs.getString("longitude");
```

```
      }

      if(!(ARM.lttot.equals(node_lat) && ARM.lntot.equals(node_lon))
         )
      {
      rs = stmt.executeQuery("SELECT count(*) as num_record from
         POSIT");
      while(rs.next())
      {
         postot_rec = rs.getInt("num_record");
      }
      int poscount = postot_rec +1;
      stmt.executeUpdate("update POSIT SET end_datetime ='"+sqlDate
         +"' where UUID='"+ARM.nUUID+"'");
      stmt.executeUpdate("INSERT INTO POSIT(ID,"
                     +"UUID,"
                     +"LATITUDE,"
                     +"LONGITUDE,"
                     +"start_datetime,"
                     +"end_datetime)"
                + "VALUES" +"("+poscount+","
                     +"'"+ARM.nUUID+"',"
                     +"'"+ARM.lttot+"',"
                     +"'"+ARM.lntot+"',"
                     +"'"+sqlDate+"',"
                     +"'0000-00-00 00:00:00')");

      System.out.println("Writing to the database complete!!!!!!");
      }
      else
      {
      stmt.executeUpdate("update POSIT SET start_datetime = '"+
         sqlDate+"' where UUID='"+ARM.nUUID+"' AND end_datetime =
'0000-00-00 00:00:00'");
      }

   //Query for SENSOR_SN table
   }
   catch(Exception e)
   {
      e.printStackTrace();
   }
  }
```

D.1.2 SWLListen.java

SWLListen object class is called by the SWLAnnounce1.java when the message type received is report. SWLListen changes the raw sensor readings into derived values by applying the appropriate calibration. Also this program saves the sensor readings sent by the nodes in the network in the OBSERVATION table.

Listing D.2: SWLListen.java

```
import java.sql.*;
```

```
import java.util.*;
import java.lang.*;
import java.io.*;
import javax.comm.*;
//import javax.comm.*;
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

public class SWLListen
{
  Calendar cal;
  PrintWriter log;
  String logdate;
  java.sql.Timestamp  sqlDate = new java.sql.Timestamp(new java.util
     .Date().getTime());
  Connection conn;
  Statement stmt;
  ResultSet rs;
  //alert flags
  boolean ALERT_G0_HIGHTEMP=false;

  //DatumPair datumPair[32767];
  Hashtable datumPairs = new Hashtable();

  boolean[] firstMessage = new boolean[256]; // flag indicating
     weather this is the first time
                                                 // this listner got
                                                    a message from a
                                                     specific sensor
  long[] startTime = new long[256]; // actual time the most recent
     log message recieved from a sensor node
  long[] rawStartTime = new long[256]; // time stamp that
     corrisponds to the actual time

  int second = 1000;
  int minute = 60 * second;
  int hour = 60 * minute;
  int day = 24 * hour;

  int biSecond = 1024;
  int biMinute = 60 * biSecond;
  int biHour = 60 * biMinute;



  public SWLListen()
   {
      try
      {
         Class.forName("com.mysql.jdbc.Driver").newInstance();
         conn = DriverManager.getConnection( "jdbc:mysql://localhost
            /swl2","swluser", "swlsecret" );
         stmt = conn.createStatement();
```

```
   }
   catch(Exception e)
   {
         e.printStackTrace();
   }
}

public java.util.Date getDate()
{
DateFormat dateFormat = new SimpleDateFormat ("yyyy/MM/dd HH:mm:
   ss");
     java.sql.Timestamp  sqlDate = new java.sql.Timestamp(new
        java.util.Date().getTime());
return sqlDate;
}

 public void logSensorReport(int repNo, int seqNo, String report)
 {
String nuid = "";
   String suid = "";
int tr = 0;
int i=0;
StringTokenizer st = new StringTokenizer(report, ",.=");
String nodename = st.nextToken();   if(!st.hasMoreTokens())
   return;
String sensorname = st.nextToken(); if(!st.hasMoreTokens())
   return;
String msgType = st.nextToken();

int nodenum = 0;
int sensornum = 0;
if (nodename.length() > 1) nodenum = Integer.parseInt(nodename.
   substring(1));
//if (sensorname.length() > 1) sensornum = Integer.parseInt(
   sensorname.substring(1));
//(int)sensorname.charAt(1)+48;
if (sensorname.length() > 1) sensornum = (int)sensorname.charAt
   (1)+48;

String sn = new String(nodenum+"_"+repNo+"_"+seqNo);
DatumPair dp = new DatumPair();

if (datumPairs.containsKey(sn))
{  dp = (DatumPair)datumPairs.get(sn);
}
else
{  datumPairs.put(sn, dp);
}

if (msgType.charAt(0) == 't')
{     // converts the hex string to an integer
      //long timeStamp = Long.parseLong(st.nextToken().trim(),
         16);
   int dayOfUse = Integer.parseInt(st.nextToken().trim(), 16);
```

```
    int timeOfDay = Integer.parseInt(st.nextToken().trim(), 16);

    int hh = timeOfDay/biHour;
    int mm = (timeOfDay%biHour)/biMinute;
    int ss = ((timeOfDay%biHour)%biMinute)/biSecond;

    Calendar ical = Calendar.getInstance();

    if (firstMessage[sensornum] == true)
    {  firstMessage[sensornum] = false;
       startTime[sensornum] = cal.getTimeInMillis();
    }

    long currentTime = startTime[sensornum] + day*dayOfUse;
    ical.setTimeInMillis(currentTime);
    ical.set(Calendar.HOUR_OF_DAY, hh);
    ical.set(Calendar.MINUTE, mm);
    ical.set(Calendar.SECOND, ss);

    String timestamp =   ical.get(Calendar.YEAR) + "-" +
                (ical.get(Calendar.MONTH)+1) + "-" +
                ical.get(Calendar.DAY_OF_MONTH) + " " +
                ical.get(Calendar.HOUR_OF_DAY) + ":" +
                ical.get(Calendar.MINUTE) + ":" +
                ical.get(Calendar.SECOND);

    dp.setTime(timestamp);
}
else
{  dp.setData(report);
}

if (dp.isComplete())
{
     try
     {
       rs = stmt.executeQuery("SELECT UUID from SENSORNODE where
          id="+dp.getNodeNum());
       while(rs.next()) {
       nuid = rs.getString("UUID");
       }
       //System.out.println(dp.getNodeNum()+","+nuid);

       rs = stmt.executeQuery("SELECT sensor_uuid from SENSOR_SN
          where message_id='"+dp.getSensorName()+"' AND node_uuid
           ='"+nuid+"'");
       while(rs.next()) {
       suid = rs.getString("sensor_uuid");
       }
    //System.out.println(dp.getNodeNum()+","+suid);

       rs=stmt.executeQuery("SELECT count(*) as tot_read from
          OBSERVATION");
    while(rs.next()) {
```

```
        tr = rs.getInt("tot_read");
        }
     i = tr+1;
        stmt.executeUpdate("INSERT INTO OBSERVATION(id,"
              +"sensor_id,"
              +"raw_value,"
              +"derived_value,"
              +"timestamp)"
                   +"VALUES"+"("+i+",'"
              +suid+"',"
              +dp.getRawValue()+","
              +dp.getDerivedValue()+","
              +"'"+getDate()+"')");
     }
    catch(Exception e)
    {
        e.printStackTrace();
    }

    datumPairs.remove(sn); // remove it from the hashtable to avoid
        colision with the next report
  }
   }
}
 class DatumPair
 {
  long tref;
  long timeStamp;
  int nodenum;
  String nodename;
  int sensornum;
  String sensorname;
  int raw_value;
  double derived_value;
  String dataString;
  String timeString;

  boolean isTimeSet;
  boolean isDataSet;

  public DatumPair()
  {  isTimeSet = false;
     isDataSet = false;
  }

  public void setData(String report)
  {
     StringTokenizer st = new StringTokenizer(report, ".=");
     nodename = st.nextToken();
     sensorname = st.nextToken();
     String data = st.nextToken();

     nodenum = 0;
     sensornum = 0;
```

```
if (nodename.length() > 1) nodenum = Integer.parseInt(nodename
   .substring(1));
if (sensorname.length() > 1) sensornum = (int)sensorname.
   charAt(1)+48;
// Integer.parseInt(sensorname.substring(1));
raw_value = Integer.parseInt(data.trim());

derived_value = 0.0;

switch ( report.charAt(0) )
{
  case 's':
    switch (sensorname.charAt(0))
    { case 'v':
      derived_value = ((double)raw_value) / 100.0;
      // INPUT LOW POWER ALERT MESSAGE HERE
      break;
   case 's':
      double voltage = raw_value*2.5/4096;
      derived_value = 100.0*(voltage/2200)/0.65e-6;
      //derived_value = voltage;
      break;
   case 'h':
      derived_value = ((-3.9559e-6*raw_value*raw_value+6.1797e
         -2*raw_value-67.681)+0.5);
      break;
   default:
      derived_value = ((double)raw_value) / 100.0;
      break;
    }
  break;
  case 'g':
   switch (sensorname.charAt(0))
   {  case 'v':
         derived_value = 0.6*1024/raw_value;
         break;
      case 's':
         switch (sensorname.charAt(1))
         {
            case '2':
            derived_value = raw_value*3.1/1024/0.15178;
            break;
            case '3':
            derived_value= raw_value*3.1/1024/0.15234;
            break;
            case '4':
            derived_value= raw_value*3.1/1024/0.15201;
            break;
            case '5':
            double adc = raw_value*3.1/1024;
            double current = (3.1-adc)/13000;
            double resistance = adc/current;
            derived_value = 298.15*3977/(3977+298.15*(Math.log
               (resistance)-9.2103));
```

```
                derived_value = derived_value - 273.15;

                // ALERT SECTION
                //if (derived_value > 21) { alertHighTemp(nodenum,
                    true); }
                //else      { alertHighTemp(nodenum, false); }
                break;
                default:
                break;
             }
             break;
          default:
             derived_value = ((double)raw_value) / 100.0;
             break;
       }
       break;
    default:
       System.out.println("Alert: " + report);
       break;
    }

    isDataSet = true;
 }

 public void setTime(String time)
 {   timeString = time;
     isTimeSet = true;
 }

 public boolean isComplete()
 {  return (isTimeSet && isDataSet);
 }

 public int getNodeNum() { return nodenum; }
 public String getNodeName() { return nodename; }
 public int getSensorNum() { return sensornum; }
 public String getSensorName() { return sensorname; }
 public int getRawValue() { return raw_value; }
 public double getDerivedValue() { return derived_value; }
 public String getTimeStamp() { return timeString; }
}

class DPKey
{   public int reportNo;
 public int seqNo;

 public DPKey(int r, int s)
 {  reportNo = r;
    seqNo = s;
 }
}
```

D.2 Server Code

D.2.1 Nodes.php

This program is an interface between the Network2.html program and the database. Nodes.php is called by Network2.html, then gets the network information from the database and sends back an XML file to the Network2.html.

Listing D.3: Nodes.php

```
<?php

$username = "root";
$password = "swl214ca";
$database = "swl2";
$server = "localhost";

//start XML file, create parent node
$dom = new DOMDocument("1.0");
$node = $dom->createElement("markers");
$parnode = $dom->appendChild($node);

//Opens a connection to MySQL Server
$connection = mysql_connect(localhost, $username, $password);
if(!$connection) { die('Not Connected: ' . mysql_error());}

//set the active MySQL database
$db_selected = mysql_select_db($database, $connection);
if (!$db_selected) {
  die ('Can\'t use db : ' . mysql_error());
}

// Select all the rows in the markers table
$query1 = "SELECT sn.UUID, sn.id, sn.manufacturer, sn.model, sn.name
    , p.latitude, p.longitude, p.height_above_sea,
p.height_above_ground, p.start_datetime, p.end_datetime FROM
    SENSORNODE sn, POSIT p WHERE sn.UUID=p.UUID AND
p.end_datetime='0000-00-00 00:00:00'";
$result1 = mysql_query($query1);
if (!$result1) {
  die('Invalid query: ' . mysql_error());
}

$query2 = "SELECT s.manufacturer, s.description, s.model, s.name, s.
    unit, s.UUID, ssn.node_uuid from SENSOR s, SENSOR_SN ssn
WHERE s.UUID = ssn.sensor_uuid";

$result2 = mysql_query($query2);
if (!$result2) {
  die('Invalid query: ' . mysql_error());
}

 $i=0;

 $j=0;
```

```
while($row1 = @mysql_fetch_assoc($result1))
{
     $A1=$row1['UUID'];
     //echo $A1;

$UUID[$i] = $row1['UUID'];
   $ID[$A1] = $row1['id'];
     $Manu[$A1] = $row1['manufacturer'];
$Model[$A1] = $row1['model'];
$Name[$A1] = $row1['name'];
$lat[$A1] = $row1['latitude'];
$lon[$A1] = $row1['longitude'];
$habsea[$A1] = $row1['height_above_sea'];
$habgd[$A1] = $row1['height_above_ground'];
$stdat[$A1] = $row1['start_datetime'];
$enddat[$A1] = $row1['end_datetime'];

$sttimef2[$A1] = strtotime($stdat[$A1]);

$endtimef2[$A1] = strtotime($enddat[$A1]);
$i=$i+1;
     //echo "$ProjectName[$A1]";
}



while($row2 = @mysql_fetch_assoc($result2))
{
     $A2 = $row2['node_uuid'];
     //echo $A2;

$anID[$j] = $row2['node_uuid'];
   $sUUID[$j][$A2] = $row2['UUID'];
     $sManu[$j][$A2] = $row2['manufacturer'];

$sDescrip[$j][$A2] = $row2['description'];
$sModel[$j][$A2] = $row2['model'];
$sName[$j][$A2] = $row2['name'];

$sUnit[$j][$A2] = $row2['unit'];

$j=$j+1;
     //echo "$ProjectName[$A1]";
}
//echo $j;

for($k=0; $k < $i; $k++)
{
$U = $UUID[$k];
$latdeg[$U] = substr($lat[$U], 0, 2);
$latmin[$U] = substr($lat[$U],3, 5);
$latsec[$U] = substr($lat[$U],6, 5);
$latdir[$U] = substr($lat[$U], -1);
$londeg[$U] = substr($lon[$U], 0, 2);
```

```
   $lonmin[$U] = substr($lon[$U],3, 5);
   $lonsec[$U] = substr($lon[$U],6, 5);
   $londir[$U] = substr($lon[$U],-1);
   $convertlat[$U] = convert($latdeg[$U], $latmin[$U], $latsec[$U],
      $latdir[$U]);
   $convertlon[$U] = convert($londeg[$U], $lonmin[$U], $lonsec[$U],
      $londir[$U]);
  }

  function convert($d, $m, $s, $dir)
  {
   $mm = $m + ($s/(1000*60));
   $dd = $d + ($mm/60);

   if($dir=='N' || $dir=='E')
   {
      return($dd);
   }
        else
   {
      $dd=-$dd;
      return($dd);
        }
   }

header("Content-type: text/xml");

// Iterate through the rows, adding XML nodes for each
  for($x=0;$x < $i;$x=$x+1)
  {
      $l=0;
      $node = $dom->createElement("marker");
      $newnode = $parnode->appendChild($node);
        // ADD TO XML DOCUMENT NODE
      $UUID1 = $UUID[$x];
       //echo $UUID1;
      $newnode->setAttribute("ID",$ID[$UUID1]);
      //echo "$ProjectName[$ASN1]";
      $newnode->setAttribute("Manufacturer",$Manu[$UUID1]);
      $newnode->setAttribute("Model", $Model[$UUID1]);
      $newnode->setAttribute("Name", $Name[$UUID1]);
      $newnode->setAttribute("Latitude", $convertlat[$UUID1]);
      $newnode->setAttribute("Longitude", $convertlon[$UUID1]);
      $newnode->setAttribute("habsea", $habsea[$UUID1]);
      $newnode->setAttribute("habgd", $habgd[$UUID1]);
      $newnode->setAttribute("stdat", $stdat[$UUID1]);
      $newnode->setAttribute("enddat", $enddat[$UUID1]);

      $newnode->setAttribute("sttimef2", $sttimef2[$A1]);
      $newnode->setAttribute("endtimef2", $endtimef2[$A1]);


      for($y=0; $y<$j;$y=$y+1)
     {
```

```
   $UUID2 = $anID[$y];
   //echo $UUID2;

   if($UUID2 == $UUID1)
   {
     $newnode->setAttribute("sUUID$l",$sUUID[$y][$UUID2]);
     //echo $sUUID[$UUID2];
     $newnode->setAttribute("sManufacturer$l",$sManu[$y][$UUID2]);
          $newnode->setAttribute("sDescription$l",$sDescrip[$y][
             $UUID2]);
          $newnode->setAttribute("sModel$l",$sModel[$y][$UUID2]);
          $newnode->setAttribute("sName$l",$sName[$y][$UUID2]);
          $newnode->setAttribute("sUnit$l",$sUnit[$y][$UUID2]);
          $l=$l+1;
   }
    }
    $newnode->setAttribute("NumSensor", $l);
  }

echo $dom->saveXML();
?>
```

Below is the sample response sent by the Nodes.php to the Network2.html. The file is in XML format and contains information of one node in the network.

Listing D.4: Nodes.php - Sample Response in XML format

```
<markers>
   <marker ID="2" Manufacturer="Crossbow Technology Inc." Model="
      MPR400CB" Name="RSN2" Latitude="45.9497347222"
   Longitude="-66.6427716667" habsea="116" habgd="2" stdat
      ="2010-07-30 15:08:58" enddat="0000-00-00 00:00:00"
   lastheard="1280513535" sUUID0="391e816927484bcb" sManufacturer0="
      Crossbow Tech Inc." sDescription0="Voltage"
   sModel0="PP-T-24" sName0="GV1" sUnit0="vUnit" sUUID1="5867
      a59e24f844f3" sManufacturer1="Hamamatsu Corp."
   sDescription1="Solar Radiation" sModel1="S1133-14" sName1="SR1"
      sUnit1="W_m2" sUUID2="c8d72dbabba8443d"
   sManufacturer2="Omega Technologies Company" sDescription2="
      AirTemperature" sModel2="PP-T-24" sName2="AT1" sUnit2="degc
   Unit" NumSensor="3"/>
</markers>
```

D.2.2 Observation.php

Called by the GetObservation.html and last10obs.html programs, this server code gets the 10 latest observations for a particular sensor and sends back to the html programs in XML format.

Listing D.5: Observation.php

```
<?php

$posted = $_GET;
$NodeName= $posted["node"];
```

```
$SensorName = $posted["sensor"];
//$NodeName="RSN3";
//$SensorName= "GV1";

$username = "root";
$password = "swl214ca";
$database = "swl2";
$server = "localhost";

//start XML file, create parent node
$dom = new DOMDocument("1.0");
$node = $dom->createElement("markers");
$parnode = $dom->appendChild($node);

//Opens a connection to MySQL Server
$connection = mysql_connect(localhost, $username, $password);
if(!$connection) { die('Not Connected: ' . mysql_error());}

//set the active MySQL database
$db_selected = mysql_select_db($database, $connection);
if (!$db_selected) {
  die ('Can\'t use db : ' . mysql_error());
}

// Select node UUID from Sensor Node
$query1 = "SELECT UUID from SENSORNODE WHERE name='$NodeName'";
$result1 = mysql_query($query1);
if (!$result1) {
  die('Invalid query: ' . mysql_error());
}
while($row1 = @mysql_fetch_assoc($result1))
   {
       $nUID=$row1['UUID'];
   }
 //echo "$nUID";

//Select sensorUUID from the SENSOR_SN table.
$query2 = "SELECT UUID from SENSOR where name='$SensorName'";
$result2 = mysql_query($query2);
if (!$result2) {
  die('Invalid query: ' . mysql_error());
}
   while($row2 = @mysql_fetch_assoc($result2))
   {
        $suid = $row2['UUID'];
   }
  //echo $suid;

$query3 = "SELECT count(*) as tot_obs FROM OBSERVATION where
   sensor_id='$suid'";
$result3 = mysql_query($query3);
if (!$result3) {
  die('Invalid query: ' . mysql_error());
}
```

```
   while($row3 = @mysql_fetch_assoc($result3))
   {
        $nread = $row3['tot_obs'];
   }
   //echo $nread;
 $i=0;
  if($nread>10)
  {
   $query4 = "SELECT timestamp, derived_value from OBSERVATION where
       sensor_id='$suid' order by timestamp desc limit 10";
   $result4 = mysql_query($query4);
   if (!$result4) {
      die('Invalid query: ' . mysql_error());
   }
      while($row4 = @mysql_fetch_assoc($result4))
      {
            $Timeob[$i] = $row4['timestamp'];
      $val[$i] = $row4['derived_value'];
      $i=$i+1;
      }
  }else {
   $query5 = "SELECT timestamp, derived_value from OBSERVATION where
       sensor_id='$suid'";
   $result5 = mysql_query($query5);
   if (!$result5) {
      die('Invalid query: ' . mysql_error());
   }
      while($row5 = @mysql_fetch_assoc($result4))
      {
            $Timeob[$i] = $row5['timestamp'];
      $val[$i] = $row5['derived_value'];
      $i=$i+1;
      }
  }

header("Content-type: text/xml");

// Iterate through the rows, adding XML nodes for each

  for($x=0;$x < $i;$x=$x+1)
  {
      $l=0;
      $node = $dom->createElement("marker");
      $newnode = $parnode->appendChild($node);
        // ADD TO XML DOCUMENT NODE
      $newnode->setAttribute("TimeDate",$Timeob[$x]);
      $newnode->setAttribute("Observation",$val[$x]);
  }

echo $dom->saveXML();

?>
```

D.2.3 getdatedval.php

Called by DatedObservations.html, getdatedval.php gets observations for a sensor between a particular time period.

Listing D.6: getdatedval.php

```
<?php

$posted = $_GET;
$startdat = $posted["startdat"];
$enddat = $posted["enddat"];
$NodeName= $posted["node"];
$SensorName = $posted["sensor"];
//$NodeName="RSN3";
//$SensorName= "GV1";
//$startdat = "16-JUN-2010";
//$enddat = "20-JUN-2010";
$username = "root";
$password = "swl214ca";
$database = "swl2";
$server = "localhost";

//$st = (date("Y-m-d",strtotime("'$startdat'"))) . ' ' . '00:00:00';
//$et = (date("Y-m-d",strtotime("'$enddat'"))) . ' ' . '00:00:00';
//echo $st;
//start XML file, create parent node
$dom = new DOMDocument("1.0");
$node = $dom->createElement("markers");
$parnode = $dom->appendChild($node);

//Opens a connection to MySQL Server
$connection = mysql_connect(localhost, $username, $password);
if(!$connection) { die('Not Connected: ' . mysql_error());}

//set the active MySQL database
$db_selected = mysql_select_db($database, $connection);
if (!$db_selected) {
  die ('Can\'t use db : ' . mysql_error());
}

// Select node UUID from Sensor Node
$query1 = "SELECT UUID from SENSORNODE WHERE name='$NodeName'";
$result1 = mysql_query($query1);
if (!$result1) {
  die('Invalid query: ' . mysql_error());
}
while($row1 = @mysql_fetch_assoc($result1))
   {
       $nUID=$row1['UUID'];
   }
 //echo "$nUID";

//Select sensorUUID from the SENSOR_SN table.
$query2 = "SELECT UUID from SENSOR where name='$SensorName'";
```

```
$result2 = mysql_query($query2);
if (!$result2) {
  die('Invalid query: ' . mysql_error());
}
   while($row2 = @mysql_fetch_assoc($result2))
   {
        $suid = $row2['UUID'];
   }
  //echo $suid;

  $query3 = "SELECT timestamp, derived_value from OBSERVATION where
     sensor_id='$suid' and timestamp BETWEEN '$startdat' AND '
     $enddat'";
  $result3 = mysql_query($query3);
  if (!$result3) {
   die('Invalid query: ' . mysql_error());
  }
  while($row3 = @mysql_fetch_assoc($result3))
  {
      $Timeob[$i] = $row3['timestamp'];
   $val[$i] = $row3['derived_value'];
   $i=$i+1;
  }

header("Content-type: text/xml");

 //Iterate through the rows, adding XML nodes for each

  for($x=0;$x < $i;$x=$x+1)
  {
      $l=0;
      $node = $dom->createElement("marker");
      $newnode = $parnode->appendChild($node);
        // ADD TO XML DOCUMENT NODE
      $newnode->setAttribute("TimeDate",$Timeob[$x]);
      $newnode->setAttribute("Observation",$val[$x]);
  }

echo $dom->saveXML();

?>
```

D.3 Browser Code

D.3.1 Network2.html

This is the central web application 2 screen. Network2.html embeds Google Maps API into the html code. It shows the nodes in the network on the map and color codes each node according to the node's activity in the network.

Listing D.7: Network2.html

```
<!-- Sample HTML code, embedding java script to draw maps on the web
    browser -->
<script type="text/javascript" src="sarissa.js"></script>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www
   .w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-
   microsoft-com:vml" xml:lang="en">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf
       -8"/>
    <title>Google Maps</title>
    <h1><center>Wireless Sensor Networks</center></h1>
    <!--<hr width="100%" size="3" />-->
    <link rel="stylesheet" type="text/css" href="web.css" />

    <!--<style type="text/css">-->
    <!--</style>-->


    <script src="http://maps.google.com/maps?file=api&v=2&key=
       ABQIAAAAqbF2N1dHygcDEx7Epod25RSonZyZQTmJ4o3VnyCttsWNdhHvMhQhBMwCOhpoYyxl
       "
            type="text/javascript"></script>

    <script type="text/javascript" language="javascript">
    //<![CDATA[
    var iconCafe = new GIcon();

    iconCafe.image = 'http://labs.google.com/ridefinder/images/
       mm_20_brown.png';

    iconCafe.shadow = 'http://labs.google.com/ridefinder/images/
       mm_20_shadow.png';

    iconCafe.iconSize = new GSize(12, 20);

    iconCafe.shadowSize = new GSize(22, 20);

    iconCafe.iconAnchor = new GPoint(6, 20);

    iconCafe.infoWindowAnchor = new GPoint(5, 1);

    var iconBlue = new GIcon();
    iconBlue.image = 'http://labs.google.com/ridefinder/images/
       mm_20_blue.png';
    iconBlue.shadow = 'http://labs.google.com/ridefinder/images/
       mm_20_shadow.png';
    iconBlue.iconSize = new GSize(12, 20);
    iconBlue.shadowSize = new GSize(22, 20);
    iconBlue.iconAnchor = new GPoint(6, 20);
    iconBlue.infoWindowAnchor = new GPoint(5, 1);

    var iconRed = new GIcon();
```

```
 iconRed.image = 'http://labs.google.com/ridefinder/images/
    mm_20_red.png';
 iconRed.shadow = 'http://labs.google.com/ridefinder/images/
    mm_20_shadow.png';
 iconRed.iconSize = new GSize(12, 20);
 iconRed.shadowSize = new GSize(22, 20);
 iconRed.iconAnchor = new GPoint(6, 20);
 iconRed.infoWindowAnchor = new GPoint(5, 1);

 var iconGreen = new GIcon();
 iconGreen.image = 'http://labs.google.com/ridefinder/images/
    mm_20_green.png';
 iconGreen.shadow = 'http://labs.google.com/ridefinder/images/
    mm_20_shadow.png';
 iconGreen.iconSize = new GSize(12, 20);
 iconGreen.shadowSize = new GSize(22, 20);
 iconGreen.iconAnchor = new GPoint(6, 20);
 iconGreen.infoWindowAnchor = new GPoint(5, 1);

 var customIcons;
 customIcons = iconRed;
 <!-- customIcons["2"] = iconRed; -->
 <!-- customIcons["3"] = iconGreen; -->
 var u=0;
 var http_request = false;
function makeRequest(url, parameters) {

   http_request = false;
   if (window.XMLHttpRequest) { // Mozilla, Safari,...
      http_request = new XMLHttpRequest();
      if (http_request.overrideMimeType) {
         http_request.overrideMimeType('text/xml');
      }
   } else if (window.ActiveXObject) { // IE
      try {
         http_request = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (e) {
         try {
            http_request = new ActiveXObject("Microsoft.XMLHTTP")
                ;
         } catch (e) {}
      }
   }
   if (!http_request) {
      alert('Cannot create XMLHTTP instance');
      return false;
   }
   http_request.onreadystatechange = alertContents;
   http_request.open('GET', url + parameters, true);
   http_request.send(null);
}


function alertContents() {
```

```
if (http_request.readyState == 4) {
   if (http_request.status == 200) {
 if (GBrowserIsCompatible()) {
   var map = new GMap2(document.getElementById("map"));
   map.addControl(new GMapTypeControl());
<!--map.setUIToDefault();-->
map.enableScrollWheelZoom();
   map.setCenter(new GLatLng(45.949728,-66.642719), 19);
map.addControl(new GLargeMapControl());
     var xml = http_request.responseXML;
          var markers = xml.documentElement.
             getElementsByTagName("marker");
var numnode = markers.length;
var suid = new Array(numnode);

var sManu = new Array(numnode);

var sDescrip = new Array(numnode);

var sModel = new Array(numnode);

var sName = new Array(numnode);

var sUnit = new Array(numnode);

          for (var i = 0; i < markers.length; i++) {
      if ((markers[i].getAttribute("Latitude")) != "" && (
         markers[i].getAttribute("Longitude")) != "" ) {
                 var NodeID = markers[i].getAttribute("ID");
                 var NManu  = markers[i].getAttribute("
                    Manufacturer");
                 var NModel = markers[i].getAttribute("Model")
                    ;
           var NodeName = markers[i].getAttribute("Name");

        var point = new GLatLng(parseFloat(markers[i].
           getAttribute("Latitude")), parseFloat(markers[i].
           getAttribute("Longitude")));
           var NHtabsea = markers[i].getAttribute("habsea");
           var NHtabgd = markers[i].getAttribute("habgd");
           var Nstdat = markers[i].getAttribute("stdat");

        var stdatf2 = markers[i].getAttribute("sttimef2");

        var Nenddat = markers[i].getAttribute("enddat");
        var lt = parseFloat(markers[i].getAttribute("Latitude
           "));
        var ln = parseFloat(markers[i].getAttribute("Longitude
           "));

        var ns = markers[i].getAttribute("NumSensor");
        suid[i] = new Array(ns);
        sManu[i] = new Array(ns);
        sDescrip[i] = new Array(ns);
```

```
              sModel[i] = new Array(ns);
              sName[i] = new Array(ns);
                   sUnit[i] = new Array(ns);
              for(var j=0; j < ns; j++) {

                suid[i][j] = markers[i].getAttribute("sUUID"+j);

                <!--document.writeln("SUUID"+j+": "+suid[i][j]);-->
                sManu[i][j] = markers[i].getAttribute("sManufacturer
                   "+j);

                sDescrip[i][j] = markers[i].getAttribute("
                   sDescription"+j);

                sModel[i][j] = markers[i].getAttribute("sModel"+j);

                sName[i][j] = markers[i].getAttribute("sName"+j);

                sUnit[i][j] = markers[i].getAttribute("sUnit"+j);

              }

                       var marker = createMarker(point, NodeID,
                          NManu, NModel, NodeName, NHtabsea,
                          NHtabgd, Nstdat, Nenddat, lt, ln,
stdatf2, ns, suid, sManu,                          sDescrip, sModel
   , sName, sUnit, i);
                       map.addOverlay(marker);
            }
          }
      }
        }
  }
  }

  function do_xml() {

     makeRequest('Nodes.php', '?test=2');
  }

   function load() {
  do_xml();
  setInterval(do_xml, 30000);
   }


   function getIcon(stdatf2, curTime)

  {

  var hh = (curTime.valueOf()*0.001) - stdatf2;

  var diff = Math.floor(hh / (60 * 60*24));
```

```
<!--document.write(diff);-->

<!--var alpha = Math.max(alphamin,);-->

if(diff > 6) {
    return iconGrey;
}

if(diff > 4) {
    return iconBlue;
}

else if(diff>2) {
    return iconGreen;
}

else {
    return iconRed;
}

 }



 function geturl(suid, NodeName, name)

{
document.writeln(name);
var url = "http://localhost/xampp/Maps/GetObservation2.html?"+
   NodeName+","+name;
     return url;

}


 function createMarker(point, NodeID, NManu, NModel, NodeName,
    NHtabsea, NHtabgd, Nstdat, Nenddat, lt, ln, stdatf2, ns,
    suid, sManu, sDescrip, sModel, sName, sUnit,     i)     {

<!--document.writeln("sName1: "+sName[1][3]);-->
curTime = new Date();

var url1 = new Array(10);
<!-- document.write("I am here");-->

customIcon1 = getIcon(stdatf2, curTime);

<!--document.write(Date().toString());-->
      var marker = new GMarker(point, customIcon1);
      var html = '<div style="text-align:left">'+ "<b>Node:</b>"
         + NodeID +"<br/>Manufacturer: "+ NManu +"<br/>Model:"+
         NModel+"<br/>Name: "+
NodeName+"<br/>Location:         " + lt + " " + ln + "<br/>Ht
   above sea:" + NHtabsea+ "<br/>Ht above gd:"+NHtabgd+ "<br/><b
```

```
      >Start datetime:</b>"+ Nstdat +"<br/>Node end date:"+ Nenddat
                  +"<br/>Num Sensors Attached: "+ns+'</div>';


            for(var j=0; j< ns; j++)
            {
        var name = sName[i][j];

     <!--var url = geturl(suid[i][j], NodeName, name);-->
        var url = ("?"+sName[i][j]+'&'+NodeName);

        url1[u] = url;
        <!--if(url!="")-->


                html+= '<div style="text-align:left">'+"<br/>Sensor
                    UUID"+j+": "+suid[i][j]+", Manufacturer:"+sManu[
                    i][j]+", Description:"+sDescrip[i][j]+",Model:"+
                    sModel[i][j]+", Name:"+name+", Unit:"+sUnit[i][j
                    ]+"<a href=GetObservation2.html"+url+">Show
                    Observations</a>"+'</div>';
                         <!--onClick=getObs('NodeName','sName[i]')
                            ;-->
         u=u+1;
             }

      <!--html += '<div style="text-align:left">'+"<br/><a href=\""
         + "\">More Info</a>"+'</div>';-->

         GEvent.addListener(marker, 'click', function() {
         marker.openInfoWindowHtml(html);
      });

      return marker;
    }
    //]]>
  </script>

  </head>

  <body onload="load()" onunload="GUnload()">
    <center>
    <!-- <div id="map" style="width: 750px; height: 550px"></div>
       -->
    <div id="map_header" style="width: 400px; height: 20px"><p><b>
       IB214m09.cs.unb.ca, Wireless Sensor Network</b></p></div>
    <div id="map" style="width: 800px; height: 600px"></div>
    </center>
  </body>
</html>
```

D.3.2 GetObservation2.html

Shows the 10 latest observations of each node in the network in a tabular form.

Listing D.8: GetObservation2.html

```
<!doctype html public "-//w3c//dtd html 3.2//en">
<html>
<head>

  <title>Get Observation</title>
  <h1><center>Show Observations</center></h1>
  <meta name="keywords" content="">
  <meta name="Generator" content="HTMLpad">
  <meta http-equiv="Content-type" content="text/html; charset=UTF-8"
      />
  <script type="text/javascript" src="calendarDateInput.js">
  </script>
</head>

<body bgcolor="FAEBD7">

<SCRIPT LANGUAGE="javascript">
      var data = new Array(2);
      var input_data;
      var j=1;
      var query = window.location.search;
      // Skip the leading ?, which should always be there,
      // but be careful anyway
      if (query.substring(0, 1) == '?') {
            query = query.substring(1);
      }
      input_data = query.split('&');
      for (i = 0; (i < input_data.length); i++) {
            data[i] = (input_data[i]);
      }
      document.write("<b>Node: " + data[1] + "<br/>Sensor:"+data
         [0]+"</b>\n");
      do_xml();
      <!--setInterval(do_xml, 5000);-->

  function createhttpinstance()
  {
   http_request = false;
        if (window.XMLHttpRequest) { // Mozilla, Safari,...
            http_request = new XMLHttpRequest();
          if (http_request.overrideMimeType) {
             http_request.overrideMimeType('text/xml');
          }
          } else if (window.ActiveXObject) { // IE
          try {
               http_request = new ActiveXObject("Msxml2.XMLHTTP");
          } catch (e) {
             try {
                  http_request = new ActiveXObject("Microsoft.
                     XMLHTTP");
```

```
            } catch (e) {}
         }
      }
      if (!http_request) {
           alert('Cannot create XMLHTTP instance');
           return false;
      }
  }

  function do_xml(){
      var param = "?node="+data[1]+"&sensor="+data[0];
      makeRequest('Observation.php', param);
  }

  function makeRequest(url, parameters) {
      createhttpinstance();

      http_request.onreadystatechange = alertContents;
      http_request.open('GET', url + parameters, true);
      http_request.send(null);
   }


   function alertContents() {
      if (http_request.readyState == 4) {
         if (http_request.status == 200) {
           var xml = http_request.responseXML;
                var markers = xml.documentElement.
                   getElementsByTagName("marker");
      var tot = markers.length;
      var TimeStamp = new Array(tot);
      var val = new Array(tot);
                for (var i = 0; i < markers.length; i++) {
         TimeStamp[i] = markers[i].getAttribute("TimeDate");
                   val[i]  = markers[i].getAttribute("Observation");
         addRow(TimeStamp[i],val[i]);
         }
        }
   }
   }

</SCRIPT>

<br/><br/>

<center>
<TABLE id="dataTable" width="350px" border="1">
<CAPTION><b> Last 10 Observations</b></CAPTION>
   <TR>
      <TH>TimeDate</TH>
      <TH>Value</TH>
   </TR>

</TABLE>
```

```
<script type="text/javascript">

  function addRow(TimeStamp,val) {

   var table = document.getElementById('dataTable');
   var rowCount = table.rows.length;
   var row = table.insertRow(rowCount);
   tabBody=document.getElementsByTagName("tbody").item(0);
   root=document.createElement("tr");
        c1 = document.createElement("td");
        c2 =document.createElement("td");
        textnode1=document.createTextNode(TimeStamp);
        textnode2=document.createTextNode(val);
        c1.appendChild(textnode1);
        c2.appendChild(textnode2);
        root.appendChild(c1);
        root.appendChild(c2);
        tabBody.appendChild(root);
  }

</script>
<br/>

 <script type="text/javascript">

  function getdatedobs(start_dat, end_dat)
  {
     url = 'DatedObservations.html?'+start_dat+"&"+end_dat+"&"+data
        [1]+"&"+data[0];
     window.open(url,'mywindow','width=800,height=1000,toolbar=yes,
        location=yes,directories=yes,status=yes,menubar=yes,
        scrollbars=yes,copyhistory=yes,resizable=yes');
   }
</script>

<script type="text/javascript">
function exportobs()
   {
   url = 'last10obs.html?'+data[1]+"&"+data[0];
   window.open(url,'mywindow','width=500,height=300,toolbar=yes,
      location=yes,directories=yes,status=yes,menubar=yes,
      scrollbars=yes,copyhistory=yes,resizable=yes');
   }
</script>

  <input type="button" onClick="exportobs()" value="Export
     Observation">
<br/><br/>

  <form>
  <label FOR="orderdatei">Choose Start Date:</label>
  <script>DateInput('orderdatei', true, 'YYYY-MM-DD')</script>
```

```
  <label FOR="orderdate_e">Choose End Date:</label>
  <script>DateInput('orderdate_e', true, 'YYYY-MM-DD')</script>
<br/>
  <input type="button" onClick="getdatedobs(this.form.orderdatei.
     value, this.form.orderdate_e.value)" value="Get Observations">
  </form>
</center>
<a style="color:#617f10" href="http://ib214m09.cs.unb.ca/html/sos/
   server.php?request=GetCapabilities&service=SOS"><b>SOS
   GetCapabilities metadata</b></a>
<br/>
<a style="color:#617f10" href="http://www.w3schools.com"><b>SOS
   DescribeSensor metadata</b></a>
<br/>
<a style="color:#617f10" href="http://www.w3schools.com"><b>SOS
   GetObservation metadata</b></a>



</body>
</html>
```

D.3.3 last10obs.html

Shows the last 10 observations of a sensor in a Comma Separated Version (CSV) format.

Listing D.9: last10obs.html

```
<!doctype html public "-//w3c//dtd html 3.2//en">
<html>
<head>

  <title>Last 10 Observations</title>
  <h1><center>Show Observations</center></h1>
  <meta name="keywords" content="">
  <meta name="Generator" content="HTMLpad">
  <meta http-equiv="Content-type" content="text/html; charset=UTF-8"
      />

</head>

<body bgcolor="aqua">

<SCRIPT LANGUAGE="javascript">
   var data = new Array(2);
   var input_data;
   var j=1;

   var query = window.location.search;
   // Skip the leading ?, which should always be there,
   // but be careful anyway
   if (query.substring(0, 1) == '?') {
      query = query.substring(1);
```

```
 }
 input_data = query.split('&');
 for (i = 0; (i < input_data.length); i++) {
    data[i] = (input_data[i]);
 }
 document.write("<b>Node: " + data[0] +"<br/>Sensor: "+data[1]);
 do_xml();

function createhttpinstance()
{
 http_request = false;
      if (window.XMLHttpRequest) { // Mozilla, Safari,...
          http_request = new XMLHttpRequest();
        if (http_request.overrideMimeType) {
           http_request.overrideMimeType('text/xml');
        }
        } else if (window.ActiveXObject) { // IE
        try {
             http_request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
           try {
                http_request = new ActiveXObject("Microsoft.
                   XMLHTTP");
           } catch (e) {}
        }
     }
     if (!http_request) {
          alert('Cannot create XMLHTTP instance');
          return false;
     }
}

function do_xml(){
 var parameters = "?node="+data[0]+"&sensor="+data[1];
 url = 'Observation.php';
 makeRequest(url, parameters);
}

function makeRequest(url, parameters) {
    createhttpinstance();

      http_request.onreadystatechange = alertContents;
    http_request.open('GET', url + parameters, true);
    http_request.send(null);

 }

 function alertContents() {
    if (http_request.readyState == 4) {
       if (http_request.status == 200) {
         var xml = http_request.responseXML;
              var markers = xml.documentElement.
                 getElementsByTagName("marker");
    var tot = markers.length;
```

```
      TimeStamp = new Array(tot);
      val = new Array(tot);
               for (l=0; l < markers.length; l=l+1) {
         TimeStamp[l] = markers[l].getAttribute("TimeDate");
                  val[l]  = markers[l].getAttribute("Observation");
         document.write("TimeDate: "+TimeStamp[l]+", value: "+val[l
            ]);
         document.write("</br>");
         }
        }
   }
   }
</SCRIPT>

<br/><br/>

<center>
 </br><b>NO READINGS CURRENTLY FROM THE SENSORS</b>
</center>

</body>
</html>
```

D.3.4 DatedObservations.html

Shows the observations of a sensor in between a particular time period in CSV format.

Listing D.10: DatedObservations.html

```
<!doctype html public "-//w3c//dtd html 3.2//en">
<html>
<head>

  <title>Observations</title>
  <h1><center>Show Observations</center></h1>
  <meta name="keywords" content="">
  <meta name="Generator" content="HTMLpad">
  <meta http-equiv="Content-type" content="text/html; charset=UTF-8"
      />

</head>

<body bgcolor="aqua">

<SCRIPT LANGUAGE="javascript">
   var data = new Array(2);
   var input_data;
   var j=1;

   var query = window.location.search;
   // Skip the leading ?, which should always be there,
   // but be careful anyway
   if (query.substring(0, 1) == '?') {
```

```
    query = query.substring(1);
 }
 input_data = query.split('&');
 for (i = 0; (i < input_data.length); i++) {
    data[i] = (input_data[i]);
 }
 document.write("<b>Node: " + data[2] +"<br/>Sensor: "+data[3]+"<
    br/>Start Date: "+data[0]+"<br/>End Date: "+data[1]+"</b>");
 do_xml();

function createhttpinstance()
{
 http_request = false;
      if (window.XMLHttpRequest) { // Mozilla, Safari,...
          http_request = new XMLHttpRequest();
        if (http_request.overrideMimeType) {
           http_request.overrideMimeType('text/xml');
        }
        } else if (window.ActiveXObject) { // IE
        try {
             http_request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
           try {
                http_request = new ActiveXObject("Microsoft.
                   XMLHTTP");
           } catch (e) {}
        }
     }
     if (!http_request) {
          alert('Cannot create XMLHTTP instance');
          return false;
     }
}

function do_xml(){
 var parameters = "?startdat="+data[0]+"&enddat="+data[1]+"&node
    ="+data[2]+"&sensor="+data[3];
 url = 'getdatedval.php';
 makeRequest(url, parameters);
}

function makeRequest(url, parameters) {
    createhttpinstance();

      http_request.onreadystatechange = alertContents;
    http_request.open('GET', url + parameters, true);
    http_request.send(null);

 }

 function alertContents() {
    if (http_request.readyState == 4) {
       if (http_request.status == 200) {
         var xml = http_request.responseXML;
```

```
                var markers = xml.documentElement.
                   getElementsByTagName("marker");
      var tot = markers.length;
      TimeStamp = new Array(tot);
      val = new Array(tot);
               for (l=0; l < markers.length; l=l+1) {
         TimeStamp[l] = markers[l].getAttribute("TimeDate");
                  val[l]  = markers[l].getAttribute("Observation");
         document.write("TimeDate: "+TimeStamp[l]+", value: "+val[l
            ]);
         document.write("</br>");
         }
        }
   }
   }
</SCRIPT>

<br/><br/>

<center>
 </br><b>NO OBSERVATION DURING THE SPECIFIED TIME INTERVAL</b>
</center>

</body>
</html>
```

Appendix E

SOS code

E.1 SOS Server - server.php

All SOS code is in server.php program.
The following extract of the program shows definition of all the constant configurations values in the program.

Listing E.1: SOS - Defining the constant values

```
ini_set("memory_limit","256M");
ini_set("max_execution_time","600"); // 10 minutes

define('DBID','sos');
define('NID_URN','swl');  // Per RFC 2141, may contain only upper/
   lower case letters, numbers and '-' (1-32 characters).
define('NID_URN_PATT',preg_quote(NID_URN)); // Used in regular
   expression pattern matching.
define('ORG','UNB FCS Test');
define('ORG_ACRONYM','MyOrg');
define('ORG_URN','myorg');  // Per RFC 2141, may contain only upper/
   lower case letters, numbers and any of ("%" | "(" | ")" | "+" |
   "," | "-" | "." | ":" | "=" | "@" | ";" | "$" | "_" | "!" | "*"
   | "'")
define('ORG_URN_PATT',preg_quote(ORG_URN)); // Used in regular
   expression pattern matching.
define('ORG_URL','http://ib214m09.cs.unb.ca/html/sos/server.php?
   service=SOS&request=GetCapabilities');
define('TITLE',ORG." SOS");
define('SOS_URL',$_SERVER["SCRIPT_URI"]);
define('ABSTRACTTEXT',"University of New Brunswick (UNB), Faculty of
    Computer Science (FCS)");
define('KEYWORDS','Room Temperature, Voltage, Solar');
define('INDNAME','Gunita Saini');
define('POSNAME','Student');
define('PHONE_VOICE','506-453-4566');
define('PHONE_FAX','(506) 453-3566');
define('ADDR_STREET','540 Windsor Street');
define('ADDR_CITY','Fredericton');
define('ADDR_STATE','NB');
```

```
define('ADDR_ZIP','E3B5A3');
define('ADDR_COUNTRY','Canada');
define('ADDR_EMAIL','www.unb.ca');
define('CUTOFF',gmdate('Y-m-d\TH:i:s\Z',strtotime('now -6 hours')));
    // six hours ago.
define('SRSNAME','urn:ogc:def:crs:epsg::4326');

// In supported_versions array, always define most recent version as
    first element.
$supported_versions = array('1.0.0');
$supported_sections = array('ServiceIdentification','ServiceProvider
   ','OperationsMetadata','Contents');
$getobservation_formats = array('text/xml;schema=\"swl/0.6.1\"','
   application/swl+xml;version=0.6.1','text/csv','text/tab-
   separated-values','application/vnd.google-earth.kml+xml');
$describesensor_formats = array('text/xml;subtype=\"sensorML
   /1.0.0\"');
// Old property name => new MMI property name
$supported_ob_props = array(
   'airtemperature'   => 'Air Temperature',
   'voltage'      => 'Battery Voltage',
   'waterlevel'      => 'Water Level'
);

$params = array();
$exceptions = array();
$dom = null;

require_once('dbinfo.php');
```

Whenever an SOS request is made, the function getParams() is called in server.php. This function parses the request, checks if it is valid. Also, checks if it is a DescribeSensor, GetCapability or GetObservation request and saves the values provided by the user in the request.

Listing E.2: SOS - GetParams()

```
function getParams() {
   global $exceptions,$supported_versions,$supported_sections,
      $supported_ob_props,$getobservation_formats,
      $describesensor_formats,$dom;
   $params = array('service' => null, 'request' => null, 'eventtime'
       => null, 'responseformat' => null);

   function checkBbox($llon,$llat,$ulon,$ulat) {
      // Assumes numeric input since this function is not called
         unless the arguments pass the regex test in getParams
      $err = '';
      if ($llon < -360 || $llon > 360) { $err .= ' Lower Longitude
         is out of range (-360,360).'; }
      if ($llat < -90 || $llat > 90) { $err .= ' Lower Latitude is
         out of range (-90,90).'; }
      if ($ulon < -360 || $ulon > 360) { $err .= ' Upper Longitude
         is out of range (-360,360).'; }
      if ($ulat < -90 || $ulat > 90) { $err .= ' Upper Latitude is
```

```
        out of range (-90,90).'; }
    if ($err) { return $err; }
    if (($ulon-$llon) > 360) { $err .= ' Longitude range must not
        exceed 360 degrees.'; }
    if ($llon >= $ulon) { $err .= ' Lower Longitude must be less
        than Upper Longitude.'; }
    if ($llat >= $ulat) { $err .= ' Lower Latitude must be less
        than Upper Latitude.'; }
    return $err;
}

if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    $_GET = array_change_key_case($_GET, CASE_LOWER);
    if (array_key_exists('service',$_GET)) {
        if ($_GET['service'] == 'SOS') {
            $params['service'] = $_GET['service'];
        } else {
            $exceptions[] = array('InvalidParameterValue','service',
                $_GET['service']);
        }
    } else {
        $exceptions[] = array('MissingParameterValue','service',
            null);
    }
    if (array_key_exists('request',$_GET)) {
        if (preg_match('/^GetCapabilities|DescribeSensor|
            GetObservation$/',$_GET['request']) == 1) {
            $params['request'] = $_GET['request'];
        } else {
            $exceptions[] = array('OperationNotSupported',
                preg_replace('/[^a-z0-9\_]/i','',$_GET['request']),
                null);
        }
    } else {
        $exceptions[] = array('MissingParameterValue','request',
            null);
    }
    if ($params['request'] == 'GetCapabilities') {
        // parse params for GetCapabilities.
        $negotiatedversion = null;
        if (array_key_exists('acceptversions',$_GET)) {
            if (preg_match('/^\d{1,2}\.\d{1,2}\.\d{1,2}(\,\d{1,2}\.\
                d{1,2}\.\d{1,2})*$/',$_GET['acceptversions']) == 1)
                {
                foreach (split(',',$_GET['acceptversions']) as
                    $version) {
                    if (in_array($version,$supported_versions)) {
                        $negotiatedversion = $version;
                        break;
                    }
                }
                if ($negotiatedversion === null) {
                    $exceptions[] = array('VersionNegotiationFailed',
                        null,null);
```

```
            }
         } else {
            $exceptions[] = array('InvalidParameterValue','
               acceptversions',$_GET['acceptversions']);
         }
      } else {
         $negotiatedversion = $supported_versions[0];
      }
      $params['version'] = $negotiatedversion;
      $params['sections'] = array();
      if (array_key_exists('sections',$_GET)) {
         if (preg_match('/^[a-z]+(\,[a-z]+)*$/i',$_GET['sections
            ']) == 1) {
            foreach (split(',',$_GET['sections']) as $section) {
               if ($section == 'All') {
                  $params['sections'] = $supported_sections;
                  break;
               }
               if (in_array($section,$supported_sections)) {
                  $params['sections'][] = $section;
                  continue;
               }
               $exceptions[] = array('InvalidParameterValue','
                  sections',$_GET['sections']);
               break;
            }
         } else {
            $exceptions[] = array('InvalidParameterValue','
               sections',$_GET['sections']);
         }
      } else {
         $params['sections'] = $supported_sections;
      }
   }
   if ($params['request'] == 'DescribeSensor') {
      // parse params for DescribeSensor.
      if (array_key_exists('outputformat',$_GET)) {
         if (in_array($_GET['outputformat'],
            $describesensor_formats)) {
            $params['outputformat'] = $_GET['outputformat'];
         } else {
            $exceptions[] = array('InvalidParameterValue','
               outputFormat',$_GET['outputformat']);
            $params['outputformat'] = null;
         }
      } else {
         $exceptions[] = array('MissingParameterValue','
            outputFormat',null);
         $params['outputformat'] = null;
      }
      if (array_key_exists('procedure',$_GET)) {
         //    urn:swl:sensor:wmo:21415::tsunameter0:
         $matches = array();
         if (preg_match('/^urn:'.NID_URN_PATT.':sensor:wmo:([a-z0
```

```
            -9]+)::([A-Za-z0-9\-\.\_ ]+):$/',$_GET['procedure'],
            $matches)) {
            $params['procedure'] = array('urn' => $matches[0], '
               stn' => $matches[1], 'sensor' => $matches[2]);
         } elseif (preg_match('/^urn:'.NID_URN_PATT.':station:wmo
            :([a-z0-9]+):$/',$_GET['procedure'],$matches)) {
            $params['procedure'] = array('urn' => $matches[0], '
               stn' => $matches[1], 'sensor' => null);
         } else {
            $exceptions[] = array('InvalidParameterValue','
               procedure',$_GET['procedure']);
            $params['procedure'] = null;
         }
      } else {
         $exceptions[] = array('MissingParameterValue','procedure
            ',null);
         $params['procedure'] = null;
      }
   }
   if ($params['request'] == 'GetObservation') {
      // parse params for GetObservation.
      if (array_key_exists('responseformat',$_GET)) {
         if (in_array($_GET['responseformat'],
            $getobservation_formats)) {
            $params['responseformat'] = $_GET['responseformat'];
         } else {
            $exceptions[] = array('InvalidParameterValue','
               responseFormat',$_GET['responseformat']);
            $params['responseformat'] = null;
         }
      } else {
         $exceptions[] = array('MissingParameterValue','
            responseFormat',null);
            $params['responseformat'] = null;
      }
      if (array_key_exists('offering',$_GET)) {
         $matches = array();
         if (preg_match('/^urn:'.NID_URN_PATT.':(network):'.
            ORG_URN_PATT.':([a-zA-Z0-9]+)$/',strtolower($_GET['
            offering']),$matches) == 1) {
            if ($matches[2]  != 'all') {
               $exceptions[] = array('InvalidParameterValue','
                  offering',$_GET['offering']);
               $params['offering'] = null;
            } else {
               $params['offering'] = array('type' => $matches[1],
                   'id' => $matches[2]);
               if (array_key_exists('procedure',$_GET)) {
                  $matches = array();
                  if (preg_match('/^urn:'.NID_URN_PATT.':station:
                     wmo:([a-zA-Z0-9]+):$/',$_GET['procedure'],
                     $matches) == 1) {
                     $params['procedure'] = $matches[1];
                  } else {
```

```
                $exceptions[] = array('InvalidParameterValue
                    ','procedure',$_GET['procedure']);
            }
        }
    }
} elseif (preg_match('/^urn:'.NID_URN_PATT.':(station):
    wmo:([a-zA-Z0-9]+):$/',strtolower($_GET['offering'])
    ,$matches) == 1) {
    $params['offering'] = array('type' => $matches[1], '
        id' => $matches[2]);
} else {
    $exceptions[] = array('InvalidParameterValue','
        offering',$_GET['offering']);
}
} else {
    $exceptions[] = array('MissingParameterValue','offering
        ',null);
}
if (array_key_exists('featureofinterest',$_GET)) {
    $matches = array();
    if (preg_match('/^BBOX\:(-?\d+(\.\d+)?),(-?\d+(\.\d+)?)
        ,(-?\d+(\.\d+)?),(-?\d+(\.\d+)?)$/i',$_GET['
        featureofinterest'],$matches) == 1) {
        $err = checkBbox($matches[1],$matches[3],$matches[5],
            $matches[7]);
        if ($err) {
            $exceptions[] = array('InvalidParameterValue','
                featureofinterest',$err.' - ['.$_GET['
                featureofinterest'].']');
        } else {
            $params['featureofinterest'] = array('urn' =>
                $matches[0], 'name' => 'BBOX', 'llon'=>
                floatval($matches[1]),'llat'=>floatval(
                $matches[3]),'ulon'=>floatval($matches[5]),'
                ulat'=>floatval($matches[7]));
        }
    } elseif (preg_match('/^urn:'.NID_URN_PATT.':(event:
        tsunami::\w+:\w+)$/i',$_GET['featureofinterest'],
        $matches) == 1) {
        $params['featureofinterest'] = array('urn' =>
            $matches[0], 'name' => $matches[1]);
    } else {
        $exceptions[] = array('InvalidParameterValue','
            featureofinterest',$_GET['featureofinterest']);
    }
}
if (array_key_exists('eventtime',$_GET)) {
    if (substr_count($_GET['eventtime'],'/') > 1) {
        $exceptions[] = array('InvalidParameterValue','
            eventTime',$_GET['eventtime']);
    }
    foreach (split('/',$_GET['eventtime']) as $dt) {
        $ts = readISO8601($dt);
        if ($ts) {
```

```
                $params['eventtime'][] = $ts;
            } else {
                $exceptions[] = array('InvalidParameterValue','
                    eventTime',$_GET['eventtime']);
                $params['eventtime'][] = null;
                break;
            }
        }
        if ($params['eventtime'] === null) {
            $exceptions[] = array('InvalidParameterValue','
                eventTime',$_GET['eventtime']);
//          } else {
//              if (count($params['eventtime']) == 2) {
//                  sort($params['eventtime'],SORT_NUMERIC);
//                  if (floor(($params['eventtime'][1]-$params['
    eventtime'][0])/86400) > 30) {
//                      $exceptions[] = array('InvalidParameterValue','
    eventTime','No more than thirty days of data can be requested.')
    ;
//                  }
//              }
        }
    }
    if (array_key_exists('observedproperty',$_GET)) {
        $op = $_GET['observedproperty'];
        $matches = array();
        if (preg_match('|^http://www.csc.noaa.gov/swl/schema/swl
            -DIF/swl/0.6.1/dictionaries/phenomenaDictionary.xml
            #([A-Za-z_]+)$|',$op,$matches) === 1) {
            $op = strtolower($matches[1]);
            if (array_key_exists($op,$supported_ob_props)) {
                $params['observedproperty'] = $op;
            } else {
                $exceptions[] = array('InvalidParameterValue','
                    observedProperty',$_GET['observedproperty']);
            }
        } elseif (preg_match('|^http://mmisw.org/ont/cf/
            parameter/([A-Za-z_]+)$|',$op,$matches) === 1) {
            $op = strtolower($matches[1]);
            $old_op = array_search($op,$supported_ob_props);
            if ($old_op !== false) {
                $params['observedproperty'] = $old_op;
            } else {
                $exceptions[] = array('InvalidParameterValue','
                    observedProperty',$_GET['observedproperty']);
            }
        } elseif (preg_match('/^[a-z_]+$/i',$op) == 1) {
            $op = strtolower($op);
            if (array_key_exists($op,$supported_ob_props)) {
                $params['observedproperty'] = $op;
            } elseif (($old_op = array_search($op,
                $supported_ob_props)) !== false) {
                $params['observedproperty'] = $old_op;
            } else {
```

```
                $exceptions[] = array('InvalidParameterValue','
                    observedProperty',$_GET['observedproperty']);
            }
        } else {
            $exceptions[] = array('InvalidParameterValue','
                observedProperty',$_GET['observedproperty']);
        }
        unset($op);
    } else {
        $exceptions[] = array('MissingParameterValue','
            observedProperty',null);
    }
}
} elseif ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $xmldata = urldecode(file_get_contents('php://input'));
    //if user set up a form for posting, get rid of leading
        variable name and equal sign.
    $xmldata = preg_replace('/^([a-zA-Z0-9_]+=)?/','',$xmldata);
    $dom = new DOMDocument();
    $success = @ $dom->loadXML($xmldata);
    if (! $success) {
        $exceptions[] = array('MissingParameterValue','service',
            null);
        $exceptions[] = array('MissingParameterValue','request',
            null);
        return null;
    }
    $xpath = new DOMXPath($dom);
    $xpath->registerNamespace('default','http://www.opengis.net/
        ows/1.1');
    $xpath->registerNamespace('ows','http://www.opengis.net/ows
        /1.1');
    $xpath->registerNamespace('xsi','http://www.w3.org/2001/
        XMLSchema-instance');
    $xpath->registerNamespace('ogc', 'http://www.opengis.net/ogc')
        ;
    $xpath->registerNamespace('gml', 'http://www.opengis.net/gml')
        ;
    $params['request'] = $dom->documentElement->localName;
    if (preg_match('/^GetCapabilities|DescribeSensor|
        GetObservation$/',$params['request']) != 1) {
        $exceptions[] = array('OperationNotSupported',preg_replace
            ('/[^a-z0-9\_]/i','',$_GET['request']),null);
        $params['request'] = null;
    } else {
        if ($dom->documentElement->getAttribute('service') != 'SOS
            ') {
            $exceptions[] = array('InvalidParameterValue','service',
                $dom->documentElement->getAttribute('service'));
        } else {
            $params['service'] = $dom->documentElement->getAttribute
                ('service');
        }
        if ($params['request'] == 'GetCapabilities') {
```

```
            $negotiatedversion = null;
            $noversion = true;
            $nl = $xpath->query('//default:AcceptVersions/default:
               Version');
            foreach ($nl as $version) {
               $noversion = false;
               if (in_array(trim($version->nodeValue),
                  $supported_versions)) {
                  $negotiatedversion = trim($version->nodeValue);
                  break;
               }
            }
            if ($noversion) { $negotiatedversion =
               $supported_versions[0]; }
            if ($negotiatedversion === null) {
               $exceptions[] = array('VersionNegotiationFailed',null
                  ,null);
            }
            $params['version'] = $negotiatedversion;
            $params['sections'] = array();
            $nl = $xpath->query('//default:Sections/default:Section
               ');
            foreach ($nl as $section) {
               if (trim($section->nodeValue) == 'All') {
                  $params['sections'] = $supported_sections;
                  break;
               }
               if (in_array(trim($section->nodeValue),
                  $supported_sections)) {
                  $params['sections'][] = trim($section->nodeValue);
                  continue;
               }
               $exceptions[] = array('InvalidParameterValue','
                  sections',$section);
            }
         }
         if ($params['request'] == 'DescribeSensor') {
            // parse params for DescribeSensor.
            $xpath->registerNamespace('default','http://www.opengis.
               net/sos/1.0');
            if ($dom->documentElement->getAttribute('outputFormat')
               != null) {
               if (in_array($dom->documentElement->getAttribute('
                  outputFormat'),$describesensor_formats)) {
                  $params['outputformat'] = $dom->documentElement->
                     getAttribute('outputFormat');
               } else {
                  $exceptions[] = array('InvalidParameterValue','
                     outputFormat',$dom->documentElement->
                     getAttribute('outputFormat'));
                  $params['outputformat'] = null;
               }
            } else {
               $exceptions[] = array('MissingParameterValue','
```

```
            outputFormat',null);
        $params['outputformat'] = null;
    }
    $nl = $xpath->query('//default:procedure');
    if ($nl->item(0)) {
        //    urn:swl:sensor:wmo:21415::tsunameter0:
        $matches = array();
        if (preg_match('/^urn:'.NID_URN_PATT.':sensor:wmo:([a
            -z0-9]+)::([A-Za-z0-9\-\.\_ ]+):$/',trim($nl->
            item(0)->nodeValue),$matches)) {
            $params['procedure'] = array('urn' => $matches[0],
                'stn' => $matches[1], 'sensor' => $matches
                [2]);
        } elseif (preg_match('/^urn:'.NID_URN_PATT.':station:
            wmo:([a-z0-9]+):$/',trim($nl->item(0)->nodeValue)
            ,$matches)) {
            $params['procedure'] = array('urn' => $matches[0],
                'stn' => $matches[1], 'sensor' => null);
        } else {
            $exceptions[] = array('InvalidParameterValue','
                procedure',trim($nl->item(0)->nodeValue));
            $params['procedure'] = null;
        }
    } else {
        $exceptions[] = array('MissingParameterValue','
            procedure',null);
        $params['procedure'] = null;
    }
}
if ($params['request'] == 'GetObservation') {
    // parse params for GetObservation.
    $nl = $xpath->query('//sos:responseFormat');
    if ($nl->item(0)) {
        if (in_array(trim($nl->item(0)->nodeValue),
            $getobservation_formats)) {
            $params['responseformat'] = trim($nl->item(0)->
                nodeValue);
        } else {
            $exceptions[] = array('InvalidParameterValue','
                responseFormat',trim($nl->item(0)->nodeValue))
                ;
            $params['responseformat'] = null;
        }
    } else {
        $exceptions[] = array('MissingParameterValue','
            responseFormat',null);
        $params['responseformat'] = null;
    }
    $nl = $xpath->query('//sos:offering');
    if ($nl->item(0)) {
        $params['offering'] = trim($nl->item(0)->nodeValue);
        $matches = array();
        if (preg_match('/^urn:'.NID_URN_PATT.':(network):'.
            ORG_URN_PATT.':([a-zA-Z0-9]+)$/',$params['
```

```
            offering'],$matches) == 1) {
            if ($matches[2] != 'all') {
               $exceptions[] = array('InvalidParameterValue','
                  offering',$params['offering']);
               $params['offering'] = null;
            } else {
               $params['offering'] = array('type' => $matches
                  [1], 'id' => $matches[2]);
               $nl = $xpath->query('//sos:procedure');
               if ($nl->item(0)) {
                  $params['procedure'] = trim($nl->item(0)->
                     nodeValue);
                  $matches = array();
                  if (preg_match('/^urn:'.NID_URN_PATT.':
                     station:wmo:([a-zA-Z0-9]+):$/',$params['
                     procedure'],$matches) != 1) {
                     $exceptions[] = array('
                        InvalidParameterValue','procedure',
                        trim($nl->item(0)->nodeValue));
                     $params['procedure'] = null;
                  }
               }
            }
         } elseif (preg_match('/^urn:'.NID_URN_PATT.':(station
            ):wmo:([a-zA-Z0-9]+):$/',$params['offering'],
            $matches) == 1) {
            $params['offering'] = array('type' => $matches[1],
                'id' => $matches[2]);
         } else {
            $exceptions[] = array('InvalidParameterValue','
               offering',$params['offering']);
            $params['offering'] = null;
         }
      } else {
         $exceptions[] = array('MissingParameterValue','
            offering',null);
      }
      $nl = $xpath->query('//sos:featureOfInterest');  //
         Optional feature of interest
      if ($nl->item(0)) {
         //if ($nl->item(0)->nodeValue !== null) {
         if ($nl->item(0)->childNodes->length <= 1) {
            // check for named feature of interest.
            $matches = array();
            if (preg_match('/^urn:'.NID_URN_PATT.':(event:
               tsunami::\w+:\w+)$/i',trim($nl->item(0)->
               nodeValue),$matches) == 1) {
               $params['featureofinterest'] = array('urn' =>
                  $matches[0], 'name' => $matches[1]);
            } else {
               $exceptions[] = array('InvalidParameterValue','
                  featureofinterest',trim($nl->item(0)->
                  nodeValue));
            }
```

```
            } else {
               // Or Bounding Box feature of interest.
               $nl = $xpath->query('//sos:featureOfInterest/ogc:
                  BBOX');
               if ($nl->item(0)) {
                  $nl = $xpath->query('//sos:featureOfInterest/
                     ogc:BBOX/gml:Envelope');
                  if ($nl->item(0)) {
                     $llon = $llat = $ulon = $ulat = null;
                     foreach ($nl->item(0)->childNodes as
                        $cornernode) {
                        if ($cornernode->nodeType ==
                           XML_ELEMENT_NODE) {
                           if ($cornernode->nodeName == 'gml:
                              lowerCorner' &&
                              preg_match('/^\s*(-?\d+(\.\d+)?)\s
                                 +(-?\d+(\.\d+)?)\s*$/',trim(
                                 $cornernode->nodeValue),
                                 $matches) == 1) {
                                 $llat = $matches[1];
                                 $llon = $matches[3];
                           }
                           if ($cornernode->nodeName == 'gml:
                              upperCorner' &&
                              preg_match('/^\s*(-?\d+(\.\d+)?)\s
                                 +(-?\d+(\.\d+)?)\s*$/',trim(
                                 $cornernode->nodeValue),
                                 $matches) == 1) {
                                 $ulat = $matches[1];
                                 $ulon = $matches[3];
                           }
                        }
                     }
                     if ($llon !== null && $llat !== null &&
                        $ulon !== null && $ulat !== null) {
                        $err = checkBbox($llon,$llat,$ulon,$ulat)
                           ;
                        if ($err) {
                           $exceptions[] = array('
                              InvalidParameterValue','
                              featureofinterest',$err);
                        } else {
                           $params['featureofinterest'] = array('
                              name' => 'BBOX', 'llon'=>floatval(
                              $llon),'llat'=>floatval($llat),'
                              ulon'=>floatval($ulon),'ulat'=>
                              floatval($ulat));
                        }
                     } else {
                        $exceptions[] = array('
                           InvalidParameterValue','
                           featureofinterest','Bounding Box
                           specified but could not decode lat/
                           long.');
```

```
                        }
                    } else {
                        $exceptions[] = array('InvalidParameterValue
                            ','featureofinterest','Malformed BBOX');
                    }
                }
            }
            $nl = $xpath->query('//sos:eventTime');
            if ($nl->item(0)) {
                $nl = $xpath->query('//sos:eventTime/ogc:TM_During/
                    gml:TimePeriod');
                if ($nl->item(0)) {
                    $ts = null;
                    $nl = $xpath->query('//sos:eventTime/ogc:TM_During
                        /gml:TimePeriod/gml:beginPosition');
                    if ($nl->item(0)) {
                        $ts = readISO8601(trim($nl->item(0)->nodeValue)
                            );
                        if ($ts) {
                            $params['eventtime'][] = $ts;
                        } else {
                            $exceptions[] = array('InvalidParameterValue
                                ','eventTime',trim($nl->item(0)->
                                nodeValue));
                            $params['eventtime'][] = null;
                        }
                    }
                    $nl = $xpath->query('//sos:eventTime/ogc:TM_During
                        /gml:TimePeriod/gml:endPosition');
                    if ($nl->item(0)) {
                        $ts = readISO8601(trim($nl->item(0)->nodeValue)
                            );
                        if ($ts) {
                            $params['eventtime'][] = $ts;
                        } else {
                            $exceptions[] = array('InvalidParameterValue
                                ','eventTime',trim($nl->item(0)->
                                nodeValue));
                            $params['eventtime'][] = null;
                        }
                    }
//                  if (count($params['eventtime']) == 2) {
//                      sort($params['eventtime'],SORT_NUMERIC);
//                      if (floor(($params['eventtime'][1]-$params['
    eventtime'][0])/86400) > 30) {
//                          $exceptions[] = array('InvalidParameterValue
    ','eventTime','No more than thirty days of data can be requested
    .');
//                      }
//                  }
                } else {
                    $nl = $xpath->query('//sos:eventTime/ogc:TM_Equals
                        /gml:TimeInstant/gml:timePosition');
```

```
            if ($nl->item(0)) {
               $params['eventtime'][0] = readISO8601(trim($nl
                  ->item(0)->nodeValue));
               if ($params['eventtime'][0] === null) {
                  $exceptions[] = array('InvalidParameterValue
                     ','eventTime',trim($nl->item(0)->
                     nodeValue));
               }
            }
         }
         if (!array_key_exists('eventtime',$params)) {
            $exceptions[] = array('InvalidParameterValue','
               eventTime',null);
         }
      }
      $nl = $xpath->query('//sos:observedProperty');
      if ($nl->item(0)) {
         $op = trim($nl->item(0)->nodeValue);
         $matches = array();
         if (preg_match('|^http://www.csc.noaa.gov/swl/schema/
            swl-DIF/swl/0.6.1/dictionaries/
            phenomenaDictionary.xml#([A-Za-z_]+)$|',$op,
            $matches) === 1) {
            $op = strtolower($matches[1]);
            if (array_key_exists($op,$supported_ob_props)) {
               $params['observedproperty'] = $op;
            } else {
               $exceptions[] = array('InvalidParameterValue','
                  observedProperty',trim($nl->item(0)->
                  nodeValue));
            }
         } elseif (preg_match('|^http://mmisw.org/ont/cf/
            parameter/([A-Za-z_]+)$|',$op,$matches) === 1) {
            $op = strtolower($matches[1]);
            $old_op = array_search($op,$supported_ob_props);
            if ($old_op !== false) {
               $params['observedproperty'] = $old_op;
            } else {
               $exceptions[] = array('InvalidParameterValue','
                  observedProperty',trim($nl->item(0)->
                  nodeValue));
            }
         } elseif (preg_match('/^[a-z_]+$/i',$op) === 1) {
            $op = strtolower($op);
            if (array_key_exists($op,$supported_ob_props)) {
               $params['observedproperty'] = $op;
            } elseif (($old_op = array_search($op,
               $supported_ob_props)) !== false) {
               $params['observedproperty'] = $old_op;
            } else {
               $exceptions[] = array('InvalidParameterValue','
                  observedProperty',trim($nl->item(0)->
                  nodeValue));
            }
```

```
                } else {
                    $exceptions[] = array('InvalidParameterValue','
                        observedProperty',trim($nl->item(0)->nodeValue
                        ));
                }
                unset($op);
            } else {
                $exceptions[] = array('MissingParameterValue','
                    observedProperty',null);
            }
            // more to go...
        }
    }
} else {
    $exceptions[] = array('MissingParameterValue','service',null);
    $exceptions[] = array('MissingParameterValue','request',null);
}
return $params;
}

function makeExceptionReport($exceptions) {
$dom = new DOMDocument('1.0', 'UTF-8');
$dom->preserveWhiteSpace = false;
$dom->formatOutput   = true;
$root = $dom->createElement('ExceptionReport');
$root->setAttribute('xmlns','http://www.opengis.net/ows/1.1');
$root->setAttribute('xmlns:xsi','http://www.w3.org/2001/XMLSchema
    -instance');
$root->setAttribute('xsi:schemaLocation','http://www.opengis.net/
    ows/1.1 owsExceptionReport.xsd');
$root->setAttribute('version','1.0.0');
$root->setAttribute('xml:lang','en');
$dom->appendChild($root);
foreach ($exceptions as $e) {
    $dom->lastChild->appendChild($dom->createElement('Exception'))
        ;
    $dom->lastChild->lastChild->setAttribute('exceptionCode',$e
        [0]);
    if ($e[1] !== null) {
        $dom->lastChild->lastChild->setAttribute('locator',$e[1]);
    }
    if ($e[2] !== null) {
        $dom->lastChild->lastChild->appendChild($dom->createElement
            ('ExceptionText',$e[2]));
    }
}
return $dom;
}
```

The following code is called when an SOS request is GetCapabilities.

Listing E.3: SOS - GetCapabilities

```
function getCapabilities() {

```

```
global $exceptions,$params,$dom,$dbinfo,$supported_ob_props,
   $supported_sections,$describesensor_formats,
   $getobservation_formats;

$dom = new DOMDocument();
$dom->preserveWhiteSpace = false;
$dom->formatOutput = true;

$root = $dom->createElement('Capabilities');
$root->setAttribute('xmlns:gml','http://www.opengis.net/gml');
$root->setAttribute('xmlns:xlink','http://www.w3.org/1999/xlink')
   ;
$root->setAttribute('xmlns:swe','http://www.opengis.net/swe
   /1.0.1');
$root->setAttribute('xmlns:om','http://www.opengis.net/om/1.0');
$root->setAttribute('xmlns','http://www.opengis.net/sos/1.0');
$root->setAttribute('xmlns:sos', 'http://www.opengis.net/sos
   /1.0');
$root->setAttribute('xmlns:ows', 'http://www.opengis.net/ows
   /1.1');
$root->setAttribute('xmlns:ogc','http://www.opengis.net/ogc');
$root->setAttribute('xmlns:tml','http://www.opengis.net/tml');
$root->setAttribute('xmlns:sml', 'http://www.opengis.net/sensorML
   /1.0.1');
$root->setAttribute('xmlns:myorg', 'http://www.myorg.org/features
   ');
$root->setAttribute('xmlns:xsi','http://www.w3.org/2001/XMLSchema
   -instance');
$root->setAttribute('xsi:schemaLocation','http://www.opengis.net/
   sos/1.0 http://schemas.opengis.net/sos/1.0.0/sosAll.xsd');
$root->setAttribute('version','1.0.0');
$dom->appendChild($root);
$root = $dom->documentElement;
if (in_array('ServiceIdentification',$params['sections'])) {
   $si = $dom->createElement('ows:ServiceIdentification');
   if (TITLE !== null) {
      $si->appendChild($dom->createElement('ows:Title',TITLE));
   }
   if (ABSTRACTTEXT !== null) {
      $si->appendChild($dom->createElement('ows:Abstract',
         ABSTRACTTEXT));
   }
   if (KEYWORDS !== null) {
      $si->appendChild($dom->createElement('ows:Keywords'));
      foreach (split(',',KEYWORDS) as $keyword) {
         $si->lastChild->appendChild($dom->createElement('ows:
            Keyword',trim($keyword)));
      }
   }
   $si->appendChild($dom->createElement('ows:ServiceType','OGC:
      SOS'));
   $si->lastChild->setAttribute('codeSpace','http://
      opengeospatial.net');
   $si->appendChild($dom->createElement('ows:ServiceTypeVersion
```

```
        ','1.0.0'));
    $si->appendChild($dom->createElement('ows:Fees','NONE'));
    $si->appendChild($dom->createElement('ows:AccessConstraints','
        NONE'));
    $root->appendChild($si);
}
if (in_array('ServiceProvider',$params['sections'])) {
    $sp = $dom->createElement('ows:ServiceProvider');
    if (ORG !== null) {
        $sp->appendChild($dom->createElement('ows:ProviderName',ORG
            ));
    }
    if (ORG_URL !== null) {
        $sp->appendChild($dom->createElement('ows:ProviderSite'));
        $sp->lastChild->setAttribute('xlink:href',ORG_URL);
    }
    if (INDNAME !== null || POSNAME !== null || PHONE_FAX !== null
         || PHONE_VOICE !== null || ADDR_STREET !== null ||
        ADDR_CITY !== null || ADDR_STATE !== null || ADDR_ZIP !==
        null || ADDR_COUNTRY !== null || ADDR_EMAIL !== null) {
        $sp->appendChild($dom->createElement('ows:ServiceContact'))
            ;
        if (INDNAME !== null) {
            $sp->lastChild->appendChild($dom->createElement('ows:
                IndividualName',INDNAME));
        }
        if (POSNAME !== null) {
            $sp->lastChild->appendChild($dom->createElement('ows:
                PositionName',POSNAME));
        }
        if (PHONE_FAX !== null || PHONE_VOICE !== null ||
            ADDR_STREET !== null || ADDR_CITY !== null ||
            ADDR_STATE !== null || ADDR_ZIP !== null ||
            ADDR_COUNTRY !== null || ADDR_EMAIL !== null) {
            $sp->lastChild->appendChild($dom->createElement('ows:
                ContactInfo'));
            if (PHONE_FAX !== null || PHONE_VOICE !== null) {
                $sp->lastChild->lastChild->appendChild($dom->
                    createElement('ows:Phone'));
                if (PHONE_VOICE !== null) {
                    $sp->lastChild->lastChild->lastChild->appendChild(
                        $dom->createElement('ows:Voice',PHONE_VOICE));
                }
                if (PHONE_FAX !== null) {
                    $sp->lastChild->lastChild->lastChild->appendChild(
                        $dom->createElement('ows:Facsimile',PHONE_FAX)
                        );
                }
            }
            if (ADDR_STREET !== null || ADDR_CITY !== null ||
                ADDR_STATE !== null || ADDR_ZIP !== null ||
                ADDR_COUNTRY !== null || ADDR_EMAIL !== null) {
                $sp->lastChild->lastChild->appendChild($dom->
                    createElement('ows:Address'));
```

```
            if (ADDR_STREET !== null) {
               $sp->lastChild->lastChild->lastChild->appendChild(
                  $dom->createElement('ows:DeliveryPoint',
                  ADDR_STREET));
            }
            if (ADDR_CITY !== null) {
               $sp->lastChild->lastChild->lastChild->appendChild(
                  $dom->createElement('ows:City',ADDR_CITY));
            }
            if (ADDR_STATE !== null) {
               $sp->lastChild->lastChild->lastChild->appendChild(
                  $dom->createElement('ows:AdministrativeArea',
                  ADDR_STATE));
            }
            if (ADDR_ZIP !== null) {
               $sp->lastChild->lastChild->lastChild->appendChild(
                  $dom->createElement('ows:PostalCode',ADDR_ZIP)
                  );
            }
            if (ADDR_COUNTRY !== null) {
               $sp->lastChild->lastChild->lastChild->appendChild(
                  $dom->createElement('ows:Country',ADDR_COUNTRY
                  ));
            }
            if (ADDR_EMAIL !== null) {
               $sp->lastChild->lastChild->lastChild->appendChild(
                  $dom->createElement('ows:ElectronicMailAddress
                  ',ADDR_EMAIL));
            }

         }
      }
   }
   $root->appendChild($sp);
}
if (in_array('OperationsMetadata',$params['sections'])) {
   $om = $dom->createElement('ows:OperationsMetadata');
   foreach (array('GetCapabilities','GetObservation','
      DescribeSensor') as $op) {
      $om->appendChild($dom->createElement('ows:Operation'));
      $om->lastChild->setAttribute('name',$op);
      $om->lastChild->appendChild($dom->createElement('ows:DCP'))
         ;
      $om->lastChild->lastChild->appendChild($dom->createElement
         ('ows:HTTP'));
      $om->lastChild->lastChild->lastChild->appendChild($dom->
         createElement('ows:Get'));
      $om->lastChild->lastChild->lastChild->lastChild->
         setAttribute('xlink:href',SOS_URL);
      $om->lastChild->lastChild->lastChild->appendChild($dom->
         createElement('ows:Post'));
      $om->lastChild->lastChild->lastChild->lastChild->
         setAttribute('xlink:href',SOS_URL);
      if ($op == 'GetCapabilities') {
```

```
            $om->lastChild->appendChild($dom->createElement('ows:
                Parameter'));
            $om->lastChild->lastChild->setAttribute('name','Sections
                ');
            $om->lastChild->lastChild->appendChild($dom->
                createElement('ows:AllowedValues'));
            foreach ($supported_sections as $sect) {
               $om->lastChild->lastChild->lastChild->appendChild(
                  $dom->createElement('ows:Value',$sect));
            }
            $om->lastChild->lastChild->lastChild->appendChild($dom->
                createElement('ows:Value','All'));
         } elseif ($op == 'DescribeSensor') {
            $om->lastChild->appendChild($dom->createElement('ows:
                Parameter'));
            $om->lastChild->lastChild->setAttribute('name','
                outputFormat');
            $om->lastChild->lastChild->appendChild($dom->
                createElement('ows:AllowedValues'));
            foreach ($describesensor_formats as $ofmt) {
               $om->lastChild->lastChild->lastChild->appendChild(
                  $dom->createElement('ows:Value',htmlspecialchars(
                  $ofmt)));
            }
         } elseif ($op == 'GetObservation') {
            $om->lastChild->appendChild($dom->createElement('ows:
                Parameter'));
            $om->lastChild->lastChild->setAttribute('name','
                observedProperty');
            $om->lastChild->lastChild->appendChild($dom->
                createElement('ows:AllowedValues'));
            foreach ($supported_ob_props as $sop) {
               $om->lastChild->lastChild->lastChild->appendChild(
                  $dom->createElement('ows:Value',$sop));
            }echo $laststn;
         }

      }
      $om->appendChild($dom->createElement('ows:Parameter'));
      $om->lastChild->setAttribute('name','service');
      $om->lastChild->appendChild($dom->createElement('ows:
          AllowedValues'));
      $om->lastChild->lastChild->appendChild($dom->createElement('
          ows:Value','SOS'));
      $om->appendChild($dom->createElement('ows:Parameter'));
      $om->lastChild->setAttribute('name','version');
      $om->lastChild->appendChild($dom->createElement('ows:
          AllowedValues'));
      $om->lastChild->lastChild->appendChild($dom->createElement('
          ows:Value','1.0.0'));
      $root->appendChild($om);
   }
   if (in_array('Contents',$params['sections'])) {
      $c = $dom->createElement('Contents');
```

```
$c->appendChild($dom->createElement('ObservationOfferingList')
   );
$all = array('startdate'=>'9999-12-31T23:59:59Z','stopdate
   '=>'0000-01-01T00:00:00Z','llat'=>90,'llon'=>360,'ulat
   '=>-90,'ulon'=>0,'prop'=>array(),'stations'=>array());
$mysqli = new mysqli($dbinfo[DBID]['host'], $dbinfo[DBID]['
   user'], $dbinfo[DBID]['pw'], $dbinfo[DBID]['db']);
$sql = " select lower(s.UUID) as name, s.description, p.
   longitude as lon, p.latitude as lat, s.name as shortstname
   ,  ssn.message_id as sensornumber, s.model as stname,
   date_format(min(o.timestamp),'%Y-%m-%dT%TZ') as
   firstObsDate,  date_format(max(o.timestamp),'%Y-%m-%dT%TZ
   ') as lastObsDate from SENSOR s inner join  POSIT p on s.
   UUID = p.UUID inner  join OBSERVATION o on p.UUID = o.
   sensor_id inner join SENSOR_SN ssn on o.sensor_id = ssn.
   sensor_uuid group by o.sensor_id;";
if ($mysqli) {
   if ($result = $mysqli->query($sql)) {
      $laststn = null;
      $startdate = '9999-12-31T23:59:59Z';
      $stopdate = '0000-01-01T00:00:00Z';
      // echo "I am here";
      $oplist = array();
      while ($row = $result->fetch_assoc()) {
         if ($row['name'] != $laststn) {
            $laststn = $row['name'];
            //echo $laststn;
            $all['stations'][] = $row['name'];
            $startdate = '9999-12-31T23:59:59Z';
            $stopdate = '0000-01-01T00:00:00Z';
            $oplist = array();
            $offering = $dom->createElement('
               ObservationOffering');
            $offering->setAttribute('gml:id','station-'.$row['
               name']);
            $offering->appendChild($dom->createElement('gml:
               description',htmlentities($row['description'])
               ));
            $offering->appendChild($dom->createElement('gml:
               name','urn:'.NID_URN.':station:wmo:'.$row['
               name'].':'));
            $offering->appendChild($dom->createElement('gml:
               srsName',SRSNAME));
            $offering->appendChild(boundingBox($row['lat'],
               $row['lon'],$row['lat'],$row['lon']));
            $time = $dom->createElement('time');
            $time->appendChild($dom->createElement('gml:
               TimePeriod'));
            $start = $time->firstChild->appendChild($dom->
               createElement('gml:beginPosition',$row['
               firstObsDate']));
            if ($row['lastObsDate'] < CUTOFF) {
               $stop = $time->firstChild->appendChild($dom->
                  createElement('gml:endPosition',$row['
```

```
                lastObsDate']));
            } else {
                $stop = $time->firstChild->appendChild($dom->
                    createElement('gml:endPosition'));
                $stop->setAttribute('indeterminatePosition','
                    now');
            }
            $offering->appendChild($time);
            $procedure = $offering->appendChild($dom->
                createElement('procedure'));
            $procedure->setAttribute('xlink:href','urn:'.
                NID_URN.':station:wmo:'.$row['name'].':');
            $foi = $offering->appendChild($dom->createElement
                ('featureOfInterest'));
            $offering->lastChild->setAttribute('xlink:href','
                urn:cgi:Feature:CGI:EarthOcean');
            foreach (array_values($getobservation_formats) as
                $fmt) {
                $offering->appendChild($dom->createElement('
                    responseFormat',htmlspecialchars($fmt)));
            }
            $offering->appendChild($dom->createElement('
                resultModel','om:Observation'));
            $offering->appendChild($dom->createElement('
                responseMode','inline'));
            $c->lastChild->appendChild($offering);
            if ($row['lat'] < $all['llat']) {
                $all['llat'] = $row['lat'];
            }
            if ($row['lat'] > $all['ulat']) {
                $all['ulat'] = $row['lat'];
            }
            if ($row['lon']+180 < $all['llon']) {
                $all['llon'] = $row['lon']+180;
            }
            if ($row['lon']+180 > $all['ulon']) {
                $all['ulon'] = $row['lon']+180;
            }
        }
        if ($row['firstObsDate'] < $startdate) {
            $startdate = $row['firstObsDate'];
            $start->nodeValue = $row['firstObsDate'];
        }
        if ($row['lastObsDate'] > $stopdate) {
            $stopdate = $row['lastObsDate'];
            if ($row['lastObsDate'] < CUTOFF) {
                $stop->nodeValue = $row['lastObsDate'];
            } else {
                $stop->nodeValue = null;
                $stop->setAttribute('indeterminatePosition','
                    now');
            }
        }
        if ($row['firstObsDate'] < $all['startdate']) {
```

```
                $all['startdate'] = $row['firstObsDate'];
            }
            if ($row['lastObsDate'] > $all['stopdate']) {
                $all['stopdate'] = $row['lastObsDate'];
            }
            if (! in_array($row['stname'],$oplist)) {
                $oplist[] =$row['stname'];
                $op = $foi->parentNode->insertBefore($dom->
                    createElement('observedProperty'),$foi);
                $op->setAttribute('xlink:href','http://mmisw.org/
                    ont/cf/parameter/'.$supported_ob_props[
                    strtolower($row['stname'])]);
                if (! in_array($row['stname'],$all['prop'])) {
                    $all['prop'][] = $row['stname'];
                }
            }
        }
        $result->free();
      }
      $mysqli->close();
   }

}
```

The following code is called when an SOS request is DescribeSensor.

Listing E.4: SOS - DescribeSensor

```
function describeSensor() {
   global $exceptions,$params,$dom,$dbinfo,$supported_ob_props;

   function createPosition($name,$vectorid,$lat,$lon) {
      global $dom;
      $e = $dom->createElement('sml:position');
      $e->setAttribute('name',$name);
      $e->appendChild($dom->createElement('swe:Position'));
      $e->lastChild->setAttribute('referenceFrame','urn:ogc:crs:EPSG
          :4326');
      $e->lastChild->appendChild($dom->createElement('swe:location')
          );
      $e->lastChild->lastChild->appendChild($dom->createElement('swe
          :Vector'));
      $e->lastChild->lastChild->lastChild->setAttribute('gml:id',
          $vectorid);
      $e->lastChild->lastChild->lastChild->setAttribute('definition
          ','urn:ogc:def:property:OGC:location');
      $e->lastChild->lastChild->lastChild->appendChild($dom->
          createElement('swe:coordinate'));
      $e->lastChild->lastChild->lastChild->lastChild->setAttribute('
          name','latitude');
      $e->lastChild->lastChild->lastChild->lastChild->appendChild(
          $dom->createElement('swe:Quantity'));
      $e->lastChild->lastChild->lastChild->lastChild->lastChild->
          setAttribute('axisID','Y');
      $e->lastChild->lastChild->lastChild->lastChild->lastChild->
```

```
        appendChild($dom->createElement('swe:uom'));
    $e->lastChild->lastChild->lastChild->lastChild->lastChild->
        lastChild->setAttribute('code','deg');
    $e->lastChild->lastChild->lastChild->lastChild->lastChild->
        appendChild($dom->createElement('swe:value',$lat));
    $e->lastChild->lastChild->lastChild->appendChild($dom->
        createElement('swe:coordinate'));
    $e->lastChild->lastChild->lastChild->lastChild->setAttribute('
        name','longitude');
    $e->lastChild->lastChild->lastChild->lastChild->appendChild(
        $dom->createElement('swe:Quantity'));
    $e->lastChild->lastChild->lastChild->lastChild->lastChild->
        setAttribute('axisID','X');
    $e->lastChild->lastChild->lastChild->lastChild->lastChild->
        appendChild($dom->createElement('swe:uom'));
    $e->lastChild->lastChild->lastChild->lastChild->lastChild->
        lastChild->setAttribute('code','deg');
    $e->lastChild->lastChild->lastChild->lastChild->lastChild->
        appendChild($dom->createElement('swe:value',$lon));
    return $e;
}

$stn = strtolower($params['procedure']['stn']);
echo $params['procedure']['sensor'];
if ($params['procedure']['sensor'] === null) {
    $sensor = null;
} else {
    if (preg_match('/.+\d$/',$params['procedure']['sensor']) == 1)
        {
        $sensor = strtolower(substr($params['procedure']['sensor
            '],0,-1));
        $sensornumber = substr($params['procedure']['sensor'],-1);
    } else {
        $sensor = strtolower($params['procedure']['sensor']);
        $sensornumber = '0';
    }
}
//echo "I am here";
$mysqli = new mysqli($dbinfo[DBID]['host'], $dbinfo[DBID]['user
    '], $dbinfo[DBID]['pw'], $dbinfo[DBID]['db']);
if ($mysqli) {
    if ($sensor !== null) {
        $sql = "select lower(s.UUID) as stn, p.longitude as lon, p.
            latitude as lat, s.name as shortstname, s.UUID as
            sensornumber from POSIT p inner join SENSOR s on p.UUID
             = s.UUID where lower(s.UUID) = '$stn' and s.name = '
            $sensor' and s.UUID = $sensornumber order by s.name, s.
            sensorType;";
    } else {
        $sql = "select lower(s.UUID) as stn, p.longitude as lon, p.
            latitude as lat, s.manufacturer as org from POSIT p
            left join SENSOR s on s.UUID = p.UUID where lower(s.
            UUID) = '$stn' order by s.name;";
    }
```

```
    if ($result = $mysqli->query($sql)) {
        $sensordata = $result->fetch_assoc();
        if ($sensordata && $sensor !== null) {
            $sensordata['sensorid'] = $sensordata['shortstname'].
                $sensordata['sensornumber'];
        }
        $result->free();
    }
    $mysqli->close();
}

if ($sensordata) {
    $dom = new DOMDocument('1.0', 'UTF-8');
    $dom->preserveWhiteSpace = false;
    $dom->formatOutput = true;
    $root = $dom->createElement('sml:SensorML');
    $root->setAttribute('xmlns:sml','http://www.opengis.net/
        sensorML/1.0.1');
    $root->setAttribute('xmlns:xlink','http://www.w3.org/1999/
        xlink');
    $root->setAttribute('xmlns:gml','http://www.opengis.net/gml');
    $root->setAttribute('xmlns:swe','http://www.opengis.net/swe
        /1.0.1');
    $root->setAttribute('xmlns:xsi','http://www.w3.org/2001/
        XMLSchema-instance');
    $root->setAttribute('xsi:schemaLocation','http://www.opengis.
        net/sensorML/1.0.1 http://schemas.opengis.net/sensorML
        /1.0.1/sensorML.xsd');
    $root->setAttribute('version','1.0.1');
    $dom->appendChild($root);
    $root = $dom->documentElement;
    $root->appendChild($dom->createElement('sml:member'));
    $system = $root->lastChild->appendChild($dom->CreateElement('
        sml:System'));
    if ($sensor === null) {
        $system->setAttribute('gml:id','station-'.$stn);
    } else {
        $system->setAttribute('gml:id','sensor-'.$stn.'-'.$sensor.
            $sensornumber);
    }
    $system->appendChild($dom->createElement('sml:identification')
        );
    $system->lastChild->appendChild($dom->createElement('sml:
        IdentifierList'));
    if ($sensor === null) {
        $system->lastChild->lastChild->appendChild(smlIdentifier('
            StationId',$params['procedure']['urn']));
    } else {
        $system->lastChild->lastChild->appendChild(smlIdentifier('
            SensorId',$params['procedure']['urn']));
    }
    if ($sensor === null && $sensordata['org'] !== null) {
        $contact = $system->appendChild($dom->createElement('sml:
            contact'));
```

```
            $contact->setAttribute('xlink:role','urn:ogc:def:
                classifiers:OGC:contactType:operator');
            $contact->appendChild($dom->createElement('sml:
                ResponsibleParty'));
            $contact->lastChild->appendChild($dom->createElement('sml:
                organizationName',htmlentities($sensordata['org'])));
        }

//      $system->appendChild($dom->createElement('sml:inputs'));
//      $system->lastChild->appendChild($dom->createElement('sml:
    InputList'));
//      $system->appendChild($dom->createElement('sml:outputs'));
//      $system->lastChild->appendChild($dom->createElement('sml:
    OutputList'));
        $system->appendChild($dom->createElement('sml:positions'));
        $system->lastChild->appendChild($dom->createElement('sml:
            PositionList'));
        $system->lastChild->lastChild->appendChild(createPosition('
            stationPosition','STATION_LOCATION',$sensordata['lat'],
            $sensordata['lon']));
    } else {
        $exceptions[] = array('NoApplicableCode',null,'No data found
            for this sensor');
    }
    return $dom;
}
```

E.2 GetCapabilities Response

Sample GetCapabilities response :

Listing E.5: SOS - Sample GetCapabilities Response

```
-<Capabilities xsi:schemaLocation="http://www.opengis.net/sos/1.0
    http://schemas.opengis.net/sos/1.0.0/sosAll.xsd" version
    ="1.0.0">
?
<ows:ServiceIdentification>
<ows:Title>UNB FCS Test SOS</ows:Title>
?
<ows:Abstract>
University of New Brunswick (UNB), Faculty of Computer Science (FCS)
</ows:Abstract>
?
<ows:Keywords>
<ows:Keyword>Room Temperature</ows:Keyword>
<ows:Keyword>Voltage</ows:Keyword>
<ows:Keyword>Solar</ows:Keyword>
</ows:Keywords>
<ows:ServiceType codeSpace="http://opengeospatial.net">OGC:SOS</ows:
    ServiceType>
<ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
<ows:Fees>NONE</ows:Fees>
```

```
<ows:AccessConstraints>NONE</ows:AccessConstraints>
</ows:ServiceIdentification>
?
<ows:ServiceProvider>
<ows:ProviderName>UNB FCS Test</ows:ProviderName>
<ows:ProviderSite xlink:href="http://ib214m09.cs.unb.ca/html/sos/
   server.php?service=SOS&request=GetCapabilities"/>
?
<ows:ServiceContact>
<ows:IndividualName>Gunita Saini</ows:IndividualName>
<ows:PositionName>Student</ows:PositionName>
?
<ows:ContactInfo>
?
<ows:Phone>
<ows:Voice>506-453-4566</ows:Voice>
<ows:Facsimile>(506) 453-3566</ows:Facsimile>
</ows:Phone>
?
<ows:Address>
<ows:DeliveryPoint>540 Windsor Street</ows:DeliveryPoint>
<ows:City>Fredericton</ows:City>
<ows:AdministrativeArea>NB</ows:AdministrativeArea>
<ows:PostalCode>E3B5A3</ows:PostalCode>
<ows:Country>Canada</ows:Country>
<ows:ElectronicMailAddress>www.unb.ca</ows:ElectronicMailAddress>
</ows:Address>
</ows:ContactInfo>
</ows:ServiceContact>
</ows:ServiceProvider>
?
<ows:OperationsMetadata>
?
<ows:Operation name="GetCapabilities">
?
<ows:DCP>
?
<ows:HTTP>
<ows:Get xlink:href=""/>
<ows:Post xlink:href=""/>
</ows:HTTP>
</ows:DCP>
?
<ows:Parameter name="Sections">
?
<ows:AllowedValues>
<ows:Value>ServiceIdentification</ows:Value>
<ows:Value>ServiceProvider</ows:Value>
<ows:Value>OperationsMetadata</ows:Value>
<ows:Value>Contents</ows:Value>
<ows:Value>All</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
</ows:Operation>
```

```
?
<ows:Operation name="GetObservation">
?
<ows:DCP>
?
<ows:HTTP>
<ows:Get xlink:href=""/>
<ows:Post xlink:href=""/>
</ows:HTTP>
</ows:DCP>
?
<ows:Parameter name="observedProperty">
?
<ows:AllowedValues>
<ows:Value>Air Temperature</ows:Value>
<ows:Value>Battery Voltage</ows:Value>
<ows:Value>Water Level</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
</ows:Operation>
?
<ows:Operation name="DescribeSensor">
?
<ows:DCP>
?
<ows:HTTP>
<ows:Get xlink:href=""/>
<ows:Post xlink:href=""/>
</ows:HTTP>
</ows:DCP>
?
<ows:Parameter name="outputFormat">
?
<ows:AllowedValues>
<ows:Value>text/xml;subtype=\"sensorML/1.0.0\"</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
</ows:Operation>
?
<ows:Parameter name="service">
?
<ows:AllowedValues>
<ows:Value>SOS</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
?
<ows:Parameter name="version">
?
<ows:AllowedValues>
<ows:Value>1.0.0</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
</ows:OperationsMetadata>
?
```

```
<Contents>
?
<ObservationOfferingList>
?
<ObservationOffering gml:id="network-all">
<gml:description>All stations on the SWL server</gml:description>
<gml:name>urn:swl:network:myorg:all</gml:name>
<gml:srsName>urn:ogc:def:crs:epsg::4326</gml:srsName>
?
<gml:boundedBy>
?
<gml:Envelope srsName="urn:ogc:def:crs:epsg::4326">
<gml:lowerCorner>45.949722 -66.642778</gml:lowerCorner>
<gml:upperCorner>45.949722 -66.642778</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
?
<time>
?
<gml:TimePeriod>
<gml:beginPosition>2010-06-17T20:28:14Z</gml:beginPosition>
<gml:endPosition>0000-00-00T00:00:00Z</gml:endPosition>
</gml:TimePeriod>
</time>
<procedure xlink:href="urn:swl:station:wmo:391e816927484bcb:"/>
<procedure xlink:href="urn:swl:station:wmo:5867a59e24f844f3:"/>
<procedure xlink:href="urn:swl:station:wmo:c6cf046dc1fa4e63:"/>
<observedProperty xlink:href="http://mmisw.org/ont/cf/parameter/"/>
<observedProperty xlink:href="http://mmisw.org/ont/cf/parameter/"/>
<featureOfInterest xlink:href="urn:cgi:Feature:CGI:EarthOcean"/>
<responseFormat>text/xml;schema=\"swl/0.6.1\"</responseFormat>
<responseFormat>application/swl+xml;version=0.6.1</responseFormat>
<responseFormat>text/csv</responseFormat>
<responseFormat>text/tab-separated-values</responseFormat>
<responseFormat>application/vnd.google-earth.kml+xml</responseFormat
   >
<resultModel>om:Observation</resultModel>
<responseMode>inline</responseMode>
</ObservationOffering>
?
<ObservationOffering gml:id="station-391e816927484bcb">
<gml:description>Voltage</gml:description>
<gml:name>urn:swl:station:wmo:391e816927484bcb:</gml:name>
<gml:srsName>urn:ogc:def:crs:epsg::4326</gml:srsName>
?
<gml:boundedBy>
?
<gml:Envelope srsName="urn:ogc:def:crs:epsg::4326">
<gml:lowerCorner>45.949722 -66.642778</gml:lowerCorner>
<gml:upperCorner>45.949722 -66.642778</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
?
<time>
```

```
?
<gml:TimePeriod>
<gml:beginPosition>2010-06-17T20:30:06Z</gml:beginPosition>
<gml:endPosition>0000-00-00T00:00:00 Z</gml:endPosition>
</gml:TimePeriod>
</time>
<procedure xlink:href="urn:swl:station:wmo:391e816927484bcb:"/>
<observedProperty xlink:href="http://mmisw.org/ont/cf/parameter/"/>
<featureOfInterest xlink:href="urn:cgi:Feature:CGI:EarthOcean"/>
<responseFormat>text/xml;schema=\"swl/0.6.1\"</responseFormat>
<responseFormat>application/swl+xml;version=0.6.1</responseFormat>
<responseFormat>text/csv</responseFormat>
<responseFormat>text/tab-separated-values</responseFormat>
<responseFormat>application/vnd.google-earth.kml+xml</responseFormat
   >
<resultModel>om:Observation</resultModel>
<responseMode>inline</responseMode>
</ObservationOffering>
?
<ObservationOffering gml:id="station-5867a59e24f844f3">
<gml:description>WaterLevel</gml:description>
<gml:name>urn:swl:station:wmo:5867a59e24f844f3:</gml:name>
<gml:srsName>urn:ogc:def:crs:epsg::4326</gml:srsName>
?
<gml:boundedBy>
?
<gml:Envelope srsName="urn:ogc:def:crs:epsg::4326">
<gml:lowerCorner>45.949722 -66.642778</gml:lowerCorner>
<gml:upperCorner>45.949722 -66.642778</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
?
<time>
?
<gml:TimePeriod>
<gml:beginPosition>2010-06-17T20:44:36Z</gml:beginPosition>
<gml:endPosition>0000-00-00T00:00:00Z</gml:endPosition>
</gml:TimePeriod>
</time>
<procedure xlink:href="urn:swl:station:wmo:5867a59e24f844f3:"/>
<observedProperty xlink:href="http://mmisw.org/ont/cf/parameter/"/>
<featureOfInterest xlink:href="urn:cgi:Feature:CGI:EarthOcean"/>
<responseFormat>text/xml;schema=\"swl/0.6.1\"</responseFormat>
<responseFormat>application/swl+xml;version=0.6.1</responseFormat>
<responseFormat>text/csv</responseFormat>
<responseFormat>text/tab-separated-values</responseFormat>
<responseFormat>application/vnd.google-earth.kml+xml</responseFormat
   >
<resultModel>om:Observation</resultModel>
<responseMode>inline</responseMode>
</ObservationOffering>
?
<ObservationOffering gml:id="station-c6cf046dc1fa4e63">
<gml:description>Voltage</gml:description>
```

```
<gml:name>urn:swl:station:wmo:c6cf046dc1fa4e63:</gml:name>
<gml:srsName>urn:ogc:def:crs:epsg::4326</gml:srsName>
?
<gml:boundedBy>
?
<gml:Envelope srsName="urn:ogc:def:crs:epsg::4326">
<gml:lowerCorner>45.949722 -66.642778</gml:lowerCorner>
<gml:upperCorner>45.949722 -66.642778</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
?
<time>
?
<gml:TimePeriod>
<gml:beginPosition>2010-06-17T20:28:14Z</gml:beginPosition>
<gml:endPosition>0000-00-00T00:00:00Z</gml:endPosition>
</gml:TimePeriod>
</time>
<procedure xlink:href="urn:swl:station:wmo:c6cf046dc1fa4e63:"/>
<observedProperty xlink:href="http://mmisw.org/ont/cf/parameter/"/>
<featureOfInterest xlink:href="urn:cgi:Feature:CGI:EarthOcean"/>
<responseFormat>text/xml;schema=\"swl/0.6.1\"</responseFormat>
<responseFormat>application/swl+xml;version=0.6.1</responseFormat>
<responseFormat>text/csv</responseFormat>
<responseFormat>text/tab-separated-values</responseFormat>
<responseFormat>application/vnd.google-earth.kml+xml</responseFormat
   >
<resultModel>om:Observation</resultModel>
<responseMode>inline</responseMode>
</ObservationOffering>
</ObservationOfferingList>
</Contents>
</Capabilities>
```

E.3 GetObservation Response

Sample GetObservation Response:

Listing E.6: SOS - GetObservations Response

```
<om:CompositeObservation xsi:schemaLocation="http://www.opengis.net/
   om/1.0 http://www.csc.noaa.gov/swl/schema/swl-DIF/swl/0.6.1/
   schemas/swlObservationSpecializations.xsd" gml:id="
   TemperaturePointCollectionTimeSeriesObservation">
-
<gml:description>
Voltage observations at a point at a series of times
</gml:description>
-
<gml:name>
Voltage observations at station ID 391E816927484BCB
</gml:name>
-
```

```
<gml:boundedBy>
-
<gml:Envelope srsName="urn:ogc:def:crs:epsg::4326">
<gml:lowerCorner>45.949722 -66.642778</gml:lowerCorner>
<gml:upperCorner>45.949722 -66.642778</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
-
<om:samplingTime>
-
<gml:TimePeriod gml:id="ST">
<gml:beginPosition>2010-06-24T13:25:24Z</gml:beginPosition>
<gml:endPosition>0000-00-00T00:00:00Z</gml:endPosition>
</gml:TimePeriod>
</om:samplingTime>
-
<om:procedure>
-
<om:Process>
-
<swl:CompositeContext gml:id="SensorMetadata" recDef="http://www.csc
   .noaa.gov/swl/schema/swl-DIF/swl/0.6.1/recordDefinitions/
   PointSensorMetadataRecordDefinition.xml">
-
<gml:valueComponents>
<swl:Count name="NumberOfStations">1</swl:Count>
-
<swl:ContextArray gml:id="StationArray">
-
<gml:valueComponents>
-
<swl:CompositeContext gml:id="Station1Info">
-
<gml:valueComponents>
<swl:StationName>Voltage</swl:StationName>
<swl:Organization>78e89380bffe11de</swl:Organization>
<swl:StationId>urn:swl:station:wmo:391e816927484bcb:</swl:StationId>
-
<gml:Point gml:id="Station1LatLon">
<gml:pos>45.949722 -66.642778</gml:pos>
</gml:Point>
<swl:VerticalDatum>urn:ogc:def:datum:epsg::5113</swl:VerticalDatum>
<swl:VerticalPosition xsi:nil="true" nilReason="missing"/>
<swl:Context name="TotalWaterDepth" uom="m">116</swl:Context>
<swl:Count name="Station1NumberOfSensors">1</swl:Count>
-
<swl:ContextArray gml:id="Station1SensorArray">
-
<gml:valueComponents>
-
<swl:CompositeContext gml:id="Station1Sensor1Info">
-
<gml:valueComponents>
-
```

```
<swl:SensorId>
urn:swl:sensor:wmo:391e816927484bcb::GV2391e816927484bcb:
</swl:SensorId>
<swl:Context name="SensorDepth" uom="m">-2</swl:Context>
<swl:SamplingRate uom="Hz" xsi:nil="true" nilReason="missing"/>
<swl:ReportingInterval uom="s" xsi:nil="true" nilReason="missing"/>
<swl:ProcessingLevel xsi:nil="true" nilReason="missing"/>
</gml:valueComponents>
</swl:CompositeContext>
</gml:valueComponents>
</swl:ContextArray>
</gml:valueComponents>
</swl:CompositeContext>
</gml:valueComponents>
</swl:ContextArray>
</gml:valueComponents>
</swl:CompositeContext>
</om:Process>
</om:procedure>
<om:observedProperty xlink:href="http://mmisw.org/ont/cf/parameter/
   Battery Voltage"/>
<om:featureOfInterest xlink:href="urn:cgi:Feature:CGI:EarthOcean"/>
-
<om:result>
-
<swl:Composite recDef="http://www.csc.noaa.gov/swl/schema/swl-DIF/
   swl/0.6.1/recordDefinitions/TemperaturePointDataRecordDefinition
   .xml" gml:id="VoltagePointCollectionTimeSeriesDataObservations">
-
<gml:valueComponents>
<swl:Count name="NumberOfObservationsPoints">1</swl:Count>
-
<swl:Array gml:id="TemperaturePointCollectionTimeSeries">
-
<gml:valueComponents>
-
<swl:Composite gml:id="Station1TimeSeriesRecord">
-
<gml:valueComponents>
<swl:Count name="Station1NumberOfObservationsTimes">1</swl:Count>
-
<swl:Array gml:id="Station1TimeSeries">
-
<gml:valueComponents>
-
<swl:Composite gml:id="Station1T1Point">
-
<gml:valueComponents>
-
<swl:CompositeContext gml:id="Station1T1ObservationConditions"
   processDef="#StationInfo">
-
<gml:valueComponents>
-
```

```
<gml:TimeInstant gml:id="Station1T1Time">
<gml:timePosition>2010-06-24T13:25:24Z</gml:timePosition>
</gml:TimeInstant>
</gml:valueComponents>
</swl:CompositeContext>
-
<swl:CompositeValue gml:id="Station1T1PointObservation" processDef
   ="#StationSensorInfo">
-
<gml:valueComponents>
<swl:Quantity name="Voltage" uom="V" xsi:nil="true" nilReason="
   missing"/>
</gml:valueComponents>
</swl:CompositeValue>
</gml:valueComponents>
</swl:Composite>
</gml:valueComponents>
</swl:Array>
</gml:valueComponents>
</swl:Composite>
</gml:valueComponents>
</swl:Array>
</gml:valueComponents>
</swl:Composite>
</om:result>
</om:CompositeObservation>
```

Vita

Candidate's full name: Gunita Saini
Place & date of birth: New Delhi, India, 31 August, 1985

University attended:
September 2008 - October 2010
Faculty of Computer Science
University of New Brunswick
Fredericton, New Brunswick, Canada

Bachelor of Computer Science & Engg.
August 2003 - May 2007
Beant College of Engg. & Technology
Punjab Technical University
Gurdaspur, Punjab, India

Technical Report:
Gunita Saini and Bradford G. Nickerson, "Sensor web language for dynamic sensor networks", UNB Faculty of Computer Science, TR09-196, Sept, 2009, Fredericton, N.B., Canada, 44 pages.

Poster:
Gunita Saini and Bradford G. Nickerson, "Poster: Sensor Web Language for Dynamic Sensor Networks", Proc. of the 7th UNB Computer Science Research Exposition, April 13, 2010, Fredericton, Canada.

Other Document:
Gunita Saini and Bradford G. Nickerson, "Measurand SAA Database and Web Interface architecture", UNB Faculty of Computer Science internal report for Measurand Inc., May 31, 2010, 29 pages.