UNIVERSITY OF VAASA

FACULTY OF TECHNOLOGY

TELECOMMUNICATIONS ENGINEERING

Markus Sjöblom

REMOTE MONITORING OF INDUSTRIAL FREQUENCY CONVERTERS

Master´s thesis for the degree of Master of Science in Technology submitted for inspection at Vaasa, 1st of September 2008.

Supervisor Mohammed Elmusrati

Instructor Stefan Strandberg

TABLE OF CONTENTS page

ABBREVIATIONS

AC	Alternating Current
ADSL	Asymmetric Digital Subscriber Line
API	Application Programming Interface
DC	Direct Current
DHCP	Dynamic Host Configuration Protocol
FC	Frequency Converter
FTP	File Transfer Protocol
GSM	Global System for Mobile communications
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDL	Interface Definition Language
IGBT	Insulated-Gate Bipolar Transistor
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
MAC	Media Access Control
NAT	Network Address Translator
NDR	Network Data Representation
PLC	Programmable Logic Controller
PPTP	Point to Point Tunneling Protocol
PWM	Pulse Width Modulation
RAM	Random Access Memory
RPC	Remote Procedure Call
SHA	Secure Hash Algorithm
SMS	Short Message Service
SDRAM	Synchronous Dynamic Random Access Memory
SOAP	Simple Object Access Protocol
SRAM	Static Random Access Memory
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
WAP	Wireless Application Protocol
WLAN	Wireless Local Area Network
WSDL	Web Service Definition Language
XDR	eXternal Data Representation
XML	eXtensible Markup Language

UNIVERSITY OF VAASA
Faculty of technology
Author: Markus Sjöblom
Topic of the thesis: Remote Monitoring of Industrial Frequency Converters
Supervisor: Mohammed Elmusrati
Instructor: Stefan Strandberg
Degree: Master of Science in Technology
Department: Department of Computer Science
Degree Programme: Degree Programme in Information Technology
Major of Subject: Telecommunication Engineering
Year of Entering the University: 2003
Year of Completing the Thesis: 2008 **Pages:** 69

ABSTRACT:
Frequency converters are sometimes mounted at places, in which they have the most processing power of the surrounding devices. Often the remote monitoring of frequency converters has been implemented in an external programmable logic controller PC which is connected to the Internet. Sometimes it is not reasonable to use an extra computer at the location, which means the remote connection should be directly connectible to the frequency converter. This master's thesis studies the possibility to use SOAP for a remote connection, established from the frequency converter to an external database server. The objective is to create a remote monitoring connection which is easily deployable for the end user. The implementation considers compatibility issues with firewalls, proxy servers and NAT routers.

KEYWORDS: Remote monitoring, SOAP, Frequency converter

VAASAN YLIOPISTO
Teknillinen tiedekunta
Tekijä: Markus Sjöblom
Diplomityön nimi: Taajuusmuuttajien etämonitorointi
Valvojan nimi: Mohammed Elmusrati
Ohjaajan nimi: Stefan Strandberg
Tutkinto: Diplomi-insinööri
Laitos: Tietotekniikan laitos
Koulutusohjelma: Tietotekniikan koulutusohjelma
Suunta: Tietoliikennetekniikka
Opintojen aloitusvuosi: 2003
Diplomityön valmistumisvuosi: 2008 **Sivumäärä:** 69

TIIVISTELMÄ:
Taajuusmuuttajia asennetaan joskus paikkoihin, joissa ne ovat laskentateholtaan ympäristön tehoikkaimpia laitteita. Tällaisissa ympäristöissä taajuusmuuttajien etämonitorointi on usein toteutettu erillisellä PLC-tietokoneella, johon ulkoverkosta voidaan ottaa yhteys. Joissakin tapauksissa PLC:n hankinta kohteen automaatiojärjestelmään on verrattain kallista, jolloin etäyhteys olisi järkevää hoitaa suoraan taajuusmuuttajaan, joka on kytköksissä Internetiin. Diplomityössä tutkitaan SOAP-tiedonsiirtoprotokollan käyttöä taajuusmuuttajien etäyhteyksien luonnissa. Tavoitteena on saada aikaan etäyhteys, jonka käyttöönotto on loppukäyttäjälle mahdollisimman vaivatonta. Toteutuksessa otetaan huomioon muunmuassa yhteensopivuus palomuurien, proxy-palvelinten, sekä NAT-reitittimien kanssa.

AVAINSANAT: Etämonitorointi, SOAP, Taajuusmuuttaja

1 INTRODUCTION

Frequency Converters (FCs) are widely used to drive motors of different sizes and characteristics in many types of environments. For example in small size heating stations frequency converters are used to control ventilation pumps and fuel supply shafts, while measuring inputs from the process through external inputs like Lambda-sensors. Nowadays FC's contain strong processors and external communication ports in a way that in many cases these devices are the most intelligent ones of the surrounding equipment. With customizable applications, the frequency converters can be integrated into many process environments so efficiently, that there is no need for an external Programmable Logic Controller (PLC).

The main idea for the thesis came from the needs of a team maintaining an unmanned heating station located at the centre of Vöyri municipality in Finland. The idea here is to reduce unnecessary maintenance visits to the station, caused by lack of information in fault alarms. The same team maintains several stations at nearby area. The maintenance team does not employ anyone fulltime, but consists of approximately 15 members of local community inhabitants that take part in maintenance at their own free time. Each of these members take turns in servicing the heating stations, one at a time. When on duty, the service-person is required to take actions when the fault alarm is received. The generation of the fault alarms at the stations is handled by a small PLC connected to the FCs and relative sensors. An external Global System for Mobile Communications (GSM) module handles the sending of a Short Message Service (SMS) message to the person on duty. Practice has shown that about 90% of the alarms would not have required an immediate visit to the

station because the process could have been left running even for several hours without any service actions. According to the maintenance team information, these unnecessary visits could be avoided, if the trend history of some critical parameters would be remotely monitored. This in other hand would allow the service-person to prioritize his own work and maybe delay his visit to the heating station for some time.

Currently the remote monitoring of frequency converters is commonly handled through an external PLC module because mostly the FCs cannot communicate with the outside networks without communication ports like Ethernet. It is predictable though that when the processing power and memory of the embedded system processors increases, these devices will be equipped with Ethernet and other optional communication ports besides the more traditional ones. This means that it is technically possible in near future to establish a direct remote connection to the FC without a relatively expensive PLC in between.

The purpose of this thesis is to find a remote monitoring solution that will allow the end user to view parameter history and fault information of the FCs in a way that the installation and up keeping of the system requires minimum effort from the end user. Another goal of the thesis is to make the connecting method so that the connection is immediately established upon plugging the Ethernet cable to the device and that the connection itself does not require anything more than a regular internet connection to the station. Connecting must not require any setups to default firewall settings of a traditional Asymmetric Digital Subscriber Line (ADSL)-router, a fixed Internet Protocol (IP)-address from the Internet Service Provider (ISP) or any other special setups. Ideally this would

mean that plugging the device to a standard ADSL-modem is all the setup the end user must do.

2 FREQUENCY CONVERTERS

A frequency converter is a power electronic device capable of transforming the frequency of Alternating Current (AC) and thus capable of controlling the rotational speed of the motor connected to its output. Frequency converters are typically used to control a three-phased induction motor, which without any rotation speed control would run at fixed speed all the time. In many cases it is unnecessary for the motor to run at full speed constantly, meaning that the use of a frequency converter increases controllability and lifetime of the motor while reducing the power consumption of the whole system at the same time. Frequency converters are available for most of today's three phase motor power classes, reaching all the way from few hundred Watts to several megawatts. (Doktar 2006)

2.1 Structure of a frequency converter

The basic components of a frequency converter usually include a *rectifier circuit*, a *middle circuit* and an *inverter circuit*. The purpose of the rectifier circuit, usually same as a diode bridge, is to transfer the alternating current into Direct Current (DC). Bad supply mains tend to cause major interference to the created DC voltage. This interference is removed from the DC signal at the middle circuit with condensers and chokes. After the interference filtration, the DC voltage is converted back to AC by the inverter circuit.

In modern low voltage frequency converters the inverter circuits are mostly based on Insulated-Gate Bipolar Transistor (IGBT) technology and IGBT-modules, containing several transistor/diode pairs. Transistors of the IGBT-

module are separately controlled with individual Pulse Width Modulation (PWM) signals. Thus the PWM signals determine the output frequency of the AC signal that rotates the motor. Figure 1 assembles the basic components of a frequency converter. (Wikipedia 2008)

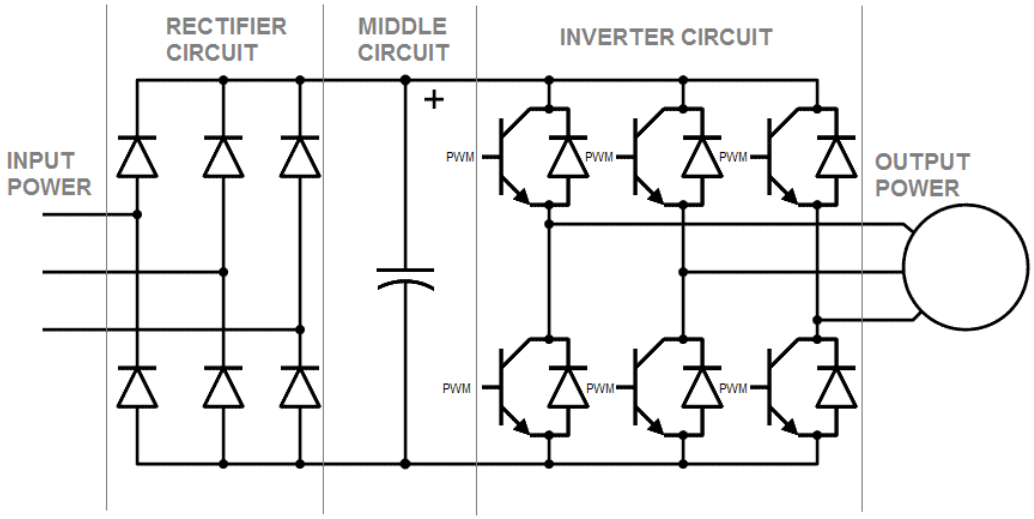


Figure 1. Basic components of a frequency converter.

In addition to the basic components, frequency converters usually contain a lot of other mandatory parts, such as a control system, built on top of a microprocessor, and some communication interfaces like serial ports and I/O's. A practical assembly for a frequency converter is shown in Figure 2. The Vacon NX drive in Figure 2 contains a control module for which the customer can build tailored applications to ensure perfect integration for multiple types of processes. Interfaces for control panel and extension boards are also included in addition to the basic FC functionality. (Vacon 2007)

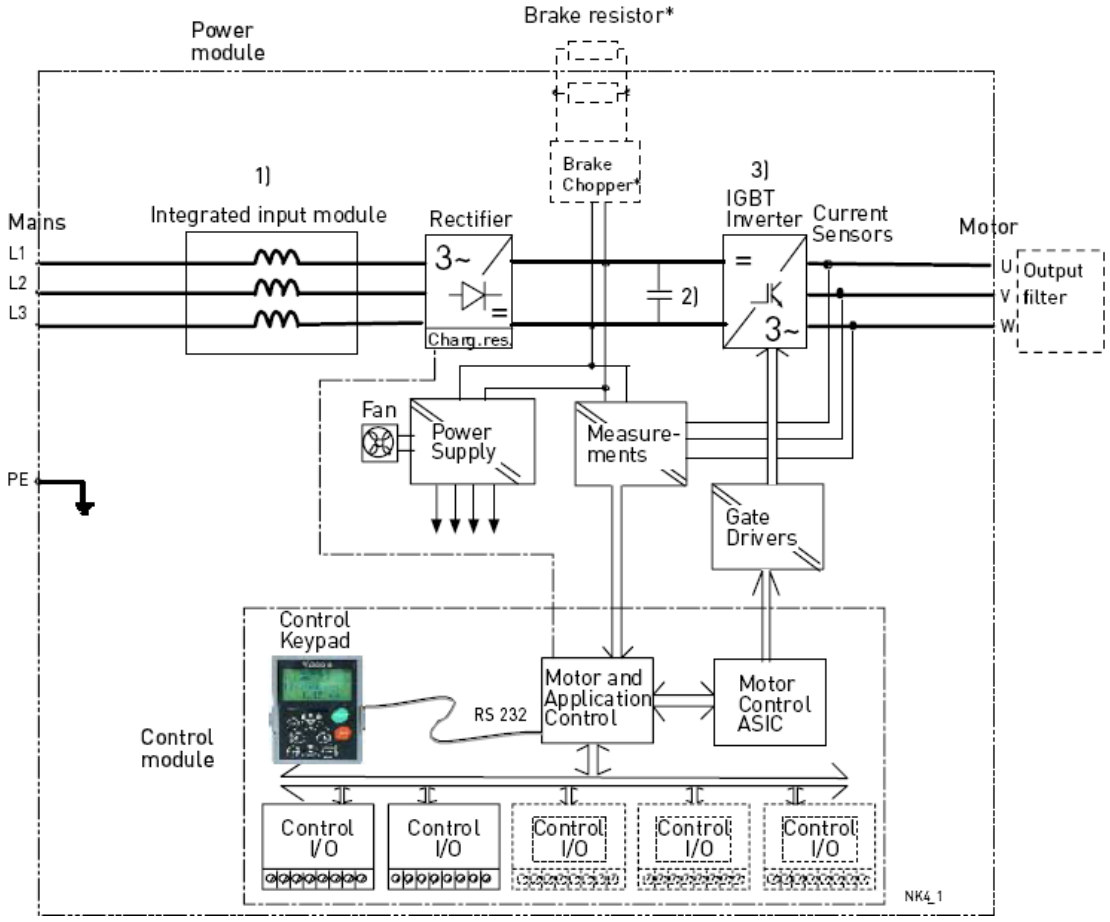


Figure 2. Vacon NX block diagram. (Vacon 2007)

2.2 Frequency converter control and communication interfaces

A frequency converter usually contains a complex system of electronics and software so that the device would offer maximum controllability and monitoring capability characteristics. The motor control unit is based on a piece of software loaded into a microprocessor. It controls the output AC power flow by performing sophisticated algorithms, which are influenced by measurements, parameter setups and external signals. Modern FC software is already so advanced that it is even possible to handle the entire process flow without an external PLC module. FCs are typically externally controlled

through a panel, I/O interface, Fieldbus or some other communication interface, like Human Machine Interface (HMI) (Zhang 2006).

Furthermore these same communication interfaces can be used for process monitoring purposes. It is possible, for instance, to collect measurement data from the driven motor through a wireless Zigbee-sensor network, and then send the collected data to a centralized diagnostics server through Ethernet (Tiainen 2006). Of course this kind of technology would require many optional peripheral interfaces on top of the basic functionalities of the FC, and the same goes for the other devices in the system too. However there are plenty of investigations going on which are focused entirely on developing and investigating frequency controller communications (Zhang 2006). At the beginning of 21st century, for instance, Wireless Application Protocol (WAP) technology was already integrated into a FC, thus enabling wireless remote monitoring of the drive (Ojanperä 2000). Frequency converter manufacturers are bringing emerging technologies to their products, so it is probably just a matter of time when Ethernet or Wireless Local Area Network (WLAN) is a stock feature in a standard FC model.

2.3 Ethernet in frequency converters

Ethernet is a great competitor to a standard serial line communication. It enables multiple simultaneous connections to a single physical port. If a device requires multiple communication protocols, like HMI and Fieldbus, to be communicating at the same time, it is possible to run these protocols over a single Ethernet port, while running the same protocols through serial port

would require multiple physical interfaces from the device. Ethernet could save space as well as costs if it was used instead of serial line communications.

Nowadays Ethernet is more likely to be used for supporting traditional communication ports rather than as a primary interface for external access. It is already available to several manufacturers FC models (e.g. ABB, Falcon and Vacon), at least as an optional feature (ABB 2008, Falcon 2008, Vacon 2008). Frequency converters contain large number of parameters and monitoring values that can be externally accessed from outside the device. The majority of modern frequency converters offer support for some Fieldbus protocol, through which the drive's data can be handled. One of the Ethernet based Fieldbus protocols that have gained popularity among FC manufacturers is Modbus TCP. Table 1 contains an example demonstration of the monitoring values and data types, which can be read in real time from a Vacon frequency converter through the Modbus TCP protocol. Values such as motor torque, power and speed can easily be accessed through Fieldbus. Outside the table, access to monitoring the device's fault history and parameter setup is also possible through Vacon's Modbus TCP registers. (Vacon 2008)

Fieldbusses are meant to be used mostly in industrial local area networks (LAN) and are not suitable for remote monitoring purposes outside the local network. Modbus TCP for example does not contain any security mechanisms for preventing third party attacks against the device's data (Modbus-IDA 2006). The purpose of this Master's thesis is to find a suitable communication protocol and methods that enable remote monitoring of FC's values. The final solution must also enable remote controlling of the devices parameters in the future as

well as take into account FC's hardware and software properties and the network environment (firewalls etc.) near the frequency converter.

Table 1. Monitoring values(Vacon 2008)

Address	Purpose	Type
10301	MotorTorque	Integer
10302	MotorPower	Integer
10303	MotorSpeed	Integer
10304	FreqOut	Integer
10305	FreqRef	Integer
10306	REMOTEIndication	Unsigned short
10307	MotorControlMode	Unsigned short
10308	ActiveFault	Unsigned short
10309	MotorCurrent	Unsigned integer
10310	MotorVoltage	Unsigned integer
10311	FreqMin	Unsigned integer
10312	FreqScale	Unsigned integer
10313	DCVoltage	Unsigned integer
10314	MotorNomCurrent	Unsigned integer
10315	MotorNomVoltage	Unsigned integer
10316	MotorNomFreq	Unsigned integer
10317	MotorNomSpeed	Unsigned integer
10318	CurrentScale	Unsigned integer
10319	MotorCurrentLimit	Unsigned integer
10320	DecelerationTime	Unsigned integer
10321	AccelerationTime	Unsigned integer
10322	FreqMax	Unsigned integer
10323	PolePairNumber	Unsigned integer
10324	RampTimeScale	Unsigned integer
10325	MsCounter	Unsigned integer

3 REMOTE MONITORING OF EMBEDDED DEVICES

Nowadays there are a lot of embedded devices around us. These devices are used to control cars and trains, to survey greenhouses and to monitor the condition of patients at the local health center. Usually the devices contain a great deal of interesting data, which is sometimes needed to be remotely monitored. Some of these remote monitoring motivations and implementations will be discussed in this chapter.

3.1 Remote monitoring motivations

Devices with moving parts that can be worn down often require periodical maintenance and repairing or replacement of damaged parts. Sometimes these devices are placed at locations which are difficult or inconvenient to access, or where the inspections would require the entire system to be shut down. Often they are located at unmanned stations or at moving platforms, like ships, which require the maintenance operations to be carefully planned beforehand. Remote monitoring could enable surveying of these moving parts, thus helping the maintenance to avoid unnecessary service visits and to reduce additional shut down costs. Remote monitoring could enable surveillance of the devices directly from the manufacturer factory, thus enabling remote technical support to be given in cases that otherwise would require a visit from a specialized service person. In some cases by remotely monitoring the device, service needs could be determined and even the statuses of entire processes could be supervised. Among the few cases mentioned above, many other special motivations and needs exist for remote monitoring.

3.2 Local collecting point based solutions

Because memory sizes in embedded systems are often small and the processing capabilities are usually limited, connecting the device into some global network (e.g. internet or telephone) could be a challenging task. If the devices data needs to be remotely viewable, it should therefore be connected to the network in some other way. One often used solution is to collect the data from all the devices to a centralized collecting point by using Fieldbus protocols and a Fieldbus master for example. In many cases the Fieldbus master has an integrated web server and support for Ethernet. Masters are usually equipped with moderate memories, enabling local data harvesting, storage and monitoring to be handled from a single centralized computer connected to the internet. Especially in larger systems a centralized solution could also offer better network security properties compared to a solution, where each device handles its own remote monitoring connection.

An example of a centralized remote monitoring system is shown in Figure 3. In the practical solution, information from the local system is collected into one central computer by using Eaton *PowerNet* software for data collection. The central computer is also equipped with Questra *Service Agent* software, which then transports the collected data into Eaton's central data storage. Clients and maintenance personnel can now access the collected data via Eaton's service pages. (Questra 2004)

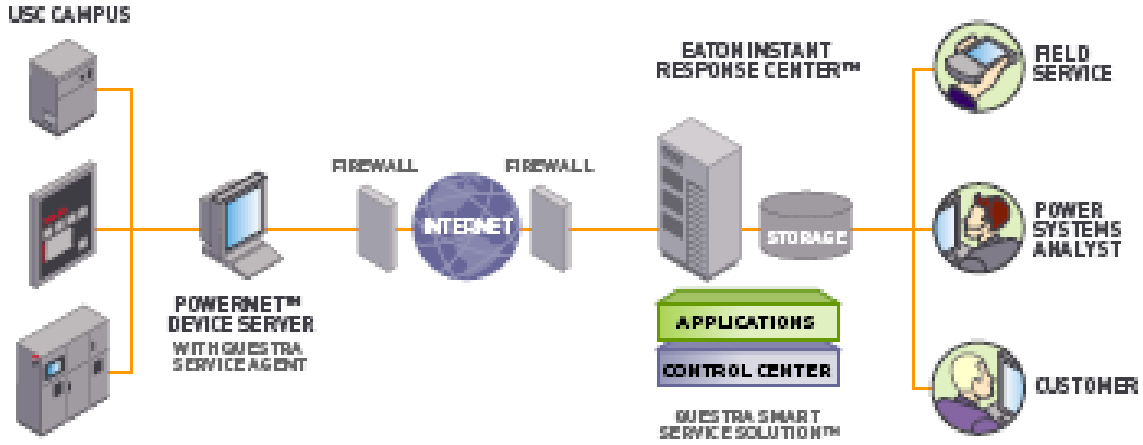


Figure 3. Questras remote monitoring solution. (Questra 2004)

However, often the extra costs of purchasing a PLC master or other centralized server might become relatively expensive especially for small systems, or for some other reasons the implementation of a centralized solution might just be too difficult. In these cases, a direct network connection for the device should be provided.

3.3 GSM based solutions

GSM radio based remote monitoring solutions have been widely used all over the world. Devices using GSM radio for external communications can usually be configured to sending alarms via SMS, and often allow parameter data viewing via a standard phone call connection. An embedded device can be made GSM compatible by integrating a radio chip directly to the devices hardware, or by using an external GSM module which can be controlled using for example standard serial port communications. An example of such a device is the in4ma GSM module, which enables storage of 40000 time stamped data

records, SMS and email alerts, and a web based remote monitoring connection (in4ma 2007).

GSM connections are well secured and the network is available throughout the world. Sometimes though a device might be installed into a location where wireless connectivity cannot be guaranteed. Such a place could for example be an elevator shaft, a basement, a mine or some other bad coverage location from a radio propagation point of view. One other downside with GSM radios is that each device would require a unique subscription from the service provider. This can lead to relatively expensive phone bills, when multiple devices are configured to sending periodical data records.

3.4 Ethernet based solutions

With Ethernet technology the devices can be globally connected to each other by IP addressing, and they can be uniquely identified from each other by their Media Access Control (MAC) addresses. With GSM routers, Internet connections could be provided even to remote locations, meaning that Ethernet based communications covers at least the same geographical area as GSM based remote monitoring solutions. One ISP provided internet connection can be shared among several users by using switches and routers equipped with Network Address Translators (NAT), which means that a single connection can be used to connect several simultaneous remote monitoring sessions. Many different Ethernet based remote monitoring solutions have been implemented for embedded devices nowadays. One often used solution is to run a web server on the device itself, and another approach is to form a connection into an

external server where the data will be stored in, and where it can be remotely accessed from later on.

3.4.1 Web server solutions

An often used approach to implement remote monitoring of a device through Ethernet is to run a web server on it. Some HTTP server softwares are available even for embedded devices, which can be configured to hosting some simple web pages. With such an approach, parameter monitoring and modifications can be made easily, without installing any special programs, using a PC and a standard web browser. An example of a commercial device implementing a web server based remote monitoring solution is ABB's *NETA-01* Ethernet module. Its web pages can be used to altering frequency converter parameters and to reading information about system status, fault logs and data logs (ABB 2007).

Implementing the web server based remote monitoring solution outside the local area network might turn out to be a challenging task. The data itself should reside on the device itself, which means that the devices memory capacity would set its own limits to the maximum size of the data storage. Other problems would be caused by firewalls, proxy servers and NAT's. A device that is located at NAT's sub network does not have its own public IP address at all, but it has a NAT allocated sub network address instead. From the outside network point of view, all the communications departing from a NAT router seems to be originated from a single device only (Ford 2005). Connection establishment from the internet to a web server located inside a NAT routers

sub network could turn out to be difficult task, and because many ISP provided ADSL connections are provided with a stock NAT router (Elisa 2006), a client based remote monitoring approach might be more convenient solution from the end users point of view.

3.4.2 Indirect monitoring

Many problems caused by NAT's and firewalls could be dodged, if the connection was opened from the device itself instead of that the connection request would come to the device from the internet. Numerous remote monitoring solutions have been implemented, in which the device connects to an external server, where it stores the data, and from which the data can be read afterwards with a standard web browser for example. The previously mentioned ABB Ethernet module for example uses email for sending monitor values to the server. Received values are then processed at the database server, and as a result of the process the maintenance personnel can be alarmed by different means (e.g. SMS messages) in problematic situations (ABB 2007). Figure 4 presents the main structure of the ABB *NETA-01* Ethernet module's remote monitoring solution implementation.

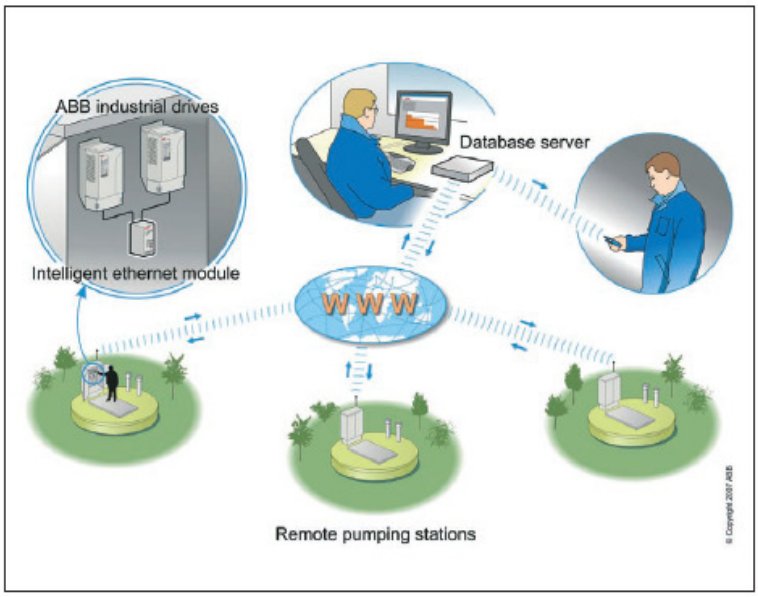


Figure 4. ABB NETA-01 remote monitoring. (ABB 2007)

In the above solution, email was used as the communication protocol between the frequency converter and the database. Many other protocols have been used in similar solutions to delivering data to the database server as well. Point-To-Point Tunneling Protocol (PPTP) (Clarke 2004) and File Transfer Protocol (FTP) (Kim 2000) among other communication protocols have been used for data transportation in existing implementations. The thesis will not cover all the communication protocols that can be used to establish remote monitoring connections, but a solution for one protocol will be presented in details.

4 REMOTE PROCEDURE CALL (RPC)

The roots of Remote Procedure Call (RPC) extend more than three decades back to the year 1976, when it was first introduced in RFC 707 (White 1976). RPC was designed as a *request/ response* protocol that mimics the procedural programming languages and was known as the *client/ server* model at the beginning. The idea of RPC was to balance the load in the network systems by distributing the process between the working computers. When designing these distributed systems, the researchers soon realized that the networking functionality should be hidden from the actual application. As a side effect of this layer model, the development of distributed systems became much easier.

RPC maps nicely to a function call in C/C++ languages. Similar to C functions, RPC is based on request/response procedure model. In C this means that after the function is called, the caller will transfer the execution control to the function and then wait for the function to complete its work before receiving the execution control back. Similarly to C, in RPC the function will return only after the remote procedure has finished and returned the result to the caller. This is the easiest way to manage concurrency in the application but does not mean that RPC would not come with a set of its own problems. Problems related to things such as remote connectivity, mutual wire representation of data, interface design for network bandwidth, and exception handling emerge. An example of an RPC call will be given in section 4.4 Example RPC Application. (Scribner 2000)

4.1 Remote endpoints

The first step in making an RPC call is to connect to the remote host. In TCP/IP the connection will be formed according to the IP address and port number. The client makes a query for a specific service to the RPC server, which then returns the port number of the requested service, if it has been registered to the Distributed Component Object Model (DCOM) endpoint mapper of the host. An example of a RPC call connection has been given in Figure 5. In the example, the client connects to the server's RPC port 135 and requests the port information of the email service. The server then returns the port number 1248 of the requested service to the client, which can then connect directly to the email service port of the host. (Sakellariadis 2008)

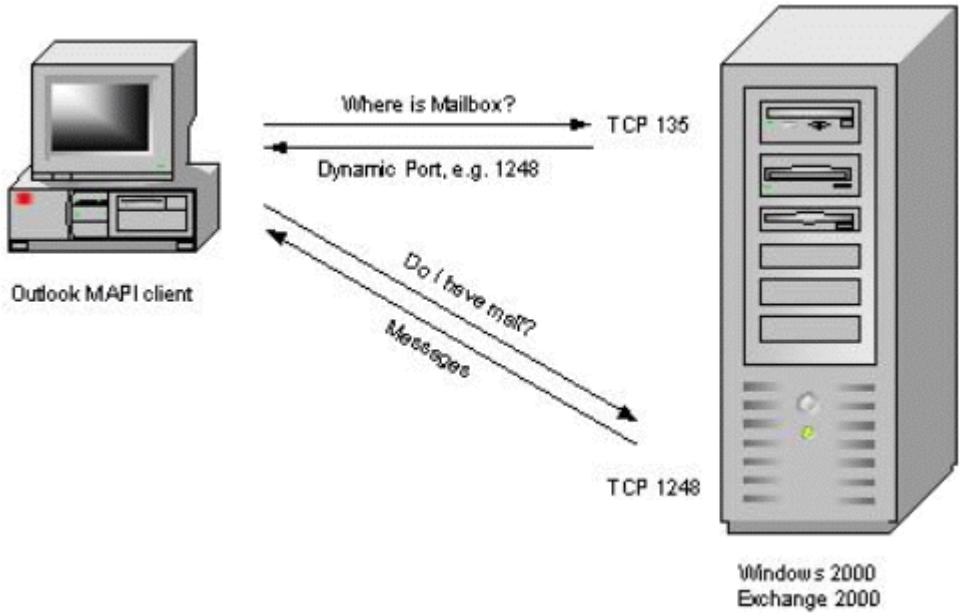


Figure 5. Example RPC query(Sakellariadis 2008)

4.2 Wire representation of data

In locally executed procedures, the parameters have been placed on the stack into unambiguous locations so that both the caller and the procedure have knowledge of the location and size of the parameters. These parameter placements have already been determined during the compiling of the code and stay intact during the life of the binary executable. A bond like this is difficult to make between two separate computers. This means that we have to have a common way to represent the data online so that both ends interpret and feed the data in correct order and format. For RPC there are a few representation methods such as the Network Data Representation (NDR) and Sun's eXternal Data Representation (XDR). The Interface Definition Language (IDL) was designed to be used for describing RPC interfaces. A demonstration example is shown in the Figure 6. (Scribner 2000)

```
[uuid(BCB31762-F16F-11d3-AB76-888888888888)]
interface myinterface {
        void withdraw_money(    [in]        uuid_t*   AccountID,
                                [in,out]    long*     Amount,
                                [out]       long*     TransactionNumber);
}
```

Figure 6. Interface Definition Language (Scribner 2000)

4.3 Interface design for networks

When using RPC it is very important to consider processing times, network congestion, system errors and slow transmission rates. It is also important to consider these matters at the design phase of the interfaces, so that for example an optimal balance between the number of remote calls and the payload of a

single call could be found. A good rule for designing a remote procedure system is that it works best for typical use rather than pathological situations. A good implementation is easy to use and serves the client in the most optimal way. (Scribner 2000)

4.4 Example RPC application

Storing of a parameter history over a long period of time in an embedded device memory is challenging. A parameter value with a size of four bytes, which is read every ten seconds, requires 24 bytes of memory storage for each recorded minute. Two weeks of parameter history for five parameters would then require memory space of

(2*7*24*60*60) (s) * (1/10) (1/s) * 5 * 4 bytes = 2419200 bytes.

Even powerful embedded device processors might have difficulties storing these amounts of data without external memory expansions. The Static Random Access Memory (SRAM) size of Atmel AT91RM9200 for example is only 16 kilobytes (ATMEL 2006). Processor memories could be expanded with an external memory such as Synchronous Dynamic Random Access Memory (SDRAM), but an alternative approach is to store the data outside the device, into an external database.

RPC technology allows routines to be driven at remote locations. An example of RPC functionality has been introduced in Figures 7 and 8. In the example code

of Figure 7 the parameter is normally stored to a parameter array located at device's Random Access Memory (RAM).

Machine A

```
int Parameterhistory[500];
int nextFreeslot=0;

void saveParameter(int Param)
{
        Parameterhistory[nextFreeslot]=Param;
        nextFreeslot++;
        nextFreeslot%=500;
}

int main()
{
        int parameter=1;

        //Save parameter to RAM
        saveParameter(parameter);

        return 0;
}
```

Figure 7. Parameter stored to local RAM.

When the array sizes are so large, that the physical restrictions of the memory sizes become a problem, the data should be stored somewhere else than into the device memory. By using RPC the parameters can be stored to a remote memory space. This kind of memory storage could for example be a MySQL database, located at a distant server. Figure 8 demonstrates identical parameter saving to Figure 7, but with RPC implementation now included.

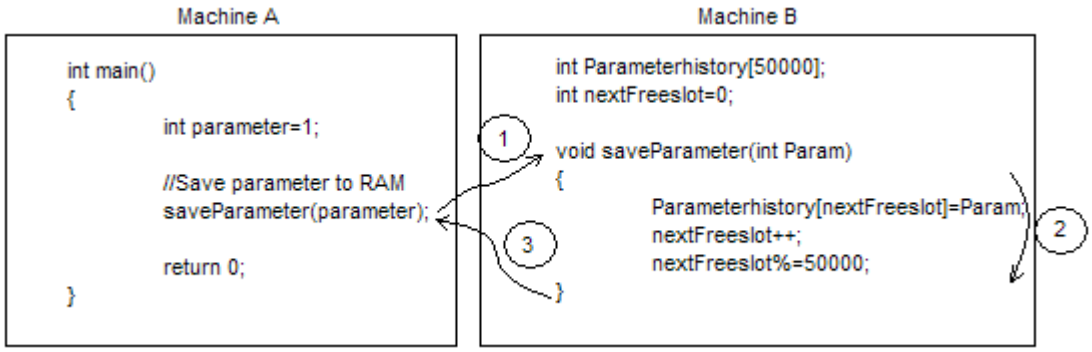


Figure 8. Parameter stored to external memory space by using RPC.

What is noticeable about the RPC usage is that the networking functionality has been completely hidden from the interface. For the application itself the *saveParameter* function looks completely same whether RPC is used or not. The processing time of the Figure 8 function might be longer than the one of Figure 7, but both of these functions block execution until the process has been completed.

5 SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

Port 135 of the RPC protocol, among the ports used by many other RPC based protocols, is often blocked by firewalls for security reasons especially in corporate networks. However, for application development it is important that programs could somehow communicate with each other through the internet. Unlike the previously mentioned RPC protocols, the communications for Hyper Text Transfer Protocol (HTTP) are usually allowed to go through most firewall and proxy server setups. Furthermore HTTP has been implemented for numerous different platforms and programming languages and for this reason it is a suitable communications protocol between multiple devices with different technologies. Unfortunately the protocol has been designed for delivering hyper text, and is not suitable for communications between applications as such. For this reason, the Simple Object Access Protocol (SOAP) specification was originally designed aside HTTP. (W3Schools 2008)

SOAP is a communications protocol that enables exchanging of eXtensible Markup Language (XML) based messages between applications. First SOAP specification was released 1999 by the name *SOAP: Simple Object Access Protocol Specification (version 1.0).* Later on version 1.1 of the protocol was released in 2000 and 1.2 in 2001 (W3C 2007). Compared to other RPC based protocols, SOAP's special characteristics are that it is based on XML language and that it fits well into today's modern internet infrastructure. Other special characteristics for the protocol are its firewall penetration ability, platform independency, programming language independency, simplicity and extensibility. Aside from the traditional RPC protocols, SOAP uses other application layer protocols for message transportation. It is mostly

recommended to use the protocol together with HTTP, from which SOAP inherits the firewall and proxy server penetration abilities. (W3Schools 2008)

5.1 Message structure

For each SOAP version, a W3C defined specification exists (W3C 2007). These specifications define for example that SOAP messages should be composed in XML format and that each message must contain at least an *envelope* element and a *body* element. SOAP messages must also contain sources for the used namespaces from which an accurate XML schema description for the elements can be found. W3C has defined several elements for SOAP, enabling rich and unambiguous communications between applications. Definitions for multiple data types, like *int*, *float* and *double*, have been made for each SOAP version, and are commonly available for public usage at W3C web site. As an example, the default namespace for the SOAP version 1.2 can be found at "http://www.w3.org/2001/12/soap-encoding".

One SOAP transaction can contain several elements in both query and response messages. The elements can be in the form of an array or a structure and furthermore an element can contain other elements in its substructure. This enables a very dynamical and extensible communication interface between the applications that are using SOAP for data transportation. It is also possible to create custom namespaces for the protocol, enabling the transportation of very unique structures and arrays between the endpoints. New transaction interfaces and message structures for SOAP can be described with the Web Service Definition Language (WSDL).

SOAP can be used together with multiple application layer communication protocols, but it is mostly recommended to be used together with HTTP. On top of a SOAP message, HTTP adds for example the required delivery information, so that a message can find its way to the right procedure call through the internet. An example of a simple SOAP query (upper) -, and response (lower) message tied to HTTP has been given below. The example SOAP RPC function *GetStockPrice* returns a stock price for the requested item. The client requested a price for an item IBM, to which the server responded with a price of 34.5. (W3Schools 2008)

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
     <soap:Body xmlns:m="http://www.example.org/stock">
          <m:GetStockPrice>
                    <m:StockName>IBM</m:StockName>
          </m:GetStockPrice>
     </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
     <soap:Body xmlns:m="http://www.example.org/stock">
          <m:GetStockPriceResponse>
                    <m:Price>34.5</m:Price>
          </m:GetStockPriceResponse>
     </soap:Body>
</soap:Envelope>
```

5.2 Web Services Description Language (WSDL)

WSDL is a description language built into XML format and can be used to describe the characteristics of a web service accurately. A WSDL file contains for example the information about the main characteristics, used communications ports and message data structures of a service. Furthermore it can be used to describe new customized namespaces and element types for the use of multiple types of procedure calls. WSDL is often used together with SOAP and the XML schema to describe web service content available on the internet. A client program can read the WSDL file, containing a list of the available services from the server, and then connect to these services by using SOAP as the communication protocol. A simple SOAP service description in WSDL form has been given below.

```
<message name="GetStockPriceRequest">
     <part name="StockName" type="xs:string"/>
</message>
<message name="GetStockPriceResponse">
     <part name="Price" type="xs:decimal"/>
</message>
<portType name="glossaryTerms">
     <operation name="GetStockPrice">
               <input message="GetStockPriceRequest"/>
               <output message="GetStockPriceResponse"/>
</operation>
</portType>
<binding type="glossaryTerms" name="b1">
     <soap:binding style="document"
     transport="http://schemas.xmlsoap.org/soap/http" />
     <operation>
          <soap:operation
               soapAction=" http://www.example.org/stock"/>
               <input>
                         <soap:body use="literal"/>
               </input>
               <output>
                         <soap:body use="literal"/>
               </output>
     </operation>
</binding>
```

The above example WSDL file was already referenced by an earlier example at section 5.1. The *GetStockPrice* operation of the example service takes a *StockName* parameter of type string as an input, and then returns a *Price* variable of type decimal as an output response. (W3schools 2008)

5.3 Integration with databases

SOAP Application Programming Interface (API) has been implemented into numerous web server environments which also include interfaces for database accessing. PhP for example is often used for web server programming and contains libraries for the handling of both SOAP- and MySQL messages. Because of these integrated libraries, it is relatively easy for the server side programmer to store the data, received in a SOAP message, into a database. There are some considerations to be taken into account when selecting an appropriate server software environment. The speed and efficiency of SOAP message parsing and database operations for example should be carefully considered when the group of clients grows into thousands of users and the amount of data to be stored is huge. Wrong software solutions might affect negatively into the performance and capacity of the system, which means that careful comparisons between different environments should be made. Although the purpose of this master's thesis is not to go through different web server technologies, an example implementation for one environment will still be introduced in chapter 6.

5.4 Security

The SOAP specification does not define any security for the communications. However, encryption and authentication are critical for many applications. For example the remote control of a frequency converter with an unsecured communications would expose the whole process to the threat of an attack. Since the SOAP messages are transported inside the data content of another application layer protocols, the security mechanisms of these protocols can be used for securing SOAP messages as well. For example HTTP communications can be secured together with Secure Sockets Layer (SSL), which means that SOAP can also be used together with Hyper Text Transfer Protocol Secure (HTTPS) (Wikipedia 2008b). However, the usage of HTTPS is not entirely problem free. When used together with proxy servers for example, the end-to-end security will be entirely lost (Wikipedia 2008b). If the message integrity should be kept all the way from the application to another, the content should be secured before delivering it to HTTP or HTTPS.

WS-Security is an Oasis-Open published and maintained standard, which defines methods for securing web services. WS-security defines how to attach signatures and security tokens into SOAP message headers in a way that end-to-end security can be reached (Wikipedia 2008b). By using WS-security, the SOAP messages can be encrypted efficiently enough so that the usage of SSL encryption is not mandatory anymore. This means that compatibility with proxies can be achieved, and thus secure SOAP communications on top of HTTP is possible (MSDN 2008).

5.4.1 Web Services Security authentication

WS-Security provides a UsernameToken profile for SOAP client authentication. Client password and username can be transported inside the SOAP message header either in plain text or in an encrypted form. Aside from plain text representation, encrypting the client password protects the system against reply attacks and thus improves system security.

UsernameToken profile specifies the user password to be coded using two altering values that are transported to the server inside the SOAP security message. These two values must never be the same twice, so that reply attacks can be efficiently blocked at server side. The profile specifies an algorithm of the form

$$\text{Password\_Digest} = \text{Base64}(\text{SHA-1}(\text{nonce}+\text{created}+\text{password})) \quad (1)$$

to be used when coding the clients password. Algorithm 1 takes a string of three variables as an input and gives a base-64 number as an output. The input string is formed of the user's plaintext *password*, encryption *creation time* and a *nonce* or number used once in other words. The given input string is hashed using the Secure Hash Algorithm (SHA) version 1, and then transformed into base-64 numerical format. The receiver has the user's plaintext password stored in some memory space, and upon receiving the authentication header the server does exactly the same operation with the same input values and compares the output to the sent, encrypted password. If the encrypted passwords match, the user has been authenticated. WS-Security

UsernameToken element will be transported inside the SOAP message header element in the following form. (OASIS 2004)

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu= "...">
     <S11:Header>
          <wsse:Security>
               <wsse:UsernameToken>
                    <wsse:Username>NNK</wsse:Username>
                    <wsse:Password Type="...#PasswordDigest">
                         weYI3nXd8LjMNVksCKFV8t3rgHh3Rw==
                    </wsse:Password>
                    <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
                    <wsu:Created>2003-07-16T01:24:32Z</wsu:Created>
               </wsse:UsernameToken>
          </wsse:Security>
     </S11:Header>
</S11:Envelope>
```

5.5 Software for embedded devices

Only a limited amount of memory is available when dealing with embedded devices, meaning that the memory consumption of the software should be carefully taken into account. Recently, a lot of research activities have been trying to deliver SOAP into embedded communications. SOAP has already been used even in small sensor network communications (Janêcek 2004), and small footprint SOAP software is commercially available (gSOAP 2008, eSOAP 2008).

6 REMOTE MONITORING WITH SOAP

This chapter describes in detail, how the actual implementation of remote monitoring is done for the frequency converters. Outlines for the software requirements were described earlier in the introduction chapter. According to the requirements, the remote monitoring software should be operational with minimum configuration and the implementation itself should suite the modern internet infrastructure as well as possible. This means that the solution should take into account operational environments, which might include devices such as NAT's, firewalls and proxy servers. The final implementation should also be as self acting as possible so that the setup required from end user can be minimized.

Remote monitoring software was implemented with SOAP communication protocol on Vacon OPT-CI Ethernet option card. The software enables remote monitoring of six freely selectable frequency converter parameters in one second intervals. Due to software dynamics, further additions such as fault information transport, remote control, or some other remote communication features are possible without major modifications to the current implementation.

6.1 SOAP messaging

SOAP was chosen as the remote monitoring protocol due to its dynamicity, extendibility, platform independency and modern internet infrastructure suitability. With SOAP messages the parameter data transport from the frequency converter to the database server and the data handling with different

kinds of server side software can be done relatively easy. Furthermore many server side software packages that support SOAP messaging also support some database languages. This means that the data processing from the actual message to a database cell is simple. SOAP has also been used in other remote monitoring solutions, such as the previously mentioned Questra solution, meaning that a fusion to other systems with moderate changes could also be possible in the future.

SOAP messaging is neither fast nor light (Davis 2002), so it is not very suitable for dense real time data transmission. For the remote monitoring implementation, transmission of six values every second produces only 30 data units every fifth second. All of these values can be fitted into one single SOAP message, which means that a new message would be generated every five seconds. This gives both the transmitting and the receiving side plenty of time to process the request and response messages, meaning that light density remote monitoring of parameters could be implemented with SOAP.

SOAP communication is based on XML, which means that the first step to data transportation is to mold it into XML format. Communication also requires some sort of authentication mechanism for identifying the client, and also some sort of identification to separate the devices from each other. Transmitting frequency converter monitoring values over the net is not supposed to be a highly critical matter, meaning that there is no need for encrypting the actual payload of the message. For other, more critical applications, the data could be encrypted by using HTTPS or WS-Security mechanisms. The following sections describe the structure of the transmitted SOAP message.

6.1.1 Data representation in XML format

Values that can be read from the frequency converter can be of very different types, depending on their actual use purposes on the device. In table 1 we see that the monitoring values consists of three different data types. Parameters which can contain both positive and negative values are represented in *integer* format, while exclusively positive values are represented in *unsigned integer* format. *Boolean* format can be used for representing the value of a digital input, and accurate decimal values can be represented in *float* format. The data type sizes can also vary from one single bit, all the way to 64 bits and even more. The actual data type of the value must thus be included in the SOAP message so that the receiver knows how to handle the data.

In Vacon frequency converters the remotely monitored values and parameters are called IDs. IDs on the FC are separated from each other with an ID number. Because the remote monitoring messaging was implemented in a way that one single SOAP message can contain multiple units from different moments, the transmitted ID unit must thus include an ID number, ID value and a timestamp from the record moment. With these elements, one single value could be represented in XML format in the following way.

```
<ID>
     <TimeStamp>12:00:00.00</TimeStamp>
     <IDNumber>1</IDNumber>
     <IDValue>100</IDValue>
</ID>
```

On the example above, the ID 1 value at 12 o'clock was 100, but the representation does not yet show any types of individual elements. XML Schema documentation standard can be used for describing different kinds of

XML structures and types. With XML Schema documentation, unique namespaces together with custom element structures and data descriptions can be unambiguously used to describe XML data. The previous ID unit can be described with XML Schema *complexType* structure in following way.

```
<xsd:schema tns='www.example.com'
     xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
     <xsd:complexType name='ID'>
          <xsd:all>
               <xsd:element name='TS' type='xsd:time'/>
               <xsd:element name='Nr' type='xsd:int'/>
               <xsd:element name='ERROR' type='xsi:string'
                    minOccurs='0' maxOccurs='1'/>
               <xsd:element name='BOOL' type='xsd:boolean'
                    minOccurs='0' maxOccurs='1'/>
               <xsd:element name='BYTE' type='xsd:byte'
                    minOccurs='0' maxOccurs='1'/>
               <xsd:element name='SHORT' type='xsd:short'
                    minOccurs='0' maxOccurs='1'/>
               <xsd:element name='SINT' type='xsd:int'
                    minOccurs='0' maxOccurs='1'/>
               <xsd:element name='UBYTE' type='xsd:unsignedByte'
                    minOccurs='0' maxOccurs='1'/>
               <xsd:element name='USHORT' type='xsd:unsignedShort'
                    minOccurs='0' maxOccurs='1'/>
               <xsd:element name='UINT' type='xsd:unsignedInt'
                    minOccurs='0' maxOccurs='1'/>
               <xsd:element name='REAL4' type='xsd:double'
                    minOccurs='0' maxOccurs='1'/>
          </xsd:all>
     </xsd:complexType>
</xsd:schema>
```

On the XML Schema description above a *tns* namespace, which includes an ID complex type, was defined. Because the element names increase the payload, the name lengths have been kept minimal. By using XML Schema documentation, it is now possible to map the data types of each individual element. Timestamp *TS* was defined as a *time* type defined at 2001 W3 XMLSchema, while ID number *Nr* is of type *int* of the same documentation. Multiple elements were defined for the actual ID value so that the same ID complex type element could be used to transmit different types of monitoring

values. Boolean data for instance can be transmitted inside the *BOOL* element, and an error message of string format can be transmitted inside the *ERROR* element. The occurrences of ID value elements have been limited with *minOccurs* and *maxOccurs* parameters so it is now possible to transmit an ID element in the following format.

```
<tns:ID>
     <tns:TS>12:00:00.00</TS>
     <tns:Nr>1</tns:Nr>
     <tns:SINT>100</tns:SINT>
</tns:ID>
```

All the data types of the used elements can be unambiguously identified from the above example by referencing the XML Schema documentation. A WSDL document that describes the web service is in XML Schema format, and can contain the descriptions for all the elements used in the communication. The WSDL document of the remote monitoring service is attached in Appendix 1.

6.1.2 Authentication and identification

If the remote monitoring messages would not include any authentication information, a third party attacker could easily send false information to the database by performing a so called reply attack (W3C 2001). Authentication information, such as a username and a password, can be added to the message so that the service provider can identify the client and accept the message content. A message with fixed, plain text identity information is however as exposed to reply attacks as a message with no identity information at all. This means that the identity information should alternate between sent messages. Furthermore the identity information should alternate in a way, that even if the

attacker would know the authentication alternation method, it would not be possible to predict the next message identity information even while knowing the contents of previous messages.

WS-Security *UsernameToken* element provides a standard password hashing mechanism for SOAP messaging. The remote monitoring service provider assigns a fixed username-password pair to the client, which then creates a new hashed password to every message according to Algorithm 1. An example user 101 with password 102 sends a message at 11th of July 2008. According to Algorithm 1, a message that was created at 1.31.58 o'clock, and which was provided with nonce +8JRA04FAAA=, would then have a hashed password

Base64(SHA-1(+8JRA04FAAA=2008-07-11T01:31:58.000102)) =
XUFmDr88Qxgi/q6afcuGcJCeVPs.

The next message which will be sent five second later with nonce XdZRAFkGAAA= would have a password

Base64(SHA-1(XdZRAFkGAAA=2008-07-11T01:32:03.000102)) =
B3nLuZ+LkaqYnRyb5TpFlQr8m7c=.

The *dataTime* timestamps in the above examples were enhanced with red color to highlight the Algorithm content. The examples demonstrate how the hashed password will be generated, and how the two generated passwords have nothing in common. The receiver must know the clients username-password pair in order to authenticate the user. The receiver performs the same Algorithm 1 hash procedure with the received nonce and timestamp

information, together with the known plain text password. If the received password corresponds to the result of the formula outlet, the client is authenticated.

The WS-Security UsernameToken profile defines that the used nonce must be unique in each message. This rule causes the hashed password to be different even if the same timestamp would appear in consecutive messages. Without breaking the SHA-1 encryption, the attacker must now guess the clients password in order to access the service. This can be made harder by selecting an appropriate length and form for the password. UsernameToken profile provides protection against reply attacks, but does not prevent so called man-in-the-middle attacks (MSDN 2001), where the attacker can capture the message, alter the body of the SOAP message, and forward tampered data content with valid authentication details to the server. This type of attacks are not considered to be a risk on implemented remote monitoring application, but should be considered for example in remote control applications. In such cases, a higher level security mechanism must be considered. The difference between reply and man-in-the-middle attacks has been visualized in Figure 9.

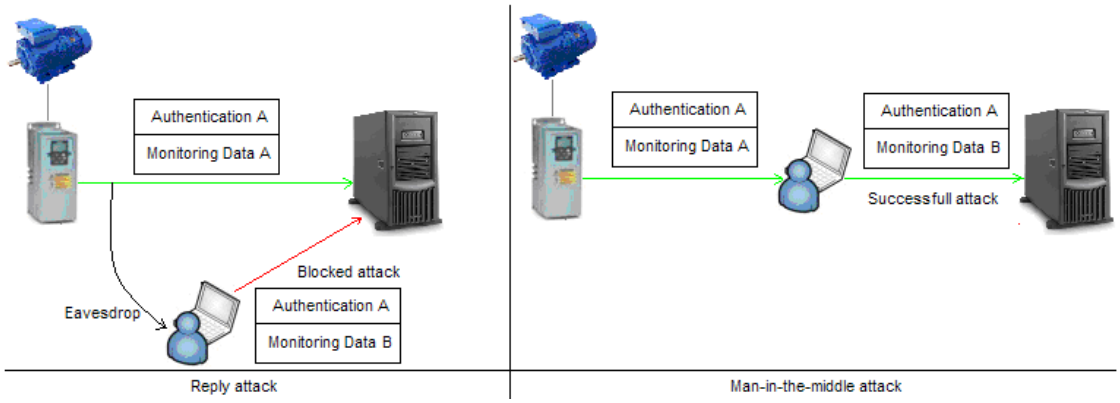


Figure 9. WS-Security UsernameToken against different types of attacks

In addition to identifying the user, the messages arriving from different devices must be separated from each other. If it is assumed that the same user credentials can be used in multiple devices, the devices can be distinguished from each other for example by adding the device's MAC-address to the SOAP message. This method together with UsernameToken profile will be used in the remote monitoring implementation.

6.1.3 Service description in WSDL format

A full WSDL description of the ID storing service is attached in Appendix 1. Briefly explained, the remote monitoring RPC function in the WSDL file has been defined to take the device properties- and n amount of ID elements in an array format as parameter inputs. This means that the amount of stored IDs in one message can alter between messages. This gives the possibility to store more IDs at times when the connection has been down for a while for instance. In the response message, the service returns a numerical response 1 if the storing was successful and some negative feedback if the procedure failed. If for instance the authentication failed, the response will be -1. The SOAP message header has been defined to contain a WS-Security element, which in this case is a UsernemeToken element. Furthermore the WSDL description file contains a binding to the service port, or more precisely to the Uniform Resource Locator (URL) address of the remote monitoring service.

With the help of a WSDL description file, it is easier to build the server software. PhP's SOAP extension for instance can automatically parse the RPC

parameter structure directly from the description file, thus reducing the amount of mandatory coding. A PhP example, which demonstrates the usage of a WSDL description file, has been given below. In the example, the Appendix 1 *StoreIDs* RPC service will be taken into use. The SOAP extension automatically parses the input parameters from the incoming message and then calls the actual procedure. Inside the function, these parameters can then be used in the same way as any other PhP objects, meaning that they can easily be stored to a database for instance. When the actual function call has been finished, the SOAP extension will automatically generate a response message according to the WSDL description file and send it back to the client.

```
<?php
function StoreIDs($DeviceInfo,$IDArray)
{
     ...
     //Store parameters to database etc.
     ...
     return 1;
}
ini_set("soap.wsdl_cache_enabled", "0");
$server = new SoapServer("RemoteConnection.wsdl");
$server->addFunction("StoreIDs");
$server->handle();
?>
```

The thesis implementation covers only an ID storing service, but it is relatively easy to add new RPC functionalities in the future. A fault information storing could be added to the WSDL file as a new function, and the same description file could then be used to cover all the SOAP RPC services. Adding a new service does not require any changes to the old software structures, meaning that software updates caused by the adding of a new service can be kept minimal. From the PhP software point of view the fault info storing would only require porting of a new RPC function from the WSDL file and creating a functional content for the procedure. It is also possible in certain limits to mold

the existing XML elements. Additional sub elements, like device software information or serial number, could be added into the device info element for example. If the updates are correctly defined into the WSDL file, no changes to the old, operational server software are required. This means that backwards compatibility can be maintained even when adding new content to the service interface.

6.2 Frequency converter hardware description

Remote monitoring application was implemented on Vacon OPT-CI option card. The Ethernet compatible option card can be connected to Vacon NX series frequency converters, which makes it possible to add the remote monitoring feature as a post option even to older devices. The card has been equipped with an ATMEL AT91RM9200 microcontroller, 64MB of SDRAM memory, 32MB of flash memory and with a 10/100Mbps Ethernet. The OPT-CI card is shown at Figure 10.

Figure 10. Vacon OPT-CI Ethernet Option Board (Vacon 2008).

6.3 Frequency converter software description

Previously mentioned Modbus TCP protocol has been implemented on Vacon OPT-CI option card. This makes it already possible for the user to read and write ID values from the drive over Ethernet. Both SOAP-, and Modbus TCP protocols use a TCP socket interface for communication, meaning that the existing software structure of the option card gives a good base for SOAP RPC communications.

To enable remote monitoring calls, application layer software which provides an interface for SOAP communication is required on top of the TCP stack. SOAP communication requires data transformation to XML format, and back to data again, so the software must contain features of an XML parser. Furthermore the software must cover at least HTTP client communication properties, which makes the SOAP software relatively large already. Several small-memory-foot-printed HTTP softwares, as well as some SOAP software implementations can already be found on the marked nowadays. Writing the communications software entirely from the beginning would require so much work and testing that a complete SOAP/HTTP communications stack was decided to be used in the implementation. The ID storing SOAP RPC application was then written on top of this stack.

The frequency converter software consists of several readable values from power-, current- and torque measures to analog- and digital inputs. These values have been read from the device to the option card in real time through a special option bus interface. The option card software has also included some parameters which could have been altered directly from the frequency

converter's control panel. The remote monitoring implementation was built on the top of the existing option card software, thus taking advantage of the existing TCP/IP communications- and option bus interfaces.

The software implementation is shown in Figure 11. It was realized in a way that the ID storing application was split into two real time operating system tasks. The application has six ID address parameters which the user can freely alter from the FC's control panel. An independent, periodically every second driven, task fetches the parameter defined ID values and data types from the device. A timestamp from the moment of the fetch is added to the ID value, so that it would be possible for instance to draw a time dependent trend graph from the collected data later on. The gathered data is then added to a circular buffer, where it can be read at an appropriate time.

Storing the IDs from the circular buffer into an external memory is done in the second task. Because it is not reasonable to create new SOAP calls for every ID to be stored, the task will wait until an appropriate amount of IDs are available at the buffer. The task will read the ID units one at a time from the circular buffer, transform these into Subchapter 6.1.1 shown textual XML format, and finally the units will be placed into an XML array. Among the ID address parameters, the user can also modify authentication details from the panel. Based on the given details, a WSS UsernameToken element will be created and the devices MAC address will also be transformed into XML format. When the SOAP RPC call's required information has been gathered, a connection to the actual remote service can be formed. Because the frequency converter might be located in such a network location, that all the HTTP connections go through a proxy server, the user must be able to modify the proxy server address among

the actual database server address from the panel. Using the gathered information, a SOAP RPC call will then be set up and the task will then stay waiting for the response message. If the call was successful the task will return to sleep until the next data call, but in some exceptional circumstances the errors must be handled. If the authentication details for instance were invalid, there is no need to create further calls because the user was not identified. In situations where the call could not be established, for instance due to some network error, the ID values could temporarily be stored into flash memory from where they could be restored when the connection is back to working again. The above description of the software functionality periodically repeats as long as the remote monitoring feature has been turned on.

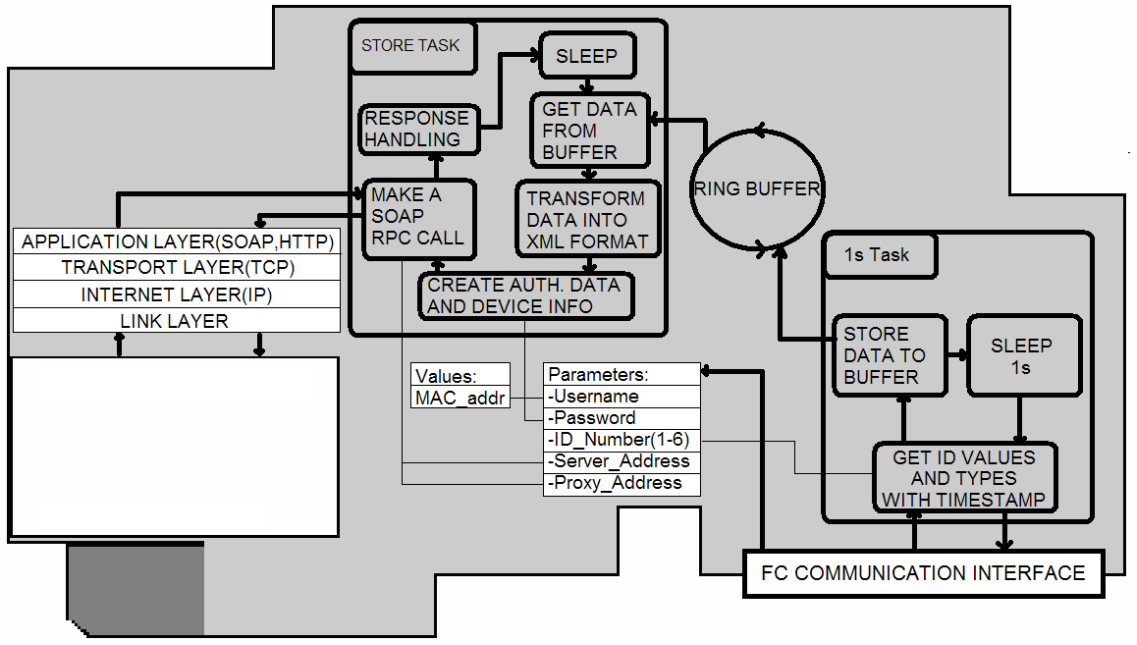


Figure 11. Software description for ID storing.

6.4 Server software

The essential purpose of the master's thesis is not to present different server side solutions, but one example implementation for one server platform will still be introduced. A database server was built into a Windows XP PC computer, that was running an Apache 2.2 server software and a MySQL database. The web interfaces were written using PhP programming language. Fundamental purpose of the server software is to store the FC's transmitted ID values into a database, and to draw a web browser monitorable trend view graph from the stored values for the end user.

6.4.1 Service interface for frequency converters

Frequency converters establish a SOAP RPC connection to the Appendix 1 WSDL document defined URL, or more precisely to the address *www.uwasa.fi/~l82962/RemoteConnection/wsdl/RemoteConnection.php*. The PhP code behind the target URL receives the SOAP messages, checks the header content to authenticate the user and finally performs the WSDL defined RPC function call implementation for ID storing. The server software interface for the frequency converter communications has been visualized at Figure 12.

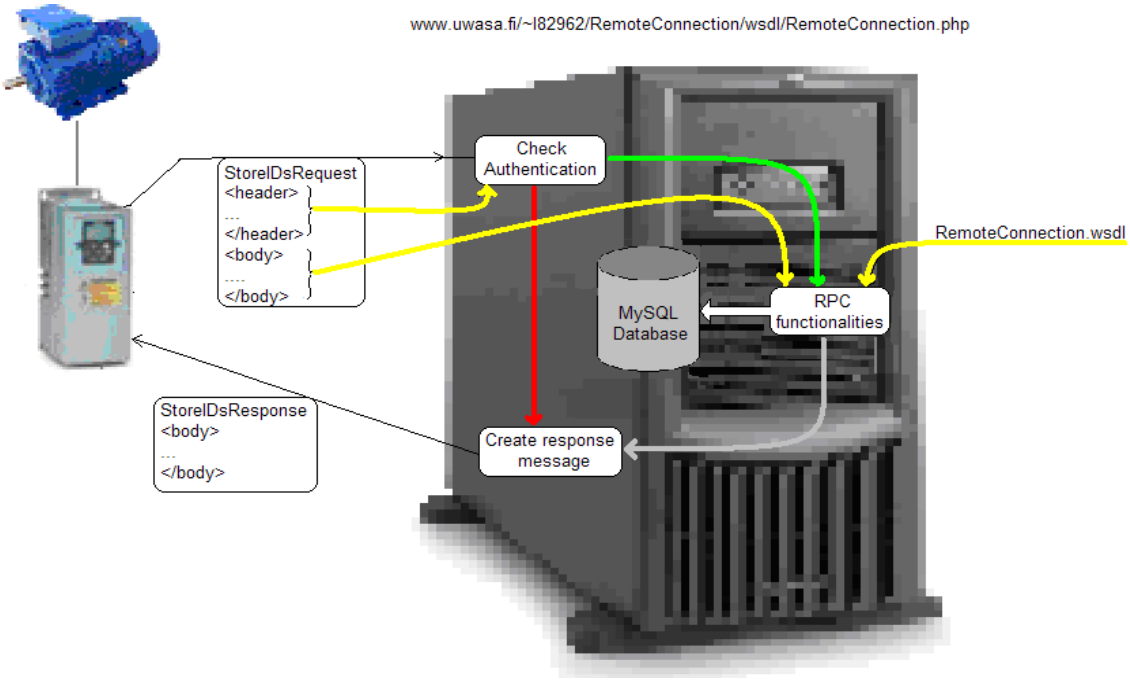


Figure 12. Server software interface for frequency converters.

6.4.2 Service interface for remote monitoring

For the actual remote monitoring purposes, the database server contains a standard web page for displaying the stored data in graphical format. At the example implementation, the graphical remote monitoring view was made to display device specific ID values in a standard web browser. The user can select a specific device according to received MAC addresses, after which all the values that have been received from the selected address will be displayed graphically and sorted according to the ID numbers. The graph's x-axis displays time, while the y-axis displays the received ID values. Negative changes between consecutive ID values are highlighted with blue horizontal lines, while positive changes are highlighted with red horizontal lines respectively to enhance small, but significant changes in binary values for instance. An example remote monitoring case will be introduced at chapter 7.

7 EXPERIMENTS AND RESULTS

This chapter introduces an example use case which demonstrates the usage of the implemented remote monitoring feature. The complete remote monitoring system still missed the final database server implementation at the time of writing this chapter. Due to this cause, the system could not be tested in real use case environments at this stage. The server side software used in this demonstration is only a temporary implementation built for demonstrational and test purposes.

7.1 Example case and environment

The built-up use case with its surrounding environment has been visualized at Figure 13. The frequency converter, that controls a motor, is connected to a corporate intranet. The devices at the intranet fetch their IP addresses from a Dynamic Host Configuration Protocol (DHCP) server, which is also the FC's primary task before establishing any other TCP connections. All the TCP communication that departs from the intranet goes through a proxy server, which in this case also communicates with the database server, located at the internet. The remote monitoring client at the figure can observe the ID data that the frequency converter is transmitting, with a standard web browser from anywhere, by viewing the web site located at the database server.

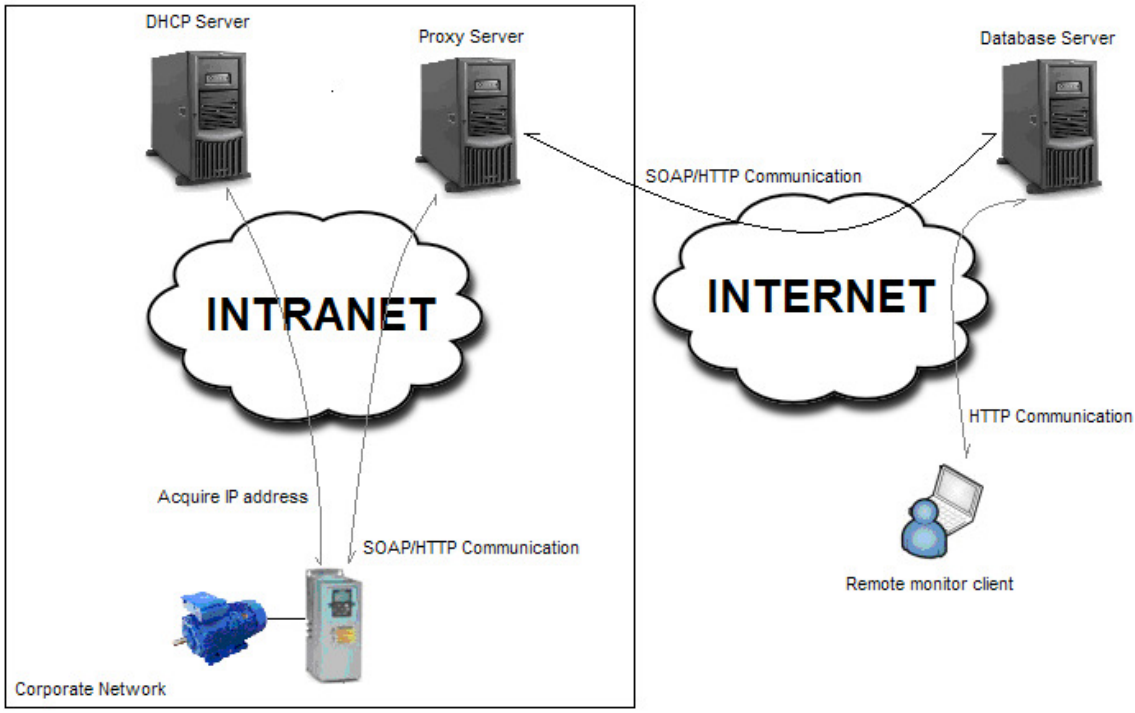


Figure 13. Example use case.

7.1.1 Frequency converter control environment

The frequency converter that is used in the example use case is a Vacon NXS (Vacon 2007) drive, which is connected to a small three phased induction motor. Furthermore the FC has been equipped with a Vacon OPT-CI option card supplied with the implemented remote monitoring application and a gateway to Ethernet.

The FC is locally controlled with an I/O simulator shown at Figure 14. The simulator has six digital and two analog inputs which can be used to control the frequency converter. *AI1* analog switch shown at the figure is used to alter the device's reference frequency. At system software, the position of the switch has been scaled to gain values between 0 and 100, and the current value can be

externally read through ID 59. The switches *DIA1* and *DIA2* are used to control the rotational direction of the motor. When the *DIA1* switch is high, the reference frequency that directly influences to the output frequency is positive and the direction of the motor rotation is clockwise. Setting the *DIA2* switch high causes the motor to run at counter clockwise direction recursively. When both of the switches are low, the motor is stopped. Turning the *DIA3* switch on, causes a bogus fault event, which forces the motor to halt immediately. The FC has been programmed in a way that the engine wont start running again, until the source of the fault has been removed and the event has been acknowledged with a reset signal. In this case the reset signal can be given by toggling *DIB6* switch on. The last switch used in the example is *DIB4*, which forces the reference frequency to be set at a predefined value. The positions of the digital inputs *DIA1-DIA3* can be read via ID 15, and positions of *DIB4-DIB6* via ID16 recursively.

A few predefined reference-, minimum- and maximum values exist on the frequency converter. Maximum output frequency has been set to 50Hz and the predefined reference frequency behind *DIB4* switch is 10Hz. Acceleration time for the engine from stand-still to maximum speed is 15 seconds, while braking the speed back to zero takes 10 seconds recursively. Output frequency that is fed to the engine can be monitored through ID 1 and the reference frequency, which is the target for the output frequency, can be viewed via ID 25. A calculated motor temperature can be read through ID 9.

Figure 14. Vacon I/O Simulator

7.1.2 State machine representation of demo control and events

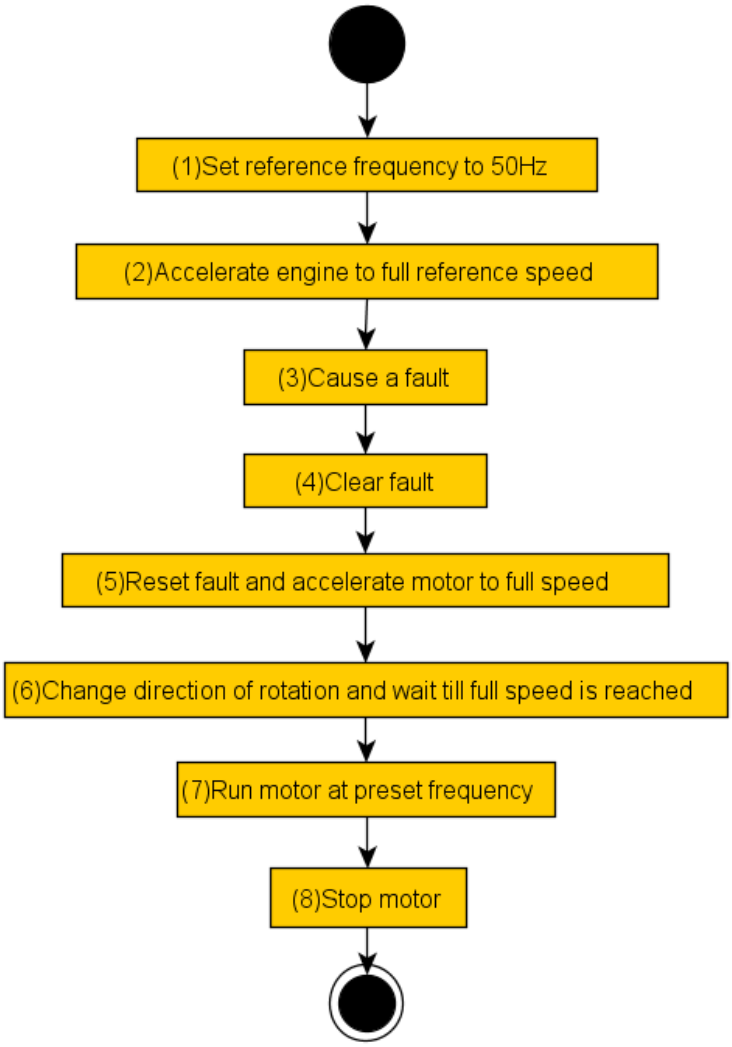


Figure 15. State representation of demo control.

The FC will be driven according to the sequence graph shown at Figure 15. Firstly the reference frequency will be set to 50Hz by turning the analog input *AI1* fully on. Motor will then be set to accelerate towards the reference frequency by setting the *DIA1* digital input high. After 15 seconds of acceleration, a bogus fault will be caused with *DIA3* switch. The source of the fault will be removed immediately and the fault will then be reset from the

software by toggling the *DIB6* switch on. After the motor has reached its full speed again, the direction on the rotation will be altered by changing the positions of switches *DIA1* and *DIA2*. The FC will now slowly stop the engine and then accelerate it again at reverse direction till it has reached the reference frequency again. Finally, before stopping the engine completely, the reference speed will be set to 10Hz by turning the *DIB4* switch on.

7.1.3 Remote monitoring of demo case

Before the user can view any values remotely, the desired IDs must first be set to the implemented application as parameters. Some interesting ID values at the presented drive sequence are output frequency 1, reference frequency 25, analog switch position 59 and digital switch positions 15 and 16. Furthermore the application will be set sending the calculated motor temperature behind ID 9. The application must also be configured with the service provider requested authentication details and with a proxy server address, so that the sent messages would found a way out from the intranet. The software is operational after it has been configured with the mentioned parameters and starts the periodical sending of given ID values to the database server. The remote monitoring client can now view the process of the sequence in real time with a web browser, regardless of his location.

The remote monitoring view, generated by the temporary database server, is shown at Appendix 2. The sequence of Figure 15 can be accurately followed from the monitor view, where the events 1-8 have been marked afterwards to simplify interpretation. At phase 1, the alteration of the analog switch causes

the reference frequency to rise at 50Hz. When the *DIN1* switch at phase 2 is then turned on, the motor starts a steady acceleration until it has finally reached the target reference frequency. The caused bogus fault at phase 3 causes the motor to halt, and it will remain stopped until the removal of the fault at phase 4 and the given reset signal at phase 5 make it accelerate again. Position change of the digital switches at phase 6 cause a sign change at the reference frequency. This causes the motor to be decelerated to a stop and then accelerate towards the reference frequency at reverse speed. Change of *DIB4* switch position at phase 7 changes the nominal reference frequency value to be dropped from 50Hz to 10Hz, to which the motor will be decelerated before the final stop at phase 8.

8 CONCLUSIONS AND FUTURE WORK

Support and maintenance of frequency converters could often be made more dynamic and flexible if the devices could be remotely monitored. Especially on locations, such as unmanned stations or scattered systems, the maintenance personnel could plan and schedule their upcoming service visits better, if they would be able to monitor the data and states of the remote automation processes. This thesis studied the suitability of SOAP communication as a remote connection protocol. Furthermore the goal was to find and implement a dynamic and easily deployable remote connectivity method for an embedded systems device.

From configuration point of view, SOAP applies well for making effortless remote connections partly due to its HTTP compatibility. SOAP inherits its excellent proxy server- and firewall penetration abilities from HTTP protocol. It is possible to form remote connections from places, where the usage of a regular web browser is possible. Only infrastructures where SOAP messaging has been limited with devices, which also monitor the content of HTTP messages, are excluded. For data protection, an Oasis Open maintained WS-Security standard provides security elements for SOAP messaging. Furthermore the standard covers authentication methods, of which a digest access authentication was implemented in the thesis. WS-Security elements can be used to creating end-to-end secure SOAP connections, meaning that messages can even be encrypted without using lower layer encryption mechanisms. It is therefore possible to securely exchange application data even on top of unsecured transmission protocols like HTTP.

SOAP communication is relatively slow, so it does not suite for transmitting dense real time data that well. SOAP can be used to delivering data even in real time for sparse measurements, thus enabling accurate remote monitoring of values that do not require frequent sampling rate. The thesis presented a practical remote monitoring implementation based on SOAP messaging. Implementation enables the frequency converter to periodically send monitor able data to a centralized database server, from where a remote monitor client can view the received values. A SOAP client initiated RPC communication removes the need for a solid IP-address, meaning that connections can be formed even from devices behind NAT-routers, commonly used in ordinary ADSL subscriptions. When using a fixed database server, the mandatory configuration for enabling remote connections can be minimized, meaning that service deployment for end user is effortless. The client can monitor frequency converter processes with a standard browser through web sites located at the database server, excluding the need for the user to install any special computer programs for this purpose. Aside user friendly deployment for the service, other positive features for using a centralized server exist. Especially when dealing with embedded systems, a lot more data can be stored to an external database memory than to the devices own limited memory space. Centralized solution is also easy to maintain, to update for new services and allows features, like data fusion, to be implemented in the future. SOAP messaging is suitable for communicating between devices with highly different hardware and software platforms. Data sorting and storing to a database cell from the message at the server side turned out to be relatively easy, effortless and dynamic.

More remote services, like remote control ability and fault information transportation, can be implemented aside the presented remote monitoring application in the future. Due to SOAP's dynamicity, old services can be updated and new services can be added without losing the backwards compatibility to existing solutions. More than one connection can be established from a single device, meaning that several services can be operational even at the same time. Further investigation would be required to see if SOAP messaging by itself is reliable enough to handle alert information delivery without any other alternate transport medium like an SMS message. So far the implemented service can only be used to providing some additional information about the states of a process, when the actual alerts to the user must still be delivered in some other way. Further investigation would also be required to find an optimal balance between a single SOAP message payload size, and the amount of message transactions, while considering message delivery times and the level of network congestion at the same time.

Altogether, SOAP is a reliable communications protocol, which can provide secure, dynamic and easily deployable remote connection services for embedded devices.

9 REFERENCES

ABB (2007). *Case notes: ABB drives and remote monitoring reduce costs in wastewater pumping system*[online]. [cited 4.6.2008]. Available at: <http://search.abb.com/library/ABBLibrary.asp?DocumentID=CD27&LanguageCode=en&DocumentPartId=1&Action=Launch>

ABB (2008). *ABB drives: Product guide for low voltage drives*[online]. [cited 27.5.2008]. Available at: <http://library.abb.com/global/scot/scot201.nsf/veritydisplay/d577948349c81125c125741d004d6489/$File/EN_Productguide_forlowvoltagedrivesREVF.pdf>

ATMEL (2006). *ARM920T-based Microcontroller AT91RM9200*[online]. [cited 20.5.2008]. Available at: <http://www.atmel.com/dyn/resources/prod_documents/1768s.pdf>

Clarke M & Jones RW & Bratan T & Larkworthy A (2004). *Providing remote patient monitoring services in residential care homes*[online]. [cited 1.8.2008]. Available at: <http://www.health-informatics.org/hc2004/P34_Clarke.pdf>

Davis Dan & Parashar Manish (2002). *Latency Performance of SOAP Implementations*[online]. [cited 22.7.2008]. Available at: <http://www.caip.rutgers.edu/TASSL/Papers/p2p-p2pws02-soap.pdf>

Doktar, Andreas (2006). *Utveckling Av Metoder För Mjukvarestimering Av Temperaturen I Kraftelektronik,* Master's Thesis. Turku, Finland: Åbo Akademi. 84 s.

Elisa (2006). *Zyxel 660H/HW -verkkopäätteen asetukset*[online]. Available at: <http://tuki.elisa.fi/asiakastuki/elisa.do?id=hen_as_yhka_internet,dokumenttisivu_adsl_0013.htm>

eSOAP (2008). *eSOAP Datasheet*[online]. [cited 30.5.2008]. Available at: <http://esoap.ultimodule.com/bin/esoap/templates/default.asp?_resolutionfile=templatespath|default.asp&area_3=pages/datasheet>

Falcon (2008). *FN Series™ UPS PLUS® - 3kVA to 40kVA Brochure*[online]. [cited 27.5.2008]. Available at: <http://www.falconups.com/FN_3-6kVA_-2TXI_N+1_Brochure_Final.pdf>

Ford Bryan & Srisuresh Pyda & Kegel Dan (2005). *Peer-to-Peer Communication Across Network Address Translators*[online]. [cited 4.6.2008]. Available at: <http://www.bford.info/pub/net/p2pnat.pdf>

gSOAP (2008). *gSOAP: SOAP C++ Web Services*[online]. [cited 30.5.2008]. Available at: < http://www.cs.fsu.edu/~engelen/soap.html>

in4ma (2007). *The in4ma pc*[online]. [cited 4.6.2008]. Available at: <http://www.in4ma.co.uk/downloads/in4ma%20pc%20v2%20technical%20specification.pdf>

Janêcek Jan (2004). *Efficient SOAP processing in embedded systems*. Prague: Czech Technical University in Praque.

Kennard Scribner & Stiver Mark C. (2000). *Understanding SOAP.* Indiana: Sams Publishing.

Kim Sang-Oh & Chun Jae-Kun (2000). *Remote Monitoring and Control of Agricultural Storage Facility using Internet*[online]. [cited 1.8.2008]. Available at: <http://zoushoku.narc.affrc.go.jp/ADR/AFITA/afita/afita-conf/2000/part06/p179.pdf>

Modbus-IDA (2006). *MODBUS Messaging on TCP/IP Implementation Guide V1.0b*[online]. [cited 28.5.2008]. Available at: <http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf>

MSDN (2005). *Implementing Direct Authentication with UsernameToken in WSE 3.0* [online]. [cited 23.7.2008]. Available at: < http://msdn.microsoft.com/en-us/library/aa480575.aspx>

MSDN (2008). *Understanding WS-Security*[online]. [cited 30.5.2008]. Available at: <http://msdn.microsoft.com/en-us/library/ms977327.aspx>

OASIS (2004). *Web Services Security UsernameToken Profile 1.0*[online]. [cited 8.7.2008]. Available at: <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>

Ojanperä, Reijo (2000). *Markkinajohtajuus vaatii tutkimusta*[online]. [cited 27.5.2008]. Available at: <http://www.prosessori.fi/uutiset/uutinen2.asp?id=36849>

PHP (2008a). *PHP: MySQL functions*[online]. [cited 30.5.2008]. Available at: <http://fi.php.net/mysql>

PHP (2008b). *PHP: SOAP - Manual*[online]. [cited 30.5.2008]. Available at: <http://fi.php.net/soap>

Questra (2004). *Case Study: Remote Monitoring Gives Eaton Customers Immediate Data On Power System Operation*[online]. [cited 4.6.2008]. Available at: <http://www.questra.com/collateral/collateral_files/Case_Study_Eaton.pd>

Sakellariadis Spyros (2002). *Protecting Windows RPC Traffic*[Online]. [cited 21.5.2008]. Available at: <http://www.microsoft.com/technet/archive/isa/2000/maintain/rpcwisa.mspx?mfr=true>

Tiainen R. & Särkimäki V. & Ahola J. & Lindh T. (2006). *Utilization Possibilities of Frequency Converter in Electronic Motor Diagnostics.* Lappeenranta, Finland: Lappeenranta University of Technology.

Vacon (2007). *Vacon NXS/P User's Manual*[online]. [cited 27.5.2008]. Available at: <http://www.vacon.com/File.aspx?id=462819&ext=pdf&routing=396771&name=UD00701T>

Vacon (2008). *Ethernet Option Board OPT-CI User's Manual*[online]. [cited 27.5.2008]. Available at: <http://www.vacon.com/File.aspx?id=462937&ext=pdf&routing=396771&name=UD01043B>

White, James E. (1976). *RFC 707: A High-Level Framework for Network-Based Resource Sharing*. California: Stanford Research Institute.

Wikipedia (2008a). *Variable-frequency drive*[online]. [cited 27.5.2008]. Available at: < http://en.wikipedia.org/wiki/Variable-frequency_drive>

Wikipedia (2008b). *WS-Security*[online]. [cited 30.5.2008]. Available at: <http://en.wikipedia.org/wiki/WS-Security>

W3C (2001). *SOAP Security Extensions*[online]. [cited 23.7.2008]. Available at: <http://www.w3.org/TR/SOAP-dsig/ >

W3C (2007). *Lates SOAP versions*[online]. [cited 29.5.2008]. Available at: <http://www.w3.org/TR/soap/>

W3Schools (2008). *SOAP Tutorial*[online]. [cited 29.5.2008]. Available at: <http://www.w3schools.com/soap/>

Zhang H. D. & Zhou Q. Z. (2006). *Communication-based Control Mode of Frequency Converter.* Anhui, China: Anhui University of Technology

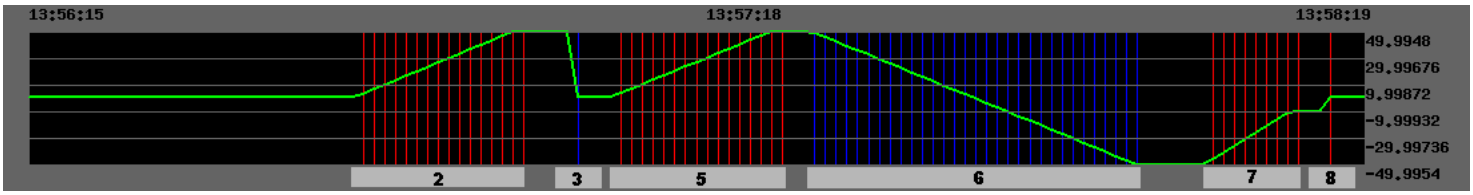
APPENDIXES

APPENDIX 1. Service description in WSDL format

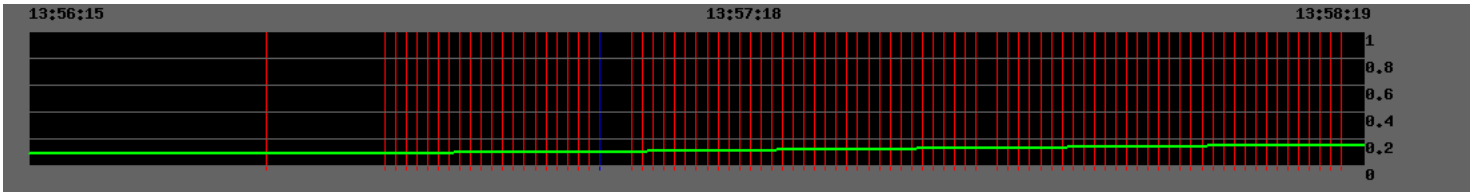
```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions name='RemoteConnection'
   targetNamespace='http://www.uwasa.fi/~l82962/RemoteConnection/wsdl/RemoteConnection.wsdl'
   xmlns='http://schemas.xmlsoap.org/wsdl/'
   xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
   xmlns:tns='http://www.uwasa.fi/~l82962/RemoteConnection/wsdl/RemoteConnection.wsdl'
   xmlns:wsse='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'>

   <wsdl:types>
   <xsd:schema
   targetNamespace='http://www.uwasa.fi/~l82962/RemoteConnection/wsdl/RemoteConnection.wsdl'
        xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
             <xsd:complexType name='DeviceInfo'>
                  <xsd:all>
                       <xsd:element name='MACAddress' type='xsd:hexBinary'/>
                  </xsd:all>
             </xsd:complexType>
             <xsd:complexType name='ID'>
                  <xsd:all>
                       <xsd:element name='TS'    type='xsd:time'/>
                       <xsd:element name='Nr'    type='xsd:int'/>
                       <xsd:element name='ERROR' type='xsd:string' minOccurs='0' maxOccurs='1'/>
                       <xsd:element name='BOOL' type='xsd:boolean' minOccurs='0' maxOccurs='1'/>
                       <xsd:element name='BYTE' type='xsd:byte' minOccurs='0' maxOccurs='1'/>
                       <xsd:element name='SHORT' type='xsd:short' minOccurs='0' maxOccurs='1'/>
                       <xsd:element name='SINT' type='xsd:int' minOccurs='0' maxOccurs='1'/>
                       <xsd:element name='UBYTE' type='xsd:unsignedByte' minOccurs='0'
                            maxOccurs='1'/>
                       <xsd:element name='USHORT' type='xsd:unsignedShort' minOccurs='0'
                            maxOccurs='1'/>
                       <xsd:element name='UINT' type='xsd:unsignedInt' minOccurs='0'
                            maxOccurs='1'/>
                       <xsd:element name='REAL4' type='xsd:double' minOccurs='0' maxOccurs='1'/>
                  </xsd:all>
             </xsd:complexType>
             <xsd:complexType name='IDArray'>
                  <complexContent>
                       <restriction base='soapenc:Array'>
                            <attribute ref='soapenc:arrayType' wsdl:arrayType='tns:ID[]'/>
                       </restriction>
                  </complexContent>
             </xsd:complexType>
        </xsd:schema>
   </wsdl:types>

   <message name='Header'>
        <part name='Security' element='wsse:Security'/>
   </message>
```

```
    <message name='StoreIDsRequest'>
         <part name='DeviceInfo' type='tns:DeviceInfo'/>
         <part name='IDArray' type='tns:IDArray'/>
    </message>

    <message name='StoreIDsResponse'>
         <part name='Response' type='xsd:int'/>
    </message>

    <portType name='RemoteConnectionPortType'>
         <operation name='StoreIDs'>
              <input message='tns:StoreIDsRequest'/>
              <output message='tns:StoreIDsResponse'/>
         </operation>
    </portType>

    <binding name='RemoteConnectionBinding' type='tns:RemoteConnectionPortType'>
    <soap:binding style='rpc'
         transport='http://schemas.xmlsoap.org/soap/http'/>
         <operation name='StoreIDs'>
              <soap:operation soapAction='StoreIDs'/>
                   <input>
                   <soap:body
                   encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
                   namespace='http://www.uwasa.fi/~l82962/RemoteConnection/wsdl/
                        RemoteConnection.wsdl'
                   use='encoded'/>
                   <soap:header use="literal" message="tns:Header" part="Security"/>
                   </input>
                   <output>
                   <soap:body
                   encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
                   namespace='http://www.uwasa.fi/~l82962/RemoteConnection/wsdl/
                        RemoteConnection.wsdl'
                   use='encoded'/>
                   </output>
         </operation>
    </binding>
    <service name='RemoteConnectionService'>
         <port name='RemoteConnectionPort' binding='tns:RemoteConnectionBinding'>
              <soap:address
              location='http://www.uwasa.fi/~l82962/RemoteConnection/wsdl/RemoteConnection.php'/>
         </port>
    </service>
</definitions>
```

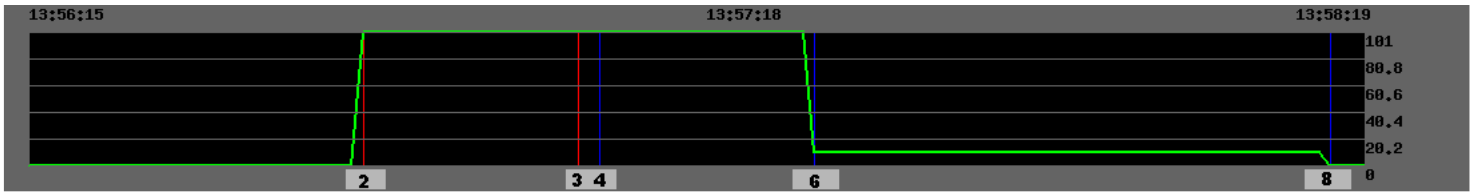
APPENDIX 2. Remote monitoring view of an example case



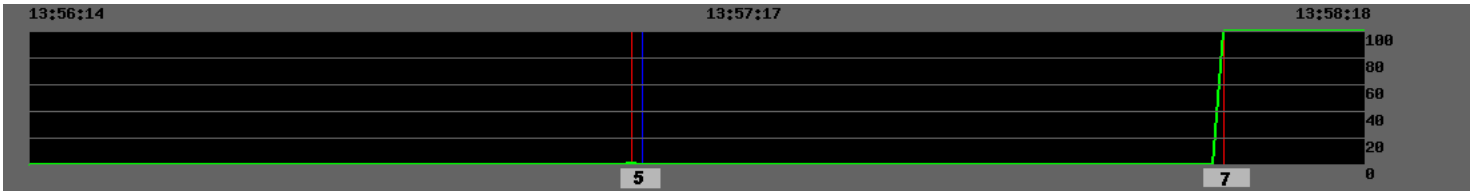
ID1



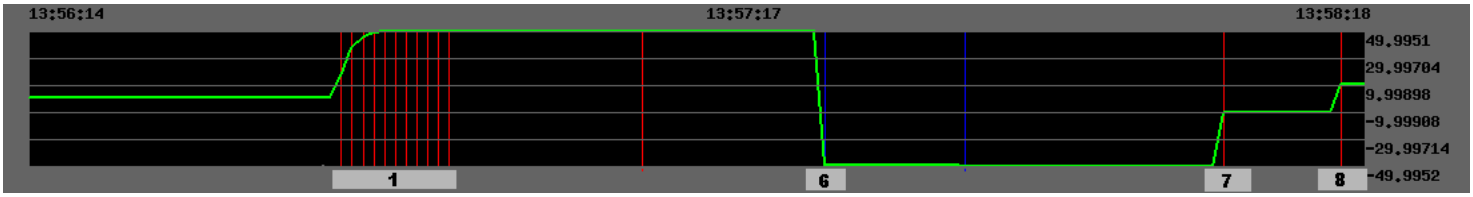
ID9



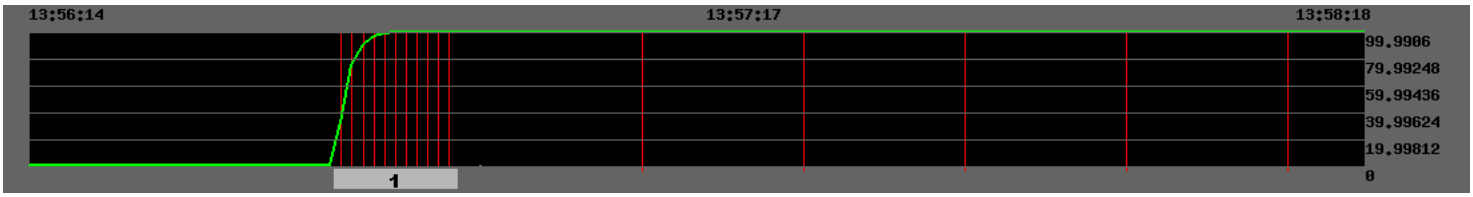
ID15



ID16



ID25



ID59