A Web Server for Sensors

YE TIAN

KTH Information and Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

A Web Server for Sensors

Ye Tian

2011-09-03

Examiner: Gerald Q. Maguire Jr.

KTH Royal Institute of Technology

Stockholm, Sweden

Abstract

This thesis describes the project “A new Web Server for sensors”. The project has created a demonstration web service that can receive data from sensors (both fixed sensors and movable sensors) and distribute the received information in the form of web pages. These web pages can provide forms that enable the user to enter commands which are to be given to sensors. The aim of this thesis project was to design and evaluate web-based application which could utilize sensor data. In this project, we focused on two aspects: (1) web access to sensors and (2) the potential mobility of sensors. The web server provides web server mediated access to the sensors. Additionally, this project examined how to integrate a sensor with a mobile device, such as a personal data appliance. The web server provides an easy access mechanism to users who want to use and control sensors. Those users can flexibly use their web browser to access to sensors through our web server. Moreover, a sensor could move, for example because it is integrated with a personal data appliance. The mobility of sensors increases the sensing scope of sensors because the sensors are not fixed in position. Such sensors can sense the environment along the path that they are moved.

To achieve the goals of this thesis project, we analyzed what are the basic parts and functions that should exist for sensors in the web server. Furthermore, the thesis analyzed how a sensor can be integrated with a personal data appliance, for instance, how to supply the power to sensor; and how to synchronize data between the sensor and personal data appliance. As a result of this project, a web server with some of the necessary functions was developed. An approach of how to integrate a sensor with a personal data appliance is specified in this thesis.

The thesis begins with an analysis of some existing solutions. Their advantages were used to specify the requirements for our own solution. The thesis describes the design and implementation of this proposed solution. Next the thesis describes the testing and evaluation of this solution in the context of this project. The thesis ends with some conclusions and suggests future work.

Key words: 6LoWPAN, sensor, mobile, web service, HTTP, UDP

Sammanfatting

Denna avhandling beskriver projektet "En ny webbserver för sensorer". Projektet har skapat en tjänst demonstration nätet som kan ta emot data från sensorer (både fasta sensorer och rörliga sensorer) och distribuera fått information i form av webbsidor.Dessa webbsidor kan ge former som gör det möjligt för användaren att skriva in kommandon som ges till sensorer. Syftet med detta examensarbete var att designa och utvärdera webbaserad applikation som kan använda sensordata. I detta projekt har vi fokuserat på två aspekter: (1) webb tillgång till sensorer och (2) de potentiella rörlighet av sensorer. Webbservern ger Web Access Server förmedlas till sensorer. Dessutom undersökte detta projekt hur man kan integrera en sensor med en mobil enhet, t.ex. en personuppgifter apparat. Webbservern ger en enkel tillgång mekanism för användare som vill använda och styra sensorer. Dessa användare kan flexibelt använda sin webbläsare för att få tillgång till sensorer via vår webbserver. Dessutom kan en sensor röra sig, exempelvis genom att den är integrerad med en personuppgifter apparat.Rörligheten av sensorer ökar avkänning omfattningen av sensorer eftersom sensorerna inte är fasta på plats. Sådana sensorer kan känna miljön längs den väg som de flyttas.

För att nå målen med denna avhandling projektet analyserade vi vad som är grundläggande delar och funktioner som bör finnas för sensorer i webbservern.Dessutom analyseras i avhandlingen hur en sensor kan integreras med en personuppgifterna apparat, till exempel, hur man levererar energi till sensorn, och hur man synkronisera data mellan sensorn och personuppgifter apparaten. Som ett resultat av detta projekt var en webbserver med några av de nödvändiga funktioner utvecklas.En strategi för hur man kan integrera en sensor med en personuppgifterna apparat som anges i denna avhandling.

Avhandlingen inleds med en analys av några befintliga lösningar. Deras fördelar har använts för att specificera kraven för vår egen lösning. Avhandlingen beskriver utformningen och genomförandet av den föreslagna lösningen. Nästa avhandlingen beskriver testning och utvärdering av denna lösning i samband med detta projekt.Avhandlingen avslutas med några slutsatser och föreslår framtida arbete.

Nyckelord: 6LoWPAN, sensor, mobile, web service, HTTP, UDP

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

6LoWPAN	IPv6 over Low power Wireless Personal Area Network
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
FCS	Frame Check Sequence
FFD	Full Function Device
GPIO	General Purpose Input Output
GTS	Guaranteed Time Slot
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
M2M	Machine to Machine
MAC	Medium Access Control
Mime	Multipurpose Internet Mail Extensions
PHY	Physical Layer
RFD	Reduced Function Device
REST	Representational State Transfer
RPC	Remote Procedure Call
RPL	Routing Protocol for Low Power and Lossy Networks
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
WADL	Web Application Description Language
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSDL	Web Services Description Language

1 Introduction

Today a combination of wired and wireless networks reaches nearly every place in the world. Additionally, wide area wireless networks have become popular all over the world. Personal area networks, especially wireless personal area networks, are an important area both commercially and as a research area. Such personal area networks are used for monitoring and controlling devices within a very local area. In addition to these personal networks which are used for interconnecting sensors on different parts of the body, in clothing, etc. wireless networks with somewhat longer range links are used for sensor systems include those in: healthcare centers, intelligent buildings, smart houses, and so on. In these systems, sensors are used to sense something in their environment and actuators are used to execute the commands of the system's owners (for example, to close a blind when there is too much sun coming into a room during a presentation). Facilitating access and control of these sensors and actuators is a key issue in these systems. One possibility is to create a web service for these sensors and actuators. Such a web service could be used by a user to view the information collected by sensors and to control actuators. Such a web service could provide tasks to collect data from many sensors and automatically control one or more actuators; for example, the user might set a desired temperature range, then the software would automatically adjusting the flow of air and heat to maintain the temperature at these sensors within this temperature range. The goal of this thesis project is to identify the basic parts of such a web service for a collection of sensors that are connected to the Internet via one or more gateways. For the remainder of this thesis we will assume that the 6LoWPAN (IPv6 over low power wireless personal networks) protocol is used to communication with these sensors and actuators.

A generic network diagram of the integrated sets of sensors and actuators together with their web service is shown in figure 1. This network diagram shows a (set of) personal area networks (PANs), Wireless Local Area Networks (W LANs), a wired intranet, and internet. A PAN interconnects sensors and actuators to a PAN gateway, thus the sensors and actuators connect to other networks through this PAN gateway. We assume that the WLAN is in turn connected to the intranet and internet by a switch or a router. As a result the sensors and actuators can be accessed (indirectly through the web service or directly) from a host connected to the internet or locally though the WLAN. Due to the potentially very large numbers of sensors and actuators, if these devices are to each be directly accessible to a host on the Internet, then it is essential that these devices utilize IPv6. Note that for policy reasons a given sensor or actuator might only be reachable from some specific Internet hosts or intranet hosts. Thus we separate the potential for communication (enabled by IPv6's addressability) from the ability to communicate (controlled by policy).

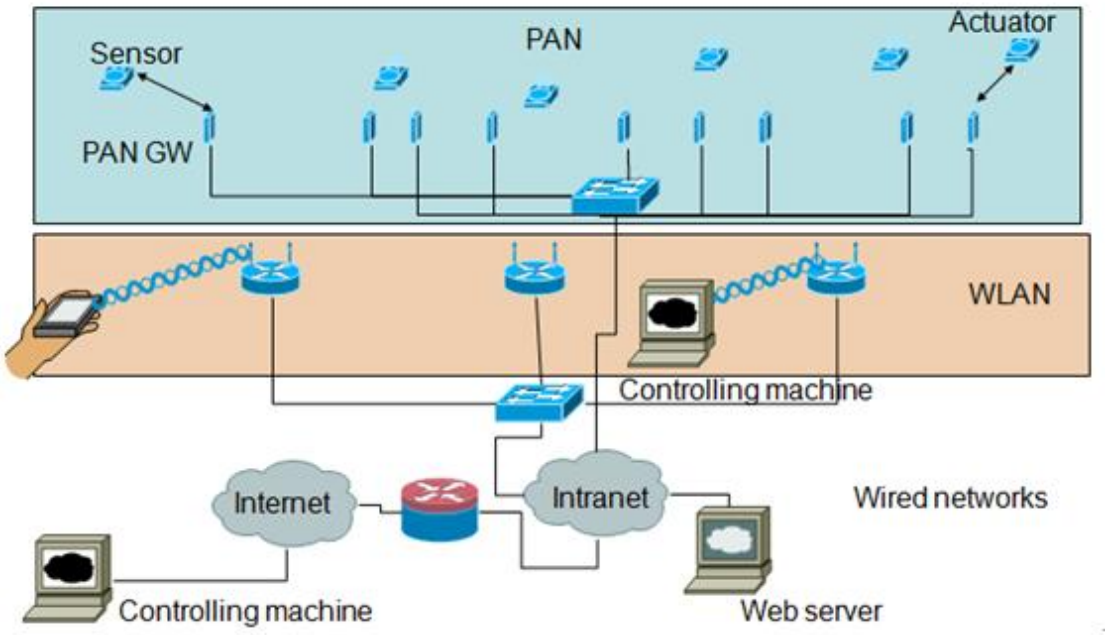


Figure 1: PAN, WLAN, and wired networks for sensing and control via a web service

Traditionally, a host with a specific program is used to control sensors and actuators. This program receives and processes the data from sensors and sends commands to actuators. This host can be placed in the same network as the sensors and actuators or it can be placed elsewhere, more generally this host could be connected via the internet. This approach of directly communicating with the sensors and actuators is a very efficient method to control sensors and actuators. However, there are limitations of this method. There are problems in scalability, mobility, and flexibility. One of the issues of scalability is the communications traffic between this host and all of the sensors and actuators - as IPv4 simply cannot address the numbers of devices that will be needed, this is addressed by utilizing IPv6. The host that is utilizing these sensors and actuators may change its point of network attachment; for example, due to the user's movement or to different users being responsible for the operation of these sensors and actuators at different times. Thus a solution should support mobile users and multiple users. With regard to flexibility, when using a single application if there are several different types of sensors and actuators, then the user(s) have to install multiple programs to control all of these sensors and actuators or these applications need to support plug-ins to dynamically extend the application to support new types of sensors and actuators. For these reasons, adopting a flexible and preferably standardized sensing and control methodology is very important. In later sections we will discuss the standards that we have considered.

Web services have had a huge success, especially in enterprise Machine to Machine (M2M) internet systems. Almost all internet users use one or more web services. Thus combining the concept of a web service with sensors and actuators

would result in a system that is readily accepted by users. Using a web service to control sensors and actuators is also more flexible than using fixed purpose applications. Using a web service, users can use their web browser to view and control sensors and actuators. Note that this approach enables IPv4 communication between the web browser and web server, while the web server(s) communicate via IPv6 with the sensors and actuators. Therefore this approach offers great flexibility to users and brings the benefits of sensors and actuators to a larger audience quickly (as it does not require IPv6 communication to be implemented and available in every user's device and all intermediate networks).

Although a combination of web services and sensors and actuators is a good solution, problems still remain. Many technologies have been developed to solve these problems. This thesis begins by providing some background information concerning typical web services, then focuses on the specific problems of web services for sensors and actuators. The thesis also provides relevant background information about the other technologies used for solving the remaining problems. I designed a web service based on these web service methods and the traditional control method. In addition to this web service, this project also developed a program that simulates fixed sensors and a mobile device with a sensor board (as integrating a sensor with a mobile device is another research aspect of this thesis), thus emulating a mobile sensor that communicates with the web server. The actual interfacing of a sensor to a mobile device (in this case a HP iPAQ personal data appliance) was done by Pontus Olvebrink in another project[1]. Further details of this are given in section 3.3.

My web server consists of two parts: (1) a receiving and decoding program for data from a sensor or actuator and (2) a web server that implements services for sensors and actuators. Processing begins when packets are received from sensors or actuators. The receiving and decoding program receives and decodes datagram contains application layer messages from the device. The server saves relevant information into a database. For this thesis project we have used a Microsoft SQL server 2005 database[2]. Subsequently the web server reads information from the database and generates suitable web pages. The user can control the sensors and actuators via the web server by using these web pages as their user interface to the sensors and actuators.

The structure of thesis is as follows: chapter 2 introduces the background necessary to understand the rest of this thesis. Chapter 3 presents a study and research about a sensor board. Chapter 4 describes the software implementation done during this thesis project. Chapter 5 presents the results, measurements collected during the project, and an evaluation of this data. The last chapter presents some conclusions and suggests future work.

2 Background

This chapter introduces the concepts needed to understand this thesis. This chapter gives an overview of the different technologies used during this thesis project.

2.1 Problems statement

As mentioned in chapter 1, a web service is a means to flexibly access sensors and actuators. However, problems still remain, especially for the sensors and actuators which are based on IPv6 over Low Power Wireless Personal Area Network (6LoWPAN) protocol[3] (see section 2.2). The first problem is that all web services are based on reliable session transport protocols, such as TCP[4], however 6LoWPAN uses UDP[5]. Another problem is 6LoWPAN emphasizes low power consumption by the sensor nodes. This is necessary because the sensors and actuators have to conserve power, as they are frequently battery operated. As a result the sensors and actuators typically use small packets size to transfer data in order to reduce their power consumption. As a result the size of these packets is very small; in fact, the size is only 127 bytes for a 6LoWPAN packet. These two problems limit the direct integration of web services with 6LoWPAN sensors and actuators. The thesis clearly describes these two problems in sections 2.2 and 2.3. In the section 2.3, the thesis introduces some solutions to solve these problems.

Another research area examined in this project concerns the mobility of sensors. Sensing technology and wireless and mobile communication technologies are becoming increasingly popular. An important trend is the merger of sensing and wireless communication technologies. Therefore, we have already also researched how to integrate a sensor with a mobile device. (Note in this research, the mobile device is a HP iPAQ Pocket PC h 5500 series personal digital appliance.) Because the sensor can connect to the web service via the wireless network of the attached mobile device, such an integrated sensor can combine web service, sensing technology, and wireless and mobile technology together to realize one or more applications. The result is a combination that can provide additional functions to web users, the sensor’s users, or other types of users.

2.2 6LoWPAN protocol

IPv6 over low power wireless personal area network (6LoWPAN) is a set of standards defined by the Internet Engineering Task Force (IETF) (in RFC4919 [[6]], RFC4944[7]). The 6LoWPAN protocol was created to facilitate IPv6 communication

by embedded Internet devices. The majority of such embedded Internet devices raise challenges such as following:

1. Power and duty-cycle problem: a battery-powered wireless device needs to utilize low duty cycles in order to operate for as long a lifetime as possible.
2. Lack of Multicast: Wireless embedded radio technologies such as IEEE 802.15.4, do not typically support multicast.
3. Limited bandwidth and frame size problems: Low power wireless embedded devices typically have limited bandwidths, thus in order to minimize their power consumption they utilize a small data frame size.

Figure 2 shows a comparison between the IPv6 protocol stack with 6LoWPAN and a typical IP protocol stack. 6LoWPAN can be viewed as a part of the data link layer. 6LoWPAN cooperates with and can operate over the IEEE 802.15.4 protocol. IEEE 802.15.4 defines both a physical layer and medium access control sub layer. The IEEE 802.15.4 standard will be introduced in next section.

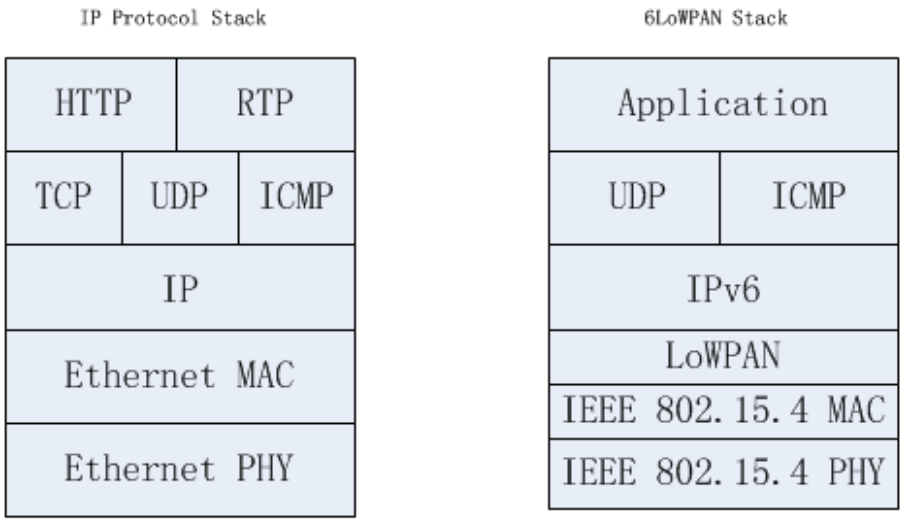


Figure 2: IP protocol stack and 6LoWPAN protocol stack

The most important feature of 6LoWPAN is that its maximum data frame size is only 127 bytes, thus it is much smaller than the IPv6 maximum transmission unit of 1280 bytes in common case (The jumbo packet length can be larger than 1500 bytes, but less than 9000 bytes). The small frame size reduces power consumption, thus saves battery-power. 6LoWPAN's features also include:

1. Support for both 64-bit and 16-bit IEEE 802.15.4 addressing
2. Useful with low-power link layers such as IEEE 802.15.4, narrowband ISM, and power-line communications
3. Efficient header compression of IPv6 base and extension headers, and UDP header

4. Network auto configuration using neighbor discovery
5. Unicast, multicast, and broadcast support, multicast packets are compressed and mapped to broadcast packets
6. Support for IP routing (e.g. IETF Routing Protocol for Low Power and Lossy Networks (RPL) [8])
7. Support of a link-layer mesh (e.g. IEEE 802.15.5[9])

2.2.1 IEEE 802.15.4

The first version of the IEEE 802.15.4[10] standard was released in January 2003 by the IEEE 802.15 task group 4. This standard defined a protocol by which compatible wireless devices could communicate by using low-data-rate, low power, and short-range radio frequency transmissions in a wireless personal area network (WPAN). The IEEE 802.15.4 standard focuses on WPAN. A WPAN differs from a WLAN in that the range of communication links with a WPAN are typically shorter than the maximum ranges of a WLAN and the maximum data rates of a WPAN are much lower than the maximum data rates of a WLAN. The IEEE 802.15.4 standard is optimized for the case of small, power-efficient, and inexpensive solutions.

The standard defines both a physical layer and medium access control (MAC) sub layer for low-data rate wireless connectivity with fixed, portable, and moving devices with no battery or very limited battery consumption requirements. The typical range of an IEEE 802.15.4 device is 10 meters. The IEEE 802.15.4 standard uses carrier sense multiple access with collision avoidance (CSMA-CA) as the medium access mechanism. Details of the various physical layers are outside the scope of this thesis.

Note that IEEE 802.15.4 includes two device types: full function device (FFD) and reduced function device (RFD). The FFD can operate in three modes serving as a PAN coordinator, coordinator, or an end device. Because such a device can support all three modes and all IEEE 802.15.4 functions, it is called a full function device. A RFD is intended for extremely simple applications such as a light switch or temperature sensor. RFDs do not need to send a large amount of data. FFDs can talks to RFDs or other FFDs. However, a RFD can talk only to FFD. Additionally a RFD may only associate with a single FFD at a time.

IEEE 802.15.4 defines 4 types of frames. These four types are: Beacon, Data, ACK, and MAC.

1. A beacon frame is used by a coordinator to transmit beacons
2. A data frame is used for all transfers of data
3. An acknowledgment (ACK) frame is used for confirming successful frame reception
4. A MAC command frame is used for handling all MAC peer entity control

transfers

2.2.2 6LoWPAN compression

The maximum 6LoWPAN frame size is 127 octets (IEEE 802.15.4 defines the frame size. This is a key limitation of the 6LoWPAN protocol) as 25 octets are necessary for the MAC headers. Additionally, there is an optional, but highly recommended for link layer security overhead, for example AES-CCM-128 [11] that would add 21 octets of overhead. This means if we apply such a security method, only 81 octets are available for the higher layers in a single IEEE 802.15.4 frame. Because 6LoWPAN uses IPv6, the IPv6 header requires at least 40 octets. The UDP protocol adds 8 octets of header. Finally, in the common case (without any security application), 53 octets are left for carrying the data. This is shown in figure 3 (note: the MAC header includes 21 octets of MAC header and 4 octets of frame check sequence, the L field is an 8 bit dispatch value). If we apply AES-CCM-128 encryption, then we only have 32 octets to carry the actual data. Due to the overhead of all these headers, performing header compression is compelling for 6LoWPAN. While header compression is desirable and even feasible in many cases, it is not required.

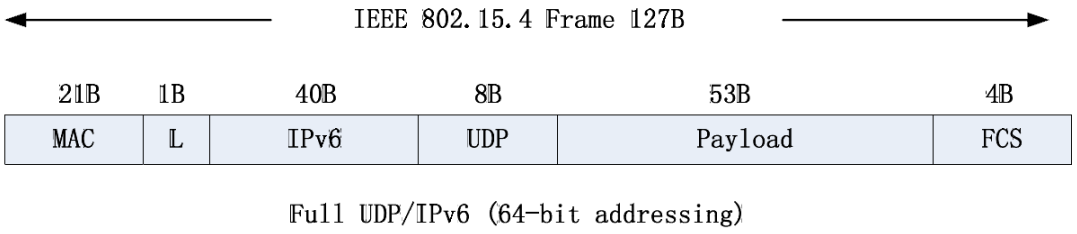


Figure 3: Basic format of 6LoWPAN packet

After header compression, the available payload can be 108 octets as shown in figure 4. However, this header compression can only be used **inside** a 6LoWPAN network. If the sensors send a packet to an external host, the PAN gateway of 6LoWPAN will expand this compressed packet to an uncompressed packet since the external host will not have sufficient information to understand the compressed header (another reason is that in order for the packet to reach an external host it will likely have to be forwarded by routers; these routers will also lack the information necessary to understand the compressed header).

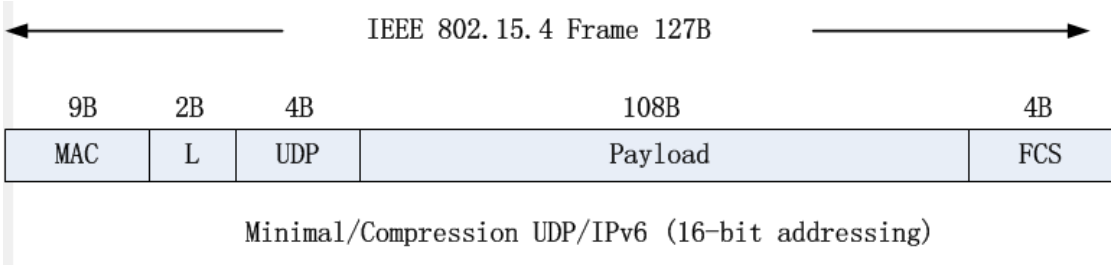


Figure 4: The minimal/compression format

The process of 6LoWPAN header compression is very complex. The thesis just briefly introduces the basic idea of 6LoWPAN header compression. The idea of 6LoWPAN header compression is that 6LoWPAN elides the commonly used fields as much as possible. For example, the header compression of IPv6 header field, the 6LoWPAN compresses the 64 bit network prefix for both source and destination addresses to one bit each when they carry the well-know link-local prefix. 6LoWPAN compresses the Next Header field to two bits when the packet uses UDP, or ICMP. 6LoWPAN also use the same approach to process the other header fields.

2.3 Typical web services for sensors and actuators

Today a typical web service is built upon the HTTP protocol. HTTP uses a reliable transport protocol such as TCP or SCTP as its transport protocol. In this thesis we assume the use of HTTP over TCP. Note that Transport Layer Security (TLS) can be used in conjunction with TCP to provide end-to-end security between the user's web browser and the web server. There two main types of web services: service-based web services (simple object access protocol (SOAP) model) [12] and resource-based web services (representational state transfer (REST) model)[13].

Service-based web services use XML following the SOAP format to implement remote procedure-calls (RPCs) between clients and (SOAP) servers. The web services description language (WSDL) [14] is used to describe the SOAP message. This model is popular for enterprise machine to machine systems. A SOAP interface (a Universal Resource Locator, URL) consists of a SOAP Uniform Resource Identifier (URI) that implements several RPCs (also called methods). An example of a SOAP URI is: http://sensor.com/test. This URI's methods could include:

getSensorStatus(sensorID)
getSensorValue(sensorID)
setConfig(parameter,value)

In this example the web server supporting http://sensor.com/test implements the three methods shown above. As a result the browser user can get the sensor's status or

value via this SOAP URI. Additionally, the browser user could set the value of a configuration parameter associated with this sensor.

In contrast to SOAP, the REST paradigm models objects as HTTP resources (a good analogy is a noun), each with a URL accessible using standard HTTP methods. The web application description language (WADL) [15] is used to describe REST. The REST model is normally used on the internet between websites (it is also used to provide dynamic web pages). The content of REST HTTP messages can be any Multipurpose Internet Mail Extensions (MIME) [16] content, although XML is common in machine to machine applications. Two examples are:

http://sensor.com/sensors/value

http://sesnor.com/sensors/sleeptime

In the REST model, the GET function is used on all resources to request the value, and the POST function is used to set the value. Figure 5 shows the structure of web services.

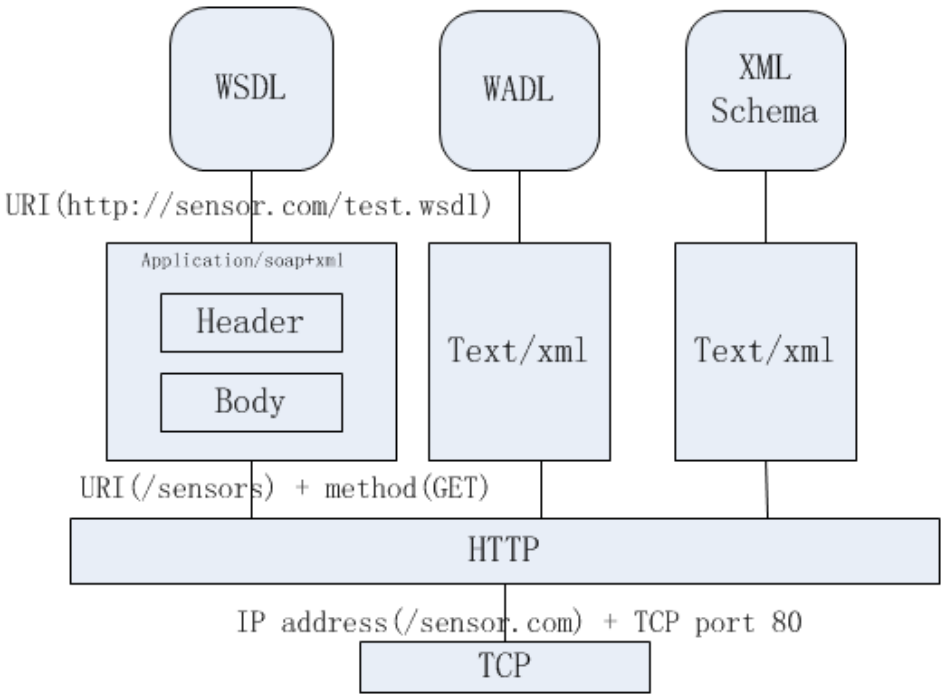


Figure 5: The structure of web services

As mentioned previously, in the SOAP model a URL defines a service (URI + method). SOAP uses XML to encode its header and body. In contrast, in the REST model, GET with a URL is used to access the resource. REST designs make use of well-know XML.

Using the above information, we can define a web services for sensors and actuators with 6LoWPAN.

The size of XML payload is large because XML has to describe the URI, the methods, and so on. An example of XML is:

```
POST    /sensorwebserver HTTP/1.1
Host: webserver.sensor.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 454

<?xml version=”1.0”?>
<soap:Envelope
xmlns:soap=”http://www.ws.org/2001/12/soap-envelope”
soap: encodingSytle=”http://www.ws.org/2001/12sopa-encoding”>

<soap:Body xmlns:m=”http://www.sensor.com/sopa.wsdl”>
      <m:GetSensor>
            <m:SensorID>0x1a</m:SensorID>
      </m:GetSensor>
</soap:Body>
</soap:Envelope>
```

Even this short example requires 454 bytes. However, in 6LoWPAN, the maximum size of a packet is 127 bytes. This means this XML message would require several fragments to transmit. To solve this problem, there are two approaches: a gateway approach and a compression approach (see next sections).

2.3.1 Gateway approach

In the gateway approach, a web service gateway is implemented at the edge of the 6LoWPAN. This gateway looks like a traditional host to hosts in the rest of the Internet. However, a proprietary protocol can be used between the gateway (which might be located at the web server) and the sensors and actuators. The gateway converts the content and control messages of these devices into a standard format. If the gateway is located at the web server the gateway could directly generate web pages. This gateway approach is widely used with non-IP wireless embedded network protocols, such as ZigBee[17]. One drawback of this approach is that a proprietary protocol is needed between the sensors and the gateway. This creates scalability and flexibility problems. There may be problems when a new type of sensor is added, since the existing gateways might not have support for this kind of sensor - again

requiring an upgrade of the affected gateways. Figure 6 shows this gateway approach. The sensors communicate with the wireless gateway. The gateway enables these sensors to indirectly communicate with a web server.

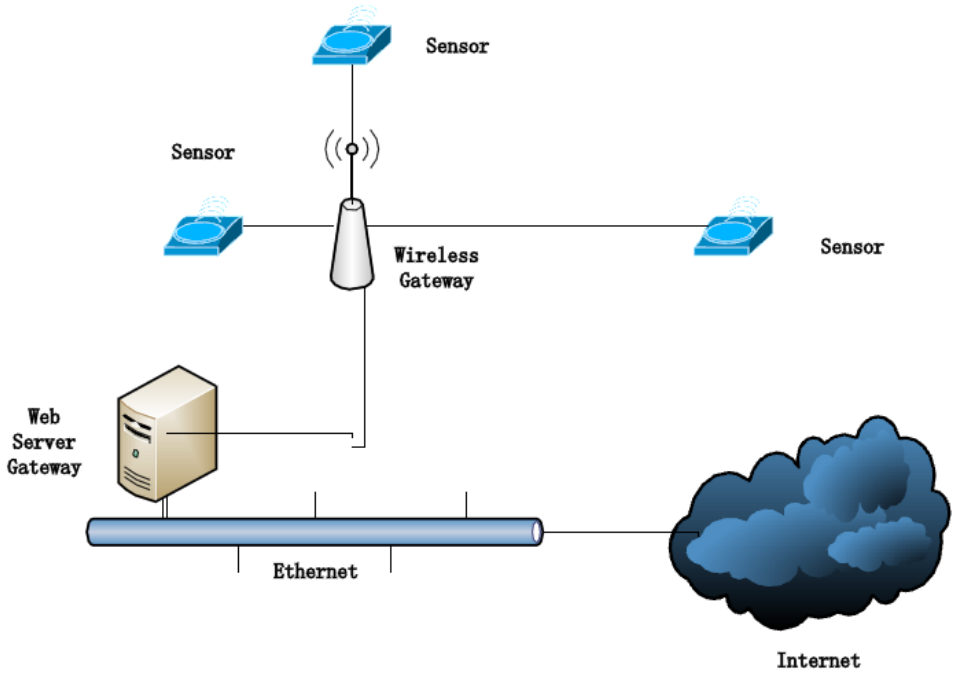


Figure 6: A diagram of web service gateway

2.3.2 Compression approach

In the compression approach, the web service format and protocols are compressed to a size suitable for using over 6LoWPAN. There are two alternative approaches: an end-to-end approach and a proxy approach. In the end-to-end approach, the compression/decompression happens in both server and client. In the proxy approach an intermediate node performs the compression/decompression transparently, thus the clients and servers can use standard web services protocols. Several technologies and standards exist for performing XML compression. Two of the most common are:

1. The WAP Binary XML (WBXML) [18]
2. Binary XML (BXML) [19]

In 2006, the W3C proposed standardization of the efficient XML interchange (EXI) format, which implements a compact binary encoding of XML[20]. In March 2011, the “Efficient XML interchange (EXI) Format 1.0” was released.

The compression approach does not require extra equipment, but requires that the

sensors and the server support the selected compression method. This means we there must be a compatible compression algorithm in the sensor and web server. However, this compression software might be a burden for the sensors, because the sensors (especially 6LoWPAN sensors) are designed to have low power consumption and the compression can require lots of computation. Another problem occurs on the server side, where the compression software is also needed. This approach is not flexible, since if we want to add a new server for the sensors, then we have to make sure that it implements all of the compression algorithms that have be used by the sensors.

2.3.3 Solution for HTTP and TCP

As mentioned in section 2.1, another problem of 6LoWPAN sensors is that all web services are based on a reliable session transport protocol, such as TCP, however 6LoWPAN uses UDP. The reason for using UDP in 6LoWPAN is that a request-response protocol such as HTTP would require that battery-operated, that wants to spend most of their time sleeping, be listening for requests. This incompatibility limits the direct integration of web services with 6LoWPAN sensors and actuators. Moreover, the gateway and compression approaches do not solve the HTTP and TCP problem. Hence we must recognize that HTTP and TCP are not suitable for use over 6LoWPAN. An alternative approach is illustrated by Sensinode's Nano Web Services (NanoWS) [21] that applies XML compression together with an efficient binary transfer protocol over UDP. This solution has been specifically designed by Sensinode for 6LoWPAN. An ideal long-term solution will be the standardization of a combination of XML (such as OpenGIS Sensor Model Language Encoding Standard (SensorML)[22]) with a binary encoding bound to a suitable UDP-based protocol.

2.4 Problem solutions

Considering the above problems, we designed a new architecture to provide web server for sensors and actuators. A network diagram is shown in figure 7. In this diagram, the sensors connect to the PAN gateway, and the gateway connects to a switch or router. The switch connects to a local area network and finally interconnects to an internet. As a result of this architecture we can put the web server anywhere. The only requirement is that if users outside of the intranet are going to access these web services, then the web server needs to have a connect to the internet. In order for the sensors to communicate with the web server the sensors send UDP datagrams to software running at the web server. This architecture is very simple. We do not need any extra equipment other than the PAN gateway which we assume will implement all of the gateway functions between the 6LoWPAN devices and the internet. We use UDP datagrams between the sensors and gateway; as this suits the case for 6LoWPAN

sensors since they use UDP. This architecture follows the gateway approach and locates the gateway at the PAN gateway. The web server can be located anywhere in the internet. Only one problem remains: how do the sensor and web server understand the content of datagrams that they will exchange? As the web server utilizes HTTP over TCP protocol it cannot process these sensor's UDP datagrams directly. In sections 2.3.2 and 2.3.3 we described several potential approaches. In our solution, we added a receiving and decoding program at the web server. This program processes the datagrams come from sensors and provides this information to the web server. This is similar to the compression approach, but rather than simply performing decompression this code can support a more general transformation from the received datagram to the format that will be used within the web service. Therefore, we need to install specific software in the web server to transform the messages from each different type of senor to the internal format used by the web service. However, this approach has the advantage that we do not need to make any changes in sensors. Unfortunately, this requires conversion software for each different type of sensor - which may be specific to the web service. This suggests that in the longer term that this software should probably output a standard format XML encoded message corresponding to each sensor's message - then only one type of software is needed per web service to convert this format into the internal format used by the specific web service.

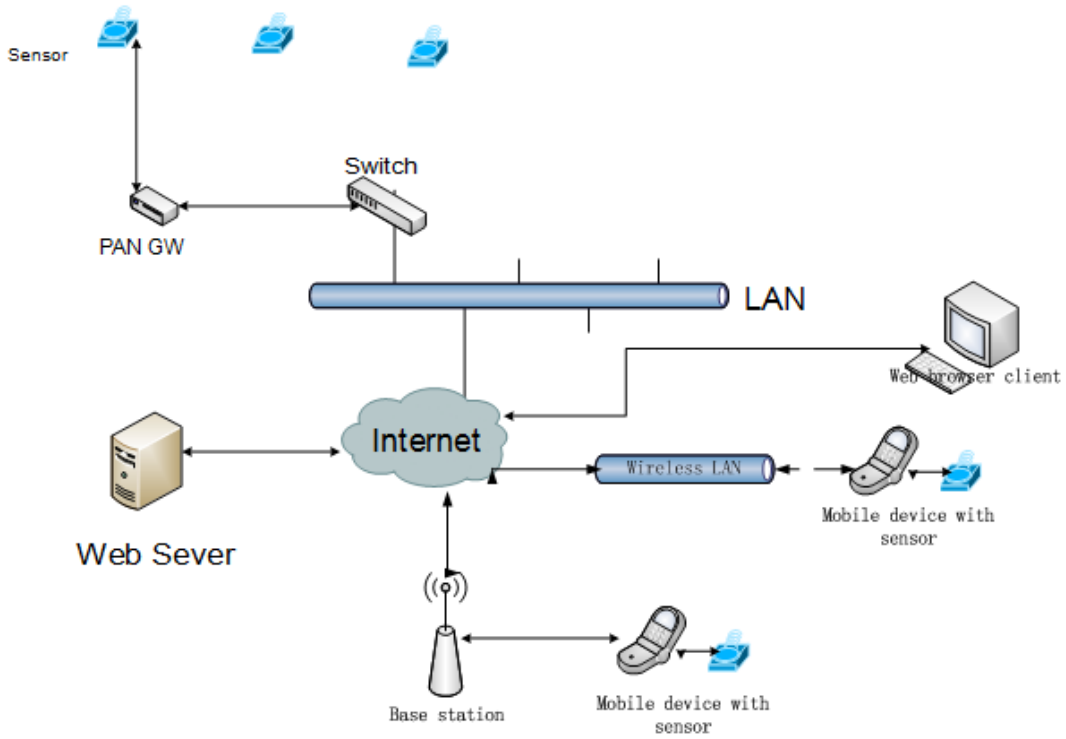


Figure 7: Architecture of the problem solution

As was stated in section 2.1, we also researched how to integrate a senor with a mobile device. We investigated this integration since we would like to enable the web service to utilize such movable sensors. Following the model used above this requires that we write a receiving and decoding program to take the sensor messages from these mobile devices and transform these messages into the internal format used by the web service. Thus our receiving and decoding program should also process these datagrams. In the figure 7, the movable sensors act like other sensors, but they can directly generate UDP over IPv6 packets (i.e., they do not have to be concerned with 6LoWPAN). The mobile device with sensor(s) can connect to the internet via WLAN or a wide area wireless network. Note that this software to emit sensors values in UDP over IPv6 packets can also be used to emulate 6LoWPAN sensors. This traffic can be used to test the web server, both for functional correctness and to understand how the server will perform given a specific load of sensor reports. In general, the project consists of three parts: (1) web server; (2) fixed sensors; and (3) movable sensors.

The thesis also compares the existing solutions and our project solution to see the advantages and disadvantages of these solutions. The comparison result is shown in table 1.

Table 1: Comparison of sensor access solutions

	Traditional access solution	Gateway approach	Compression approach	The solution of this thesis
Access approach	Users use a specific host to access sensors	Users access the web service gateway and get sensor information from web service gateway	Users access the web service and get sensor information from web service	Users access the web service and get sensor information from web service
Requirements for service	A specific program is needed for host.	A specific protocol	A compression method	A receiving and decoding program
Requirements for sensors	None	A specific protocol	A compression method	None
Requirements for users	A specific host	Web Brower	Web Brower	Web Brower
Communication between sensors and server	Based on the specific program	Based on the specific protocol	Based on TCP	Based on UDP
Extension for new type sensor	Require the specific program to change the configuration for new type sensors	Require the new sensor to apply the specific protocol	Require the new sensor to apply the compression method	Require the receiving and decoding program to change the configuration

2.5 Protocols in project

In this project, we have used UDP between the sensors and web server and have used HTTP and TCP protocol between the web server and web browser users. 6LoWPAN is used within the PAN network. As 6LoWPAN was described earlier in section 2.2, here we briefly introduce the other protocols.

2.5.1 UDP

UDP is a core part of the Internet Protocol Suite. It is a connectionless and unreliable transport layer protocol. UDP does not use handshaking; hence it does not by itself provide any reliability. UDP also does not implement flow or error control. UDP uses port numbers to multiplex and demultiplex data from the application layer.

2.5.2 TCP

The Transmission Control Protocol is a stream connection oriented and reliable transport protocol. TCP provides handshaking, flow control, and error control. These features are very well suited for transporting HTTP (see next subsection).

2.5.3 HTTP

The Hypertext Transfer Protocol (HTTP) [23] is an application level protocol for distributed, collaborative, hypermedia information. It is a request and response (also called client – server) protocol. A client sends a request to the server in the form of a request method, URI, and protocol version. The server responds with a status line including the message protocol version and a success or error code, potentially followed by data. This protocol is very popular as it is widely used by web browser users to view web pages.

2.6 Programming languages and tools

The thesis project involves a web server, movable sensors, and fixed sensors. In addition to the web server itself, on the web server side, we have developed a receiving and decoding program for the UDP messages from the sensors. This program was written using Java[24]. The programming language used is independent of the requirements of this program (beyond the fact that the programming language must be able to deal with UDP datagrams and XML), hence I simply choose my favorite programming language. The web server self was designed using the ASP.net [25]programming language. ASP.net was selected as it is a very popular programming language to develop a web site. The fixed sensors were being developed in another project at KTH (see [29]). However, these fixed sensors were not yet ready when I started this project. Therefore, I have written an application to simulate the fixed sensors. This simulator was written using C programming to facilitate it being combined with the code currently being developed for the 6LoWPAN gateway (see the thesis by Luis Maqueda Ara[26]). For the movable sensors, the sensors connect to the

mobile device via a serial interface, thus we developed a program that runs in the mobile device to read data from sensors via the serial interface and sends the sensor's value to the web server via UDP. This program was written using C # programming since the mobile device used in this project was a HP pocket PC. This HP pocket PC runs under Windows pocket 2003 operating system. C# and the Windows pocket 2003 OS were both developed by Microsoft.

To facilitate our development in these above programming languages, we have used the following tools:

1. Netbeans 6.9 for Java programming and
2. Microsoft Visual Studio for C, C #, and ASP.net programming.

In addition to the above components, we also need a database. In this project, we used Microsoft's SQL server 2005. To provide the web service we used Microsoft's Internet Information Service (IIS). IIS provides the run-environment for the web server.

2.7 Equipment utilized in this thesis project

We used a workstation as the web server, a mobile device, and some sensors in this project. The workstation was a common personal computer (specifically a Dell™ Studio XPS™ 1640 laptop with an Intel Core Duo P8600 CPU running at 2.4 GHz, with 4.0 GB of memory and a 100 Mbps Ethernet interface). Chapter three describes the sensors used in more detail. Therefore this section introduces only the mobile device.

A HP iPAQ Pocket PC h5500 PDA was used in this project. A picture of this device is shown in figure 8. This HP PDA runs Microsoft's Windows Pocket 2003 operating system. The iPAQ 5550 has a build-in Bluetooth interface and a build-in IEEE 802.11b WLAN interface. The Bluetooth interface can be used to connect to additional devices. The WLAN interface can detect nearby WLAN access points and be used for high speed wireless networking. The WLAN antenna is inside a rubberized bump on the top. These two wireless interfaces enable the HP iPAQ to access the internet via access points. This version of the HP iPAQ does not have a 3rd generation (3G) mobile network interface. However, this does not affect the results of the project as today there is little difference (from the point of maximum throughput) between a WLAN and a 3G network connection. The HP iPAQ has a function called "Automatically connect to non-preferred networks". This function will associate the HP iPAQ with at an access point, if there is at least one access point in the surrounding area.

Figure 8: HP iPAQ Pocket PC h550

The HP iPAQ also has a serial communication interface. This interface is used to connect to the sensors that were used in this project. Details of how the serial communication works are given in chapter 3. Table 2 presents the specifications of the HP iPAQ.

Table 2: HP iPAQ Pocket PC h5500 specifications

Processor	Intel XScale 400 MHz
Random Access Memory (RAM)	128MB SDRAM
Operating system	Microsoft Windows Pocket PC 2003
Read Only Memory (ROM)	48 MB ROM
SDIO slot	SD memory and SDIO card support
Expansion	PC Card with backup battery
Display	240 x 320 pixels, 64K-color support, TFT
Weight	206.8 g (without Expansion Pack)
Battery	Lithium Ion Polymer rechargeable. 1250mA
Networking	Wi-Fi 802.11b and Bluetooth

3 Sensor board

Sensing technology is widely used in many fields such as industrial monitoring systems, home and building automation, healthcare, defense, and real time environmental monitoring and forecasting. Today sensing is also becoming widely used in personal area networks, for instance, for personal health and fitness monitoring. Sensing technology has greatly improved in recent years in terms of physical size and weight, power consumption, reliability, cost, functionality, etc. Today a majority of sensors are wireless. These sensors use a wireless link to connect to each other, to a control host, or via a gateway to remote computers. A wide variety of sensors are used, for instance, a temperature sensor can be used to detect the temperature of the environment and the humidity sensor to measure the level of humidity. In addition to these types of sensors, many other sensors capable of measuring acceleration, light, magnetic field strength, pressure, and so on are being used in an increasing variety of application. In this thesis project, we use a temperature sensor embedded on a sensor board (a Wasa board). However, we do not directly use the Wasa board. Instead we used two improved sensor boards (based on the basic Wasa board design). This chapter introduces the WASA board and the two improved sensor boards. The thesis briefly describes how these boards work, and how they connect to the web server.

3.1 Wasa board

The Wasa board [27] is a growing family of open hardware and software embedded controller boards that are intended to allow people to extend existing computing platforms by adding sensors, radios, or other embedded control functionality with few constraints. The first Wasa board was designed by Professor Mark T. Smith of KTH. The Wasa board has some on-board sensors (for temperature and light level) and a number of General Purpose Input Output (GPIO) ports to enable students to easily add new sensors. The Wasa board uses a USB interface to emulate a serial port to connect the board to a computer, since most current laptops do not have a serial interface. Additionally, the USB interface also provides power to the board; therefore an external power supply is not need. Figure 9 shows a picture of a Wasa board.

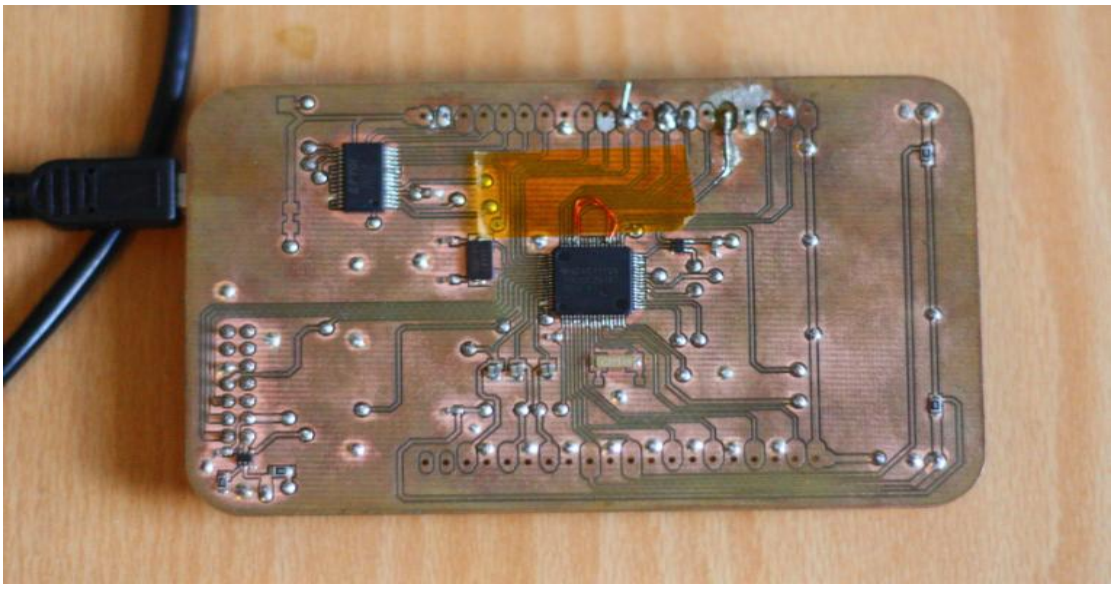


Figure 9: Wasa board

To manually operate the Wasa board, we can use a terminal program to communicate with the board via the USB interface. The Wasa board accepts and interprets AT commands[28]. A program running on the board processes these special AT commands that have been defined for this board.

3.2 6LoWPAN sensor board

The 6LoWPAN sensor board was designed by Joaqu ń Juan Toledo as part of his master's thesis project[29]. Luis Maqueda Ara has utilized a version of this board (without sensors) for his master's thesis project concerning creating a 6LoWPAN gateway. This 6LoWPAN gateway project has realized a 6LoWPAN gateway that connects 6LoWPAN sensors using IEEE 802.15.4 to the internet via an Ethernet interface. The 6LoWPAN sensor board was developed based on Wasa board. However, the new board adds a wireless interface (specifically IEEE 802.15.4) and uses the 6LoWPAN protocol to communicate with the gateway and adds a battery to provide power. With both of these additions the 6LoWPAN sensor board does not need to be connected to the power mains and can communicate its data via the wireless link. The 6LoWPAN sensor board retains the USB interface that can be used to load software into the microcontroller and for debugging. In my thesis project, a program emulates the 6LoWPAN sensor board since the 6LoWPAN sensor gateway project and sensors boards were only being developed when my thesis project started. At this time there are only a limited number of prototypes. However, the use of a simulator for the sensor nodes does not affect the results since the simulator produces UDP datagrams with data in the same data format as the 6LoWPAN sensors do (after being forwarded by the gateway).

3.3 Low Power RS -232 sensor board

The Low Power RS-232 sensor board was built by Pontus Olvebrink in his thesis project (see the board in figure 10). This Low Power RS-232 sensor board is also based on Wasa board. The difference is that the Low Power RS-232 sensor board connects via a RS-232 cable to a computer or a mobile device (in this case the HP iPAQ). An innovation of this board is that the board and its sensors are power by the RS-232 handshaking lines, thus the board gets its power from the attached computer. This sensor board demonstrated that is feasible to equip such a mobile device with a variety of sensors. In this section, the thesis introduces how the sensor board communicates via the serial interface, what problems occurred when developing the software to run on the HP iPAQ, and how to these problems were solved.

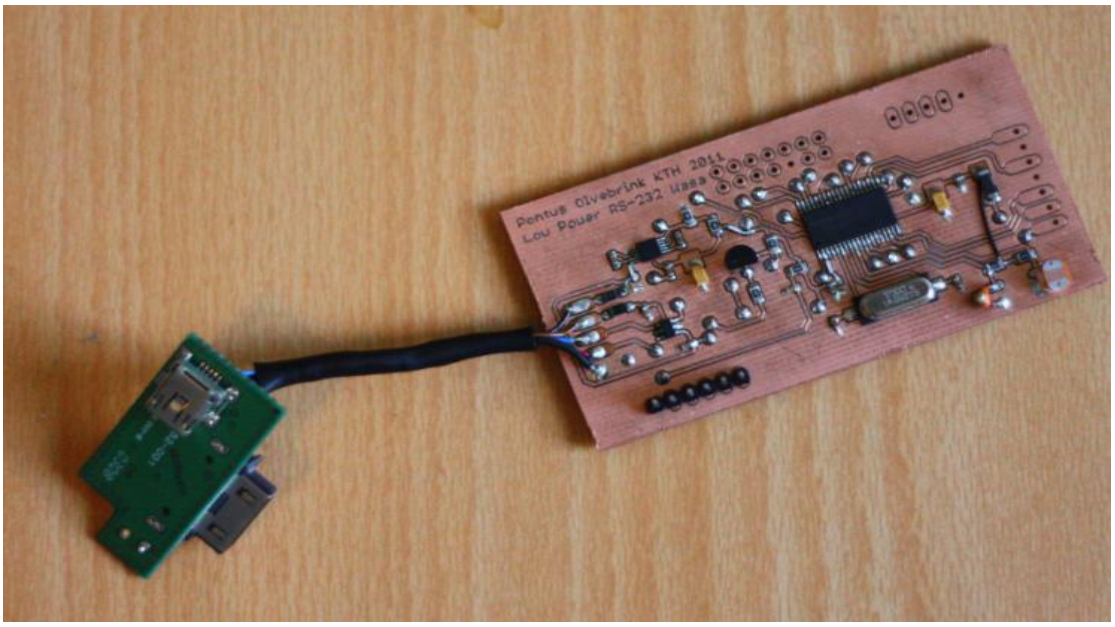


Figure 10: Low Power RS-232 sensor board

The Wasa board uses a Texas Instruments MSP430 [30] serial microcontroller. This microcontroller supports two Universal Asynchronous Receiver-Transmitter (UART) interfaces. Each UART interface has two pins: one pin is used to transmitter output data and the other pin is used to receive input data. The USB interface of Wasa board utilizes one UART interface to connect to the board to a computer. To connect the microcontroller with a HP iPAQ we use the other UART interface. Although, the HP iPAQ has both a USB interface and serial interface, we cannot use the USB interface because it was designed to be connected to a USB bus master (unfortunately, the HP iPAQ that we used only implements a USB slave interface). Additionally, the slave USB interface of the HP iPAQ cannot provide power to the sensor board. Therefore, we used the serial port of the HP iPAQ to connect to the second UART interface of the Wasa board. This raises two issues: (1) we have to ensure that the HP iPAQ has an ability to power the sensor board and (2) the signaling voltage of

MSP430 UART differs from the RS-232 signaling levels used by the HP iPAQ - hence we need to convert the voltage levels to the appropriate levels for the interfaces of the respective devices

3.3.1 Serial data exchange

RS-232 is a widely used serial communication standard. It uses voltages lower than (-5v) and higher than (+5v) levels in such a way that (-5v or lower means a logical 1 and +5v or greater means a logical 0). The logical representation is opposite the sign of the voltage levels. However, the UART interface in MSP430 microcontroller uses the 5v to represent a (1) and 0v to represent a (0). Therefore, we have to convert the voltages to the appropriate levels.

Several years earlier a KTH student, Alejanro Arcos clearly described this problem and presented a solution in his thesis[31]. A MAXIM MAX3241 transceiver [32] can do the voltage level shifting. The MAX3241 transceiver can run at 3.3 V (the same voltage as the MSP 430 microcontroller). We connect the UART interface of the MSP430 with the MAX3241 transceiver and some necessary capacitors such that when the output signals of MSP430 chip go through the MAX3241 transceiver, the voltage of the output signals will be shifting by MAX3241 transceiver from the microcontroller's UART voltage level to a RS-232 voltage level for the HP iPAQ. This final RS-232 output signal arrives at the HP iPAQ's RS-232 interface. Pontus Olvebrink's Low Power RS-232 sensor board uses the same design idea to solve this problem, although he used a MAX3188 transceiver. The different MAXIM transceivers support different transmission rates. Pontus also provides for the conversion from RS-232 levels to the voltage level used by the microcontroller's UART interface for the reverse direction. Details can be found in Pontus' thesis [1].

3.3.2 Power of sensor board

Arcos also determined the ability of the HP iPAQ to provide the power to a sensor board via the RS-232 handshaking lines. The Low Power RS-232 sensor board obtains power from the RS-232 interface of the HP iPAQ - although Pontus Olvebrink solves it using a different circuit design than that proposed by Alejanro Arcos. Details can be found in Pontus' thesis. However, the result is that we can use the HP iPAQ to power the sensor board via the RS-232 handshaking lines.

4 Software developed in this project

This chapter presents a description of all of the software developed in this thesis project. The thesis describes the functionalities of each of the different programs and how they fit into the complete design used in this thesis project. This chapter begins with a requirements analysis. A result of this requirements analysis is a list of the elementary functions that we have to implement.

4.1 Requirements analysis

The overall system architecture of this thesis project was shown in figure 7 in section 2.4. From that figure we can see that: the data come from sensors (6LoWPAN sensors or sensors connected to the HP iPAQ) and will later be presented to web browser users via pages provided by an internet connected web server. Additionally, the web browser user can use the web server to control the sensors. Based upon these requirements we extract the basic functions listed below:

The 6LoWPAN sensor (or the program emulating the sensor):

1. The sensor should send data to web server.
2. The sensor should receive commands from web server, execute these commands, and send the result to the web server.

The sensor connected with the HP iPAQ:

1. A program running on the HP iPAQ reads sensor values from sensor board via the serial port.
2. This program sends the sensor data to the web server.
3. The program running on the HP iPAQ should receive commands from web server, execute these commands, and send a reply to the web server.
4. The program running on the HP iPAQ must set up the serial interface appropriately.
5. The program running on the HP iPAQ encapsulates and dencapsulates its messages to the application running on the web server.

The web server:

1. A program is needed to receive sensor data from the above sources and save this information in a database.
2. The web server should provide a page to enable users to register via their web browser.
3. The web server should provide a login page to a web browser user
4. The web server should provide a search function to allow the web browser

user to search for a specific sensor's current information.

5. The web server should provide a search function to the web browser user to enable them to search for a specific sensor's historical information, i.e., the record of earlier sensor values and the time when each of these values were stored in the database.
6. The web server should provide a means to send commands to enable the web browser user to operate each specific sensor.
7. The web server should provide a logout function to enable a user to log out from the web service.

Besides the above basic required functions, this thesis project also provides an additional function that can download the sensor information history in an XML formated file when the web browser is viewing historical sensor information. The whole project was developed based on these requirements. Additional details will be given in later sections.

4.2 Program to emulate a 6LoWPAN sensor

For testing purposes a C program was written to send simulated 6LoWPAN sensor packets to a specific IP destination address and port. In my experiments I have used the IP address 213.100.19.167 and UDP port 9999. Another UDP port could be configured before the program runs. IPv4 was used because my experimental environment did not support IPv6. However the actual 6LoWPAN sensor uses IPv6. This program runs on a computer running the Linux operating system. The program was developed in C to facilitate it being combined with the code currently being developed for the 6LoWPAN gateway by Luis Maqueda Ara.

The program takes two arguments: a destination IP address and UDP port number. The program sends UDP packets to this destination from a randomly allocated UDP source port. Each UDP packet contains simulated 6LoWPAN sensor data.

As introduced in section 2.2.2, 6LoWPAN sensors can use a compressed header format within the sensor network. However, if the packet is sent to an external host, then the PAN gateway has to expand the compressed header producing an IP datagram. In this thesis project, the web server is assumed to be located external to the sensor network, thus when the web server receives packets from the 6LoWPAN sensors, these packets are not compressed. For this reason the sensor emulator simply sends a UDP datagram. Each of these UDP data contains less than 108 bytes of user payload, to ensure that the emulated packet could have originated from an IEEE 802.15.4 equipped 6LoWPAN sensor node.

Luis Maqueda Ara has designed and implemented the software for the 6LoWPAN

gateway and Sergio Floriano Sanchez has designed and implemented the software for the 6LoWPAN sensor board. Sergio Floriano Sanchez also defined the format of the 6LoWPAN packet used for this sensor board. Three packet formats are relevant here: Query, Order, and Report.

1. A query command (QUERY) is sent to the sensor to request sensor readings. It takes a variable number of parameters (0 to 4) specifying which sensors readings are requested. For example, a QUERY command asking for the temperature and battery readings would be as follows:

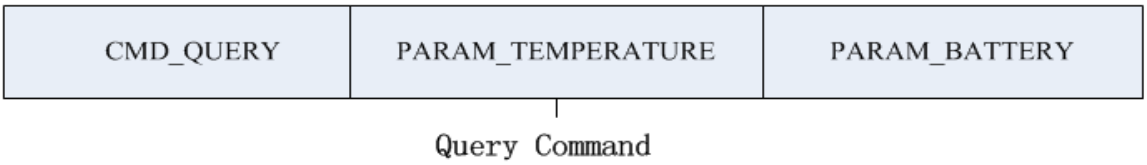


Figure 11: A Query command for both temperature and battery readings

For this command we send 057 in decimal, because in Sergio Toledo Sanchez's program, he defines CMD_QUERY as 0, PARAM_TEMPERATURE as 5, and PARAM_BATTERY as 7. The program running in the microcontroller on the sensor board parses this command and performs the request operation. Each of the commands and options has their own decimal code. Table 3 shows some of these definitions of these components (further details can be found in Sergio Floriano Sanchez's master's thesis[33]).

Table 3: Definitions of 6LoWPAN sensor message components

Name	Definition
CMD_QUERY	0
CMD_ORDER	1
CMD_REPORT	2
ORDER_LED_ON	0
ORDER_LED_OFF	1
PARAM_LED_GREEN	1
PARAM_LED_YELLOW	2
PARAM_LED_RED	3
PARAM_LIGHT	4
PARAM_TEMPERATURE	5
PARAM_HUMIDITY	6
PARAM_BATTERY	7

Notice that because these encoding are used in different fields, the values used to encode commands and parameters can reuse the same decimal values. The software running on the actual 6LoWPAN sensor board parsers and processes all of these messages. All of these definitions utilize a C char type, thus it is actually possible to

use characters other than '0' to '9' to encode commands and parameters.

2. An ORDER command works in a similar way, but it specifies the order in which the sensor should perform the commands. An example to turn the green light emitting diode (LED) is shown in Figure 12.

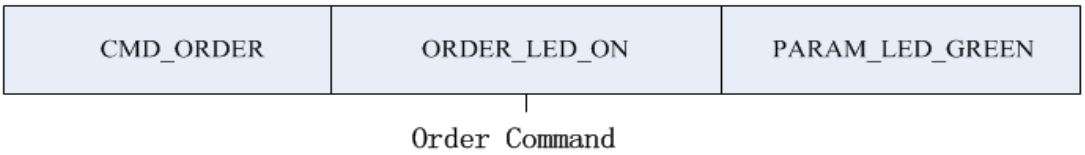


Figure 12: An Order command to turn on the board's green LED

3. A report message is used to report the values of sensor reading. This is used to reply to the Query command.

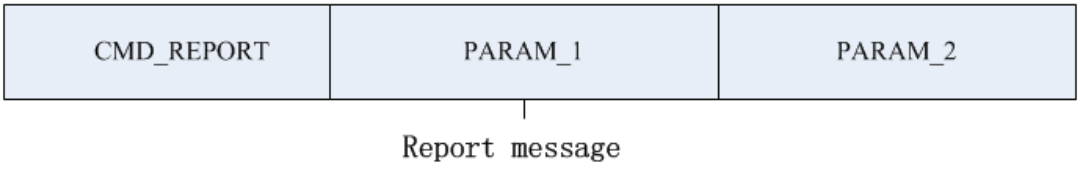


Figure 13: A generic report message example

These three types of user payload packet formats are used in the communication between the 6LoWPAN sensor board and web server. The program which emulates the 6LoWPAN sensor board starts by generating a CMD_REPORT message and sends it to the web server after processing processed the two arguments to the program (the destination IP address and destination UDP port number). This initial message is sent to inform the web server of the IP address and port number of the emulated sensor. When the program receives a CMD_QUERY command, it generates a simulated packet in reply. Notice that the third format "CMD_REPORT" can carry a maximum of 4 sensor readings. These four values have the same length (2 binary bytes). Therefore, the receiving and decoding program running on the web server side process each 2 byte binary value by (encoding and decoding these two bytes as four hexadecimal digits in little endian order. Note that the MSP430 microcontroller uses little endian order).

4.3 ComServer program running on the HP iPAQ

We call the program which runs on the HP iPAQ ComServer. The "com" indicates the serial port for serial communication, as we have used the serial port to connect the HP iPAQ and sensor board. This program reads sensor values from the Low Power RS-232 sensor board via the serial interface. This program also responds to requests for data from the web server and can order the sensor board to perform various operations based upon commands from the web server. The ComServer

program acts as a bridge connecting the sensor and the web server. This program was written in the C# programming language.

ComServer is more complex than the program that emulates a 6LoWPAN sensor because the ComServer program has to perform additional processing. First, the ComServer utilizes a graphical interface to enable the user to setup the serial communication and to enter the UDP port number and IP address of the program running on the web server. The serial communication configuration includes the serial port number, baud rate, number of data bits, parity, number of stop bits, and what form of flow control (if any) will be used. Figure 14 shows the graphic interface used to enter these settings.

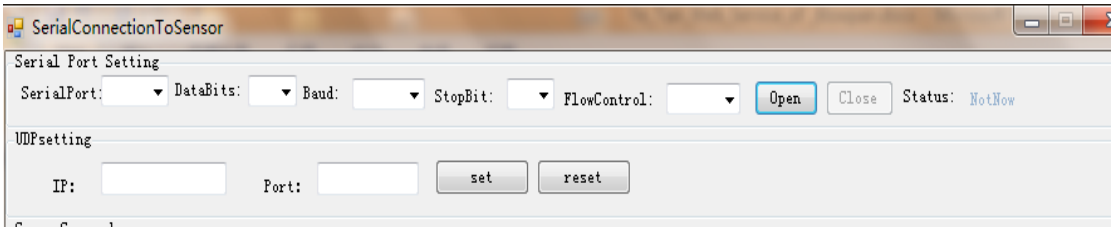


Figure 14: Serial port setting and UDP setting

In the serial port setting area, the SerialPort (pull down) field is used to set the serial port number. The program automatically checks for the serial ports of the device and shows the available port numbers in a menu list. If there is no available port, then a message will be displayed in the message box as shown in figure 15. The default number of data bits is 8. The default baud rate is 9600 baud. The number of stop bits is one. By default there is no flow control. For the Low Power RS-232 sensor board, we use the default settings except for the baud rate, the required baud rate for the Low Power RS-232 sensor board is 115200 baud. Notice when you have changed the serial port's settings you have to close the current serial connection and re-open the serial interface, otherwise the new settings will not be applied.

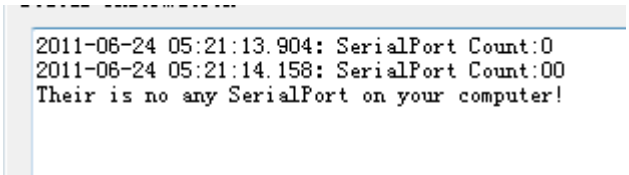


Figure 15: A status message indicating that there is no available serial port

When the sensor board is connected, the program will show messages in a status area as shown in figure 16.

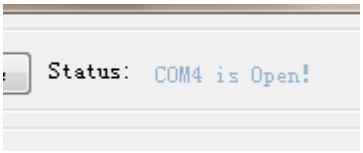


Figure 16: The ComServer status message indicating that the Com4 serial port is openned

The UDP setting is very simple. Simply input the IP address of web server and destination UDP port number in the associated fields, and then click the "set" button.

Next we can choose a sensor command to execute. In the ComServer, the program only defines two sensor commands (as examples). The first command checks the connection status of the sensor board. The second command reads sensor values from the sensor board (in our experiments the program reads the temperature sensor's values).

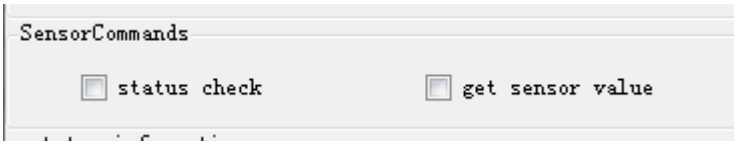


Figure 17: ComServer sensor commands

Because the Low Power RS-232 sensor board is based on the Wasa board it uses the same AT commands as same as the Wasa board. Thus the "status check" to check the connection status of sensor board simply sends the string AT<CR> (where <CR> is an abbreviation for the carriage return). If the connection is working, then the sensor board will reply "OK". If something is wrong with the connection, the program will either receive a reply message saying "ERROR" or the program will not get any response. To handle the last case we need to implement a timeout - so that after some period of time the program can know that the sensor board is not responding (for example if the sensor board is not plugged in). The "get sensor value" button is used to read the temperature sensor's value. The ComServer sends a command to a specific register (S- register 200, this logical register is used to operate the temperature sensor on the board). The resulting command to read the temperature sensor is: ATS200?<CR>.

Notice that all AT commands must begins with "AT". "S200" is the S-register number of MSP 430 microcontroller and a question mark is used to indicate a read request. Thus "S200?" means read the value of S-register 200. The reply to this command is: <CR><LF>DATA<CR><LF> <CR><LF>OK<CR><LF>.

Where <LF> is an abbreviation for 'line feed' and <CR> is an abbreviation for 'carriage return'; while DATA is a four digit hexadecimal value encoding the measured temperature from the temperature sensor. The final OK says that the

command completed without error.

Now that we have verified that all of the serial port settings are correct, we can click the start button to run the ComServer program, causing the program to enter a loop processing commands from the program running on the web server. The output from the program is displayed in the status information box (as shown in Figure 18).

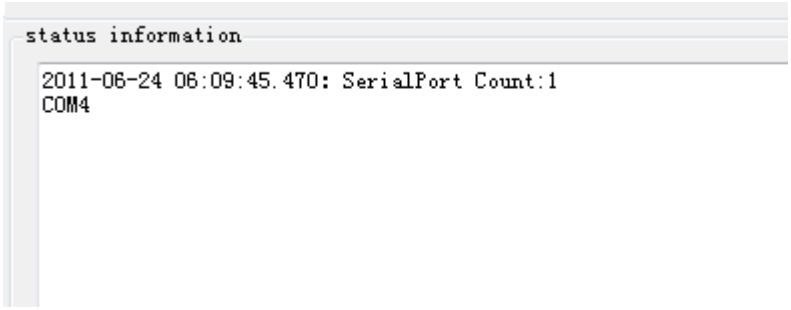


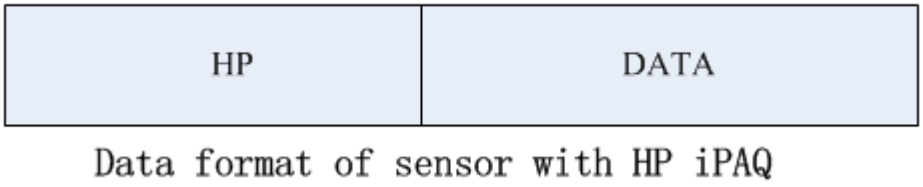
Figure 18: ComServer status information box

When the ComServer program starts, the program creates two threads one to process commands sent by the web server and the other to process commands via the graphical user interface.

1. Thread 1 opens a UDP connection; and listens to this UDP connection for command packets from the web server. If a command is received from the web server, then the thread executes this command and sends a reply to web server. (We assume for our tests that the command from the web server will be to read the temperature value from the sensor board.). The thread continues to listen to the UDP connection until a command comes or the program is terminated.
2. Thread 2 first checks what kind of command was made via the graphical user interface. If the user chooses “status check”, then the thread will execute this command once and show the result OK or ERROR in the status information box. After this the thread will be terminated. If the “get sensor value” command is chosen by user, then the thread will periodically execute the command (ATS200?). If the program successfully reads the sensor value, then the program will send this sensor value to the web server via a UDP packet. This assumes that the destination UDP port number and IP were successfully configured via the graphical user interface. In this case the program will continue to periodically get the temperature and send a UDP datagram with the latest value to the program running at the web server until the user terminates the ComServer program.

In our experiments we have used two different types of sensor sources. For this reason the receiving and decoding program running at the web server has to determine the format of the sensor packet which it receives. Because we only have two different

packet formats we add a specific element “HP” before the DATA to indicate that this data comes from the HP iPAQ. This format is shown in figure 19.



Data format of sensor with HP iPAQ

Figure 19: Data format of sensor coming from the HP iPAQ

4.4 Receiving and decoding program at the web server

The receiving and decoding program is written in Java. Java was selected because it is widely used and is my favorite programming language. However, this program could have been written in many other programming languages that supports receiving and sending UDP datagram. There is no need for a graphic interface for the receiving and decoding program. For the purposes of this project we have hardcoded the receiving UDP port number in the program. We do this because this receiving and decoding program will serve many clients and the UDP port number should be a fixed value, otherwise the clients will not know what UDP port number to send packets to. Note that this could be a well know port number, but I have not requested a port number assignment from the Internet Assigned Number Authority. The program receives and decodes packets sent by the sensors. As noted in the previous section, there are two different types of sensor sources that have been used in this project. In order for the program to determine the type of data source (i.e., a 6LoWPAN sensor board or the HP iPAQ) all of the UDP datagrams sent from the HP iPAQ include an "HP" element.

The program starts by opening a UDP socket to receive datagrams. Next the program enters a loop to receive and decode the received packet. For each packet, the program uses the Java API functions getAddress(), getPort(), and getLength() [34] to extract the IP address and UDP port number of sender, and the length of payload (respectively). The program parses the encapsulated payload from the sensor into several separate parts.

During parsing, the program first checks whether there is a “HP” element at in the start of the payload. If there is, then this packet comes from an HP iPAQ acting as a mobile sensor otherwise the packet comes from a 6LoWPAN sensor (or software emulating such a sensor). If this packet is from HP iPAQ, the program will parse and process the rest of the payload. If there is a DATA value, then this source's IP address, sensor ID, and value will be recorded in the database. If this packet is from a

6LoWPAN sensor, then the program will check to see whether it is a CMD_REPORT, if so then the program parses and decodes the data value. If the payload does not start with a CMD_REPORT, then the receiving and decoding program will discard this packet since no other types of packets are processed in the context of this thesis project. Note that there is a possibility that the packet might be a fake and malicious packet, however, currently the program does not make any attempt to validate the received packets, and hence this program may fail in unpredictable ways. Better handling of these types of packets should be addressed in future work.

Finally, the program saves the data values (and associated information) into the database. Details of the database will be introduced in next section.

4.5 Web server

This web server will collect and distribute information from 6LoWPAN sensors and Low Power RS-232 sensors connected to HP iPAQs. An authorized user can use their web browser to view a sensor's information. The user may also be able to control the selected sensor via the web server. This section introduces the web server's structure, the underlying database, the functionality of the prototype server, and so on.

The web server is written in ASP.net programming and developed using Microsoft Visual Studio. The following technologies were used: JAVA script, HTML, and Microsoft SQL server 2005. The run-time environment utilizes is Microsoft's IIS 6.0 with Microsoft SQL server 2005.

4.5.1 Structure

The structure of web server is the classic implementation pattern Model – View - Controller, frequently abbreviated MVC[35]. MVC is a software architecture that is a widely used architectural pattern in software engineering. A typical diagram presenting the Model, View, and Controller relationships is shown in figure 20.

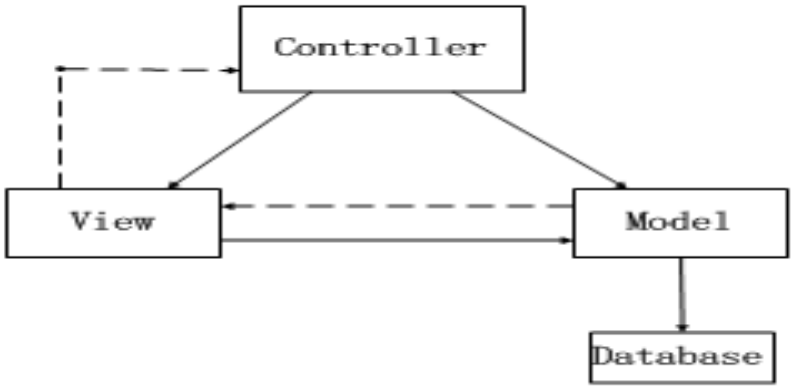


Figure 20: MVC model

The functions of those three components are:

1. "Models" in a MVC based application are the components of the application that are responsible for maintaining state. Often this state persists inside a database (for example: in this web server, we have a UserInfo.cs file that is used to represent web browser user information extracted from the database).
2. "Views" in a MVC based application are the components responsible for the application's user interface. The web browser user uses the View interface to communicate with the program.
3. "Controllers" in a MVC based application are the components responsible for handling end user interaction, manipulating the model, and ultimately choosing a view to render via the user interface. In a MVC application the view is only concerned with displaying information. It is the controller that handles and responds to user input and interaction.

More specifically for this thesis project, the Entity folder in web server represents the Model. This folder includes several files, including: UserInfo.cs, InviteInfor.cs, and Status.cs. The code in these files are used to manage the behavior of the application and data, to respond to requests for information about a sensor's state (usually from the view), and to respond to instructions to change state (usually from the controller). This code also manages the connection to database. The DataContext folder in the web server contains the controller programming files used to receive user input and initiate a response by calling model objects. Finally, the *.aspx files (such as default.aspx and main.aspx) are used to display the graphic interface and results to web browser users.

In addition to the above files and folder, in the web server, there is a Web.config file. This file contains parameters for the server, for example, the URI of the database

connection, along with the username and password that are required to access the database. These parameters are easier to modify if we place them in Web.config, rather than hard coding them in the software. The following sections of the thesis explain these files in details.

4.5.1.1 Views

This project uses *.aspx files to represent the graphic interface. There are five related *.aspx files used in the project.

1. Default.aspx is used to represent a login page to a web browser user if he or she is already registered as a user in the web server. To login to the web server, a username and password are required. If user fails to input either the username or password, this page will display a red colored notice to user. If the web browser user is not registered in the web server, the user can click the “Sign Up” link to switch to a registration page. This registration page is displayed by the Register.aspx file (described next). Figure 21 shows a screenshot for Default.aspx. Note that this screenshot was captured using Microsoft Visual Studio; hence the Default.aspx page shown in the figure is the developer's view and not the user's view of this page. The actual results of executing these *.aspx pages are shown in Chapter 5.

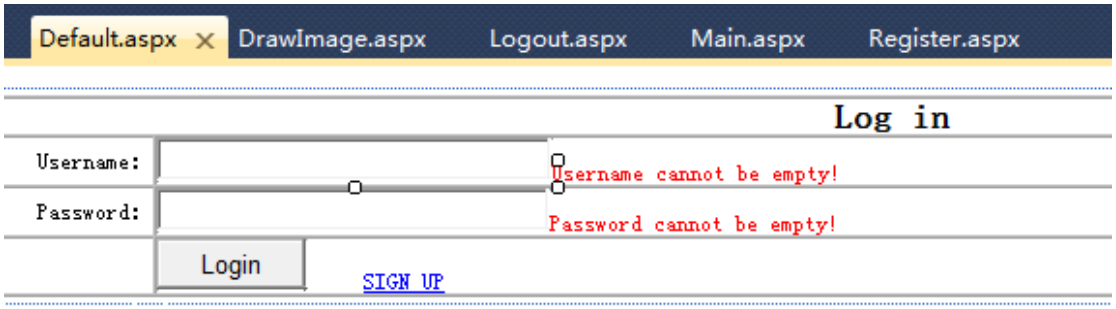


Figure 21: Defalut.aspx

.

2. Register.aspx enables unregistered web browser users to sign up via the web server to access sensor data. Registration information includes: username, password, repeat password, email address, and an Invited Code. Because the web server distributes sensor information and this information might be private (especially for the HP iPAQ user) we protect this privacy by requiring that each new user enter a valid Invited Code when registering. The Invited Code is assigned by the administrator of the web server. The administrator only gives these codes to people who should have access to sensor data via the web server. Using this Invited Code avoids un-authorized people registering with the web server. Figure 22 shows a screenshot of Register.aspx page.

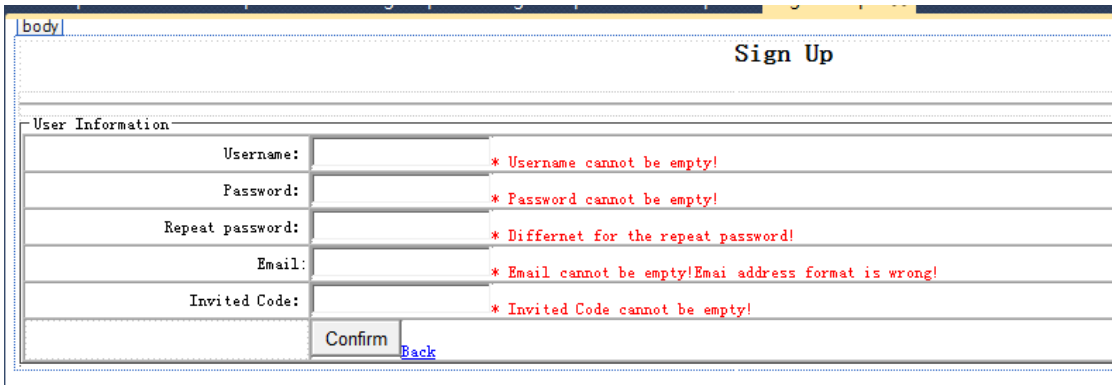


Figure 22: Register.aspx

3. Main.aspx generates the page that the web browser users when they successfully login. This Main.aspx relates to several other *.aspx pages, for instance, DrawImage.aspx and Logout.aspx. The Main.aspx works with these two files together. Figure 23 shows a screenshot of main.aspx.

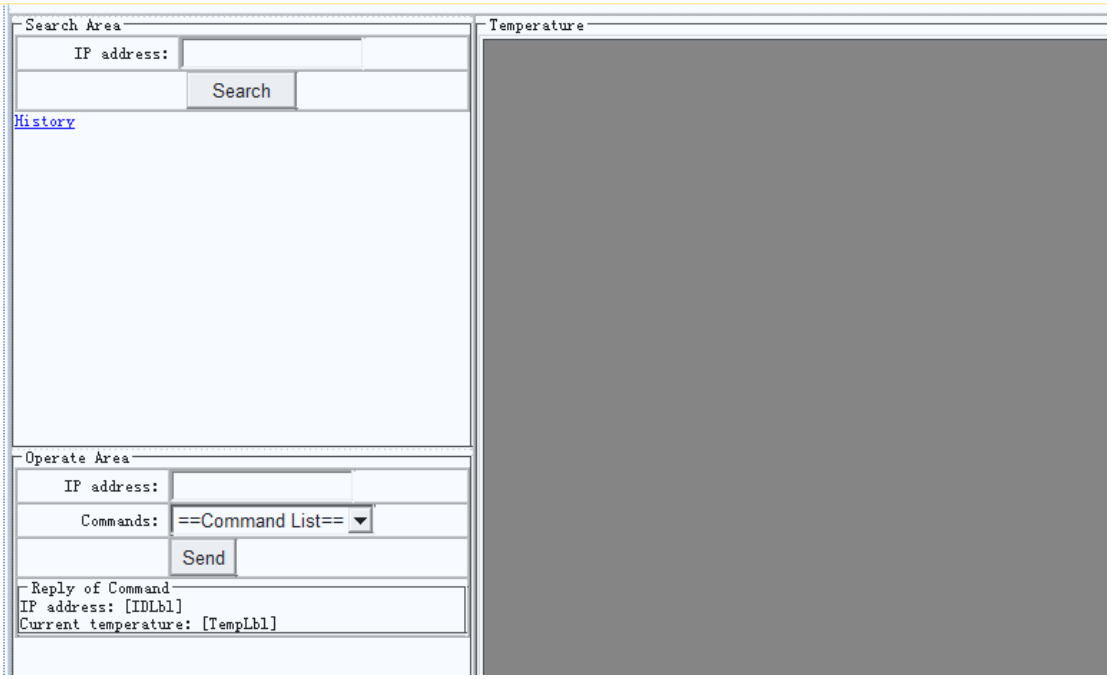


Figure 23: Main.aspx

Figure 23 on the left side at the top shows the username and a logout link which leads to Logout.aspx. Under the username, there is a search area. The users can input the IP address of a sensor to look for information from this sensor. If sensor values exist for IP address in the database, then Main.aspx presents these values on the right side. The display of results is generated by DrawImage.aspx. The label on the panel says "Temperature" because in this project we have assumed that we should display temperature values because we only have collected temperature values. If the search result is null, this means the target sensor does not exist, thus the Main.aspx

page will show a notice message of this to user. In the bottom of left side, there is an Operate Area that can be used to send a command to the sensor. In our project, we have only implemented one command (read the temperature value), hence the command list has only one entry; adding additional commands is straightforward. The user clicks the send button to send the command. Notice that because we have two types of sensors, when the send process starts, the web server will look for the sensor in the database to determine the type of the targeted sensor, and then it will send a suitable command to the sensor based on the type of sensor. The reply to this operation is displayed below the send button. Below the search button, there is link named "History". This link is used to switch to another web page that enables the user to search the history of records of a sensor. In contrast the main.aspx only presents the latest sensor values. The page history.aspx will be introduced in next.

Each time Main.aspx is viewed, it first checks that the user is in a valid session (i.e., that the user has logged in). If the user has not logged in then a notice asks the potential user to login, and then automatically switches to the Default.aspx page. Note that the Main.aspx page cannot be directly viewed by inputting a URI such as http://example.com/Main.aspx.

4. History.aspx provides a search function to enable users to search the history of values reported by a sensor. This page also provides a download to enable users to download the historical records of a sensor after successfully searching. The search template is shown in figure 24. There are three fields: IP address, Date from, and Date end. The IP address specifies the sensor (i.e., it acts as the sensor ID). While Date from and Date end define a time interval as a search condition.

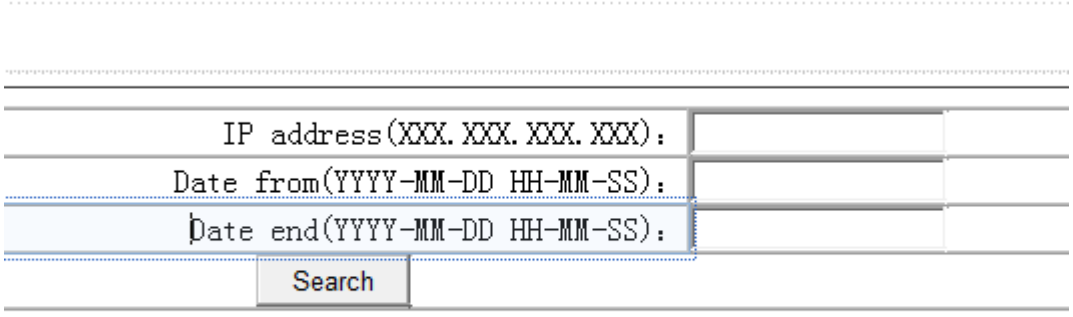


Figure 24: History.aspx

5. Logout.aspx is used to log out from the web server.

6. DrawImage.aspx is used to generate a picture to display the data information of a sensor.

Note that Logout.aspx and DarwImage.aspx do not work independently, but rather they have to work with the Main.aspx. The DrawImage.aspx will be blank by default and when there is no search result. Main.aspx, History.aspx, and DrawImage.aspx will ensure that the user is in a valid session (i.e., that the user has logged in) when they are viewed.

4.5.1.2 Model

As previously said, the Model in an MVC architecture is used to build a Data model. The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state. In this web server we have 5 data models.

1. The web browser user information data model uses UserInfo.cs to build the data model for web browser users. This data model contains the username, password, and email address of the web browser user.
2. The Invited code data model uses InvitedInfo.cs to build the data model for the Invited Code. In this prototype the length of the Invited Code is a four digit code.
3. Sensor information data model uses SensorInfo.cs to build the data model for a sensor. This model is used to process the sensor information, including the IP address, and UDP port number
4. The sensor type data model uses SensorType.cs and Type.cs to build the data model. This model computes the sensor source type (i.e. is the data from a 6LoWPAN sensor or Low Power RS-232 sensor attached to a HP iPAQ).
5. The sensor reading value data model uses SensorData.cs to build the data model. This model processes the received values from a sensor.

4.5.1.3 Controller

The controller programming files exist in the DataContext folder in the web server. There is a one-to-one correspondence between the controller and model. Thus for every model we have in the project we must have one controller. Because the controller is used to process the model and view parts in MVC architecture, then in this web server there are five controllers:

1. UserDataContext.cs is used in Register.aspx to add a user to the set of allowed users of this web server. Another function in the Register.aspx page

determines whether the username exists or not. If the username exists, then the user is asked to choose another username. UserDataContext.cs is also used to determine the login session of a user when the Main.aspx page is viewed. This controller generally works with web browser user information data model.

2. InvitedDataContext.cs is used to process the Invited Code (in the Invited code data model) when the user registers. It checks to see if this Invited Code exists in database. If this code exists, then it returns a true value to Register.aspx, otherwise it responds with a false value (Note that the user can only register successfully when the reply is true.). Once an Invited Code has been used, the InvitedDataContext.cs deletes this Invited Code from the database to avoid someone re-using this Invited code.
3. ServerInfoDataContext.cs is used to find the IP address of the sensor in the database when the user operates the sensors (i.e., works with the sensor information data model). This codes searches based upon the IP address of the sensor using the IP address value that the user input in the "Operate Area". If the IP address exists in the database, then the code sends the command to the specific sensor, otherwise sends a warning message to user to indicate that a sensor with this IP address does not exist.
4. SensorTypeDataContext.cs is used to determine the source type of sensors. When a user sends a command to a sensor, the web server first checks the sensor type (6LoWPAN or Low Power RS-232) to determine the specifics of the command that will be used.
5. SensorDataContext.cs is used to search the sensor reading values in the database according to the IP address that has been entered. It always works with the sensor data model. When the web browser user searches the latest sensor reading will be displayed in main.aspx or a history record in history.aspx.
6. Microsoft's SqlHelper.cs [36] is used to establish the connection to the Microsoft SQL Server 2005 database. SqlHelper.cs is not a component of the MVC architecture, but it is rather a general purpose C# class developed by Microsoft to utilize the SQL database. SqlHelper.cs gets the database username, password, and URI from Web.config and establishes an SQL connection to the database. This controller contains the code that is used to read, search, and save information from/to the database when there is a request. SqlHelper.cs is a general open source class to operate a database.

4.5.1.4 Configuration file

The configuration file does not belong to the MVC architecture but it is used to set some common parameters of the web server. Only one configuration file exists in the web server – Web.config. The Web.config file contains information about the

database: database connection URI, database name, username, and password for using database. In addition to the database related information, the Web.config file also defines some parameters about temperature. The temperature value of a sensor is presented as a picture on a web page. These temperature values are drawn on an X-Y plot. In operation, the receiving program will receive a large number of temperature values from a sensor. This means the web server has to show some or all of these values on a web page. However, showing more than some limited number of values is impractical due to the limited screen size of the web browser. Additionally older values might not be useful to user. Therefore, we added a displaying parameter in the Web.config file for the temperature value that limits the number of temperature values displayed in the plot. For instance, if we set the limits to 20, then the plot will only contain the latest 20 temperature values.

4.5.2 Database

Five database tables exist in the web server. These five databases are used to process three kinds information:

1. Sensor information is stored in three tables: SensorDataInfo, SensorInfo, and SensorTypeInfo. These three tables store all information about the sensors. They all use the Sensor's IP address as the key. These three tables are shown in Figure 25.
2. Browser User information is stored in one table: UserInfo. This table stores the register browser user's information. This table is shown in Figure 26.
3. The Invited code is stored in one table InviteInfo. This table stores the un-used invited codes.

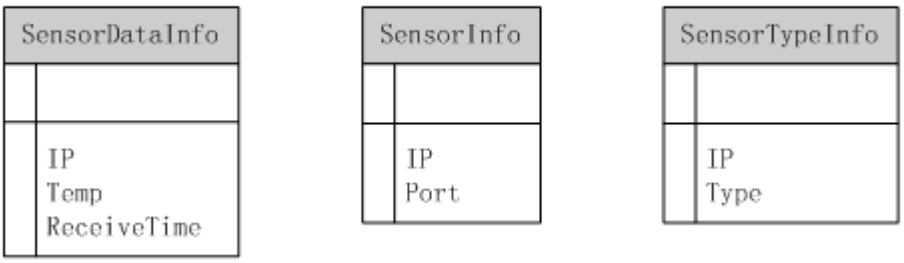


Figure 25: Three database tables for sensor information

Figure 26: The database table for browser registers user information

Figure 27: The database table for invited code

4.5.3 Functionalities

Sections 4.1 and 4.5 mentioned nearly all of the functions of the web server. There are seven functions in total: login, sign up, view current sensor information, view sensor history information, send commands, download history records, and logout. Figure 28 is a use-case diagram showing the relationships between of the functions.

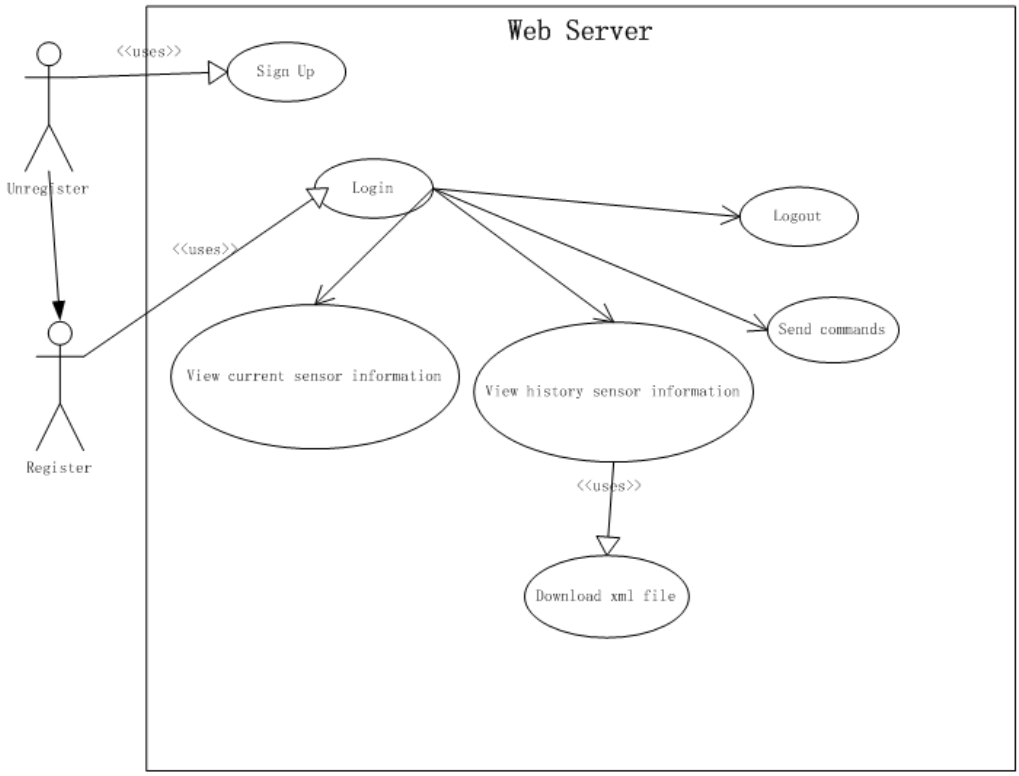


Figure 28: Use-case diagram for web server

It is easy to understand this use-case diagram. A normal browser user has to use the sign up function (with a valid invited code) to become a registered user of the web server. This invite code indicates that after authentication this user has been authorized to use the search function to view the current sensor information or history information (the historical records can be downloaded in an xml file). An authenticated user can also operate the sensor by sending a command. The logout function is used to safely leave the web server when the user wants to leave. A side effect of logging out is to invalidate the session information, so that no one else can hijack this session.

5 Results and measurements

This chapter presents the measurements and results of testing conducted during this project. These measurements were used to test the correctness of the operation of the different software components that were implemented. Additionally, these measurements were used to evaluate the performance of each of these software components.

5.1 Results of functional testing of the software

This section shows some screen dumps illustrating the correct functioning of each part of the whole project. These screen dumps should also to help the reader understand how this software meets the goals of this project. The project consists of four software components: the sensor simulator program (to emulate 6LoWPAN sensor), the ComServer program (to communicate with a Low Power RS232 sensor connected to a HP iPAQ), a receiving and decoding program, and the web server. Of these four parts, the simulator program and the receiving and decoding program do not have a graphical user interface. Therefore the following sections will focus on showing the ComServer program and web server. Later in the measurements section 5.2, the measurements and testing will also include the simulator program and receiving and decoding program.

5.1.1 ComServer program

Because I found it hard to capture screen dumps when the ComServer program was running on an HP iPAQ, I have captured screen dumps running this same software on a laptop computer running Microsoft Windows 7 connected with a Wasa board. Another reason is that Pontus who is the owner of the Low Power RS232 sensor was testing the sensor board a part of his thesis project at the same time as I was doing the measurement. Note that because the Low Power RS-232 sensor board and the Wasa board share the same command and response structure the program runs just the same as it would when run on the HP iPAQ connected to a Low Power RS-232 sensor board (with the only major difference being the use of a USB interface in the case of the Wasa board and the serial interface in the case of the Low Power RS232 board, but as mentioned in Chapter 3 the USB interface on Wasa board is to simulate the serial interface. Therefore the transmission rate of Wasa board is same as the Low Power RS232 sensor board).

Figure 29 shows the ComServer program executing the command “status check”. As described in section 4.3, if only the “status check” action is chosen by the user, then the program simply send the string “AT” to request the sensor board (in this case the Wasa board) to report its status. In this figure, the sensor board is connected and functioning, so it replies “OK” to the ComServer.

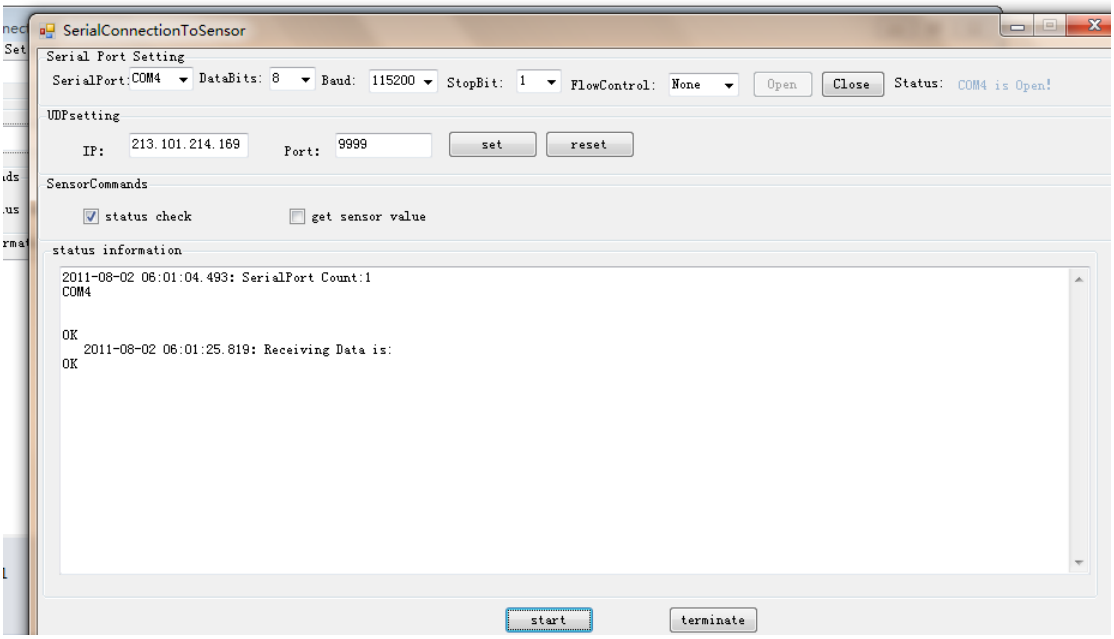


Figure 29: ComSever performing a “status check”

The figure 30 shows the ComProgram executing the command “get sensor value”. In this case, the program receives the sensor’s data from the sensor board via the serial port, and then transfers this data to the web server. The IP address and UDP port number were inputted in “UDPsettting” area.

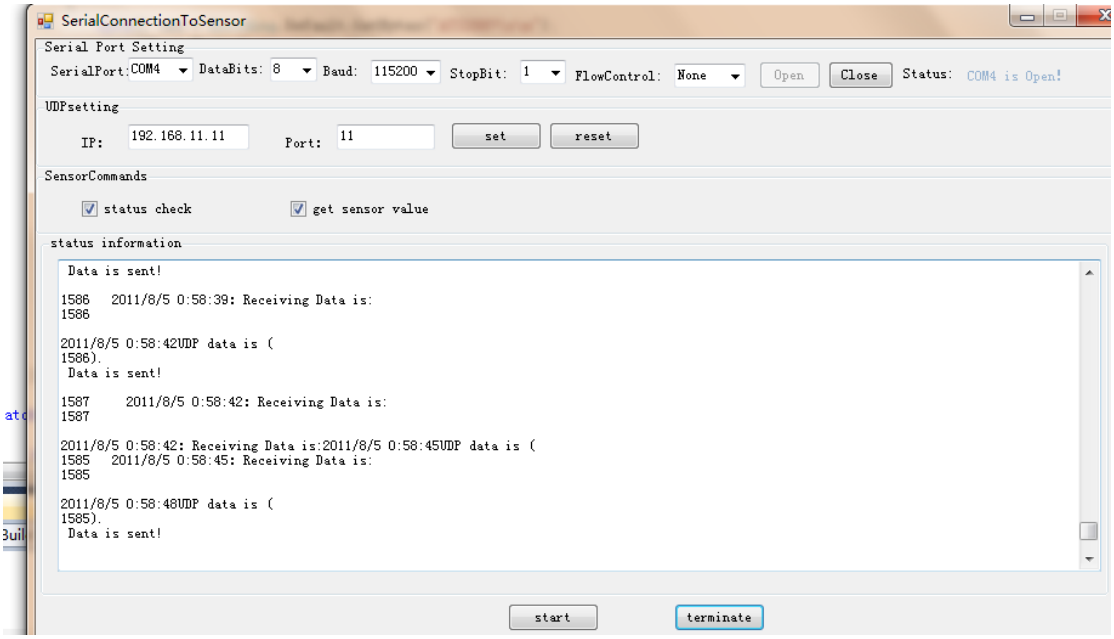


Figure 30: ComServer performing the command “get sensor value”

Figure 31 shows how the ComServer responds to receiving a command from the web server. In figure 31, we can see that the ComServer program received a command from web server, executed the command, and generated a reply to web server.

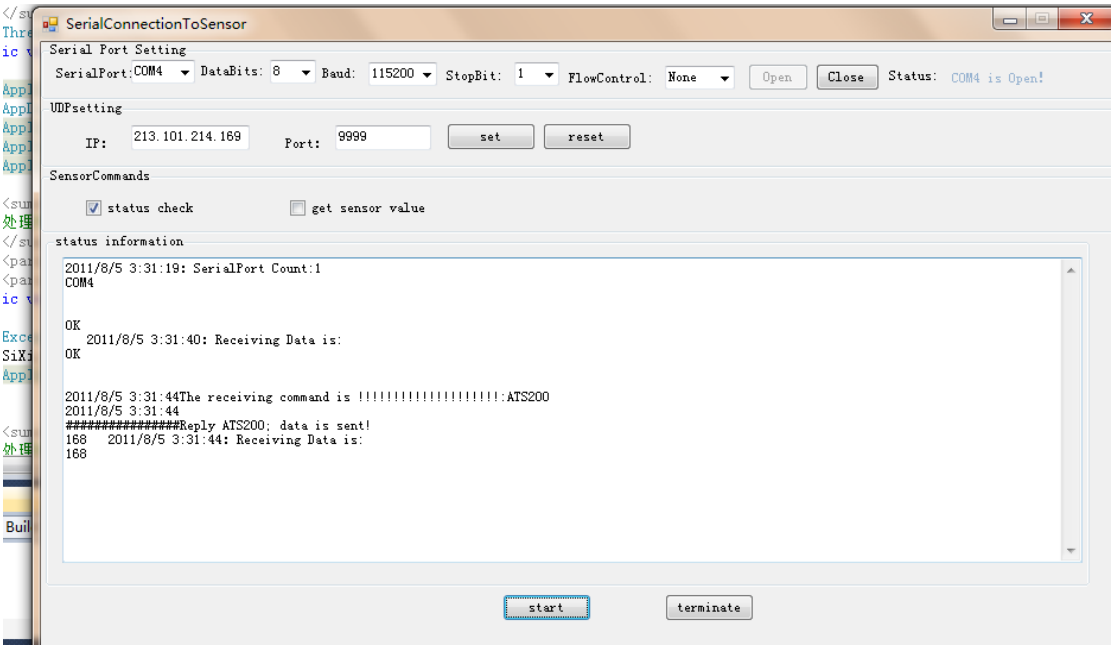


Figure 31: ComServer after receiving a command to read a sensor

If user wants to terminate the ComServer, the user simply presses the terminate button (as shown in figure 32). If the user wants to restart the program, the user has to close the serial port and open again, and then press the start button to restart the program.

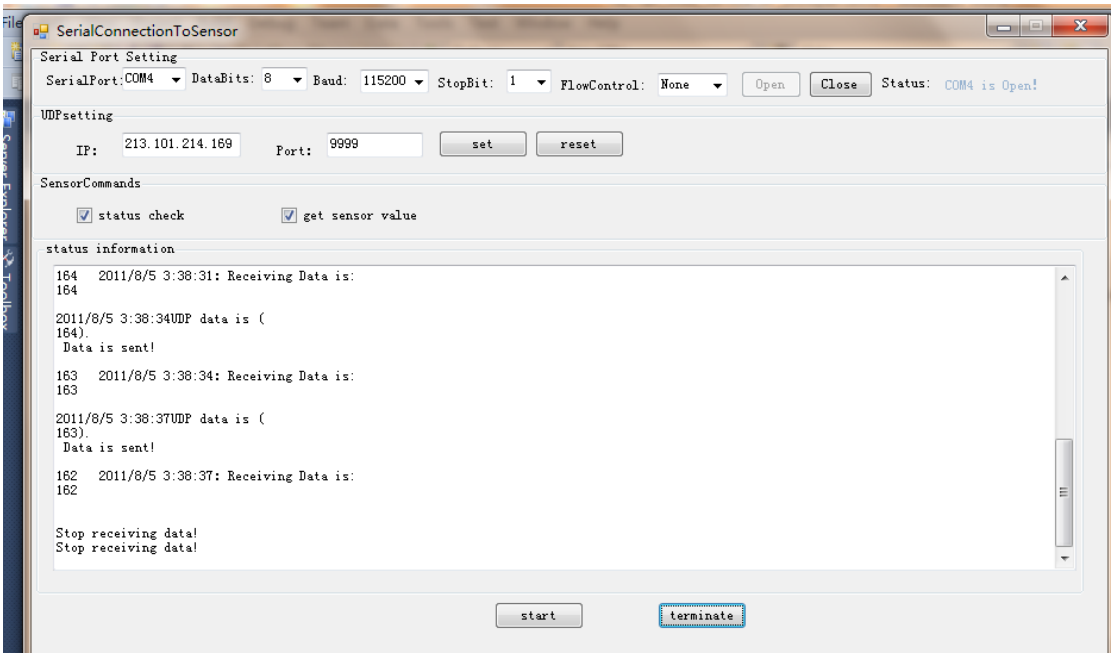


Figure 32: ComServer is terminated

5.1.2 Web server

This section shows screen dumps of user interactions with the web server. These screen dumps include all the situations that a browser user might experience.

A browser user wanting to use our web server first has to log in firstly. Figures 33 and 34 show some situations that a user might face during the login process. Figure 33 shows the situation if either (or both) username or password is not entered by the user. Figure 34 shows what happens when the username or password is wrong. The web server always displays a message to indicate what the problem is.

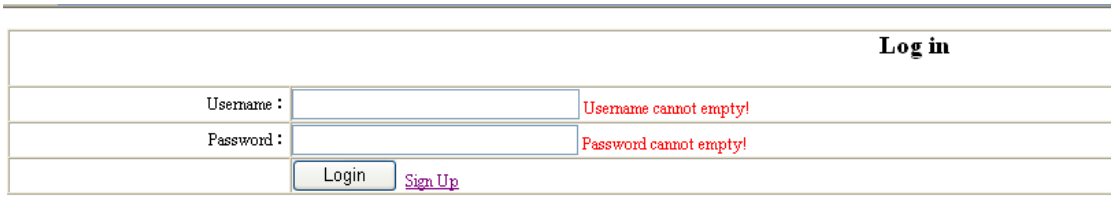


Figure 33: Login error with username or password is empty

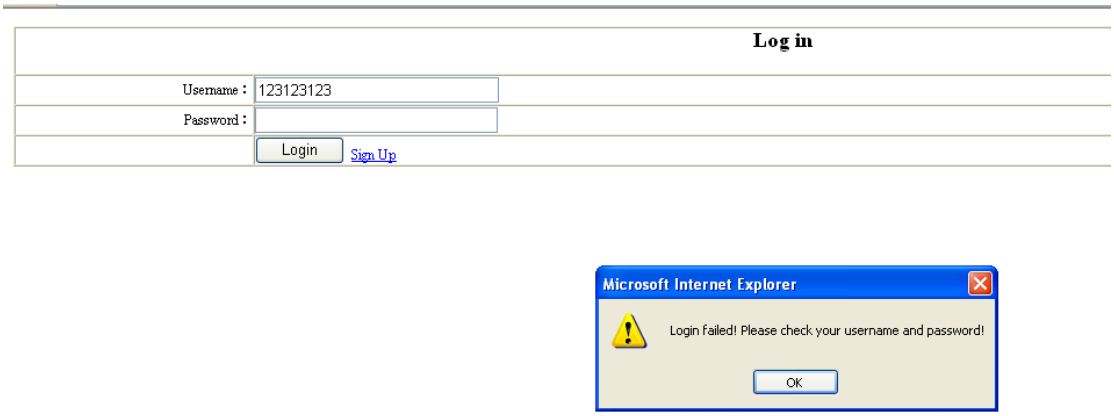


Figure 34: Login error with username or password is wrong

If the browser user is not registered with our web server, the user must first sign up to gain access to our web server. Figures 35, 36, and 37 show some examples of sign up errors. Figure 38 shows what happens when this sign up is successful. The error in figure 35 shows that the user forgot to input one or more values that are necessary in order to successfully register. The error in figure 36 is that the user entered an inconsistent password when he repeating the password. Figure 37 indicates that the user entered an incorrect invited code.

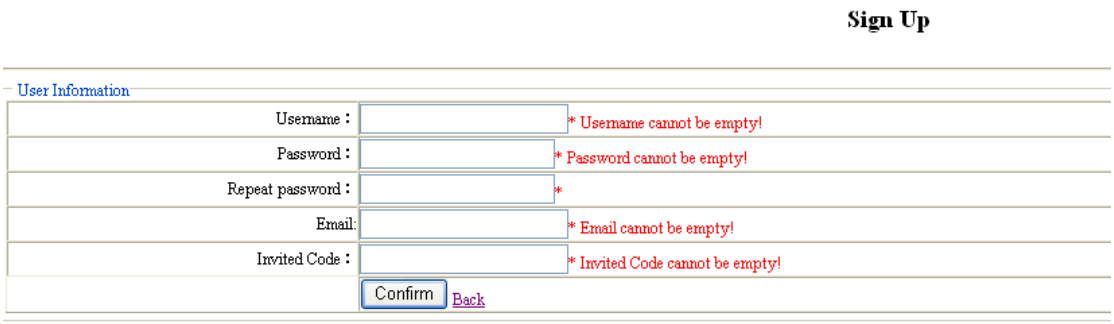


Figure 35: Sign up error with empty parameters

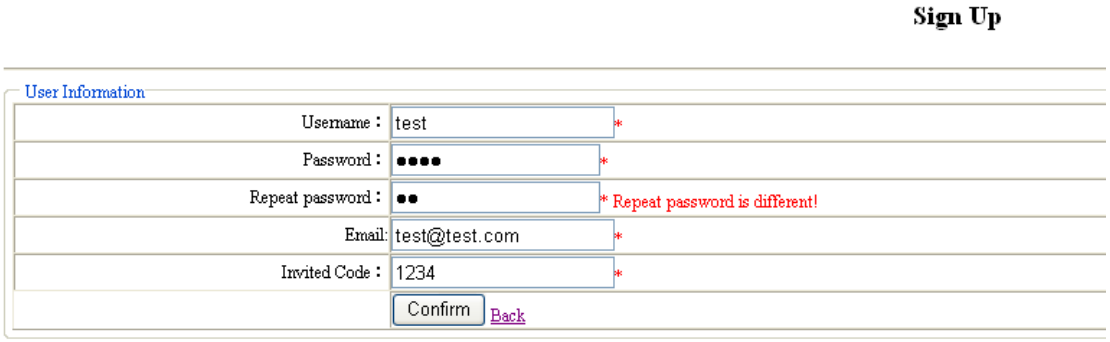


Figure 36: Sign up error with repeat password error

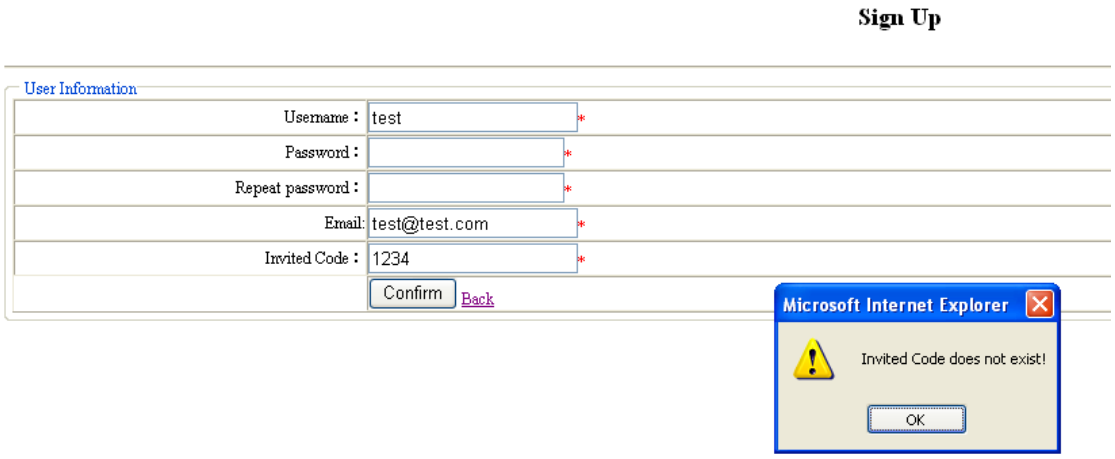


Figure 37: Sign up error with invalid invited code

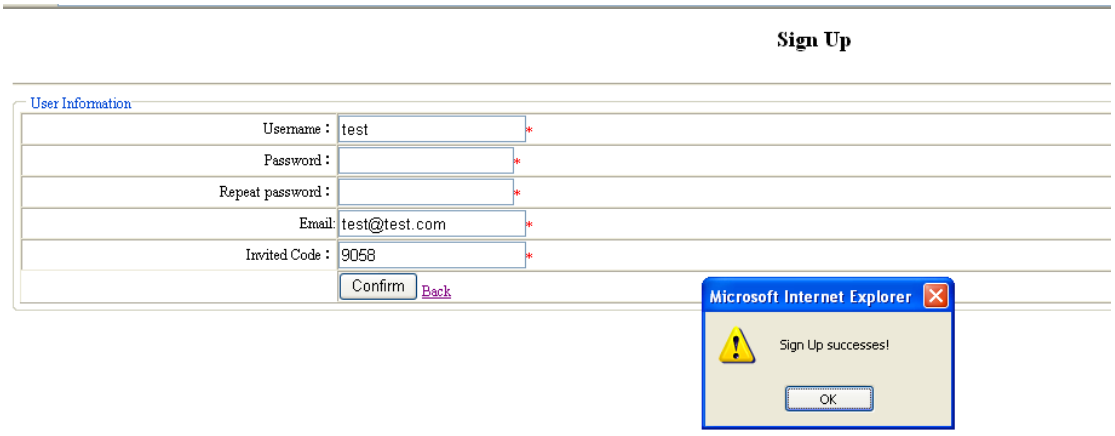


Figure 38: Sign up successes

Figure 39 displays an initial page after the user successfully logs in. The temperature area is blank because there is not yet any data to present. In figure 40, if a user presses the search button in the left top without inputting any IP address, then the web server will display a notice to user. If the user presses the send button in left bottom without inputting an IP address, the web server also displays a notice reminding the user of the information that needs to be specified. If the user does not choose a command, the web server will also display a notice as shown in figure 41.

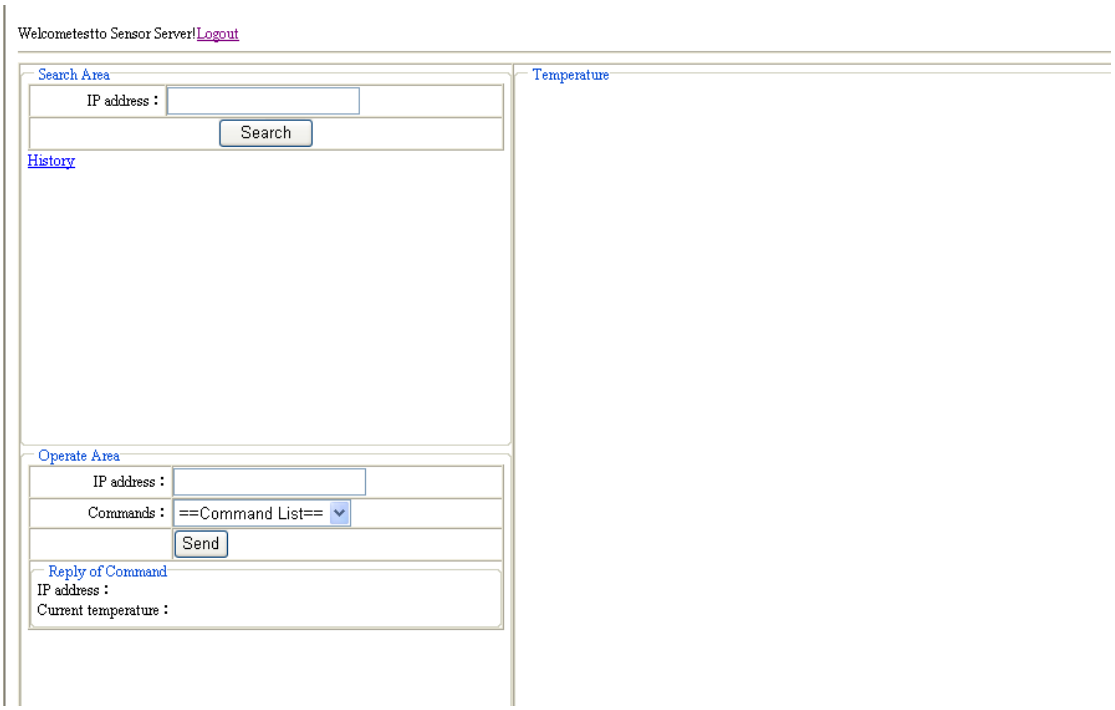


Figure 39: Initial main.aspx page of web server

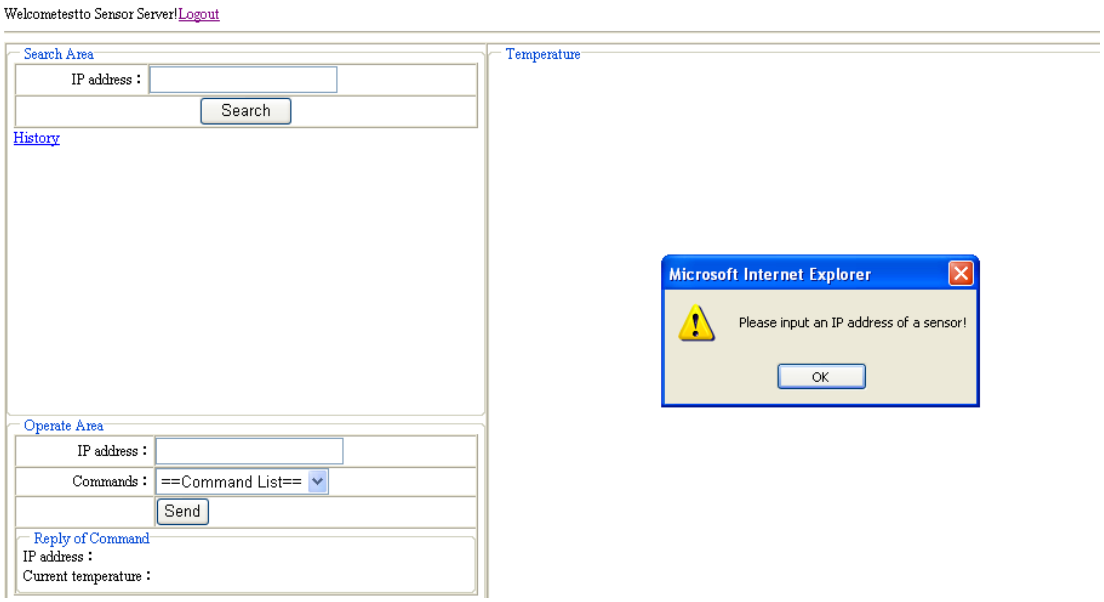


Figure 40: A notice message of web server to user

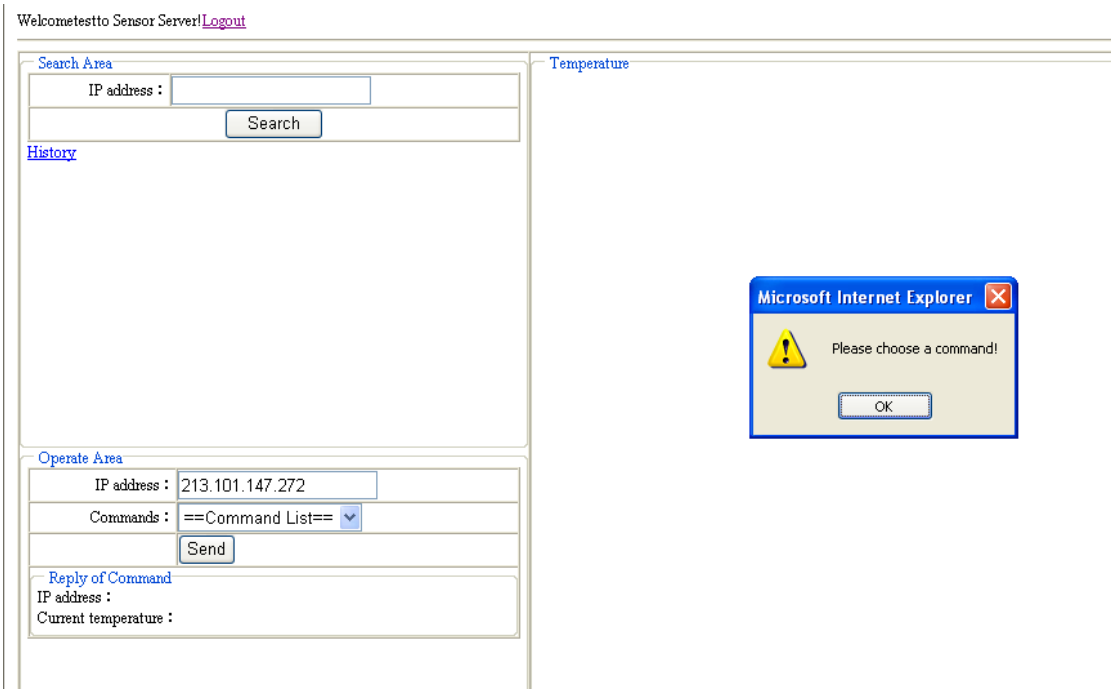


Figure 41:A notice message of web server to remain user to choose a command

Figure 42 presents an example temperature plot. The web server has searched for temperature records in the database for the indicated sensors, and drawn a picture to present on the web page. In this example, the picture shows only the most recent 10 records (if an administrator of web server wants to display more records in the plot, the administrator can modify the setting in the Web.config file). Note that the temperature values shown here are uncablibrated readings of a thermistor and should be converted to a specific temperature scale, such as degrees Centigrade (however, this calibration and conversion is outside the scope of this thesis project). However, if a user wants to see more or all records of a sensor, this user can on click the history link to view additional data. Figure 43 shows an example result. The historical records of a sensor are shown in a table. Above this table is a download link that enables the user to download these records in an XML format. Note that this download link is only visible after a user pressed the search button.

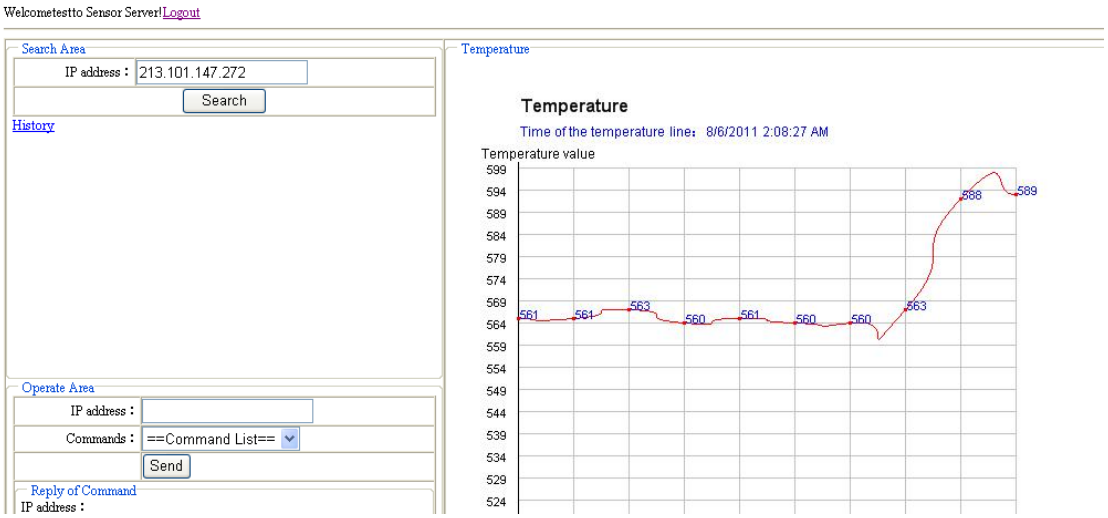
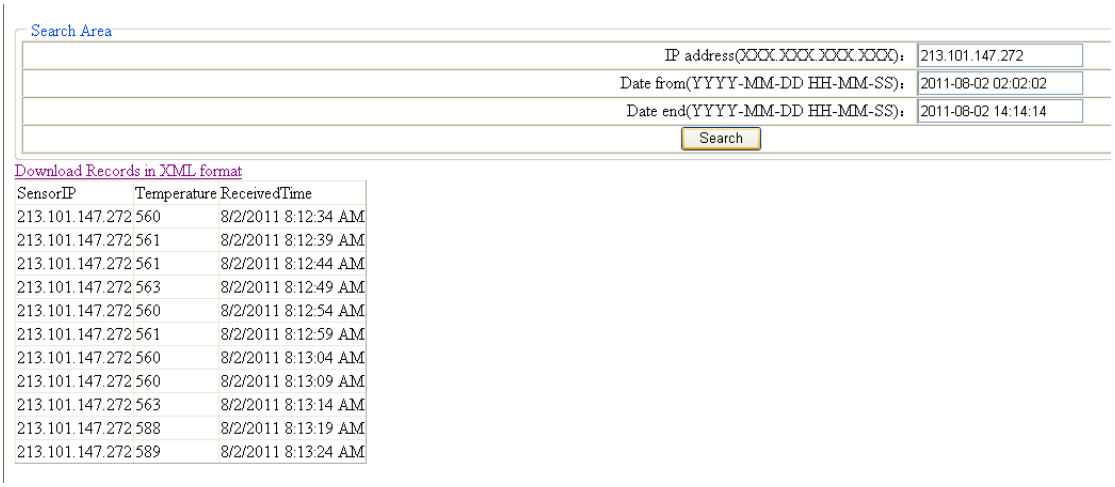


Figure 42: Recent temperature values of target sensor



SensorIP	Temperature	ReceivedTime
213.101.147.272	560	8/2/2011 8:12:34 AM
213.101.147.272	561	8/2/2011 8:12:39 AM
213.101.147.272	561	8/2/2011 8:12:44 AM
213.101.147.272	563	8/2/2011 8:12:49 AM
213.101.147.272	560	8/2/2011 8:12:54 AM
213.101.147.272	561	8/2/2011 8:12:59 AM
213.101.147.272	560	8/2/2011 8:13:04 AM
213.101.147.272	560	8/2/2011 8:13:09 AM
213.101.147.272	563	8/2/2011 8:13:14 AM
213.101.147.272	588	8/2/2011 8:13:19 AM
213.101.147.272	589	8/2/2011 8:13:24 AM

Figure 43: A history page example of web server

Figure 44 shows an example of sending a command to a specific sensor. The result is shown on the bottom left of the web page. The user has to input an IP address of a sensor and choose a command. After pressing the send button, the command is sent to the specific sensor by web server. If the web server gets a reply from that sensor, the web server will display the IP address of that sensor and the temperature value in the “Reply of Command” area. Otherwise the “Reply of Command” area will be blank. Additionally, the web server also saves this reply’s result into the database.

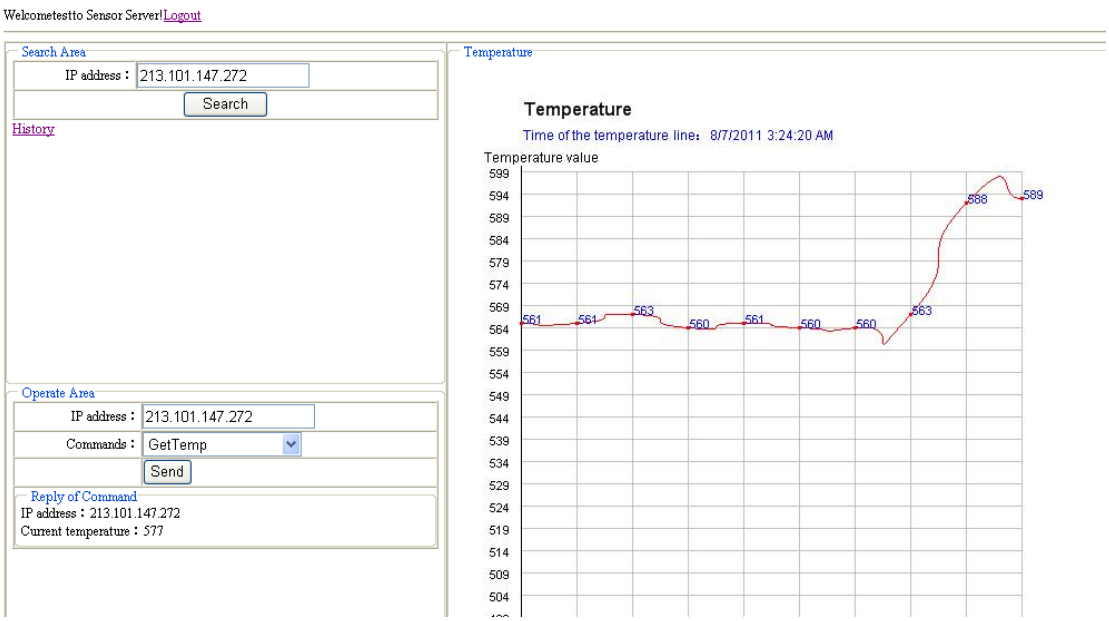


Figure 44: An example of sending a command to specific sensor

5.2 Measurements

In this section, measurements of the performance of each of the software components used in the project will be presented. As mentioned in previous sections, the process begins with the sending of a sensor data packet (by the simulator program or the ComServer program); next the web server's receiving & decoding program receives and decodes the packet and updates the database. The web server can display the sensor values on a web page. Therefore, three measurements are relevant to the performance of the system: (1) we measure the time required to generate a sensor packet and send it. In this thesis, we assume only two types of sensors: a 6LoWPAN sensor and a RS-232 low power sensor. A 6LoWPAN sensor will be emulated by a simulator program. Because we are simulating this data rather than actually using a sensor we do not see any reason to measure this delay - as it will be basically limited by the maximum rate that we can generate UDP datagrams. (2) We measure the delay between when the sensor sends a packet and when it has been received and decoded. (3) We measure the time required to store the values into the tables in the database (within the web server's receiving and decoding program). The measurement environment and equipment consists of:

1. Rather than using a low power sensor board and HP iPAQ we have used a Dell™ Studio XPS™ 1640 laptop: Intel Core 2 P8600 2.4 GHz CPU; 4 GB memory; 10/100M Ethernet card; Windows 7 operating system; and Microsoft Visual Studio 2010. The ComServer program running on this machine communicates over a USB interface with a Wasa board.
2. The web server is running on a Dell™ D 610 laptop configured with Intel Pentium M 1.6 GHz CPU; 512 MB memory; 10/100M Ethernet card;

Windows XP operating system; Microsoft Visual Studio 2005; and Microsoft SQL Server 2005. The receiving & decoding program and web server are both running on this machine.

3. These two machines are connected to a Netcore NR205S family router via a crossover cable. The two DELL laptops and the Netcore router form a local area network. The bandwidth of each Ethernet interface is 100M bps. The average packet transfer delay of the Netcore router is specified by the vendor as less than 300 μs[37].

5.2.1 Measuring performance of ComServer

The performance of the ComServer program is primarily determined by how much time is required to generate a sensor packet and send it out. The measurement method was to set a timer inside the ComServer program so that a timer records a startTime when the ComServer program starts to read the sensor data and endTime when the ConServer program sends the sensor data. The interval between startTime and endTime is the time required for reading and sending a sensor data value. Actually this interval is very short. To facilitate measuring this time, we set a 3 seconds break after the ComServer program sends the packet. Therefore, we have to subtract 3 seconds from each time record during our analysis. Figure 45 shows an example of such a measurement.

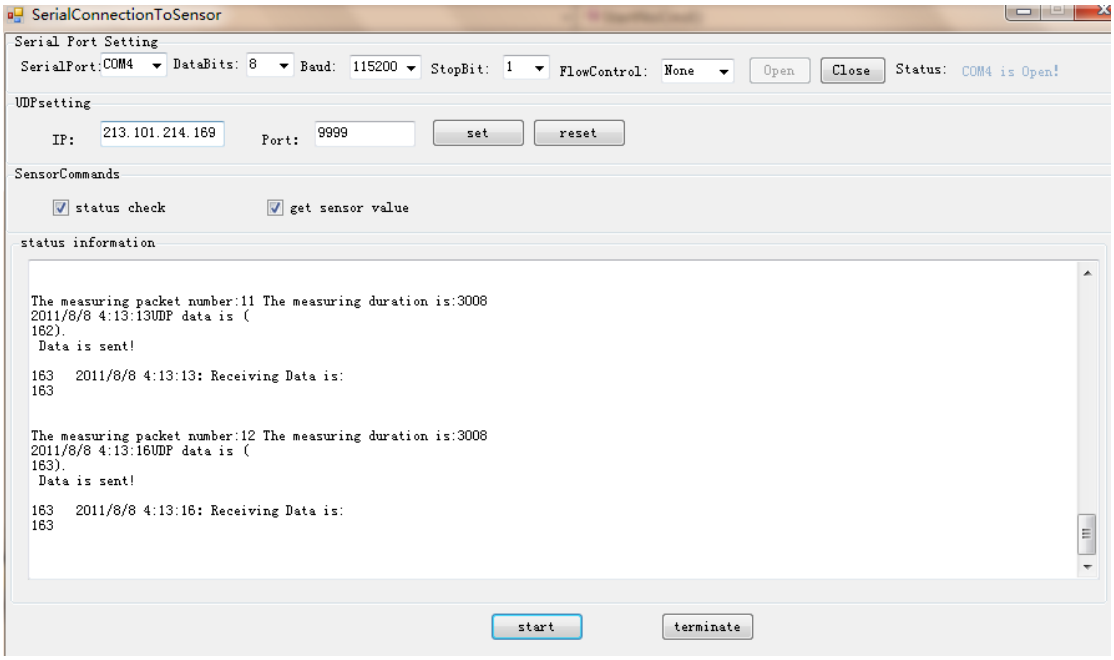


Figure 45: Time required to generating a sensor packet and sending out in ComServer

We repeated this measurement 5 times (i.e., 5 rounds), in each measurement the ComServer read and generated 20 sensor packets and sent them out. We calculated the average value over 20 measurements. We made 20 measurements because the time we

want to measure is very short and variable. We want to collect enough data to calculate a statistical value for the time interval required to perform and send a single measurement. Table 4 shows the records for this measurement (here we have already subtracted the extra 3 seconds). Note that the median value is 21 ms. Details of the analysis of these measurements are given in section 5.2.4.

Table 4: Measuring the performance of the ComServer

	Round 1	Round 2	Round 3	Round 4	Round 5
Average time for each round	8 ms	21 ms	51 ms	23 ms	15 ms
Average time for total 5 rounds	23.6ms (±14.7 ms)				

5.2.2 Measuring the performance of the receiving and decoding program in the web server

This measurement measures the how much time is needed between when the receiving and decoding program receives a sensor packet and when it stores the sensor's value into the database. This is quite similar to the previous measurement. Therefore we use the same method of measurement. Figure 46 shows an example of this measurement. We repeated the measurement 5 times (i.e., 5 rounds), and each time received and decoded 20 packets. Table 5 shows the average of these measurements. Details of the analysis of these measurements are given in section 5.2.4.

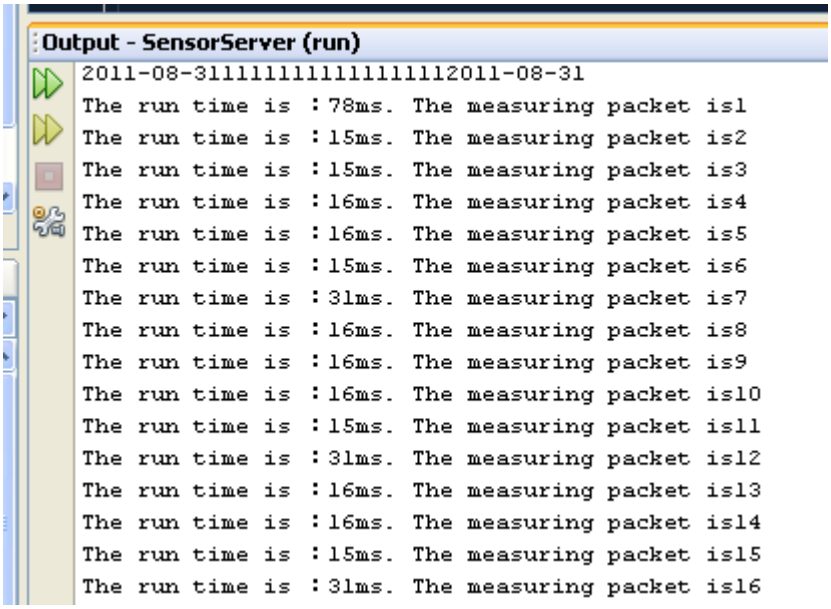


Figure 46: Measuring performance of receiving and decoding program

Table 5: Measuring the performance of the receiving and decoding program

	Round 1	Round 2	Round 3	Round 4	Round 5
Average time for each round	30.4 ms	28.5 ms	19.8 ms	18.9 ms	23.3 ms
Average time for total 5 rounds	24.18ms (±4.58 ms)				

5.2.3 Delay between when the sensors send a packet and when it has been received and decoded

Figure 47 presents the transfer process of packets flowing between the sensors and the web server's receiving and decoding program. Packet 1 is sent at time point G1; and arrives at the receiving and decoding program at time point R1 (This time point is when the packet has just received at the receiving and decoding program, and does not include the decoding time). The time between G1 and G2 is simply the processing time required by the sensor to generate a sensor packet. The transfer time of packet 1 is: **Delay1 = R1 - G1 – delay of router.** By parity of reasoning, the transfer time of rest packets is:

Delay2 = R2 – G2 – delay of router
Delay3 = R3 – G3 – delay of router
Delay4 = R4 – G4 – delay of router
Delay5 = R5 – G5 – delay of router
Delay6 = R6 – G6 – delay of router
…

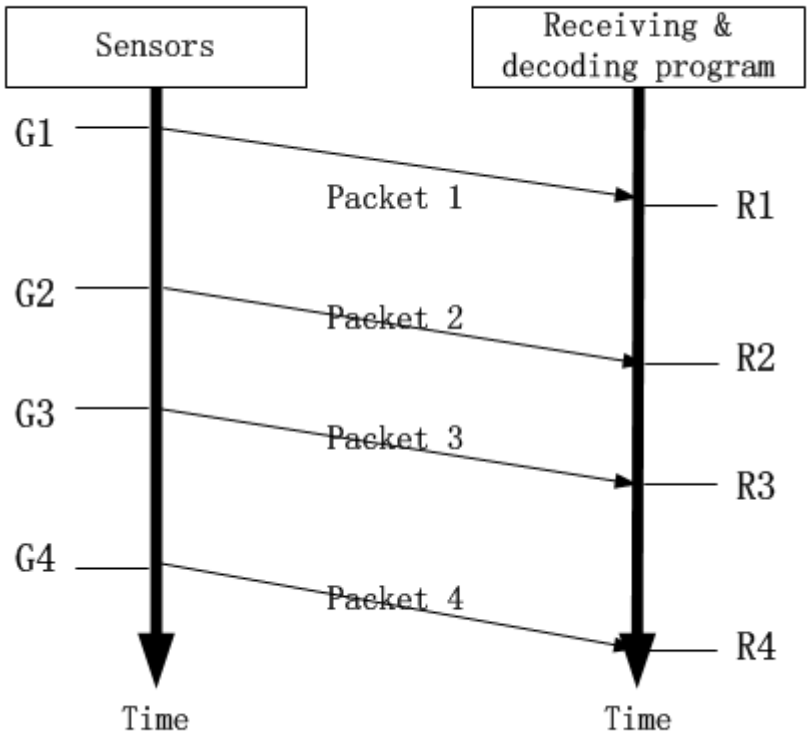


Figure 47: Sending process flow

Actually, the transmission delay when the sensors send a packet and when it has been received and decoded is very difficult to measure in millisecond level, since the data packet size in this transmission is very small and the transmission rate of the laptop's network card and router is very fast (100M bps). All the packets can be completely transferred less than 1 ms. The result were improved by a ping measurement (use the ping command "ping [IP address] –l [Size of packet]").

5.2.4 Analysis of these measurements

We can find that the standard deviation in the results of measurements of ComServer's performance is large. The variance of measurement results is very high. The highest result is 51 ms but lowest result is just 8 ms. In the measurements of performance of the receiving and decoding program, the results are more stable. We consider the reason might be related to Wasa board. The ComServer was running on a DELL laptop with a Wasa board. The receiving and decoding program was running on another DELL laptop. The common point of these two program's hardware is that both of them use a DELL laptop. Actually, the laptop belongs to ComServer has a better performance rather than the receiving and decoding program's (The two laptops were introduced in previously). However, a difference is the ComServer is also with a Wasa board. The Wasa board is a very simple board. It is used to teach the knowledge

to students. And the Wasa board is not very stable. The process and respond time is very variable. Therefore, we consider this is the reason why the variance of the ComServer's measurement results is very high. Another reason is that we calculate these measurements in microseconds. However, because the delay is so short we need to make additional measurements and use a more precise measurement technique.

The whole process (generating a sensor packet, transferring, receiving and decoding), it takes totally 48.78 ms (23.6ms + 24.18 ms + 1 ms (or less than 1ms)) on average. From our measurements we know that it only takes about 23 ms to generate and send the sensor packet, thus we can get a conclusion in theory that the rest of the time the sensor could be sleep (or in some other sort of low power mode), hence the power consumption of the sensor can be quite low - even with a duty cycle of once per second the sensor could be consuming minimal power for nearly 98% of the time (assuming that the time to wake up from this low power mode and to restart the radio transceiver is negligible - however, this is unlikely to be true, hence the actual fraction of time in lower power mode will be less than 98%). However, in practical this is not possible, because the hardware cannot just work around 23 ms and sleep 977 ms in one second. The duty cycle is too short.

The transferring delay is very short, hence the transfer delay in practice will be negligible in a local area network, but may have to be considered in a wide area network.

The time to receive and decode a sensor packet is the longest interval based upon our measurements. The average value is 24.18 ms for processing 60 bytes of data. This means in 1 second, the receiving and decoding program can process a maximum of 41 packets. Therefore, if 41 or more sensors send their reports to web server at the same time, the receiving and decoding program cannot process all the packets within this one second period. This means that either the average rate of sensor packets has to be lower than 41 per second or the receiving and decoding software needs to be modified to speed it up. In practice there would be an even larger problem because both the receiving & decoding and web service are running on the same computer, thus if sensor packets come at an average rate of more than 41 per second the web server will not have enough time to execute - hence the sensor web service will not be able to provide good performance to the user.

6 Conclusions and Future work

6.1 Conclusions

The aim of this thesis project was to design, implement, and evaluate a web application for sensors. More specifically we have developed, a web access solution for sensors that enables users to read and control sensor via a web site and we have shown how it is possible to integrate sensor with a PDA or other mobile device - thus enabling portable (or mobile) sensors. The web server accomplished the primary goal of being able to display sensor information via a web page to a user via their web browser. Communication from the user's web browser to the web server can use HTTP or HTTP over TLS (the later offers greater security). This project integrates a number of other solutions into a simple web based system. The end users can use their browsers to view and control the sensors without needing to install any additional software. Because the receiving & decoding program communicates with each sensor via UDP, this protocol can be adapted to the specific sensors and gateway that are used in practice. The sensor web server gets sensor information from a database. Additionally, this, the thesis project showed how the HP iPAQ pocket PC can utilize a sensor board connected to it via a serial port.

This thesis described the development and implement process of this web service. The author implemented the software by applying his knowledge of software engineering, the MVC software architecture, the techniques of Java programming, ASP.net programming, and networking knowledge.

This thesis project successfully achieves the primary goal of collecting and distributing sensor information from a sensor to web browser users. It is an example application that combines mobile technology, web technology, and sensing technology. Based on this web service, a future developer could add additional types of sensor to this web server. For instance, in the current web server, we utilized only a temperature sensor. In order to add a humidity sensor or a sensor offers map orientation or location is relatively straightforward. The next step is to combine the sensor data of all of these sensors. For example, the web server might know the location of the sensor, thus it can associate the temperature and humidity level with that place. If the web server were to utilize Goolge Maps technology, then the browser user could view a page that presents temperature and humidity measurements from a number of places, for example to use this with micro weather prediction. This thesis project not only built a generic web server, but it also introduces a way to develop a web based on the sensor applications.

6.2 Future work

Although this thesis project is completed, there are many improvements that could be made. Before this web server can be used outside of a lab environment, there is a lot of work that needs to be done in order to address issues concerning the administration of users & sensors, the security of the system, how to better support browser users, and so on. Some of the problem s that are more urgent are described below.

The first thing we have to consider is security. The web server actually has two interfaces: one interface is for the web browser user and another for sensors. The security between the web server and web browser user or sensors is very important. A hacker might steal information from the web server, or send a fake packet to corrupt or crash the web server. Any of these actions could cause valuable data to be lost. Therefore, it will be important to add suitable security to this communication. For example, an encryption function on the communication link between sensor and web server or between web browser and web server would improve the security of the communication. If this communication also included authentication, then it would be possible to improve the security even more.

The second issue is to make the web server friendly. In the current web server, there is no specific administrator page. If the administrator wants to manage the web server, for instance, to add a new sensor, the administrator has to manually modify the codes or the database. This is not user friendly. For the web browser user, the web server also needs to provide more functions such as the ability to change password and profile. Another problem for web browser user is in each time a user wants to search or operate on a sensor; the user has to input an IP address of a sensor. Since an IP address is not easy to remember (particularly in the case of IPv6 addresses), a naming scheme for sensors has to be added to the web server.

Additional functions can be added to the web server. The web server will also need to be modified to make it easier to add new types of sensors.

References

[1] Pontus Olvebrink, Low Power Sensor Platform for PDA: Wasa Board v1.4 revisited, Bachelor's thesis, Royal Institute of Technology (KTH), School of Information and Communications Technology, TRITA-ICT-EX-2011:183, 22 June 2011.

[2] Wikipedia. Microsoft SQL Server. http://en.wikipedia.org/wiki/Microsoft_SQL_Server . Latest view on June 27th, 2011.

[3] Zach Shelby and Carsten Bormann. 6LoWPAN: The Wireless Embedded Internet. A John Wiley and Sons, Ltd, Publication, 2009. [Pages 20,134,136].

[4] J.Postel. Transmission Control Protocol. RFC 793, RFC Editor, September 1981, Obsoletes RFC761, Updated by RFC1122, RFC3168, RFC6093. http://www.rfc-editor.org/rfc/rfc793.txt.

[5] J.Postel. User Datagram Protocol. RFC768, Information Sciences Institute. August 1980. http://www.rfc-editor.org/rfc/rfc768.txt.

[6] N. Kushalnagar, G. Montenegro, C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, RFC Editor. August 2007. http://www.rfc-editor.org/rfc/rfc4919.txt.

[7] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, RFC Editor. September 2007. http://www.rfc-editor.org/rfc/rfc4944.txt.

[8] P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik,and JP. Vasseur. RPL: IPv6 Routing Protocol for Low power and Lossy Networks draft-ietf-roll-rpl-19. Internet-Draft. Expires: September 14th, 2011. http://tools.ietf.org/html/draft-ietf-roll-rpl-19.

[9] IEEE 802 working group, Part 15.5: Wireless Personal Area Network (WPAN) Mesh Networking. http://www.ieee802.org/15/pub/TG5.html.

[10] IEEE 802 working group, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE Computer Society, Standard specification, [WWW], http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf.

[11] AES CCM Encryption and Decryption. http://www.inno-logic.com/resourcesEncryption.html . Latest view on February 15th, 2011.

[12] Don Box, David Ehnebuske, Gopal kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, Dave Winer. W3C. Simple Object Access Protocol. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/. May 08th, 2011.

[13] Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures, Doctoral Dissertation, Information and Computer Science, University of California, Irvine, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[14] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl. March 15th, 2001

[15] Marc Hadley. Web Application Description Language. http://www.w3.org/Submission/wadl/. August 31st , 2009

[16] N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, Internet Request for Comments, ISSN 2070-1721, RFC 2045, RFC Editor, November 1996, Updated by RFCs 2184, 2231, 5335, http://www.rfc-editor.org/rfc/rfc2045.txt.

[17] ZigBee. http://www.zigbee.org/Specifications.aspx. Latest view on December 12th, 2010.

[18] Open Mobile Alliance. Binary XML Content Format Specification. Tech. rep, WAP-192-WBXML-20010725-a, 2001.

[19] Open Geospatial Consortium. Binary Extensible Markup Language(BXML) Encoding Speicfication. Tech.rep., 03-002r9,2006.

[20] Efficient XML Interchange(EXI) primer. http://www.w3.org/TR/exi-primer/. Latest view on March 23rd, 2011.

[21] NanoService - Sensinode Ltd, Sensinode's NanoService™. [Online]. Available: http://www.sensinode.com/EN/products/nanoservice.html. Latest view on: August 25th, 2011.

[22] Mike Botts, and Alexandre Robin. OpenGIS® Sensor Model Language (SensorML) Implementation Specification. OGC 07-000, Open Geospatial Consortium Inc.. July 17th, 2007. http://www.opengeospatial.org/standards/sensorml.

[23] R.Fielding, J.Gettys, J.Mogul, H.Frystyk, L.Masinter, P.Leach, and t.Berners-Lee. Hypertext Transfer Protocol- HTTP/1.1. RFC 2616, Internet Engineering Task Force. June 1999. Obsoletes RFC2068, Updated by RFC2817,RFC5785, RFC6266. http://datatracker.ietf.org/doc/rfc793/?include_text=1.

[24] Java. http://en.wikipedia.org/wiki/Java_(programming_language). Latest view on March 12th, 2011.

[25] ASP.NET. http://www.w3schools.com/aspnet/default.asp. Latest view on May 17th, 2011.

[26] Luis Maqueda Ara, A 6LoWPAN gateway, Master's thesis, Royal Institute of Technology (KTH), School of Information and Communications Technology, work in progress.

[27] M.T.Smith. "The Wasa Board Project" Swedish Royal Institute of Technology, Stockholm, Sweden. http://web.it.kth.se/~msmith/wasa/wasa_board_project.html. Latest view on May 19th, 2011.

[28] AT command guide. http://linmodems.technion.ac.il/pctel-linux/Pctel.ATCommand.Guide.6.23.00.pdf. Latest view on May 12th, 2011.

[29] Joaqu ń Juan Toledo, A 6LoWPAN sensor board, Master's thesis, Royal Institute of Technology (KTH), School of Information and Communications Technology, work in progress.

[30] An introduction to the TI MSP430 low-power microcontrollers. http://mspgcc.sourceforge.net/manual/c68.html Latest view on December 12th, 2010.

[31] Alejanro Arcos. A context-aware application offering map orientation. Royal Institute of Technology (KTH), School of Information and Communications. 2010.

[32] MAX3241E. http://www.maxim-ic.com/datasheet/index.mvp/id/1780. Latest view on February 22nd, 2011.

[33] Sergio Floriano Sanchez. A 6LoWPAN sensor board software implementation, Master's thesis, Royal Institute of Technology (KTH), School of Information and Communications Technology, work in progress.

[34] Java API. http://download.oracle.com/javase/6/docs/api/. Latest view on June 16th, 2011.

[35] Model View Controller. http://c2.com/cgi/wiki?ModelViewController. Latest view on July 17th, 2011.

[36] SqlHelper. http://www.sharpdeveloper.net/source/SqlHelper-Source-Code-cs.html. Latest view on July 17th, 2011.

[37] NETCORE NR205 PLUS home broadband router, product web page, Shen Zhen Harten Techonology Co., June 13th , 2009. http://www.hartenisi.com.cn/en/products_detail.asp?productid=751.

Appendix A: The code of program to emulate 6LoWPAN sensor

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <netinet/in.h>
#include <unistd.h>
#include <errno.h>
#include <resolv.h>
#include <arpa/inet.h>
#define __LITTLE_ENDIAN_BITFIELD

struct LoWPAN {
	int cmd;
	unsigned short parameter1;
	unsigned short parameter2;
	unsigned short parameter3;
	unsigned short parameter4;
};

unsigned char usage[] = {"usage: udpsend remote_ip remote_port\r\n"
	"		for example:\r\n"
	"		udpsend 172.18.1.128 10000\r\n"};

int main(int argc, char *argv[]) {
	struct LoWPAN *pudphdr;
	unsigned char *pbuf;
	int port;
	struct sockaddr_in servaddr, cliaddr;
	unsigned char buff[127];
	int len = 0;
	if (argc < 3) {
		printf(usage);
		return 1;
	}
	memset(buff, 0, sizeof (buff));
```

```
        int snd = socket(AF_INET, SOCK_DGRAM, 0);
        if (!snd)
                return 1;
        bzero(&servaddr, sizeof (servaddr));
        servaddr.sin_family = AF_INET;
        port = atoi(argv[2]);
        servaddr.sin_port = htons(port);
        if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0) {
                printf("[%s] is not a valid IPaddress\n", argv[1]);
                return 1;
        }
        pbuf = buff;
        pudphdr = (struct LoWPAN*) pbuf;
        pudphdr->parameter1 = htons(1025);
        pudphdr->parameter2 = htons(61617);
        pudphdr->parameter3 = htons(25);
        pudphdr->parameter4 = htons(25);
        pbuf += sizeof (struct LoWPAN);
        len = pbuf - buff;
        sendto(snd, buff, len, 0, (struct sockaddr*) &servaddr, sizeof (servaddr));
        close(snd); // close the socket
        printf("The packet cotent is : %p", *pudphdr);
        return 0;
}
```

Appendix B: The code of ComServer program

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;
using System.Threading;


namespace COMServer
{
    public partial class Form1 : Form
    {
        protected bool isInTimerFun = false;
        string ReceiveData = "";
        private System.Threading.Thread thread = null;
        private System.Threading.Thread otherThread = null;

        public delegate void _SafeAddtrTextCall(string text);

        private delegate void ShowMSgDelegate(string msg);

        private delegate void ShowExceptionMSgDelegate(Exception msg);

        Mutex mu = new Mutex(false);

        Mutex muExceptionLog = new Mutex(false);

        public Form1()
        {
            InitializeComponent();
        }
```

```
        #region event
        private void Form1_Load(object sender, EventArgs e)
        {

            string portNum = "SerialPort Count:";

            try
            {

                portNum += SerialPort.GetPortNames().Length.ToString();
                for (int i = 0; i < SerialPort.GetPortNames().Length; i++)
                {
                    portNum += "\r\n" + SerialPort.GetPortNames().GetValue(i).ToString() + "\r\n";


ComboBoxPortNum.Items.Add(SerialPort.GetPortNames().GetValue(i).ToString());

                }
                SetMSG(portNum);
                ComboBoxPortNum.SelectedIndex = 0;
                comboBoxDateBits.SelectedIndex = 3;
                comboBoxParity.SelectedIndex = 2;
                comboBoxStopBit.SelectedIndex = 0;
                comboBoxBondRate.SelectedIndex = 3;
                OpenSerialPort();

            }
            catch
            {
                portNum += "0\r\nTheir is no any SerialPort on your computer!";
                SetMSG(portNum);
            }
        }


        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            try
            {
                if (serialPort1 != null)
```

```
            {
                if (serialPort1.IsOpen)
                {
                    serialPort1.Close();
                }
                serialPort1.Dispose();
            }
        }
        catch (Exception ex)
        {
            SetMSG("Close error！ ê?" + ex.Message);
        }
    }

    #endregion

    #region open serial port

    protected void OpenSerialPort()
    {
        if (ComboBoxPortNum.Text.ToString() != "")
        {
            serialPort1.PortName = ComboBoxPortNum.Text.ToString().Trim();
            try
            {
                if (!serialPort1.IsOpen)
                {
                    serialPort1.Open();
                }

                LBPortState.Text = serialPort1.PortName + " is Open！ ê?";
                if (comboBoxBondRate.Text.ToString() != "")
                {
                    serialPort1.BaudRate = int.Parse(comboBoxBondRate.Text.ToString());
                }
                if (comboBoxDateBits.Text.ToString() != "")
                {
                    serialPort1.DataBits = int.Parse(comboBoxDateBits.Text.ToString());
                }
                if (comboBoxStopBit.Text.ToString() != "")
                {
```

```
            switch (comboBoxStopBit.Text.ToString())
            {
                case "1":
                    serialPort1.StopBits = StopBits.One;
                    break;
                case "1.5":
                    serialPort1.StopBits = StopBits.OnePointFive;
                    break;
                case "2":
                    serialPort1.StopBits = StopBits.Two;
                    break;
            }
        }
        if (comboBoxParity.Text.ToString() != "")
        {
            switch (comboBoxParity.Text.ToString())
            {
                case "Even":
                    serialPort1.Parity = Parity.Even;
                    break;
                case "Mark":
                    serialPort1.Parity = Parity.Mark;
                    break;
                case "None":
                    serialPort1.Parity = Parity.None;
                    break;
                case "Odd":
                    serialPort1.Parity = Parity.Odd;
                    break;
                case "Space":
                    serialPort1.Parity = Parity.Space;
                    break;
            }
        }
        btnCloseSerialPort.Enabled = true;
        btnOpenSerialPort.Enabled = false;

    }
    catch (Exception e2)
    {
        LBPortState.Text = e2.Message;
```

```
                serialPort1.Dispose();
            }
        }
        else
        {
            LBPortState.Text = "Please choose a serial port！ ê?";
        }
    }

    private void btnOpenSerialPort_Click(object sender, EventArgs e)
    {
        OpenSerialPort();
    }

    private void btnCloseSerialPort_Click(object sender, EventArgs e)
    {

        if (ComboBoxPortNum.Text.ToString() != "")
        {

            try
            {
                if (serialPort1.IsOpen)
                {
                    serialPort1.Close();

                }

                LBPortState.Text = serialPort1.PortName + " is closed！ ê?";

                btnOpenSerialPort.Enabled = true;
                btnCloseSerialPort.Enabled = false;

            }
            catch (Exception e2)
            {
                LBPortState.Text = e2.Message;
                serialPort1.Dispose();
            }

        }
```

```
            else
            {
                LBPortState.Text = "Please choose a serial port！ ê?";
            }
        }

        #endregion

        #region

        private void safeAddtrText(string text)
        {

            if (this.InvokeRequired)
            {
                _SafeAddtrTextCall callALL =
                    delegate(string s)
                    {
                        this.txtBoxMSG.AppendText(s + "      ");
                        SetMSG("Receiving Data is:" + s    + "\r\n");
                    };
                this.Invoke(callALL, text);
            }
            else
            {
                txtBoxMSG.AppendText(text);
                SetMSG("Receiving Data is:" + text    + "\r\n");
            }

        }

        #endregion

        #region
        private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {

            while (serialPort1.BytesToRead > 0)
            {
```

```
            if (false)
            {
                int data = 0x00;
                try
                {
                    data = serialPort1.ReadByte();
                }
                catch (Exception e2)
                {
                    SetMSG(e2.Message);
                    break;
                }

                ReceiveData += data.ToString("X").PadLeft(2, '0');
                ReceiveData = ReceiveData.ToUpper();
            }
            else
            {
                ReceiveData += serialPort1.ReadExisting();
            }
        }
        UDPClient.MainContent = ReceiveData;
        safeAddtrText(ReceiveData);
        ReceiveData = "";

    }
    #endregion

    #region

    private void CLSTxtBoxAllReceive_Click(object sender, EventArgs e)
    {

    }

    #endregion

    #region
    public string WriteData(string cmd)
    {
        string sResult = "";
```

```
            if (serialPort1 == null)
            {

                sResult = "Serial port does not initial！ê?";
                return sResult;

            }
            if (!serialPort1.IsOpen)
            {

                sResult = "Serial port ： ê¤" + serialPort1.PortName +" open failed";
                return sResult;
            }
            try
            {
                string sendvalue = "";
                sendvalue = cmd;

                if (false)
                {
                    sendvalue = sendvalue.Replace(" ", "");
                }

                if (false)
                {
                    int sendLength = sendvalue.Length / 2;
                    byte[] StrBuffer = new byte[sendLength];
                    string hexstring = "";
                    int k = 0;
                    for (int i = 0; i < sendvalue.Length; )
                    {
                        try
                        {
                            hexstring = sendvalue.Substring(i, 2);
                        }
                        catch (Exception ex)
                        {
                            sResult += "Data format error, the data must be hex data： ê¤" +
ex.Message;
                            return sResult;
                        }
```

```
                    int j;
                    j = int.Parse(hexstring, System.Globalization.NumberStyles.HexNumber);
                    StrBuffer[k] = (byte)j;
                    i += 2;
                    k++;
                }

                serialPort1.Write(StrBuffer, 0, StrBuffer.Length);

            }
            else
            {
                serialPort1.Write(sendvalue);
                System.Threading.Thread.Sleep(50);
            }
        }
        catch (Exception ex)
        {
            sResult = ex.Message;
        }
        return sResult;
    }
    #endregion

    #region
    protected void SetMSG(string msg)
    {
        ShowMSgDelegate showmsgDelegate = new
ShowMSgDelegate(SetmsgDelegateTargetFun);
        if (txtBoxMSG.InvokeRequired)
        {
            txtBoxMSG.BeginInvoke(showmsgDelegate, msg);
        }
        else
        {
            if (txtBoxMSG.Text.Length > 1024 * 512)
            {
                txtBoxMSG.Text = string.Empty;
            }
            txtBoxMSG.AppendText(DateTime.Now.ToString() + "：  ê" + msg + "\r\n");
        }
```

```
            mu.WaitOne();
            SiXi.Logs.Log.WriteDebugLog(msg);
            mu.ReleaseMutex();
        }

        protected void SetmsgDelegateTargetFun(string msg)
        {
            if (txtBoxMSG.Text.Length > 1024 * 512)
            {
                txtBoxMSG.Text = string.Empty;
            }
            txtBoxMSG.AppendText(DateTime.Now.ToString() + "：  ê¤" + msg + "\r\n");
        }

        protected void SetExceptionMSG(Exception msg)
        {
            ShowExceptionMSgDelegate showmsgDelegate = new
ShowExceptionMSgDelegate(SetExceptionmsgDelegateTargetFun);
            if (txtBoxMSG.InvokeRequired)
            {
                txtBoxMSG.BeginInvoke(showmsgDelegate, msg);
            }
            else
            {
                if (txtBoxMSG.Text.Length > 1024 * 512)
                {
                    txtBoxMSG.Text = string.Empty;
                }
                txtBoxMSG.AppendText(DateTime.Now.ToString() + "：  ê¤" + msg.ToString() + "\r\n");
            }
            muExceptionLog.WaitOne();
            SiXi.Logs.Log.WriteDebugLog(msg.ToString());
            muExceptionLog.ReleaseMutex();
        }

        protected void SetExceptionmsgDelegateTargetFun(Exception msg)
        {
            if (txtBoxMSG.Text.Length > 1024 * 512)
            {
                txtBoxMSG.Text = string.Empty;
            }
```

```
            txtBoxMSG.AppendText(DateTime.Now.ToString() + "：  ĝo" + msg.ToString() + "\r\n");
        }
        #endregion

        private void button5_Click(object sender, EventArgs e)
        {
            UDPClient client = new UDPClient(this.IPText.Text, int.Parse(this.PortTxt.Text), serialPort1,
txtBoxMSG);
            otherThread = new Thread(client.StartByCmd);
            otherThread.Start();
            if (checkBox4.Checked == true)
            {
                string sResult = "";
                sResult = WriteData("at\r\n");
                if (sResult != "")
                {
                    SetMSG(sResult);

                }
            }
            if (checkBox3.Checked == true)
            {

                thread = new Thread(client.StartNoCmd);
                thread.Start();
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            //if (thread != null)
            //    thread.Abort();
            //if (otherThread != null)
            //    otherThread.Abort();
            txtBoxMSG.AppendText("\r\nStop receiving data!");
            UDPClient.Stop();
            if (serialPort1.IsOpen)
            {
                serialPort1.Close();
            }
        }
```

```
        private void checkBox3_CheckedChanged(object sender, EventArgs e)
        {

        }

        private void comboBoxParity_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void comboBoxStopBit_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void comboBoxBondRate_SelectedIndexChanged(object sender, EventArgs e)
        {

        }


    }
}
```

Form1.Designer.cs

```
namespace COMServer
{
    partial class Form1
    {
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
```

```
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    #region Windows

    private void InitializeComponent()
    {
        this.components = new System.ComponentModel.Container();
        this.serialPort1 = new System.IO.Ports.SerialPort(this.components);
        this.groupBox2 = new System.Windows.Forms.GroupBox();
        this.comboBoxParity = new System.Windows.Forms.ComboBox();
        this.btnOpenSerialPort = new System.Windows.Forms.Button();
        this.ComboBoxPortNum = new System.Windows.Forms.ComboBox();
        this.comboBoxBondRate = new System.Windows.Forms.ComboBox();
        this.label1 = new System.Windows.Forms.Label();
        this.label6 = new System.Windows.Forms.Label();
        this.label2 = new System.Windows.Forms.Label();
        this.label5 = new System.Windows.Forms.Label();
        this.LBPortState = new System.Windows.Forms.Label();
        this.comboBoxStopBit = new System.Windows.Forms.ComboBox();
        this.btnCloseSerialPort = new System.Windows.Forms.Button();
        this.label3 = new System.Windows.Forms.Label();
        this.label4 = new System.Windows.Forms.Label();
        this.comboBoxDateBits = new System.Windows.Forms.ComboBox();
        this.groupBox4 = new System.Windows.Forms.GroupBox();
        this.button4 = new System.Windows.Forms.Button();
        this.button2 = new System.Windows.Forms.Button();
        this.PortTxt = new System.Windows.Forms.TextBox();
        this.label7 = new System.Windows.Forms.Label();
        this.IPText = new System.Windows.Forms.TextBox();
        this.label8 = new System.Windows.Forms.Label();
        this.groupBox6 = new System.Windows.Forms.GroupBox();
        this.checkBox3 = new System.Windows.Forms.CheckBox();
        this.checkBox4 = new System.Windows.Forms.CheckBox();
        this.txtBoxMSG = new System.Windows.Forms.TextBox();
        this.groupBox3 = new System.Windows.Forms.GroupBox();
        this.button3 = new System.Windows.Forms.Button();
        this.button5 = new System.Windows.Forms.Button();
        this.groupBox2.SuspendLayout();
```

```
            this.groupBox4.SuspendLayout();
            this.groupBox6.SuspendLayout();
            this.groupBox3.SuspendLayout();
            this.SuspendLayout();

            this.serialPort1.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.serialPort1_DataReceived);

            this.groupBox2.Controls.Add(this.comboBoxParity);
            this.groupBox2.Controls.Add(this.btnOpenSerialPort);
            this.groupBox2.Controls.Add(this.ComboBoxPortNum);
            this.groupBox2.Controls.Add(this.comboBoxBondRate);
            this.groupBox2.Controls.Add(this.label1);
            this.groupBox2.Controls.Add(this.label6);
            this.groupBox2.Controls.Add(this.label2);
            this.groupBox2.Controls.Add(this.label5);
            this.groupBox2.Controls.Add(this.LBPortState);
            this.groupBox2.Controls.Add(this.comboBoxStopBit);
            this.groupBox2.Controls.Add(this.btnCloseSerialPort);
            this.groupBox2.Controls.Add(this.label3);
            this.groupBox2.Controls.Add(this.label4);
            this.groupBox2.Controls.Add(this.comboBoxDateBits);
            this.groupBox2.Dock = System.Windows.Forms.DockStyle.Top;
            this.groupBox2.Location = new System.Drawing.Point(0, 0);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(904, 49);
            this.groupBox2.TabIndex = 23;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Serial Port Setting";

            this.comboBoxParity.FormattingEnabled = true;
            this.comboBoxParity.Items.AddRange(new object[] {
            "Even",
            "Mark",
            "None",
            "Odd",
            "Space"});
            this.comboBoxParity.Location = new System.Drawing.Point(526, 20);
            this.comboBoxParity.Name = "comboBoxParity";
            this.comboBoxParity.Size = new System.Drawing.Size(59, 20);
            this.comboBoxParity.TabIndex = 11;
```

```
            this.comboBoxParity.SelectedIndexChanged += new
System.EventHandler(this.comboBoxParity_SelectedIndexChanged);

            this.btnOpenSerialPort.Location = new System.Drawing.Point(596, 18);
            this.btnOpenSerialPort.Name = "btnOpenSerialPort";
            this.btnOpenSerialPort.Size = new System.Drawing.Size(51, 23);
            this.btnOpenSerialPort.TabIndex = 2;
            this.btnOpenSerialPort.Text = "Open";
            this.btnOpenSerialPort.UseVisualStyleBackColor = true;
            this.btnOpenSerialPort.Click += new System.EventHandler(this.btnOpenSerialPort_Click);

            this.ComboBoxPortNum.FormattingEnabled = true;
            this.ComboBoxPortNum.Location = new System.Drawing.Point(76, 15);
            this.ComboBoxPortNum.Name = "ComboBoxPortNum";
            this.ComboBoxPortNum.Size = new System.Drawing.Size(54, 20);
            this.ComboBoxPortNum.TabIndex = 3;

            this.comboBoxBondRate.FormattingEnabled = true;
            this.comboBoxBondRate.Items.AddRange(new object[] {
            "1200",
            "2400",
            "4800",
            "9600",
            "19200",
            "115200"});
            this.comboBoxBondRate.Location = new System.Drawing.Point(276, 17);
            this.comboBoxBondRate.Name = "comboBoxBondRate";
            this.comboBoxBondRate.Size = new System.Drawing.Size(58, 20);
            this.comboBoxBondRate.TabIndex = 15;
            this.comboBoxBondRate.SelectedIndexChanged += new
System.EventHandler(this.comboBoxBondRate_SelectedIndexChanged);

            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(11, 20);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(71, 12);
            this.label1.TabIndex = 4;
            this.label1.Text = "SerialPort:";

            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(235, 20);
```

```
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(35, 12);
            this.label6.TabIndex = 14;
            this.label6.Text = "Baud:";

            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(713, 22);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(47, 12);
            this.label2.TabIndex = 5;
            this.label2.Text = "Status:";

            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(340, 22);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(53, 12);
            this.label5.TabIndex = 13;
            this.label5.Text = "StopBit:";

            this.LBPortState.AutoSize = true;
            this.LBPortState.ForeColor = System.Drawing.SystemColors.ActiveCaption;
            this.LBPortState.Location = new System.Drawing.Point(766, 24);
            this.LBPortState.Name = "LBPortState";
            this.LBPortState.Size = new System.Drawing.Size(41, 12);
            this.LBPortState.TabIndex = 6;
            this.LBPortState.Text = "NotNow";

            this.comboBoxStopBit.FormattingEnabled = true;
            this.comboBoxStopBit.Items.AddRange(new object[] {
            "1",
            "1.5",
            "2"});
            this.comboBoxStopBit.Location = new System.Drawing.Point(399, 17);
            this.comboBoxStopBit.Name = "comboBoxStopBit";
            this.comboBoxStopBit.Size = new System.Drawing.Size(38, 20);
            this.comboBoxStopBit.TabIndex = 12;
            this.comboBoxStopBit.SelectedIndexChanged += new
System.EventHandler(this.comboBoxStopBit_SelectedIndexChanged);

            this.btnCloseSerialPort.Enabled = false;
            this.btnCloseSerialPort.Location = new System.Drawing.Point(653, 18);
```

```
this.btnCloseSerialPort.Name = "btnCloseSerialPort";
this.btnCloseSerialPort.Size = new System.Drawing.Size(54, 23);
this.btnCloseSerialPort.TabIndex = 7;
this.btnCloseSerialPort.Text = "Close";
this.btnCloseSerialPort.UseVisualStyleBackColor = true;
this.btnCloseSerialPort.Click += new System.EventHandler(this.btnCloseSerialPort_Click);

this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(133, 18);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(59, 12);
this.label3.TabIndex = 8;
this.label3.Text = "DataBits:";
this.label3.Click += new System.EventHandler(this.label3_Click);

this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(443, 24);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(77, 12);
this.label4.TabIndex = 10;
this.label4.Text = "FlowControl:";

this.comboBoxDateBits.FormattingEnabled = true;
this.comboBoxDateBits.Items.AddRange(new object[] {
"5",
"6",
"7",
"8"});
this.comboBoxDateBits.Location = new System.Drawing.Point(193, 15);
this.comboBoxDateBits.Name = "comboBoxDateBits";
this.comboBoxDateBits.Size = new System.Drawing.Size(39, 20);
this.comboBoxDateBits.TabIndex = 9;

this.groupBox4.Controls.Add(this.button4);
this.groupBox4.Controls.Add(this.button2);
this.groupBox4.Controls.Add(this.PortTxt);
this.groupBox4.Controls.Add(this.label7);
this.groupBox4.Controls.Add(this.IPText);
this.groupBox4.Controls.Add(this.label8);
this.groupBox4.Dock = System.Windows.Forms.DockStyle.Top;
this.groupBox4.Location = new System.Drawing.Point(0, 49);
```

```
this.groupBox4.Name = "groupBox4";
this.groupBox4.Size = new System.Drawing.Size(904, 57);
this.groupBox4.TabIndex = 28;
this.groupBox4.TabStop = false;
this.groupBox4.Text = "UDPsetting";

this.button4.Location = new System.Drawing.Point(423, 19);
this.button4.Name = "button4";
this.button4.Size = new System.Drawing.Size(75, 23);
this.button4.TabIndex = 15;
this.button4.Text = "reset";
this.button4.UseVisualStyleBackColor = true;

this.button2.Location = new System.Drawing.Point(342, 19);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(75, 23);
this.button2.TabIndex = 14;
this.button2.Text = "set";
this.button2.UseVisualStyleBackColor = true;

this.PortTxt.Location = new System.Drawing.Point(248, 21);
this.PortTxt.Name = "PortTxt";
this.PortTxt.Size = new System.Drawing.Size(81, 21);
this.PortTxt.TabIndex = 13;

this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(204, 30);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(41, 12);
this.label7.TabIndex = 12;
this.label7.Text = "Port： ĝ";

this.IPText.Location = new System.Drawing.Point(76, 21);
this.IPText.Name = "IPText";
this.IPText.Size = new System.Drawing.Size(100, 21);
this.IPText.TabIndex = 11;

this.label8.AutoSize = true;
this.label8.Location = new System.Drawing.Point(36, 30);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(29, 12);
```

```
            this.label8.TabIndex = 11;
            this.label8.Text = "IP：  ĝo";

            this.groupBox6.Controls.Add(this.checkBox3);
            this.groupBox6.Controls.Add(this.checkBox4);
            this.groupBox6.Dock = System.Windows.Forms.DockStyle.Top;
            this.groupBox6.Location = new System.Drawing.Point(0, 106);
            this.groupBox6.Name = "groupBox6";
            this.groupBox6.Size = new System.Drawing.Size(904, 55);
            this.groupBox6.TabIndex = 29;
            this.groupBox6.TabStop = false;
            this.groupBox6.Text = "SensorCommands";

            this.checkBox3.AutoSize = true;
            this.checkBox3.Location = new System.Drawing.Point(210, 26);
            this.checkBox3.Name = "checkBox3";
            this.checkBox3.Size = new System.Drawing.Size(120, 16);
            this.checkBox3.TabIndex = 1;
            this.checkBox3.Text = "get sensor value";
            this.checkBox3.UseVisualStyleBackColor = true;
            this.checkBox3.CheckedChanged += new
System.EventHandler(this.checkBox3_CheckedChanged);

            this.checkBox4.AutoSize = true;
            this.checkBox4.Location = new System.Drawing.Point(38, 26);
            this.checkBox4.Name = "checkBox4";
            this.checkBox4.Size = new System.Drawing.Size(96, 16);
            this.checkBox4.TabIndex = 0;
            this.checkBox4.Text = "status check";
            this.checkBox4.UseVisualStyleBackColor = true;

            this.txtBoxMSG.Location = new System.Drawing.Point(14, 20);
            this.txtBoxMSG.Multiline = true;
            this.txtBoxMSG.Name = "txtBoxMSG";
            this.txtBoxMSG.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.txtBoxMSG.Size = new System.Drawing.Size(860, 257);
            this.txtBoxMSG.TabIndex = 0;

            this.groupBox3.Controls.Add(this.txtBoxMSG);
            this.groupBox3.Location = new System.Drawing.Point(4, 161);
            this.groupBox3.Name = "groupBox3";
```

```
this.groupBox3.Size = new System.Drawing.Size(880, 297);
this.groupBox3.TabIndex = 26;
this.groupBox3.TabStop = false;
this.groupBox3.Text = "status information";

this.button3.Location = new System.Drawing.Point(526, 464);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(75, 23);
this.button3.TabIndex = 31;
this.button3.Text = "terminate";
this.button3.UseVisualStyleBackColor = true;
this.button3.Click += new System.EventHandler(this.button3_Click);

this.button5.Location = new System.Drawing.Point(389, 464);
this.button5.Name = "button5";
this.button5.Size = new System.Drawing.Size(75, 23);
this.button5.TabIndex = 30;
this.button5.Text = "start";
this.button5.UseVisualStyleBackColor = true;
this.button5.Click += new System.EventHandler(this.button5_Click);

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(904, 509);
this.Controls.Add(this.groupBox6);
this.Controls.Add(this.button3);
this.Controls.Add(this.button5);
this.Controls.Add(this.groupBox4);
this.Controls.Add(this.groupBox3);
this.Controls.Add(this.groupBox2);
this.MaximizeBox = false;
this.Name = "Form1";
this.Text = "SerialConnectionToSensor";
this.Load += new System.EventHandler(this.Form1_Load);
this.groupBox2.ResumeLayout(false);
this.groupBox2.PerformLayout();
this.groupBox4.ResumeLayout(false);
this.groupBox4.PerformLayout();
this.groupBox6.ResumeLayout(false);
this.groupBox6.PerformLayout();
this.groupBox3.ResumeLayout(false);
```

```
            this.groupBox3.PerformLayout();
            this.ResumeLayout(false);

        }

        #endregion

        private System.IO.Ports.SerialPort serialPort1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.ComboBox comboBoxParity;
        private System.Windows.Forms.Button btnOpenSerialPort;
        private System.Windows.Forms.ComboBox ComboBoxPortNum;
        private System.Windows.Forms.ComboBox comboBoxBondRate;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label LBPortState;
        private System.Windows.Forms.ComboBox comboBoxStopBit;
        private System.Windows.Forms.Button btnCloseSerialPort;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.ComboBox comboBoxDateBits;
        private System.Windows.Forms.GroupBox groupBox4;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.TextBox PortTxt;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.TextBox IPText;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.GroupBox groupBox6;
        private System.Windows.Forms.CheckBox checkBox3;
        private System.Windows.Forms.CheckBox checkBox4;
        private System.Windows.Forms.TextBox txtBoxMSG;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.Button button5;

    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace COMServer
{
    static class Program
    {

        [STAThread]
        static void Main()
        {
            Application.ThreadException += new
System.Threading.ThreadExceptionEventHandler(Application_ThreadException);
            AppDomain.CurrentDomain.UnhandledException += new
UnhandledExceptionEventHandler(CurrentDomain_UnhandledException);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }

        static void CurrentDomain_UnhandledException(object sender, UnhandledExceptionEventArgs e)
        {
            Exception error = e.ExceptionObject as Exception;
            SiXi.Logs.Log.WriteExceptionLog(error);
            Application.Exit();
        }

        static void Application_ThreadException(object sender,
System.Threading.ThreadExceptionEventArgs e)
        {
            SiXi.Logs.Log.WriteExceptionLog(e.Exception);
            Application.Exit();
        }
    }
}
```

UDPClient.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Text.RegularExpressions;
using System.Diagnostics;
namespace COMServer
{
    public class UDPClient
    {
        private static int n = 0;
        private static Socket serverSocket = null;
        private string ip;
        private static bool isStop = false;
        private int port = 0;
        private System.IO.Ports.SerialPort serialPort = null;
        private System.Windows.Forms.TextBox listBox = null;
        public static string MainContent = "";
        public UDPClient(string _ip, int _port, System.IO.Ports.SerialPort _serialPort,
System.Windows.Forms.TextBox _listBox)
        {
            ip = _ip;
            port = _port;
            serialPort = _serialPort;
            listBox = _listBox;

        }
        public void StartByCmd()
        {

            try
            {
                if (!isStop)
                {
                    int recv;
                    byte[] data = new byte[1024];
                    IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9999);
```

```
                    Socket newsock = new Socket(AddressFamily.InterNetwork,
                                    SocketType.Dgram, ProtocolType.Udp);
                    newsock.Bind(ipep);
                    IPEndPoint sender = new IPEndPoint(IPAddress.Any, 9999);
                    EndPoint Remote = (EndPoint)(sender);

                    recv = newsock.ReceiveFrom(data, ref Remote);

                    newsock.SendTo(data, data.Length, SocketFlags.None, Remote);

                    while (true)
                    {
                        string cmd = Encoding.Default.GetString(data, 0, recv);
                        listBox.AppendText(System.DateTime.Now.ToString() + "The receiving
command is !!!!!!!!!!!!!!!!!!!:" + cmd + "\r\n");
                        if (cmd == "ATS200?\r\n")
                        {
                            serialPort.Write("ATS200?\r\n");
                            // System.Threading.Thread.Sleep(50);
                            if (!string.IsNullOrEmpty(MainContent))
                            {
                                byte[] sendMessage = Encoding.Default.GetBytes(MainContent);//
Encoding.Default.GetBytes(serialPort.ReadLine());
                                serverSocket.SendTo(sendMessage, Remote);
                                listBox.AppendText(System.DateTime.Now.ToString() +
"@@@@@@@@@@@@@Reply" + cmd + "data is sent!");
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                if (serverSocket != null)
                    serverSocket.Close();
            }
            finally
            {
                if (serverSocket != null)
                    serverSocket.Close();
            }
```

```
        }
        public void StartNoCmd()
        {

            Stopwatch sw = new Stopwatch();
            sw.Start();
            try
            {

                if (!isStop)
                {
                    IPAddress serverIP = IPAddress.Parse(ip);
                    IPEndPoint server = new IPEndPoint(serverIP, port);
                    Socket serverSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);

                    while (true)
                    {
                        n++;
                        byte[] cmd = Encoding.Default.GetBytes("ATS200?\r\n");
                        serialPort.Write("ATS200?\r\n");


                        if (!string.IsNullOrEmpty(MainContent))
                        {
                            byte[] sendMessage = Encoding.Default.GetBytes(MainContent);
                            serverSocket.SendTo(sendMessage, sendMessage.Length,
SocketFlags.None, server);
                            Match match = Regex.Match(MainContent, @"([0-9]+)");
                            listBox.AppendText(System.DateTime.Now.ToString() + "UDP data is ("
+ MainContent + "). \r\n Data is sent!\r\n");
                            System.Threading.Thread.Sleep(3000);
                            sw.Stop();
                            listBox.AppendText("\r\nThe measuring packet number:" + n + " The
measuring duration is:" + sw.ElapsedMilliseconds.ToString() + "\r\n");
                        }
                    }
                }
            }
            catch (Exception ex)
            {
```

```
                    if (serverSocket != null)
                        serverSocket.Close();
                }
                finally
                {
                    if (serverSocket != null)
                        serverSocket.Close();
                }

            }
            public static void Stop()
            {
                if (serverSocket != null)
                {
                    serverSocket.Close();
                    isStop = true;
                }
            }
        }
}
```

Appendix C: The code of receiving and decoding program

SensorReceiver.java

```
package sensorserver;

import java.io.*;
import java.net.DatagramSocket;


public class SensorReceiver {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int udpPortNumber;

        if (args.length == 1) {
            try {
                udpPortNumber = Integer.parseInt(args[0]);

                java.sql.Date date = new
java.sql.Date(System.currentTimeMillis());

                DatagramSocket updsocket = new
DatagramSocket(udpPortNumber);

                SensorReceiverThread sensorthread = new
SensorReceiverThread(updsocket);
                sensorthread.start();

                java.sql.Date date1 = new
java.sql.Date(System.currentTimeMillis());
                System.out.println(date + "1111111111111111"+ date1);

            } catch (NumberFormatException e) {
```

```
                    e.printStackTrace();
                    System.out.println("[Server Info] ERROR: The argument should be port number for    udp socket.");
                } catch (IOException ioe) {
                    ioe.printStackTrace();
                    System.out.println("[Server Info] I/O ERROR: The error occurs when opening the socket.");
                }
            }else {
                System.out.println("[Server Info] ERROR: You have to input the port number of udp socket in agument field." + "\n");
            }

    }
}
```

SensorReceiverThread.java

```
package sensorserver;

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

import db.DBManager;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.*;


public class SensorReceiverThread extends Thread {

    DatagramSocket updsocket;
    String stringHex;
    public static InetAddress sensoraddress1 = null;
    public static String sensoraddress = "";
    public static int sensorport = 0;
    public static int sensorTemp = 0;
    public static int sensorTemp1 = 0;
    public static int sensorTemp2 = 0;
    public static int sensorTemp3 = 0;
```

```
    public static int type = 0;
    public static int count = 0;

    public SensorReceiverThread(DatagramSocket updsocket) throws
SocketException {
        this.updsocket = updsocket;
    }

    @Override
    public synchronized void start() {
        super.start();
    }

    @Override
    public void run() {

        byte[] buffer = new byte[2048];

        while (true) {
            DBManager dbm = new DBManager();
            dbm.dbConnection();
            count++;
            DatagramPacket udpsensorpacket = new DatagramPacket(buffer,
buffer.length);
            long startTime = System.currentTimeMillis();

            try {
                updsocket.receive(udpsensorpacket);
                sensoraddress1 = udpsensorpacket.getAddress();
                sensoraddress = sensoraddress1.toString().substring(1);
                sensorport = udpsensorpacket.getPort();


                String msg = new String(buffer, 0, udpsensorpacket.getLength());
                udpsensorpacket.setLength(buffer.length);
                String detect = msg.substring(0, 2);

                if (detect.equals("AT")) {
                    type = 1;
                    SensorParserLowPower sensorpacketparser = new
SensorParserLowPower(msg);
```

```
                    sensorTemp = sensorpacketparser.getSensorTemp();
                    Date d = new Date();
                    long longtime = d.getTime();
                    d.toLocaleString();
                    SimpleDateFormat sdf = new
SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
                    sdf.format(longtime);

                    String query = "INSERT INTO dbo.SensorDataInfo
(IP,Temp,ReceiveTime) "
                            + "VALUES ('" + sensoraddress + "', '" +
sensorTemp + "', '" + sdf.format(longtime) + "')";
                    dbm.dbUpdate(query);

                    String query1 = "SELECT * FROM dbo.SensorInfo WHERE
IP LIKE '" + sensoraddress + "'";
                    ResultSet rs1 = dbm.dbQuery(query1);
                    try {
                        if (rs1.next()) {
                            String query2 = "DELETE FROM dbo.SensorInfo
WHERE IP LIKE '" + sensoraddress + "'";
                            dbm.dbUpdate(query2);
                            String query3 = "INSERT INTO dbo.SensorInfo
(IP,Port) "
                                    + "VALUES ('" + sensoraddress + "', '" +
sensorport + "')";
                            dbm.dbUpdate(query3);
                        } else {
                            String query4 = "INSERT INTO dbo.SensorInfo
(IP,Port) "
                                    + "VALUES ('" + sensoraddress + "', '" +
sensorport + "')";
                            dbm.dbUpdate(query4);
                        }
                    } catch (Exception e) {
                        System.err.println(e);
                    }

                    String query5 = "SELECT * FROM dbo.SensorTypeInfo
WHERE IP LIKE '" + sensoraddress + "'";
```

```
                    ResultSet rs2 = dbm.dbQuery(query5);
                    try {
                        if (rs2.next()) {
                            String query6 = "DELETE FROM
dbo.SensorTypeInfo WHERE IP LIKE '" + sensoraddress + "'";
                            dbm.dbUpdate(query6);
                            String query7 = "INSERT INTO
dbo.SensorTypeInfo (IP,Type) "
                                    + "VALUES ('" + sensoraddress + "', '" +
type + "')";
                            dbm.dbUpdate(query7);
                        } else {
                            String query8 = "INSERT INTO
dbo.SensorTypeInfo (IP,Type) "
                                    + "VALUES ('" + sensoraddress + "', '" +
type + "')";
                            dbm.dbUpdate(query8);
                        }
                    } catch (Exception e) {
                        System.err.println(e);
                    }

                } else {
                    type = 2;

                    SensorParserLowPAN sensorParserLowPAN = new
SensorParserLowPAN(msg);
                    int[] st = new int[4];
                    st = sensorParserLowPAN.getSensorTemp();
                    sensorTemp = st[0];
                    sensorTemp1 = st[1];
                    sensorTemp2 = st[2];
                    sensorTemp3 = st[3];
                    Date d = new Date();
                    long longtime = d.getTime();
                    d.toLocaleString();
                    SimpleDateFormat sdf = new
SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
                    sdf.format(longtime);

                    if (sensorTemp != 0 && sensorTemp1 != 0 &&
```

```
sensorTemp2 != 0 && sensorTemp3 != 0) {
                        } else if (sensorTemp != 0 && sensorTemp1 != 0 &&
sensorTemp2 != 0 && sensorTemp3 == 0) {
                        } else if (sensorTemp != 0 && sensorTemp1 != 0 &&
sensorTemp2 == 0 && sensorTemp3 == 0) {
                        } else if (sensorTemp != 0 && sensorTemp1 == 0 &&
sensorTemp2 == 0 && sensorTemp3 == 0) {

                            String query = "INSERT INTO dbo.SensorDataInfo
(IP,Temp,ReceiveTime) "
                                    + "VALUES ('" + sensoraddress + "', '" +
sensorTemp + "', '" + sdf.format(longtime) + "')";
                            dbm.dbUpdate(query);

                            String query1 = "SELECT * FROM dbo.SensorInfo
WHERE IP LIKE '" + sensoraddress + "'";
                            ResultSet rs1 = dbm.dbQuery(query1);
                            try {
                                if (rs1.next()) {
                                    String query2 = "DELETE FROM
dbo.SensorInfo WHERE IP LIKE '" + sensoraddress + "'";
                                    dbm.dbUpdate(query2);
                                    String query3 = "INSERT INTO
dbo.SensorInfo (IP,Port) "
                                            + "VALUES ('" + sensoraddress + "',
'" + sensorport + "')";
                                    dbm.dbUpdate(query3);
                                } else {
                                    String query4 = "INSERT INTO
dbo.SensorInfo (IP,Port) "
                                            + "VALUES ('" + sensoraddress + "',
'" + sensorport + "')";
                                    dbm.dbUpdate(query4);
                                }
                            } catch (Exception e) {
                                System.err.println(e);
                            }

                            String query5 = "SELECT * FROM
dbo.SensorTypeInfo WHERE IP LIKE '" + sensoraddress + "'";
                            ResultSet rs2 = dbm.dbQuery(query5);
```

```
                                try {
                                    if (rs2.next()) {
                                        String query6 = "DELETE FROM
dbo.SensorTypeInfo WHERE IP LIKE '" + sensoraddress + "'";
                                        dbm.dbUpdate(query6);
                                        String query7 = "INSERT INTO
dbo.SensorTypeInfo (IP,Type) "
                                                + "VALUES ('" + sensoraddress + "',
'" + type + "')";
                                        dbm.dbUpdate(query7);
                                    } else {
                                        String query8 = "INSERT INTO
dbo.SensorTypeInfo (IP,Type) "
                                                + "VALUES ('" + sensoraddress + "',
'" + type + "')";
                                        dbm.dbUpdate(query8);
                                    }
                                } catch (Exception e) {
                                    System.err.println(e);
                                }

                            } else if (sensorTemp == 0 && sensorTemp1 == 0 &&
sensorTemp2 == 0 && sensorTemp3 == 0) {
                                }

                            }
                            //Thread.yield();
                        } catch (IOException ex) {
                            System.err.println(ex);
                        }

                        long endTime = System.currentTimeMillis();
                        long runTime = endTime - startTime;
                        dbm.dbDisconnect();
                        System.out.print("The run time is ：" + runTime + "ms. The measuring
packet is" + count + "\n");
                    }
                }
            }
```

SensorParserLowPower.java

```
package sensorserver;
public class SensorParserLowPower {

	String sensorpacket = null;

	public SensorParserLowPower(String sensorpacket) {
		this.sensorpacket = sensorpacket;
	}

	public int getSensorTemp() {
		String Temp = sensorpacket.substring(2);
		int sensorTemp = Integer.parseInt(Temp);
		return sensorTemp;
	}
}
```

SensorParserLowPAN.java

```
package sensorserver;

public class SensorParserLowPAN {

	String sensorpacket = null;

	public SensorParserLowPAN(String sensorpacket) {
		this.sensorpacket = sensorpacket;

	}

	public int[] getSensorTemp() {

		int[] temp = new int[4];
		byte[] buffer = sensorpacket.getBytes();
		StringBuffer sb = new StringBuffer();
		String stringHex;
		for (int i = 0; i < sensorpacket.length(); i++) {
			stringHex = Integer.toHexString(buffer[i] & 0xFF);
			if (stringHex.length() == 1) {
				stringHex = '0' + stringHex;
```

```
                }
                sb.append(stringHex);

            }

            sb.toString();
            String[] tempA = new String[4];
            tempA[0] = sb.toString().substring(4, 8);
            tempA[1] = sb.toString().substring(8, 12);
            tempA[2] = sb.toString().substring(12, 16);
            tempA[3] = sb.toString().substring(16);

            temp[0] = Integer.parseInt(tempA[0], 16);
            temp[1] = Integer.parseInt(tempA[1], 16);
            temp[2] = Integer.parseInt(tempA[2], 16);
            temp[3] = Integer.parseInt(tempA[3], 16);

            return temp;
        }
}
```

DBManager.java

```
package db;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class DBManager {

        private Connection conn = null;
        private Statement stmt = null;
        private ResultSet rs = null;

        public void dbConnection() {
```

```
        try {

            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            String url =
"jdbc:sqlserver://localhost:1433;DatabaseName=SensorDB";
            conn = DriverManager.getConnection(url, "sa", "123456");
            //conn = ds.getConnection();
            stmt = conn.createStatement();
        } catch (Exception e) {
            System.err.println(e);
        }
        //return stmt;
    }

    public ResultSet dbQuery(String query) {
        //stmt = this.dbConnection();
        try {
            rs = stmt.executeQuery(query);
        } catch (SQLException e) {
            System.err.println(e);
        }
        return rs;
    }

    public void dbUpdate(String query) {
        try {
            stmt.executeUpdate(query);
        } catch (SQLException e) {
            System.err.println(e);
        }
    }

    public void dbDisconnect() {
        try {
            if (this.rs != null) {
                this.rs.close();
            }
            if (this.stmt != null) {
                this.stmt.close();
            }
            if (this.conn != null) {
```

```
                    this.conn.close();
                }
            } catch (SQLException e) {
                System.err.println(e);
            }
        }
}
```

Appendix D: The code of Web server

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="SensorSolution._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="font-size:12px">
    <form id="form1" runat="server">
    <div>
      <table width="100%" cellpadding="0" cellspacing="0" border="1">
      <thead>
        <tr>
          <td align="center" colspan="2">
            <h3>Log in</h3>
          </td>
        </tr>
      </thead>
        <tr>
          <td align="right">
              Username：
          </td>
          <td align="left">
            <asp:TextBox ID="UserNameTxt" runat="server" Width="204px"></asp:TextBox>
              <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
                  ControlToValidate="UserNameTxt" ErrorMessage="Username cannot
empty!"></asp:RequiredFieldValidator>
          </td>
        </tr>
        <tr>
          <td align="right">
              Password：
          </td>
```

```
            <td align="left">
              <asp:TextBox ID="PwdTxt" runat="server" TextMode="Password"
Width="203px"></asp:TextBox>
                  <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
                      ControlToValidate="PwdTxt" ErrorMessage="Password cannot
empty!"></asp:RequiredFieldValidator>
            </td>
          </tr>
          <tr>
            <td align="right">
             
            </td>
            <td align="left">
              <asp:Button ID="LoginBtn" runat="server" Text="Login" Width="77px"
                      onclick="LoginBtn_Click" />  <a href="Register.aspx">Sign Up</a>

            </td>
          </tr>
        </table>
    </div>
    </form>
</body>
</html>
```

Default.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using SensorWeb.Entity;
using SensorWeb.DataContext;
namespace SensorSolution
{
    public partial class _Default : System.Web.UI.Page
    {

        private UserDataContext _userDataContext = new UserDataContext();
        protected void Page_Load(object sender, EventArgs e)
        {
```

```
        }

        protected void LoginBtn_Click(object sender, EventArgs e)
        {
            try
            {

                UserInfo userEntiy = new UserInfo();
                userEntiy.UserName = this.UserNameTxt.Text.Trim();
                userEntiy.Pwd = this.PwdTxt.Text.Trim();
                int result=_userDataContext.Login(userEntiy);
                if (result > 0)
                {
                    Session["UserID"] = result;
                    Session["UserName"] = this.UserNameTxt.Text.Trim();
                    Server.Transfer("Main.aspx");
                }
                else
                {
                    Page.RegisterStartupScript("msg", "<script>alert('Login failed! Please check your
username and password!');</script>");
                }
            }
            catch
            {
            }
        }
    }
}
```

Default.aspx.Designer.cs

```
namespace SensorSolution {


    public partial class _Default {
```

```
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;

        protected global::System.Web.UI.WebControls.TextBox UserNameTxt;

        protected global::System.Web.UI.WebControls.RequiredFieldValidator RequiredFieldValidator1;

        protected global::System.Web.UI.WebControls.TextBox PwdTxt;

        protected global::System.Web.UI.WebControls.RequiredFieldValidator RequiredFieldValidator2;

        protected global::System.Web.UI.WebControls.Button LoginBtn;
    }
}
```

Main.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Main.aspx.cs"
Inherits="SensorSolution.Main" %>

<%@ Register Assembly="System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"
    Namespace="System.Web.UI" TagPrefix="asp" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="font-size:12px">
    <form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    Welcome<asp:Label ID="UserNameLbl" runat="server"></asp:Label>to Sensor Server!<a
href="Logout.aspx">Logout</a>
```

```
<hr />
<table width="100%" cellpadding="0" cellspacing="0" border="1">
  <tr>
    <td    width="30%" valign="top" height="600px">
     <table width="100%" cellpadding="0" cellspacing="0" height="600px">
       <tr>
         <td height="50%" valign="top">
           <fieldset style="height:300px">
             <legend>Search Area</legend>
             <table width="100%" cellpadding="0" cellspacing="0" border="1">
               <tr>
                 <td align="right">IP address：  ô</td>
                 <td align="left">
                   <asp:TextBox ID="SensorIPTxt" runat="server"></asp:TextBox>
                 </td>
               </tr>
               <tr>
                 <td align="center" colspan="2">
                   <asp:Button ID="SearchBtn" runat="server" Text="Search" Width="77px"
                               onclick="SearchBtn_Click" />
                 </td>

               </tr>
             </table>
             <a href="History.aspx">History</a>
           </fieldset>
         </td>
       </tr>
       <tr>
         <td valign="top">
           <fieldset style="height:300px">
             <legend>Operate Area</legend>
             <table width="100%" cellpadding="0" cellspacing="0" border="1">
               <tr>
                 <td align="right">IP address：  ô</td>
                 <td align="left">
                   <asp:TextBox ID="SensorIPTxtControl" runat="server"></asp:TextBox>
                 </td>
               </tr>
               <tr>
                 <td align="right">Commands：  ô</td>
```

```
                <td align="left">
                  <asp:DropDownList ID="CmdList" runat="server">
                    <asp:ListItem Value="-1" Text="==Command List==" Selected="True"></asp:ListItem>
                  </asp:DropDownList>
                </td>
              </tr>
              <tr>
                <td> </td>
                <td align="left">
                  <asp:Button ID="ReportBtn" Text="Send" runat="server" onclick="ReportBtn_Click" />
                </td>
              </tr>
              <tr>
                <td    colspan="2">
                <fieldset>
                  <legend>Reply of Command</legend>
                      IP address：ô¤<asp:Label ID="IDLbl" runat="server"></asp:Label><br />
                      Current temperature：ô¤<asp:Label ID="TempLbl" runat="server"></asp:Label>
                </fieldset>
                </td>
              </tr>
            </table>
            </fieldset>
          </td>
        </tr>
      </table>
    </td>
    <td valign="top">
      <fieldset>
        <legend>Temperature</legend>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
          <ContentTemplate>
        <iframe width="100%" frameborder="0" scrolling="auto" height="600px" id="MapFrame"
runat="server"></iframe>
        </ContentTemplate>
        <Triggers>
          <asp:AsyncPostBackTrigger    ControlID="Timer1" EventName="Tick" />
        </Triggers>
        </asp:UpdatePanel>
        <asp:Timer ID="Timer1" runat="server" Interval="5000" ontick="Timer1_Tick"
Enabled="false"></asp:Timer>
```

```
        </fieldset>
      </td>
    </tr>
  </table>
  </form>
</body>
</html>
```

Main.aspx.cs

```
using System;
using System.Collections.Generic;

using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using SensorWeb.DataContext;
using SensorWeb.Entity;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;
using System.Text;
namespace SensorSolution
{
    public partial class Main : System.Web.UI.Page
    {
        private SensorTypeDataContext sensorTypeDataContext = new SensorTypeDataContext();
        private SensorWeb.Entity.SensorInfo _serverInfo;
        private static string UDPResult = "";
        protected void Page_Load(object sender, EventArgs e)
        {

            if (Session["UserID"] != null && Session["UserID"].ToString().Trim() != "")
            {
                this.UserNameLbl.Text = Session["UserName"].ToString();
            }
            else
            {
                Page.RegisterStartupScript("msg", "<script>alert('Please log in
firstly!');window.location.href='Default.aspx';</script>");
```

```
            }
            string configStr =
System.Configuration.ConfigurationManager.AppSettings["cmd"].ToString();
            if (!string.IsNullOrEmpty(configStr))
            {
                string[] cmdStr = configStr.Split(',');
                if (cmdStr != null && cmdStr.Length > 0)
                {
                    for (int i = 0; i < cmdStr.Length; i++)
                    {
                        ListItem item = new ListItem(cmdStr[i].Trim(),cmdStr[i].Trim());
                        if (!this.CmdList.Items.Contains(item))
                        {
                            this.CmdList.Items.Add(item);
                        }
                    }
                }
            }
        }

        private int ToolTip(type type)
        {
            int result = 0;
            switch (type)
            {

                case type.HP:
                    result = 1;
                    break;
                case type.LoWPAN:
                    result = 2;
                    break;
            }
            return result;
        }

        protected void SearchBtn_Click(object sender, EventArgs e)
        {
            type type;
            int searchType = -1;
            int tmp=0;
```

```
                if (!string.IsNullOrEmpty(this.SensorIPTxt.Text))
            {

                    type = sensorTypeDataContext.GetSensorTypeByIP(this.SensorIPTxt.Text);
                    searchType = 1;
            }

            else
            {
                Page.RegisterStartupScript("msg", "<script>alert('Please input an IP address of a
sensor!');</script>");
                searchType = -1;
                this.MapFrame.Attributes["src"] = "";
                return;
            }
            #region

            switch (searchType)
            {

                case 1:
                    this.MapFrame.Attributes["src"] = "DrawImage.aspx?UserID=" +
this.SensorIPTxt.Text;
                    this.Timer1.Enabled = true;
                    break;
            }
            #endregion
        }

        protected void Timer1_Tick(object sender, EventArgs e)
        {
            if (this.MapFrame.Attributes["src"]!=null&&this.MapFrame.Attributes["src"].Trim() != "")
            {
                this.MapFrame.Attributes["src"] = this.MapFrame.Attributes["src"];
            }
            else
            {
                this.MapFrame.Attributes["src"] = "";
            }
        }
```

```
        private void SendUdpHP( )
        {
            IPEndPoint server = new IPEndPoint(IPAddress.Parse(_serverInfo.SensorIP),
_serverInfo.Port);
            UdpClient udpClient = new UdpClient();

                String cmd = this.CmdList.SelectedValue.Trim();
                if (cmd == "GetTemp")
                {
                    byte[] sendMessage = Encoding.Default.GetBytes("ATS200?\r\n");
                    udpClient.Send(sendMessage, sendMessage.Length, server);
                    string result = "";
                    byte[] receiveMsg = udpClient.Receive(ref server);
                    result = Encoding.Default.GetString(receiveMsg).Substring(2);
                    IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);
                    String receiveIP = RemoteIpEndPoint.Address.ToString();
                    int receivePort = RemoteIpEndPoint.Port;
                    SensorDataContext SensorDataContext = new SensorDataContext();
                    int existPort =
SensorDataContext.GetSensorPortByIP(this.SensorIPTxtControl.Text);
                    if (!string.IsNullOrEmpty(result))
                    {
                        SensorData sensorEntity = new SensorData();
                        if (receiveIP == this.SensorIPTxtControl.Text && existPort == receivePort)
                        {
                            sensorEntity.SensorIP = receiveIP;
                            sensorEntity.Temp = int.Parse(result);
                            sensorEntity.ReciveTime = System.DateTime.Now;
                            UDPResult = result;
                            new
SensorWeb.DataContext.SensorDataContext().InsertSensorData(sensorEntity);
                        }

                    }
                    udpClient.Close();
                }
        }

        private void SendUdpLowpan()
        {
```

```
                IPEndPoint server = new IPEndPoint(IPAddress.Parse(_serverInfo.SensorIP),
_serverInfo.Port);
                UdpClient udpClient = new UdpClient();
                String cmd = this.CmdList.SelectedValue.Trim();
                if (cmd == "GetTemp")
                {
                    byte[] sendMessage = Encoding.Default.GetBytes("05");
                    udpClient.Send(sendMessage, sendMessage.Length, server);
                    string result = "";
                    byte[] receiveMsg = udpClient.Receive(ref server);
                    result = Encoding.Default.GetString(receiveMsg);
                    IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);
                     String receiveIP =      RemoteIpEndPoint.Address.ToString();
                     int receivePort = RemoteIpEndPoint.Port;
                     SensorDataContext SensorDataContext = new SensorDataContext();
                     int existPort = SensorDataContext.GetSensorPortByIP(this.SensorIPTxtControl.Text);
                    if (!string.IsNullOrEmpty(result))
                    {
                        SensorData sensorEntity = new SensorData();
                        if (receiveIP == this.SensorIPTxtControl.Text && existPort == receivePort)
                        {
                            sensorEntity.SensorIP = receiveIP;
                            sensorEntity.Temp = int.Parse(result);
                            sensorEntity.ReciveTime = System.DateTime.Now;
                            UDPResult = result;
                            new
SensorWeb.DataContext.SensorDataContext().InsertSensorData(sensorEntity);
                        }
                    }
                    udpClient.Close();
                    //}
                }
            }

            protected void ReportBtn_Click(object sender, EventArgs e)
            {
                try
                {
                    if (string.IsNullOrEmpty(this.SensorIPTxtControl.Text))
                    {
```

```
                Page.RegisterStartupScript("msg", "<script>alert('Please input an SensorIP address
of sensor!');</script>");
            }
            if (this.CmdList.SelectedIndex == -1 || this.CmdList.SelectedValue == "-1")
            {
                Page.RegisterStartupScript("msg", "<script>alert('Please choose a
command!');</script>");
            }
            else
            {
                #region udp command
                _serverInfo = new
SensorWeb.DataContext.SensorInfoDataContext().GetSensorInfoBySensorIP(this.SensorIPTxtControl.Text)
;

                type type;
                if (!string.IsNullOrEmpty(this.SensorIPTxt.Text))
                {
                    type = sensorTypeDataContext.GetSensorTypeByIP(this.SensorIPTxt.Text);

                    int typeResult = ToolTip(type);

                    if (typeResult == 1)
                    {
                        Thread sendMsgThread = null;
                        try
                        {

                            sendMsgThread = new Thread(new ThreadStart(SendUdpHP));
                            sendMsgThread.Start();
                            Thread.Sleep(1000);
                            this.IDLbl.Text = this.SensorIPTxtControl.Text;
                            this.TempLbl.Text = UDPResult;
                        }
                        catch
                        {
                        }
                        finally
                        {
                            //sendMsgThread.Abort();
                        }
```

```
                    }
                    else if (typeResult == 2)
                    {
                        Thread sendMsgThread = null;
                        try
                        {

                            sendMsgThread = new Thread(new ThreadStart(SendUdpLowpan));
                            sendMsgThread.Start();
                            Thread.Sleep(1000);
                            this.IDLbl.Text = this.SensorIPTxtControl.Text;
                            this.TempLbl.Text = UDPResult;
                        }
                        catch
                        {
                        }
                        finally
                        {
                            //sendMsgThread.Abort();
                        }
                    }
                }

                    #endregion
                }
            }
            catch
            {
            }
        }
    }
}
```

Main.aspx.Designer.cs

```
namespace SensorSolution {


    public partial class Main {
```

```
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;

        protected global::System.Web.UI.ScriptManager ScriptManager1;

        protected global::System.Web.UI.WebControls.Label UserNameLbl;


        protected global::System.Web.UI.WebControls.TextBox SensorIPTxt;


        protected global::System.Web.UI.WebControls.Button SearchBtn;


        protected global::System.Web.UI.WebControls.TextBox SensorIPTxtControl;


        protected global::System.Web.UI.WebControls.DropDownList CmdList;


        protected global::System.Web.UI.WebControls.Button ReportBtn;


        protected global::System.Web.UI.WebControls.Label IDLbl;


        protected global::System.Web.UI.WebControls.Label TempLbl;


        protected global::System.Web.UI.UpdatePanel UpdatePanel1;


        protected global::System.Web.UI.HtmlControls.HtmlGenericControl MapFrame;

        protected global::System.Web.UI.Timer Timer1;
    }
}
```

Register.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Register.aspx.cs"
```

```
Inherits="SensorSolution.Register" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="font-size:12px">
    <form id="form1" runat="server">
    <div align="center">
      <h3>Sign Up</h3>
    </div>
    <hr />
    <fieldset>
      <legend>User Information</legend>
      <table width="100%" cellpadding="0" cellspacing="0" border="1">
        <tr>
          <td align="right">Username： ô</td>
          <td align="left">
            <asp:TextBox ID="UserNameTxt" runat="server"></asp:TextBox><font color="red">*</font>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
                    ControlToValidate="UserNameTxt" ErrorMessage="Username cannot be
empty!"></asp:RequiredFieldValidator>
          </td>
        </tr>
        <tr>
          <td align="right">Password： ô</td>
          <td align="left">
            <asp:TextBox ID="PwdTxt" runat="server" TextMode="Password"></asp:TextBox><font
color="red">*</font>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
                    ControlToValidate="PwdTxt" ErrorMessage="Password cannot be
empty!"></asp:RequiredFieldValidator>
          </td>
        </tr>
        <tr>
          <td align="right">Repeat password： ô</td>
          <td align="left">
            <asp:TextBox ID="RePwdTxt" runat="server" TextMode="Password"></asp:TextBox><font
```

```
color="red">*</font>
                <asp:CompareValidator ID="CompareValidator1" runat="server"
                    ControlToCompare="PwdTxt" ControlToValidate="RePwdTxt"
                    ErrorMessage="Repeat password is different!"></asp:CompareValidator>
            </td>
          </tr>
          <tr>
            <td align="right">Email:</td>
            <td    align="left">
              <asp:TextBox ID="EmailTxt" runat="server"></asp:TextBox><font color="red">*</font>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
                    ControlToValidate="EmailTxt" ErrorMessage="Email cannot be
empty!"></asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
                    ControlToValidate="EmailTxt" ErrorMessage="Email format is wrong!"

ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"></asp:RegularExpressionValidator
>
              </td>
          </tr>
          <tr>
            <td align="right">Invited Code： ê¤</td>
            <td align="left">
              <asp:TextBox ID="InviteCodeTxt" runat="server"></asp:TextBox><font color="red">*</font>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server"
                    ControlToValidate="InviteCodeTxt"
                    ErrorMessage="Invited Code cannot be empty!"></asp:RequiredFieldValidator>
            </td>
          </tr>
          <tr>
            <td></td>
            <td align="left">
              <asp:Button ID="RegisterBtn" runat="server" Text="Confirm"
                    onclick="RegisterBtn_Click" />
                <a href="Default.aspx">Back</a>
            </td>
          </tr>
        </table>
      </fieldset>
      </form>
</body>
```

```
</html>
```

Register.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using SensorWeb.DataContext;
using SensorWeb.Entity;
namespace SensorSolution
{
    public partial class Register : System.Web.UI.Page
    {

        private UserDataContext _userDataContext = new UserDataContext();
        private InviteDataContext _inviteDataContext = new InviteDataContext();
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void RegisterBtn_Click(object sender, EventArgs e)
        {
            try
            {

                UserInfo userEntiy = new UserInfo();
                userEntiy.UserName = this.UserNameTxt.Text.Trim();
                userEntiy.Pwd = this.RePwdTxt.Text.Trim();
                userEntiy.Email = this.EmailTxt.Text.Trim();

                if (!_userDataContext.IsExist(userEntiy))
                {

                    InviteInfo inviteEntiy = new InviteInfo();
                    int parseInt = 0;
                    if (int.TryParse(this.InviteCodeTxt.Text.Trim(), out parseInt))
                    {
                        inviteEntiy.InviteNum = int.Parse(this.InviteCodeTxt.Text.Trim());
```

```
                    if (_inviteDataContext.IsExist(inviteEntiy))
                    {
                        int result = _userDataContext.AddUser(userEntiy);
                        if (result > 0)
                        {
                            _inviteDataContext.DeleteInvite(inviteEntiy);
                            Page.RegisterStartupScript("msg", "<script>alert('Sign Up
successes!');window.location.href='Default.aspx';</script>");
                        }
                        else
                        {
                            Page.RegisterStartupScript("msg", "<script>alert('Sign Up
fails!');</script>");
                        }
                    }
                    else
                    {
                        Page.RegisterStartupScript("msg", "<script>alert('Invited Code does not
exist!');</script>");
                    }
                }
                else
                {
                    Page.RegisterStartupScript("msg", "<script>alert('Invited Code is 4 digit
number!');</script>");
                }
            }
            else
            {
                Page.RegisterStartupScript("msg", "<script>alert('Username already
exists!');</script>");
            }
        }
        catch
        {
        }
    }
}
```

Register.aspx.Designer.cs

```
namespace SensorSolution {


    public partial class Register {

        protected global::System.Web.UI.HtmlControls.HtmlForm form1;

        protected global::System.Web.UI.WebControls.TextBox UserNameTxt;

        protected global::System.Web.UI.WebControls.RequiredFieldValidator RequiredFieldValidator1;

        protected global::System.Web.UI.WebControls.TextBox PwdTxt;

        protected global::System.Web.UI.WebControls.RequiredFieldValidator RequiredFieldValidator2;

        protected global::System.Web.UI.WebControls.TextBox RePwdTxt;

        protected global::System.Web.UI.WebControls.CompareValidator CompareValidator1;

        protected global::System.Web.UI.WebControls.TextBox EmailTxt;

        protected global::System.Web.UI.WebControls.RequiredFieldValidator RequiredFieldValidator3;

        protected global::System.Web.UI.WebControls.RegularExpressionValidator
RegularExpressionValidator1;

        protected global::System.Web.UI.WebControls.TextBox InviteCodeTxt;

        protected global::System.Web.UI.WebControls.RequiredFieldValidator RequiredFieldValidator4;

        protected global::System.Web.UI.WebControls.Button RegisterBtn;
    }
}
```

DrawImage.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="DrawImage.aspx.cs"
Inherits="SensorSolution.DrawImage" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script type="text/javascript">
        function onLoad() {
            alert('1');
            window.location.reload();
        }
        setTimeout("onload()", 1000);
    </script>
</head>
<body onload="onLoad();">
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

DrawImage.aspx.cs

```
using System;
using System.Collections.Generic;

using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.IO;
using System.Drawing.Imaging;
using SensorWeb.Entity;
using SensorWeb.DataContext;
namespace SensorSolution
{
    public partial class DrawImage : System.Web.UI.Page
    {
```

```
        private int Dx
=int.Parse(System.Configuration.ConfigurationManager.AppSettings["Dx"].Trim());
        private int Dy = 20;
        private int maxYnum = 1;
        private int Dm = 5;
        private int maxY;
        private int tempCount = 0;
        private SensorDataContext SensorDataContext = new SensorDataContext();
        protected void Page_Load(object sender, EventArgs e)
        {

            if (Session["UserID"] != null && Session["UserID"].ToString().Trim() != "")
            {
                List<SensorData> list = new List<SensorData>();
                int topCount =
int.Parse(System.Configuration.ConfigurationManager.AppSettings["Top"].Trim());

                    if (!string.IsNullOrEmpty(Request.QueryString["UserID"]))
                {
                    list =
SensorDataContext.GetSensorDataByIP(Request.QueryString["UserID"].ToString(), topCount);

                }

                 if (maxY <= 100)
                  {
                      int max = list[0].Temp;
                      for (int h = 0; h < list.Count; h++)
                      {
                          if (max < list[h].Temp)
                              max = list[h].Temp;

                      }
                       maxYnum = max + 10;
                  }
                 else
                {
                    int L = maxY.ToString().Length;
                    int Temp = int.Parse(maxY.ToString().Substring(0, 1)) + 1;
                    if (L <= 3)
                    {
```

```
            Dm = 15;
        }
        else
        {
            Dm = 50;
        }
        for (int m = 0; m < L - 1; m++)
        {
            maxYnum *= 10;
        }
        maxYnum = Temp * maxYnum;
}

int c = 0;
if (list != null && list.Count > 0)
{
        c = list.Count;
}

int wd = 80 + Dx * (c - 1);

int hd = maxYnum / Dm * Dy + 120;

if (wd < 600) wd = 600;

if (hd < 400) hd = 250;

Bitmap img = new Bitmap(wd, hd);

Graphics g = Graphics.FromImage(img);

Pen Bp = new Pen(Color.Black);

Pen Rp = new Pen(Color.Red);

Pen Sp = new Pen(Color.Silver);

Font Bfont = new Font("Arial", 12, FontStyle.Bold);

Font font = new Font("Arial", 8);
```

```
                Font Tfont = new Font("Arial", 9);

                g.DrawRectangle(new Pen(Color.White, hd), 0, 0, img.Width, img.Height);

                LinearGradientBrush brush = new LinearGradientBrush(new Rectangle(0, 0, img.Width,
img.Height), Color.Black, Color.Black, 1.2F, true);

                LinearGradientBrush Bluebrush = new LinearGradientBrush(new Rectangle(0, 0,
img.Width, img.Height), Color.Blue, Color.Blue, 1.2F, true);

                g.DrawString("Temperature", Bfont, brush, 50, 20);
                int nums = 0;
                for (int i = 0; i < c; i++)
                {
                    nums += list[i].Temp;
                    tempCount += list[i].Temp;
                }

                string info = "Time of the temperature line：ê¤" + DateTime.Now.ToString();
                g.DrawString(info, Tfont, Bluebrush, 50, 45);

                for (int i = 0; i < c; i++)
                {
                    g.DrawLine(Sp, 50 + Dx * i, 85, 50 + Dx * i, 85 + maxYnum / Dm * Dy);

                    string st = list[c - i - 1].ReciveTime.ToShortTimeString();
                    g.DrawString(st, font, brush, 45 + Dx * i, 85 + maxYnum / Dm * Dy);
                }
                for (int i = 0; i < maxYnum / Dm; i++)
                {
                    g.DrawLine(Sp, 50, 85 + Dy * i, 50 + Dx * (c - 1), 85 + Dy * i);
                    int s = maxYnum - Dm * i;

                    g.DrawString(s.ToString(), font, brush, 20, 80 + Dy * i);
                }

                g.DrawLine(Bp, 50, 80, 50, 85 + maxYnum / Dm * Dy);

                g.DrawLine(Bp, 50, 85 + maxYnum / Dm * Dy, 55 + Dx * (c - 1), 85 + maxYnum / Dm
* Dy);
```

```
                Point[] p = new Point[c];
                for (int i = 0; i < c; i++)
                {
                    p[i].X = 50 + Dx * i;
                    p[i].Y = 85 + maxYnum / Dm * Dy - (list[c - i - 1].Temp) * Dy / Dm;
                }
                g.DrawCurve(Rp, p, 1.5F);
                for (int i = 0; i < c; i++)
                {

                    g.DrawString(list[c - i - 1].Temp.ToString(), font, Bluebrush, p[i].X, p[i].Y - 10);

                    g.DrawRectangle(Rp, p[i].X - 1, p[i].Y - 1, 2, 2);
                }

                g.DrawString("Temperature value", Tfont, brush, 15, 65);

                if (list != null && list.Count > 0)
                {
                    g.DrawString("Record time:" + list[0].ReciveTime.ToShortDateString() + "
Average value:" + tempCount / list.Count, Tfont, brush, 45, 85 + maxYnum / Dm * Dy + 15);
                }
                MemoryStream sm = new MemoryStream();
                img.Save(sm, ImageFormat.Jpeg);

                Response.Clear();
                Response.ContentType = "image/jpeg";
                Response.BinaryWrite(sm.ToArray());
            }
            else
            {
                Page.RegisterStartupScript("msg", "<script>alert('Please login
firstly!');window.location.href='Default.aspx';</script>");
            }


        }
    }
}
```

DrawImage.aspx.Designer.cs

```
namespace SensorSolution
{


    public partial class DrawImage
    {
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;
    }
}
```

Hisotry.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="History.aspx.cs"
Inherits="SensorSolution.WebForm1" %>

<%@ Register Assembly="System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"
    Namespace="System.Web.UI" TagPrefix="asp" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
      <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
     <table width="100%" cellpadding="0" cellspacing="0">
         <tr>
          <td height="50%" valign="top">
           <fieldset>
            <legend>Search Area</legend>
            <table width="100%" cellpadding="0" cellspacing="0" border="1">
             <tr>
              <td align="right">IP address(XXX.XXX.XXX.XXX)： ĝo</td>
              <td align="left">
               <asp:TextBox ID="SensorIPTxt" runat="server"></asp:TextBox>
```

```
                </td>
               </tr>
               <tr>
                <td align="right">Date from(YYYY-MM-DD HH-MM-SS)： ê¤</td>
                <td align="left">
                 <asp:TextBox ID="SensorDateFromTxt" runat="server"></asp:TextBox>
                </td>
               </tr>
               <tr>
                <td align="right">Date end(YYYY-MM-DD HH-MM-SS)： ê¤</td>
                <td align="left">
                 <asp:TextBox ID="SensorDateEndTxt" runat="server"></asp:TextBox>
                </td>
               </tr>
               <tr>
                <td align="center" colspan="2">
                 <asp:Button ID="SearchBtn" runat="server" Text="Search" Width="77px"
                          onclick="SearchBtn_Click" />
                </td>

               </tr>
              </table>

             </fieldset>
            </td>
           </tr>
           </table>
              <div id="Div1" runat="server" style="visibility:hidden">
               <a href="/123.xml">Download Records in XML format</a>
               </div>

               <asp:DataGrid id="xmldata" runat="server" />
      </form>
</body>
</html>
```

History.aspx.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using SensorWeb.DataContext;
using SensorWeb.Entity;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;
using System.Text;
using System.Data;

namespace SensorSolution
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        private static string HistoryResult = "";
        private   string result = "";
        private SensorDataContext sensorDataContext = new SensorDataContext();
        List<SensorData> list = new List<SensorData>();
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["UserID"] != null && Session["UserID"].ToString().Trim() != "")
            {
                string xmlFilePath = Server.MapPath("~/123.xml");
                if (File.Exists(xmlFilePath))
                {
                    File.Delete(xmlFilePath);

                }
            }
            else
            {
                Page.RegisterStartupScript("msg", "<script>alert('Please log in
firstly!');window.location.href='Default.aspx';</script>");
            }
        }

        protected void SearchBtn_Click(object sender, EventArgs e)
        {
            string xmlFilePath = Server.MapPath("~/123.xml");
```

```
                int searchType = -1;
                    if (!string.IsNullOrEmpty(this.SensorIPTxt.Text)
&& !string.IsNullOrEmpty(this.SensorDateFromTxt.Text)
&& !string.IsNullOrEmpty(this.SensorDateEndTxt.Text))
                    {
                        searchType = 2;
                    }
                    else
                    {
                        Page.RegisterStartupScript("msg", "<script>alert('Please input an IP address of a
sensor! And please input a serach time range');</script>");
                        searchType = -1;
                        return;
                    }

                #region
                switch (searchType)
                {
                    case 2:
                        result = sensorDataContext.GetSensorHistoryByIP(this.SensorIPTxt.Text,
this.SensorDateFromTxt.Text, this.SensorDateEndTxt.Text, xmlFilePath);
                       // this.ResultLbl.Text = result;
                         DataSet objDataSet = new DataSet();
                        DataTable dt = new DataTable();
                        objDataSet.ReadXml(xmlFilePath);
                        xmldata.DataSource = objDataSet.Tables["Sensor"].DefaultView;
                        xmldata.DataBind();
                        Div1.Style.Value = "display";
                        break;

                }

                #endregion
            }


    }
}
```

History.aspx.Designer.cs

```
namespace SensorSolution {

    public partial class WebForm1 {

        protected global::System.Web.UI.HtmlControls.HtmlForm form1;

        protected global::System.Web.UI.ScriptManager ScriptManager1;

        protected global::System.Web.UI.WebControls.TextBox SensorIPTxt;

        protected global::System.Web.UI.WebControls.TextBox SensorDateFromTxt;

        protected global::System.Web.UI.WebControls.TextBox SensorDateEndTxt;

        protected global::System.Web.UI.WebControls.Button SearchBtn;

        protected global::System.Web.UI.HtmlControls.HtmlGenericControl Div1;

        protected global::System.Web.UI.WebControls.DataGrid xmldata;
    }
}
```

Logout.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Logout.aspx.cs"
Inherits="SensorSolution.Logout" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
```

```
    </div>
    </form>
</body>
</html>
```

Logout.aspx.cs

```
using System;
using System.Collections.Generic;

using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace SensorSolution
{
    public partial class Logout : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Session.Clear();
            Server.Transfer("Default.aspx");
        }
    }
}
```

Logout.aspx.Designer.cs

```
namespace SensorSolution
{

    public partial class Logout
    {
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;
    }
}
```

Web.config

```
<?xml version="1.0"?>
<configuration>
      <appSettings>

    <add key="SQLCONN"
value="server=TIANYE-C31CA0FE\SQLEXPRESS;database=SensorDB;uid=sa;pwd=123456"/>

          <add key="Dx" value="50"/>

          <add key="Top" value="10"/>

    <add key="cmd" value="GetTemp"/>
     </appSettings>
     <connectionStrings/>
     <system.web>
          <compilation debug="true">
               <assemblies>
                    <add assembly="System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/></assemblies></compilation>

          <authentication mode="Windows"/>

    <httpHandlers>
       <remove verb="*" path="*.asmx"/>
       <add verb="*" path="*.asmx" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
       <add verb="*" path="*_AppService.axd" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
       <add verb="GET,HEAD" path="ScriptResource.axd"
type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35" validate="false"/>

    </httpHandlers>
     </system.web>
</configuration>
```

Codes in DataContext folder

InviteDataContext.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Data;
using SensorWeb.Entity;
namespace SensorWeb.DataContext
{
    public class InviteDataContext
    {
        public void DeleteInvite(InviteInfo entiy)
        {
            string sqlStr = string.Format("DELETE FROM [InviteInfo] WHERE InviteNum={0}",
entiy.InviteNum);
            int result = SqlHelper.ExecuteNonQuery(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, sqlStr);
        }

        public bool IsExist(InviteInfo entiy)
        {
            try
            {
                string sqlStr = string.Format("SELECT ID FROM [InviteInfo] WHERE
InviteNum={0}", entiy.InviteNum);
                int result = (int)SqlHelper.ExecuteScalar(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, sqlStr);
                if (result > 0)
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
            catch
            {
                return false;
            }
        }
```

```
    }
}
```

SensorDataContext.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Data;
using SensorWeb.Entity;
using System.Xml;
using System.Data;
using System.IO;
using System.Text;
namespace SensorWeb.DataContext
{

    public class SensorDataContext
    {

        public List<SensorData> GetSensorDataByIP(string IP)
        {
            List<SensorData> list = new List<SensorData>();
            string strSql = string.Format("SELECT * FROM [SensorDataInfo] WHERE IP='{0}'
ORDER BY ReceiveTime ASC", IP);
            DataSet ds = SqlHelper.ExecuteDataset(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, strSql);
            if (ds != null && ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)
            {

                for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
                {

                    SensorData entiy = new SensorData();
                    entiy.ID=int.Parse(ds.Tables[0].Rows[i]["ID"].ToString());
                    entiy.SensorIP = ds.Tables[0].Rows[i]["IP"].ToString();
                    entiy.Temp=int.Parse(ds.Tables[0].Rows[i]["Temp"].ToString());

entiy.ReciveTime=DateTime.Parse(ds.Tables[0].Rows[i]["ReceiveTime"].ToString());
                    list.Add(entiy);
                }
```

```
            }
            return list;
        }

        public List<SensorData> GetSensorDataByIP(string IP,int top)
        {
            List<SensorData> list = new List<SensorData>();

            string strSql = string.Format("SELECT TOP {1} * FROM [SensorDataInfo] WHERE
IP='{0}' ORDER BY ReceiveTime DESC", IP, top);
            DataSet ds = SqlHelper.ExecuteDataset(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, strSql);
            if (ds != null && ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)                {

                for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
                {
                    SensorData entiy = new SensorData();
                    entiy.ID = int.Parse(ds.Tables[0].Rows[i]["ID"].ToString());
                    entiy.SensorIP = ds.Tables[0].Rows[i]["IP"].ToString();
                    entiy.Temp = int.Parse(ds.Tables[0].Rows[i]["Temp"].ToString());
                    entiy.ReciveTime =
DateTime.Parse(ds.Tables[0].Rows[i]["ReceiveTime"].ToString());
                    list.Add(entiy);
                }
            }
            return list;
        }


        public int GetSensorPortByIP(string IP)
        {
            int port = 0 ;
            try
            {
                string sqlStr = string.Format("SELECT Port FROM [SensorInfo] WHERE IP='{0}'",
IP);
                port = (int)SqlHelper.ExecuteScalar(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, sqlStr);
                return port;
            }
            catch
```

```
            {
                return port;
            }
        }


        public string GetSensorHistoryByIP(string IP, string from, string end, string xmlFilePath)
        {
            if (File.Exists(xmlFilePath))
            {
                File.Delete(xmlFilePath);
            }
            string sqlStr = string.Format("SELECT * FROM [SensorDataInfo] WHERE IP='{0}' AND
ReceiveTime between '{1}' and '{2}' ", IP, from, end);

            XmlDocument doc = new XmlDocument();
            XmlNode docNode = doc.CreateXmlDeclaration("1.0", "UTF-8", null);
            doc.AppendChild(docNode);
            XmlElement root = doc.CreateElement("Sensors");
            doc.AppendChild(root);
            doc.Save(xmlFilePath);

            string result = "";
            IDataReader dr = SqlHelper.ExecuteReader(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, sqlStr);
            while (dr.Read())
            {
                result += "<tr><td>" + dr.GetString(1) + "</td></tr>" + "<tr><td>" +    dr.GetValue(2)
+ "</td></tr>" + "<tr><td>" + dr.GetDateTime(3) + "</td></tr><tr></tr>";
                string a = dr.GetString(1);
                string b = dr.GetValue(2) +"";
                string c = dr.GetDateTime(3) +"";


                XmlElement eltBook = doc.CreateElement("Sensor");
                root.AppendChild(eltBook);

                XmlElement eltTitle = doc.CreateElement("SensorIP");
                eltTitle.AppendChild(doc.CreateTextNode(a));
                eltBook.AppendChild(eltTitle);
```

```
                XmlElement eltAuthor = doc.CreateElement("Temperature");
                eltAuthor.AppendChild(doc.CreateTextNode(b));
                eltBook.AppendChild(eltAuthor);

                XmlElement eltPrice = doc.CreateElement("ReceivedTime");
                eltPrice.AppendChild(doc.CreateTextNode(c));
                eltBook.AppendChild(eltPrice);
                doc.Save(xmlFilePath);

                }

            return result;

        }

        public void InsertSensorData(SensorData entiy)
        {
            string strSql = string.Format("INSERT INTO SensorDataInfo VALUES ('{0}',{1},'{2}')",
entiy.SensorIP, entiy.Temp, entiy.ReciveTime);
            SqlHelper.ExecuteNonQuery(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, strSql);

        }
    }
}
```

SensorInfoDataContext.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Data;
using SensorWeb.Entity;
namespace SensorWeb.DataContext
{
    public class SensorInfoDataContext
    {

        public SensorInfo GetSensorInfoBySensorIP(string IP)
        {
            SensorInfo entiy = new SensorInfo();
```

```
                string strSql = string.Format("SELECT * FROM [SensorInfo] WHERE IP='{0}'", IP);
                DataSet ds = SqlHelper.ExecuteDataset(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, strSql);
                if (ds != null && ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)
                {
                    entiy.Port = int.Parse(ds.Tables[0].Rows[0]["Port"].ToString());
                    entiy.SensorIP = IP;
                }
                return entiy;
            }
        }
}
```

SensorTypeDataContext.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Data;
using SensorWeb.DataContext;
using SensorWeb.Entity;
namespace SensorWeb.DataContext
{

    public class SensorTypeDataContext
    {

        public type GetSensorTypeByIP(string IP)
        {
            try
            {
                string sqlStr = string.Format("SELECT Type FROM [SensorTypeInfo] WHERE
IP='{0}'", IP);
                int result = (int)SqlHelper.ExecuteScalar(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, sqlStr);
                type _status = type.unknow ;
                switch (result)
                {
                    case 1:
                        _status = type.HP;
                        break;
                    case 2:
```

```
                    _status = type.LoWPAN;
                    break;
                }
                return _status;
            }
            catch
            {
                return type.unknow;
            }
        }

    }
}
```

UserDataContext.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Data;
using SensorWeb.Entity;
namespace SensorWeb.DataContext
{

    public class UserDataContext
    {

        public int AddUser(UserInfo entiy)
        {
            string sqlStr = string.Format("INSERT INTO [UserInfo] VALUES ('{0}','{1}','{2}')",
entiy.UserName, entiy.Pwd, entiy.Email);
            return SqlHelper.ExecuteNonQuery(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, sqlStr);
        }

        public int Login(UserInfo entiy)
        {
            try
            {
                string sqlStr = string.Format("SELECT ID FROM [UserInfo] WHERE UserName='{0}'
AND Pwd='{1}'", entiy.UserName, entiy.Pwd);
```

```
                int result = (int)SqlHelper.ExecuteScalar(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, sqlStr);
                if (result > 0)
                {
                    return result;
                }
                else
                {
                    return 0;
                }
            }
            catch
            {
                return 0;
            }
        }

        public bool IsExist(UserInfo entiy)
        {
            try
            {
                string sqlStr = string.Format("SELECT ID FROM [UserInfo] WHERE
UserName='{0}'", entiy.UserName);
                int result = (int)SqlHelper.ExecuteScalar(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, sqlStr);
                if (result > 0)
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
            catch
            {
                return false;
            }
        }
    }
}
```

Codes in Entity folder

InviteInfo.cs

```
using System;
using System.Collections.Generic;
using System.Web;

namespace SensorWeb.Entity
{

    [Serializable]
    public class InviteInfo
    {
        #region
        private int _id;

        public int ID
        {
            get { return _id; }
            set { _id = value; }
        }
        private int _inviteNum;

        public int InviteNum
        {
            get { return _inviteNum; }
            set { _inviteNum = value; }
        }
        #endregion
    }
}
```

SensorData.cs

```
using System;
using System.Collections.Generic;
using System.Web;
```

```
namespace SensorWeb.Entity
{

    [Serializable]
    public class SensorData
    {
        #region
        private int _id;

        public int ID
        {
            get { return _id; }
            set { _id = value; }
        }
        private string _sensorIP;

        public string SensorIP
        {
            get { return _sensorIP; }
            set { _sensorIP = value; }
        }

        private int _temp;

        public int Temp
        {
            get { return _temp; }
            set { _temp = value; }
        }
        private DateTime _reciveTime;

        public DateTime ReciveTime
        {
            get { return _reciveTime; }
            set { _reciveTime = value; }
        }
        #endregion
    }
}
```

SensorInfo.cs

```
using System;
using System.Collections.Generic;
using System.Web;

namespace SensorWeb.Entity
{
    [Serializable]
    public class SensorInfo
    {
        private int _id;
        public int ID
        {
            get { return _id; }
            set { _id = value; }
        }
        private string _sensorIP;
        public string SensorIP
        {
            get { return _sensorIP; }
            set { _sensorIP = value; }
        }

        private int _port;
        public int Port
        {
            get { return _port; }
            set { _port = value; }
        }
    }
}
```

SensorType.cs

```
using System;
using System.Collections.Generic;
using System.Web;

namespace SensorWeb.Entity
{
```

```
    [Serializable]
    public class SensorType
    {
        #region
        private int _id;

        public int ID
        {
            get { return _id; }
            set { _id = value; }
        }
        private string _sensorIP;

        public string SensorIP
        {
            get { return _sensorIP; }
            set { _sensorIP = value; }
        }

        private type _type;

        public type Type
        {
            get { return _type; }
            set { _type = value; }
        }
        #endregion
    }
}
```

Type.cs

```
using System;
using System.Collections.Generic;
using System.Web;

namespace SensorWeb.Entity
{

    public enum type
```

```
    {
        LoWPAN,
        HP,
        unknow
    }
}
```

UserInfo.cs

```
using System;
using System.Collections.Generic;
using System.Web;

namespace SensorWeb.Entity
{

    [Serializable]
    public class UserInfo
    {
        #region
        private int _id;

        public int ID
        {
            get { return _id; }
            set { _id = value; }
        }
        private string _userName;

        public string UserName
        {
            get { return _userName; }
            set { _userName = value; }
        }
        private string _pwd;

        public string Pwd
        {
            get { return _pwd; }
            set { _pwd = value; }
        }
```

```
        private string _email;

        public string Email
        {
            get { return _email; }
            set { _email = value; }
        }
        #endregion
    }
}
```

TRITA-ICT-EX-2011:198

www.kth.se