Students: Greg Back, Kyle Foley, Dustin Swisher
Ohio Northern University, TJ Smull College of Engineering
Faculty Advisor: David Retterer

Sensor Network Project – Final Report

Date	Revision	Author	Comments
2010-05-12	A	All	Original Document

Table of Contents

1 Overview of this Document

This document describes the design of the Sensor Network project and its deliverables. It is based on the original project design proposal, and has been changed to reflect modifications made to the project over the course of the year. All team members had a role in developing this report.

2 Overview of the Sensor Network Project

The goals for the Sensor Network project were:

1. to design a generic sensor network **architecture** for acquisition, storage, retrieval and visualization of data, and
2. to demonstrate this architecture with an end-to-end **prototype**.

Although the final project is different in several ways from the initial design proposal, both of the goals listed above were met in the completion of the project.

3 Specifications

The Capabilities and Requirements Document for our project outlines the specifications for our project, the non-technical constraints that were imposed on our design, and our testing and verification procedure. Rather than reproducing them in their entirety, they are only summarized here. Refer to that document for the complete specifications.

The architecture developed for this project was designed meant to be as abstract and flexible. The architecture does not impose any artificial limitations on the type of data being collected, the proximity of the various major components of the system (within the same building or across the globe), or the particular technologies to be used. Each major component of the architecture is independent of the others except in the precise ways defined by their interfaces. As a result, the architecture has remained modular, allowing users to choose different components from different vendors while ensuring interoperability, as long as each component meets the defined interface standards. This is an operational necessity if the architecture is to be implemented on a large, enterprise scale.

The prototype was developed in order to demonstrate these abstract architecture principles in a concrete way, in order to verify that the architecture is valid. The concrete example is useful for demonstration purposes, but it is necessary to distinguish decisions made solely on the basis of scale (for the prototype) from those that affect the network regardless of size (for the architecture). Without the prototype, it would be impossible to demonstrate that the designed architecture is valid, yet the prototype implements additional requirements not found in the architecture (as outlined in the Capabilities and Requirements document).

3.1 Standards

The project as originally envisioned was planned to make heavy use of established standards such as the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) family of standards and best practices. This would allow maximum compatibility with existing sensors and sensor processing components. After encountering several significant difficulties in implementing the standards, the decision was made to focus on developing a working prototype than continuing to require strict adherence to the standards. Many prototype design decisions were still heavily influenced by the OGC standards, and the computational logic of each component was loosely coupled to its communication logic, allowing fully-standards-compliant communication to be easily implemented in the future.

4 System Design

This document describes both the sensor network architecture (deliverable 1) and the sensor network prototype (deliverable 2). Compared to the prototype, the architecture is more abstract and allows much more flexibility and modularity of components and technologies used. However, in order to limit the scope of work and prevent excessive abstraction, actual decisions were required when implementing the prototype.

Some of the description below is applicable only to the architecture, or only to the prototype, but the majority is meant to be common to both components. When a particular decision was made to be applied to the prototype only, that decision is clearly specified. Additionally, statements regarding the architecture which are not applicable to our prototype will be noted below.

The Sensor Network was designed to be a modular system, where there are clear boundaries and a well-defined interface between each of the subsystems. This method will allow for the highest level of interoperability between components, granted that each component meets the interface standards described below. An overview of the network architecture is defined in Figure 1, below. Each major component is described in detail in the following sections. Note that some of the component names have been changed from the design proposal in order to more clearly define their purpose.

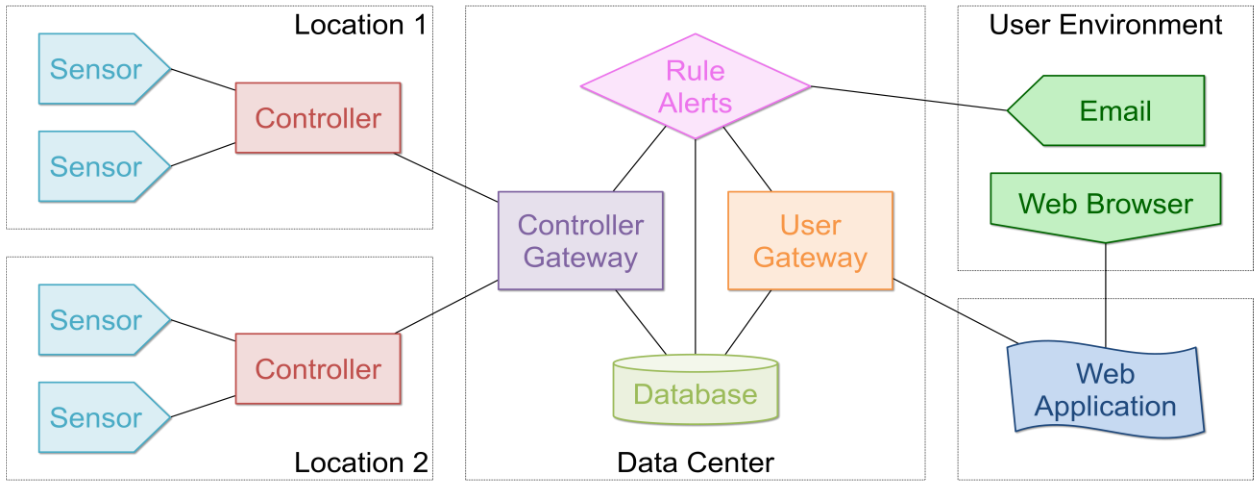

Figure 1. Sensor Network Architecture

5 Location

A *location* refers to a single environment from which data is to be collected. The components of each *location* must be local (geographically) to the environment being measured, within the limits of the communication range of the sensors (whether tethered via a cable or communicating wirelessly). Each *location* is composed of:

- One or more *sensors*.

- A *controller* to manage these sensors.

The architecture contains no fixed limit to the number of sensors or locations; this is bounded only by the particular implementation. The prototype has been designed to allow up to 64K controllers, and up to 64K sensors per controller. It currently consists of just two locations – a physical location (consisting of a hardware sensor and controller) and a virtual location. The virtual location is a C# application which simulates both a controller and its sensors.

5.1 Sensor

A s*ensor* is any device capable of gathering data and communicating that data to another device (for processing and/or persistent storage). *Sensors* will be used to collect the data to be processed by the remainder of the sensor network. A sensor may use a wired or wireless connection, use a battery or constant power source, and collect any of numerous types of data (such as temperature, humidity, motion, water depth, audio or video).

The sensor network architecture employs a distributed system for identifying each sensor. Each controller is assigned a unique controller ID number; each controller assigns each sensor at its location a locally-unique sensor ID. By combining the controller ID and sensor ID, a globally unique sensor ID can be determined. The prototype uses this combined ID for uniquely identifying each sensor. The sensor network architecture also allows for other identification systems, such as a system-wide sensor registry or the distributed manufacturer ID / unit ID system used for MAC addresses for network interfaces. The only requirement is that the system be able to uniquely identify each sensor.

The prototype sensor consists of a Cerebot II microcontroller, a LM35 temperature probe, and a RS232-to-USB connection. The LM35 outputs 10 mV per degree Celsius. This voltage is read by the analog-to-digital converter (ADC) on the Cerebot II configured for single-ended input with a reference voltage of 2.4 V. The voltage is sampled and stored as a 10-bit number and then converted to Celsius units. It is then sent to the controller via the USART port on the board, then via a USART to RS232 adapter and a RS232 to USB adapter.

5.2 Controller

Each sensor is connected to a single *controller* at a time. A sensor communicates with its designated controller using any of a variety of communication methods, either wired or wireless. The architecture does not specify any particular communication protocols between the sensor and controller, because the capabilities of the sensors can vary widely. For commercial sensors, the data received by the controller may be in a variety of formats, e.g. XML, ASCII-formatted text, or proprietary binary formats. The controller must be developed specifically to be able to communicate with a particular sensor or class of sensor. In contrast, communication between the controller and controller gateway must be explicitly specified and standardized across the system.

The sensor and sensor controller are the most tightly coupled components of the system. The controller must keep track of which sensors are connected to it, along with the details and data formats of each sensor. In many cases, commercially developed sensors would come with accompanying controllers, though it would also be possible to build a more generic controller supporting a variety of generic sensor data formats.

When reporting data measurements to the controller gateway, the controller reports the data value, units, and location of the measurement. By recording the location of each measurement, it is possible to determine if a sensor has moved. For sensors which do not report their own location (by having an integrated GPS component, for instance), the sensor location can be manually programmed into the controller, or the system can simply use the location of the controller for each sensor.

An additional function of the controller is to cache sensor measurements in case it is disconnected from the controller gateway, or is otherwise unable to forward sensor measurements. These measurements are stored with their measurement time, so that when connectivity is restored they can all be forwarded to the data center.

While developing the prototype sensor, a virtual sensor location (consisting of the controller and a software-simulated sensor) was developed in parallel to support testing of the remaining components of the system, create specific abnormal test scenarios which would be unlikely with actual sensors, as well as test the data throughput of the system. The prototype controller gateway was capable of processing 100 sensor measurements per second with no noticeable effects on performance.

6 Data Center

The *data center* is the central component of the Sensor Network architecture and prototype. While some of the other components retain local memory, all critical storage is accomplished at this central location. The data center is not a concrete component in the network, but rather a logical grouping of other key components. These components are the controller gateway, the database, the controller gateway, the user gateway, and the rule alerts.

The prototype uses a common Dell Dimension desktop computer to run all of the data center components on top of the Arch Linux operating system, Linux kernel 2.6.33.

6.1 Database

The *database* within the data center is the main data storage for the Sensor Network. Data received from the sensor controllers is inserted into the database. The database will be queried by the *user gateway* to respond to requests from the web application. The database also stores information on users, rules, and connected sensors and sensor controllers, which each other component of the data center may need to query in order to perform its duties.

The prototype uses a MySQL 5.1.45 database. It uses six tables: controller, sensor, measurement, rule, subscription, and user. Controller stores with which controllers the controller gateway is willing to communicate. The sensor table stores every known sensor that has ever provided data to the data center. Measurement stores every recorded sensor measurement so that it may later be recalled for viewing and analysis. The rule table stores every rule that the user has configured, and the subscription table stores which users are to be alerted when a rule is triggered. And finally the user table stores all the data about every user, which is a user name, password, user identification number, and an email address.

6.2 Controller Gateway

The *controller gateway* component receives data measurement messages from the *controller*. The architecture specifies that these messages be formatted using the OGC Observations and Measurements (O&M) XML schema, while the prototype uses a simplified version. The controller gateway inserts these measurements into the database (using SQL), and also forwards them to the rule alerts subsystem in order to evaluate them against the system's pre-defined rules.

An additional function for the controller gateway is to convert the measurements of each controller into a common system of units used by the database. Although this involves additional processing in the controller gateway, it permits the system to be more flexible and not force controllers to use any particular units. As long as the controller gateway understands the units used by a controller, that controller is compatible with the system.

6.3 User Gateway

The *user gateway* is responsible for querying the *database* in order to respond to requests from the *web application*, and the *user gateway* is responsible for sending rules to the rule alerts component. It communicates using OGC Web Services running on SOAP over HTTP.

The prototype merges the user gateway with the web application for the sake of simplicity, since the prototype only intends to allow one web application for the entire sensor network. The web application talks directly to the database and rule alerts to accomplish user gateway tasks. More on this in the *web application* section.

6.4 Rule Alerts

The *rule alert* component is responsible for analyzing incoming sensor measurements from the *controller gateway* using preconfigured measurement expressions stored in the *database* component. When a rule expression evaluates to true, the rule alerts finds every *user* subscribed to the rule in the *database* and emails them. Rules are created and maintained using the *web application* component.

The prototype’s rule alerts component is written in the D programming language, using DMD 2.045 compiler, and it runs on the datacenter machine (“warehouse”). Every measurement that the controller gateway sends to the database component is also sent to the rule alerts component to be analyzed. The web application does not implement a front-end to the rule alerts component. During testing rules were created by directly inserting them into the database, just as the subscribers and users.

7 Web Application

The *web application* is the component which connects the user to the sensor network. It works cooperatively with the user environment as the front-end to the sensor network. In a sense, the web application is an interface to the sensor network via a web browser.

The web application must comply with the standard sensor network client protocol. Supported operations within the protocol messages include authentication, data requests. This protocol exists only between the data center and the web application. The protocol between the web application and the user environment is HTTP.

The prototype uses the Apache 2.2.15 web server and the Tomcat 6.1 Java servlet container. It supports user authentication in addition to many user gateway responsibilities. The prototype only intended to support one web application client to the user gateway, so it was decided that the two components should be merged. For this reason, the web application directly queries the database and rule alerts components, and it runs on the same machine as the data center ("warehouse"). Another important part of the web application in the prototype is the phpMyAdmin 3.3.2, which is used as a front-end to the database for direct editing. This is not the intended use case in a more polished product.

8 User Environment

The user environment is a generalization of how any given user interacts with the sensor network. The primary method of interaction with the system is through the web application described in section 8. This is an active form of interaction with the network as the user may configure settings or retrieve data (all through a web browser). The secondary method of interaction involves the user passively receiving email alerts from the alerting engine or even directly from a sensor controller. Email alerts are configurable settings through the web application.

The prototype allows the user to search for sensors around the world using the Google Maps API. It provides an intuitive interface for searching through the potentially large number of sensors. Once found, present and past measurements recorded by a sensor are viewed using a graphical chart that is rendered inside the web browser.

When a rule is triggered by the rule alerts component to which the user is subscribed, the user receives an email about the alert, including the details of the sensor which caused the alert to trigger.

9 Deliverables

As mentioned in the project objectives, this project consists of two deliverables:

The **architecture** provided in this report describes the complete sensor network, including a description of the interfaces between each component, and allowing others to build components which would interoperate with our system. Although the prototype is limited in scope, the architecture will support scaling up to an enterprise-level sensor network with potentially hundreds of locations and potentially thousands of sensors, assuming sufficient hardware load-balancing is used.

Our **prototype** system implements this architecture on a smaller scale, as described in this document. It consists of a single microcontroller-based sensor, and the software to run both the controller and data center. When the prototype components are combined correctly, it demonstrates a functional end-to-end system capable of measuring temperature, storing it for later display, and alerting users when the temperature exceeds the specified threshold.

10 Budget

With the exception of the hardware sensors, this project is entirely software-based. The software will be developed, executed, and tested on machines already owned by the university or members of the project team. The standards being implemented are open and public. Therefore, no expenses will be required for the software aspects of the project.

The total cost of purchased materials is **$358.62**. This includes 2 microcontrollers that can be used as sensors along with the required hardware, cables, and power supplies. See Budget for Sensor Network for complete inventory.

11 Testing and Verification

This section describes the methods used to test and verify both the architecture and prototype. Several tests have been changed from the versions specified in the Capabilities and Requirements Document, reflecting feedback from the Project Review Board. For example, the Crossbow sensor was not purchased, and therefore those tests were conducted with the physical and virtual sensors described above. Other tests were deemed unnecessary during the Progress Report Review Meeting.

11.1 DT-01 – Retrieve collected temperature data

Status:	**Implemented**
Capability Tested:	CA-01, CA-02, CA-03, CP-01, CP-02, CP-03
Materials needed:	Sensor Network Prototype, User Environment
Tester:	Typical sensor-monitoring user
Steps:	1. Log in to Web Application from User Environment. 2. Select the temperature sensor from list of available sensors. 3. Retrieve current temperature at sensor location. 4. Verify accuracy of temperature using a thermometer.
Test Results:	*The attached spreadsheet contains detailed temperature measurements from the prototype which have acceptable accuracy compared to actual measured values.*

11.2 AT-01 – Acceptance Test for Architecture

Status:	**Implemented, below specification**
Capability Tested:	RAF-01, RAF-02, RAF-03, RAF-04, RAN-01
Test Results:	Status of architecture-level claims:

a) Data measurement table does not refer to specific units. (RAF-01)
IMPLEMENTED *- Database stores measurements and units in separate tables.*

b) Data Warehouse does not distinguish between physical and virtual controllers. (RAF-02)
IMPLEMENTED *- Physical and virtual controllers use same underlying libraries to communicate with gateway.*

c) Interfaces between components are clearly defined. (RAF-03)
IMPLEMENTED *– Interfaces are loosely coupled as specified above.*

d) Hardware, vendor, and sensor-to-controller communication protocols are not defined. (RAF-04)
IMPLEMENTED *- No attempt has been made to standardize sensor-to-controller protocols besides for the particular sensors used in the prototype.*

e) Compliance with OGC Sensor Web Enablement Standards. (RAN-01).
PARTIALLY IMPLEMENTED *– Architecture specified OGC-compliant communication standards. Prototype uses simplified, non-fully-compliant version.*

11.3 AT-02 – Prototype Physical Sensors

Status:	**Implemented but Below Specifications**
Capability Tested:	RPF-01, RPN-01
Test Results:	*Only one physical sensor (discussed above) has currently been implemented. As discussed with the Project Review Board during the Winter Quarter Progress Report, one sensor was determined to be sufficient to demonstrate prototype.*

11.4 AT-03 – Detection of Unresponsive Sensor

Status:	**Implemented (untested)**
Capability Tested:	RPF-06
Test Results:	*The Rules Alert subsystem supports this type of rule alerting, but the Project Review Board and project group agreed that this test was not required.*

11.5 AT-04 – Virtual Location, Throughput, and Data Visualization

Status:	**Implemented**
Capability Tested:	RPF-02, RPF-03, RPF-04
Materials needed:	Virtual Location, Data Warehouse, User Environment
Testers:	Sensor network administrator
Steps:	1. Configure the Virtual location to simulate 100 sensors. 2. Connect the Virtual location to the Data Warehouse. 3. Configure the Virtual location to generate one hour's worth of data, sending one data point per sensor every second. *(Modified)* 4. Retrieve the data from one sensor as a temporal graph. 5. Retrieve the most recent values of each sensor and display them on a map.

11.6 AT-05 – Rule Alerting

Status:	**Implemented**
Capability Tested:	RPF-05
Test Results:	N/A
Steps:	1. Configure a rule specifying an alert threshold for the measurement. 2. Configure the Virtual location to generate a measurement above the threshold. 3. Verify the receipt of a detected rule condition as an alert.

12 Results and Conclusions

As a result of this project, the group has successfully designed a flexible and scalable architecture for a sensor network and demonstrated the key aspects of this architecture in a complete, end-to-end prototype. Numerous modifications and additional features *could* be implemented (for example, additional sensor types, improved user interface, and additional functionality through the web application). However, the prototype demonstrates a proof of concept containing the essential features of the architecture.

13 References

1. Sensor Network Project Design Proposal

2. Sensor Network Project Capabilities and Requirements

14 Attachments

A. Sensor Network Project Capabilities and Requirements
B. Sensor Network Budget
C. Temperature Verification Results