Lappeenranta University of Technology
LUT School of Energy Systems
Degree Programme in Electrical Engineering

Kimmo Huoman

FAN SYSTEM MONITORING VIA CLOUD SERVICE

Examiners: Professor Jero Ahola
D.Sc. Tero Ahonen

ABSTRACT

Lappeenranta University of Technology
LUT School of Energy Systems
Degree Programme in Electrical Engineering

Kimmo Huoman

Fan system monitoring via cloud service
2016

Master's Thesis
Pages 69, pictures 16, tables 13.

Examiners: Professor Jero Ahola
D.Sc. Tero Ahonen

Keywords: variable speed drive, monitoring, fan system, cloud service, Internet of Things

The majority of electricity consumption in industrial and service sectors in the European Union is caused by electric motors, largely by fluid handling systems such as fans, pumps, and compressors. By utilising energy efficient motors and variable speed drives, the energy savings potential in these sectors is estimated to be almost 90 TWh per year, even with economically viable investments. In relation, this is more than the electricity consumption of Finland. Most of the potential savings are achieved by replacing valves, dampers, and other flow restricting devices in fluid handling systems with a rotational speed control using a variable speed drive.

The variable speed drive can serve as more than just a method of adjusting the rotational speed. It can provide an excellent, centralised source of information. Recent studies show that the variable speed drive can provide values for estimating multiple features of fluid handling systems, features currently being monitored by dedicated sensors. By reducing the amount of sensors and moving on to sensorless estimation methods, the system complexity can be reduced.

Currently variable speed drive monitoring is typically implemented only to applications where the uninterrupted operation is crucial either for safety or minimising production losses. However because of effects caused by the Internet of Things, the cost and complexity of connecting hardware devices to the Internet is constantly decreasing. By connecting variable speed drives to a cloud service, both the available storage space and processing power are practically unlimited.

In this master's thesis, a cloud service capable of retrieving data from variable speed drives is developed. A fan monitoring application is built on top of the stored data, capable of estimating features such as the produced airflow. This is achieved by reviewing the current methods for operating point estimation based on mathematical model and the manual work required to obtain the equations used in the models. As an end result, a prototype cloud service is deployed with fan system monitoring features.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
LUT Energiajärjestelmät
Sähkötekniikan koulutusohjelma

Kimmo Huoman

Puhallinjärjestelmän monitorointi pilvipalveluna
2016

Diplomityö
Sivumäärä 69, kuvia 16, taulukoita 13.

Tarkastajat: Professori Jero Ahola
Tekniikan tohtori Tero Ahonen

Hakusanat: taajuusmuuttaja, monitorointi, puhallinjärjestelmä, pilvipalvelu, esineiden internet

Suurin osa teollisuuden ja palvelusektorin sähköenergiankulutuksesta Euroopan Unionissa aiheutuu sähkömoottoreista. Pääasiassa kulutus tapahtuu nesteitä ja kaasuja käsittelevissä järjestelmissä; puhaltimissa, pumpuissa ja kompressoreissa. Energiatehokkaiden sähkömoottorien ja taajuusmuuttajien laajemmalla käyttöönotolla voidaan saavuttaa jopa 90 terawattitunnin säästöt vuosittain, joka on enemmän kuin koko Suomen vuosittainen sähkönkulutus. Suurin osa säästöpotentiaalista on saavutettavissa korvaamalla virtausta rajoittavat toimilaitteet taajuusmuuttajaan perustuvalla nopeussäädöllä.

Taajuusmuuttajaa voidaan käyttää nopeussäädön lisäksi myös yksinkertaisena, keskitettynä mittalaitteena. Viimeaikaiset tutkimukset ovat osoittaneet, että suuri osa nykyään erillisillä mittalaitteilla saadusta informaatiosta on estimoitavissa taajuusmuuttajan avulla. Korvaamalla antureita taajuusmuuttajan avulla lasketuilla estimaateilla, voidaan järjestelmää yksinkertaistaa ja parantaa siten luotettavuutta.

Nykyisellään taajuusmuuttajakäyttöjen valvontajärjestelmät on implementoitu vain käyttöihin, joissa jatkuva käyttö on olennaista joko turvallisuuden tai tuottavuuden kannalta. Yhdistämällä taajuusmuuttajakäyttö pilvipalveluun, voidaan valvontajärjestelmä kuitenkin toteuttaa edullisesti ja yksinkertaisesti. Käytössä olevien, lähes rajattomien resurssien ansiosta myös taajuusmuuttajalle liian raskaiden tai monimutkaisten estimointimenetelmien toteutus on mahdollista.

Tässä diplomityössä kuvataan pilvipalvelu, joka mahdollistaa tietojen keräyksen taajuusmuuttajilta. Esimerkkinä käyttökohteesta luodaan puhallinkäyttöjen valvontaan soveltuva järjestelmä, joka kykenee estimoimaan esimerkiksi tuotettua tilavuusvirtaa. Työssä esitellään myös tämänhetkiset puhaltimen toimintapisteen määritykseen soveltuvat matemaattiset mallit ja prosessit, joita tarvitaan mallien käyttöön. Työn lopputuloksena on prototyyppiasteella oleva pilvipalvelu, joka soveltuu puhallinjärjestelmien valvontaan.

PREFACE

This thesis was completed in the Laboratory of Digital Systems and Control Engineering in Lappeenranta University of Technology (LUT). The study was part of Efficient Energy Usage (EFEU) research program, which develops system level efficiency solutions and services for fluid handling systems and regional energy systems.

I would like to thank my examiners Professor Jero Ahola and Post-doctoral researcher Tero Ahonen for the well-suited topic and all their help during the birth of this thesis. I would also like to thank Mr. Jukka Tolvanen and Dr. Olli Alkkiomäki of ABB for their encouraging words, help with the technical issues, and for providing the demonstration fan system. Last but certainly not least, I would like to thank my family, co-workers, and friends for encouragement during these past years.

Lappeenranta, November 11th, 2015

Kimmo Huoman

TABLE OF CONTENTS

SYMBOLS AND ABBREVIATIONS

Roman letters

E	energy
J	inertia
P	power
Q	air flowrate
T	torque
d	diameter
n	rotational speed
f	frequency
p	number of magnetic poles
p	pressure

Greek letters

α	angular acceleration
δ	uncertainty
η	efficiency

Subscripts

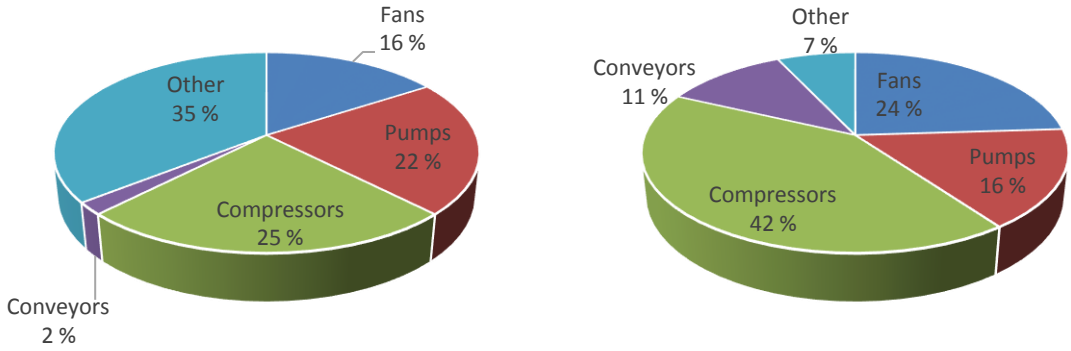
F	fan
M	motor
SO	shut-off
V	volume
avg	average
est	estimated
expected	expected value of
s	synchronous
shaft	motor shaft

Acronyms

AC	alternating current
API	application programming interface
BEP	best efficiency point
DC	direct current
DTC	direct torque control
FTP	File Transfer Protocol
EEM	energy efficient motor
HTTP	Hypertext Transfer Protocol
HVAC	heating, ventilation, and air conditioning
IGBT	insulated-gate bipolar transistor
IoT	Internet of Things
JSON	JavaScript Object Notation
M2M	machine-to-machine (communication)
MVVM	Model View ViewModel
REST	Representational State Transfer
RPC	Remote Procedure Call
SFP	specific fan power
SOAP	Simple Object Access Protocol
UI	user interface
VFD	variable frequency drive
VSD	variable speed drive
XML	Extensible Markup Language
sn	serial number

1. INTRODUCTION

The electricity consumption of electric motors is estimated to be as high as 69 % in industrial and 38 % in service sectors (de Almeida, et al., 2000). Out of these shares, the consumption of fluid handling systems account for 63 % and 83 % in industrial and service sectors, respectively (de Almeida, et al., 2003). The shares of electricity consumption of electric motors in industrial and service sectors are presented in Figure 1.1.



(a) (b)

Figure 1.1 Share of electricity used by electric motors. (a) In industrial sector and (b) in service sector. (de Almeida, et al., 2000)

As shown in Figure 1.1 the overall consumption of electric motors utilised in fluid handling systems is very large. With application of energy efficient motors (EEMs), variable speed drives (VSDs), and efficient end-use devices the economically viable savings potential is estimated to be around 67.3 TWh and 22.2 TWh in industrial and service sectors, respectively. From purely technical perspective, the savings potential is estimated to be even higher, at 92.4 TWh in industrial and 34.7 TWh in service sector. To put the numbers into perspective, the electricity consumption of the whole Finland, including industrial, service, and private sectors was 83.3 TWh. The savings potential concerning fluid handling systems is presented in Table 1.1. (de Almeida, et al., 2000; Tilastokeskus, 2015)

Table 1.1 Potential savings with application of EEMs and VSDs in fluid handling systems. (de Almeida, et al., 2000)

	Consumption [TWh]		Technical potential savings [TWh]		Economic potential savings [TWh]	
	Industry	**Service**	**Industry**	**Service**	**Industry**	**Service**
Pumps	155.4	35.5	29.3	9.0	22.4	7.5
Fans	113.6	53.8	22.1	12.5	16.2	6.5
Compres-sors	174.8	95.8	11.2	8.4	8.9	6.4
Total	**443.8**	**185.1**	**62.6**	**29.9**	**47.5**	**20.4**

As shown in Table 1.1 out of the energy consumed by electric motors in fluid handling systems, savings up to 15 % could be achieved by the usage of EEMs and VSDs. The economically viable savings add up to 11 % of the current consumption. Out of these savings, the majority comes from wider adaption of variable speed drives, as shown in Table 1.2.

Table 1.2 Potential savings with VSD application in fluid handling systems. (de Almeida, et al., 2000)

	Technical potential savings [TWh]		Economic potential savings [TWh]	
	Industry	**Service**	**Industry**	**Service**
Pumps	24.9	6.6	17.7	4.9
Fans	19.0	10.4	12.8	4.1
Compres-sors	6.6	4.5	4.3	2.4
Total	**50.4**	**21.4**	**34.8**	**11.4**

By comparing Table 1.1 and Table 1.2 it can be seen that 55 - 80 % of the savings potential are achieved by application of variable speed drives. The savings are mainly achieved by using the variable speed drive as a more efficient way to control the flow of fluids.

To achieve the savings mentioned, knowledge of the system behaviour is required. A large partition of the current fluid handling systems utilise flow restricting devices for controlling the system behaviour, which leads to low energy efficiency, especially when the system is not operated at full capacity. By utilising variable speed drives for control in such systems, a large partition of the flow restricting devices can become obsolete. Installing a variable

speed drive does not remove the need for flow restriction completely, as the systems are typically fairly complex, with different components requiring different flows or pressures.

Nowadays the variable speed drives are capable of driving the electric motor close to the motors best efficiency point (BEP) by utilising mathematical models of the motor behaviour. However in fluid handling systems the BEP of the electric motor is rarely the best efficiency point when observing the whole system. The BEP of fluid handling systems is determined by the required pressure and flow, which in turn are typically determined by an external control logic. The external control logic obtains measurements from external sensors such as airflow and pressure gauges and determines the rotational speed reference given to the variable speed drive.

The measurements used by the external control logic can be hard to obtain, especially in case the variable speed drive system is retrofitted to an existing operation. To overcome this limitation, the variable speed drive itself can be used as a soft sensor to determine the operating conditions. There has been many publications about the utilisation of VSD as a soft sensor in the recent years, many of the relating to the field of fluid handling. They have shown that the variable speed drive can be used to estimate the behaviour of a fluid handling system fairly accurately. Furthermore studies indicate that the variable speed drive can be additionally used for condition monitoring, allowing detection of efficiency reduction and possible sources of failure. (Ahonen, et al., 2011; Ahonen, et al., 2012; Tamminen, et al., 2011; Tamminen, et al., 2015a)

There are a few options for implementation of these methods. For one, they can be implemented to the variable speed drive itself. This approach is especially suitable for systems with already existing monitoring in-place, for example process data logging in industrial environments. Many of the existing industrial environments already contain monitoring and logging capabilities for variable speed drives, so adding an extra variable indicating the estimation method output is fairly simple. Another option is to implement the estimation method to the monitoring system itself. Many of the methods available utilise a fairly basic set of variables, most of which are most probably already monitored.

The third method, one this thesis focuses on, is sending the required variables over an Internet connection to a cloud service. A cloud service in essence is any resource or service that is provided over the Internet, in this case a service for monitoring fan systems. The basic structure of the system is described in Figure 1.2.

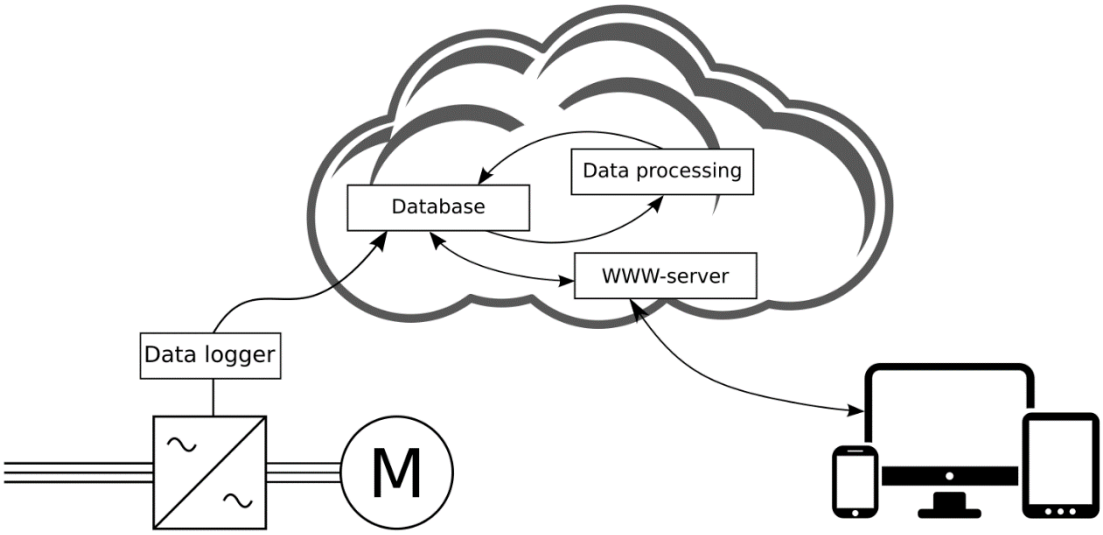


Figure 1.2 Basic structure of a monitoring system as a cloud service.

As shown in Figure 1.2, the basic operation consists of three main components; the variable speed drive, the cloud service, and an end device. The variable speed drive is equipped with a data logger capable of sending the logged variables to the database in the cloud. The database is used the preserve the logged variables. The estimation methods are then implemented to the cloud service, using the variables stored in the database. The calculated results are saved back to the database. Both the logged variables and estimation results are made available to the end user via web server.

The major advantage of this method is the practically unlimited processing power and storage available. This allows the same system to monitor a large number of variable speed drives and thus make the data available from anywhere in the world. The amount of available storage allows the system to save long periods of time, which is essential for determining the effects of system deterioration. It also enables the introduction of new and improved estimation methods, as the original values can be stored. As the estimation methods become

more and more complicated, the seemingly unlimited processing power allows implementation of methods impossible to implement on the variable speed drive itself. Furthermore as the variable speed drive monitoring can be made universal, the same data logging unit and cloud service backend can be utilised for every application, while the variables required for implementation of methods may differ. This allows application-specific methods to be developed on top of the existing data. In this thesis, the monitoring of fan systems is implemented as an example of such application-specific method.

1.1 Objectives of the work

The primary objective of this thesis is to create a demonstration of a service platform enabling remote monitoring of variable speed drives, more specifically variable speed drives used in fan systems. The secondary objectives of this thesis are to review the current state of logging the variables required from the variable speed drives and the reviewing of the current state of fan system monitoring methods.

1.2 Outline of the thesis

Chapter 2 introduces the fan systems in general. It covers the different control methods available for adjusting the fan system behaviour, the use of characteristic curves, and the estimation of fan system efficiency. Error sources in different estimation methods are also introduced.

Chapter 3 presents the current state of variable speed drive monitoring in general. This includes the existing monitoring capabilities in the VSD itself and the current state of dedicated data logger hardware.

Chapter 4 focuses on the different monitoring and estimation methods, which can be applied using the variable speed drive as a sensor. The chapter also introduces the methods for detecting life-reducing phenomena in fan systems, using only the estimates available from the variable speed drive.

Chapter 5 introduces the case study used in the testing and development of the monitoring interface described in this thesis.

Chapter 6 is dedicated to the development of a cloud service for monitoring of fan systems. The chapter focuses on the communication between the system parts; the data logger, the application programming interface, and the user interface. A comparison between different data encoding and transfer methods is described, as well as the design principles and methods used in the development of the demonstration environment.

Chapter 7 summarises the topics introduced in this thesis and discusses proposals for future work and research.

2. FAN SYSTEMS

A minimal fan system consists of two main components, a fan and an electric motor rotating the fan. This minimal system is enough for simple ventilation systems and is most commonly used to circulate air within a confined space, for rooftop ventilation, or for exhaust of gases, such as smoke and steam (U.S. Department of Energy, 2003). More commonly fan systems include some form of ducting to redirect the airflow to the locations required. In addition to the ducting itself, fan systems with ducting typically include filters to remove contaminants from the airstream. Furthermore especially in heating, ventilation and air-conditioning (HVAC) applications, heat exchangers, baffles, and outlet diffusers may be included. Heat exchangers can either recover heat energy from exhausted air or pre-heat or -cool the incoming air to increase comfort. Baffles are used to reduce the noise of the fan system by reducing turbulence of the airflow. Outlet diffusers are used to redirect and spread the airflow exiting the ductwork. An example of a typical fan system is presented in Figure 2.1. (U.S. Department of Energy, 2003)

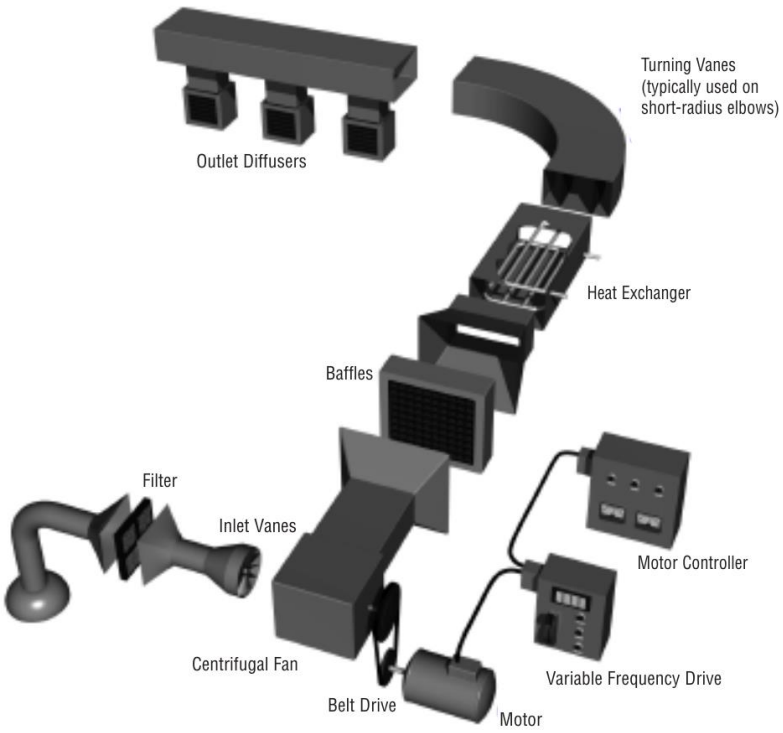


Figure 2.1 Example of a ventilation system components (U.S. Department of Energy, 2003).

The drive system of a typical fan system consists of two major components, an electric motor rotating the fan blades and some form of motor control. As the electric motors used in fan

systems are most commonly induction motors, the rotational speeds of the fan are limited to within a few percent of the synchronous motor speeds (U.S. Department of Energy, 2003). The synchronous motor speed n_s is determined by the motor supply frequency and the number of magnetic poles, and can be calculated using

$$n_s = \frac{120 \times f}{p}, \tag{2.1}$$

where "s" denotes synchronous, f is the supply frequency and p is the number of magnetic poles. In the most common, low-cost induction motors used in ventilation systems the number of magnetic poles is two, four, or six, leading to synchronous motor speeds of 3000, 1500, and 1000 rpm, respectively, when using a 50 Hz supply frequency (U.S. Department of Energy, 2003).

As the required rotational speed of a fan depends on the fan system requirements and the properties of the fan, the synchronous motor speeds are not compatible with all use cases. To work around this issue, the motor shaft is commonly linked to the fan shaft via belt, instead of a direct connection (CEATI International Inc., 2008). The belt drive acts as a transmission between the motor and fan, reducing or increasing the fan rotational speed to the required range. However, the belt drive only allows a static ratio of the rotational speed adjustment, which poses an issue when the desired operation of the fan covers a wide range of the performance curve. In addition the belt drive decreases efficiency, with the best efficiencies being up to 98 %. As the belt wears, the decrease in efficiency may be up to 3% within the first hour of operation. Furthermore the belt drive introduces a new component to the system, requiring maintenance and monitoring. The pulley diameters and pulley wearing, belt tensioners, and tension of the belt itself also affect the efficiency of belt driven systems. (Dereyne, et al., 2015; U.S. Department of Energy, 2003)

Nowadays the most common type of and electric variable speed drive is the variable frequency drive (VFD). As the speed of an alternating current (AC) motor is directly dependant on the supply frequency, a VFD can be used to alternate the motor rotational speed. The frequency of switching can be alternated by an external input, allowing for precise control

of the motor rotational speed. The basic operating circuit of a variable frequency drive is presented in Figure 2.2. (Carrier Corporation, 2005)

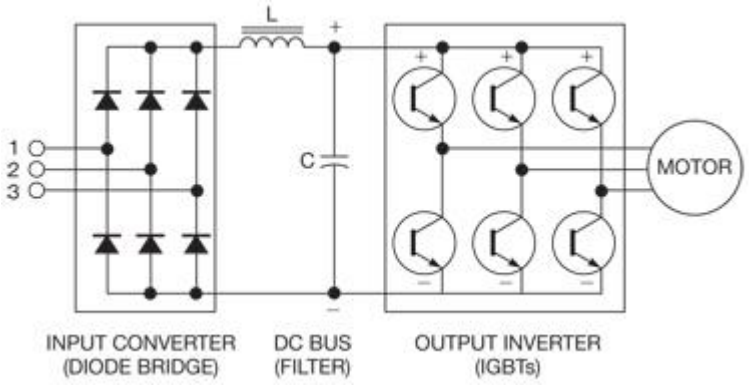


Figure 2.2 The simplified circuitry of a variable frequency drive. (Taranovich, 2012)

The pins 1, 2, 3 in Figure 2.2 are the three-phase alternating current inputs. The alternating current is converted to direct current (DC) by the diode bridge consisting of six diodes. As the DC contains ripple, it is smoothened by the filter consisting of a capacitor and an inductor. The DC is then converted back to AC by the output converter, consisting of six electronically controlled switches. The switches used in variable frequency drives are most commonly insulated-gate bipolar transistors (IGBTs). The switches are turned on and off on a high frequency, and the power fed to the motor is controlled by adjusting the duration of the on state. This method of control is called pulse width modulation. (Carrier Corporation, 2005; Taranovich, 2012)

2.1 Fan control methods

The operation of a fan at different rotational speeds can be estimated using affinity laws

$$\frac{Q_1}{Q_2} = \left(\frac{n_1}{n_2}\right) \tag{2.2}$$

$$\frac{p_1}{p_2} = \left(\frac{n_1}{n_2}\right)^2 \tag{2.3}$$

$$\frac{P_1}{P_2} = \left(\frac{n_1}{n_2}\right)^3, \tag{2.4}$$

where subscripts "1" and "2" denote different operating points, n is the fan rotational speed, Q is the fan flow rate, p is the pressure generated by the fan, and P is the fan power. As shown on the equations, the fan flow rate is directly proportional to the shaft speed, generated pressure is proportional to the square of the shaft speed, and the fan power is proportional to the cube of the shaft speed. However these equations do not take into account the possible static pressure difference, which may be present in the system. Furthermore the change in the rotational speed may affect the energy efficiency of the fan, which must be taken into account when using (2.4). (Tamminen, 2013)

A large partition of fan systems are driven on partial load at least some of the time, as ambient conditions, occupancy level of the building or production demands alter. With such systems, some method for reducing the airflow is required, as the fan system must be sized according to the maximum airflow needed. Traditionally the adjustment is reached by redirecting or restricting the airflow, with the cost of system efficiency. A comparison of different fan control methods is presented in Figure 2.3. (Ferreira, 2008; U.S. Department of Energy, 2003)

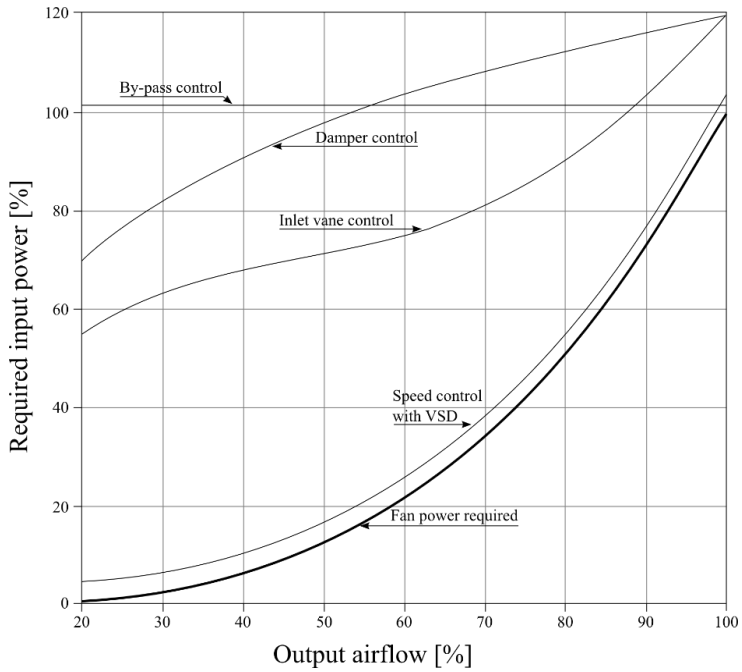


Figure 2.3 Comparison between different fan system airflow control methods (Ferreira, 2008).

Traditional fan control methods include by-pass, damper, and vane controls. By-pass control uses controllable channels to redirect part of the airflow away from the main ducting, therefore always requiring the maximum amount of input power. Dampers reduce the airflow by changing the restriction in the path of the airstream. By introducing additional resistance to the whole fan system, required power input increases dramatically when higher output airflow is required, as shown in Figure 2.3. This is caused by the fan operating point shifting away from the best efficiency point. Inlet vane control functions by introducing swirls to the airstream entering the fan. These swirls rotate in the same direction as the fan impeller, reducing the angle of attack between the incoming air and the fan blades. This in turn lowers the load on the fan and reduces fan pressure and produced airflow. (U.S. Department of Energy, 2003)

As the power required has approximately a cubic relation to the rotational speed of the fan, the use of rotational speed control is an attractive choice for controlling the output airflow of a fan system. In addition to the savings achieved by more efficient control, further savings can be achieved by directly connecting the electric motor to the fan shaft. This eliminates components such as belt drives and gears from the system, reducing system costs, power losses, and the number of failure points. As the prices of variable speed drives has decreased and their reliability has increased, they have become more and more common method for

implementing rotational speed control. (U.S. Department of Energy, 2003; Waide & Brunner, 2011)

2.2 Fan characteristic curves

Fan operation can be estimated by using characteristic curves, which provide information about the airflow rate in relation to the fan pressure production (Qp_F curve) and the airflow rate in relation to the fan shaft power consumption (QP curve). These curves are typically available from the manufacturer. An example of fan characteristic curves provided by manufacturer is shown in Figure 2.4. (Tamminen, et al., 2011)

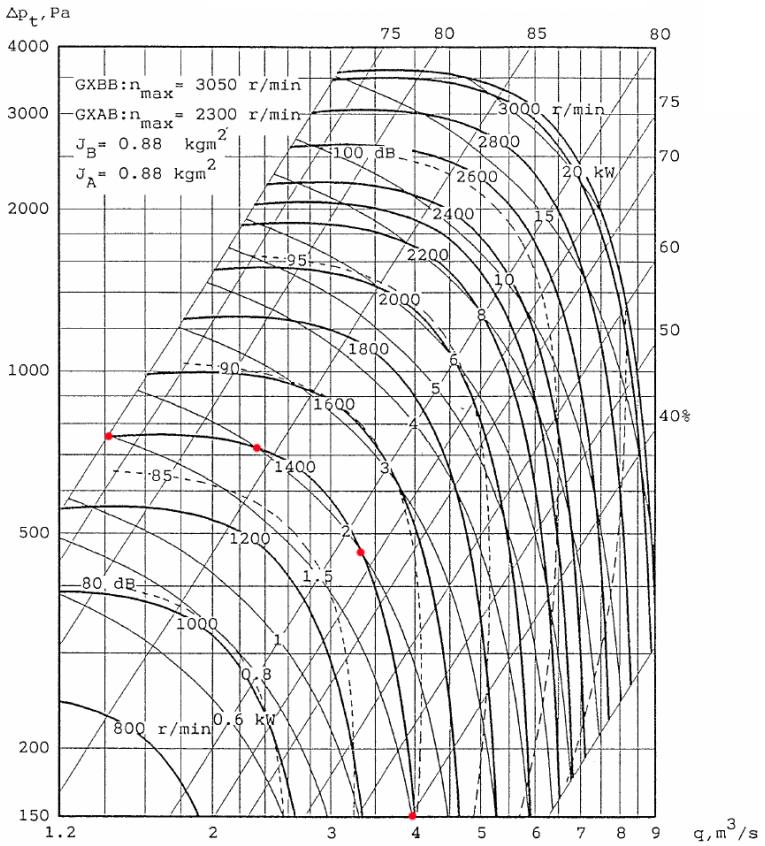


Figure 2.4 An example of fan characteristic curve provided by manufacturer (IV Produkt AB, n.d.).

As shown in the figure, fan curves provide all the essential information about the fan behaviour in a single diagram. Vertical axis on the left indicates the fan pressure and horizontal axis on the bottom indicates the airflow produced. In the top-right corner are the fan efficiency values. Note that the main axes are selected so that the efficiency lines are straight, allowing for easier optimisation of the fan control system. As the fan behaviour is highly

dependent on the shaft rotational speed and power, there are different curves for different speeds and powers. In addition, curves are provided for the fan noise level and efficiency.

Extracting the fan airflow volume in relation to shaft power at a single rotational speed from Figure 2.4 includes four steps. The procedure for extracting *QP* curve from characteristic curves is described in Figure 2.5.

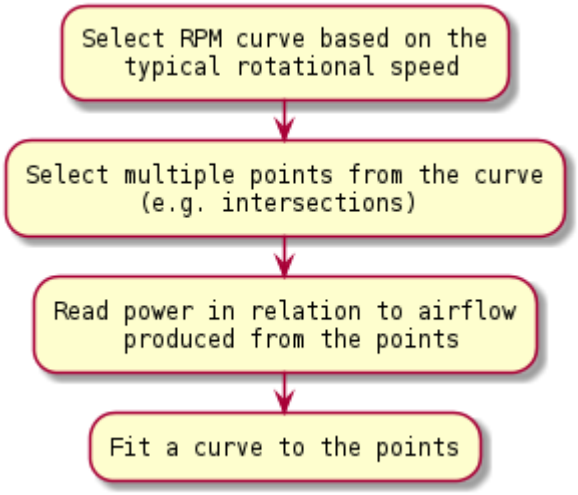


Figure 2.5 Procedure for extracting a *QP* curve.

First, a rotational speed close to the typical operational conditions is selected, in this case 1400 rpm. Second, multiple points are selected from the rotational speed line. In the Figure 2.4 red dots indicate the points, selected from the intersections of the rotational speed and power curves. Third, the required shaft power in relation to produced airflow is read from the selected rotational speed curve. The points read are presented in Table 2.1.

Table 2.1 Shaft power and air volume at 1400 rpm read from Figure 2.4.

Shaft power [kW]	**Airflow [m^3/s]**
1.50	1.40
2.00	2.35
2.00	3.35
1.50	4.00

Fourth and final step is to do a curve fit based on these points to produce an equation to estimate the airflow produced in relation to shaft power at the selected rotational speed. By using polynomial fit of third degree, the data presented in Table 2.1 produces the equation

$$Q_{1400} = -0.06308 \cdot P_{1400}^3 + 0.08475 \cdot P_{1400}^2 + 0.58682 \cdot P_{1400} + 1.84196. \quad (2.5)$$

By using affinity laws (2.2) – (2.4), the equation can be used to estimate the behaviour at different rotational speeds. First, the power estimate obtained from the variable speed drive is used to estimate power consumption at the rotational speed selected when forming the equation by using affinity law (2.4). Next, the equation formed from characteristic curves is used to calculate the airflow at the selected rotational speed. This airflow is then converted back to the measured rotational speed by using affinity law 2.2.

The main issue concerning the use of characteristic curves lies on the alternating of behaviour of the fan when using different rotational speeds. As the fan behaviour changes depending on the rotational speed, the single *QP* curve may not be enough to cover the whole operating area. For example compared to a low rotational speed, the fan may surge easier on higher rotational speeds. To overcome this issue, multiple curves can be extracted on different rotational speeds. This way the curve closest to the current rotational speed can be utilised to estimate the fan behaviour and thus provide more accurate results.

2.2.1 Error sources when using estimation based on characteristic curves

There are multiple methods for estimating the fan operating point based on a mathematical model, all of which require different sets of parameters. The most commonly utilised parameters are shaft power and fan rotational speed, both of which are nowadays estimated by even the most basic variable speed drives. The models commonly utilise the fan characteristic curves, which define the fan airflow rate in relation to the fan shaft power and the fan pressure in relation to the fan volume flow rate. The characteristic curve accuracy for all types of industrial fans (excluding jet fans) is standardised in ISO 13348:2007, as presented in Table 2.2.

Table 2.2 Manufacturing tolerance grades according to ISO 13348:2007 (International Organization for Standardization, 2007).

Tolerance grade	Volume flow rate	Fan pressure	Shaft power	Efficiency	Approximate power
AN1	± 1 %	± 1 %	+ 2 %	- 1 %	> 500 kW
AN2	± 2.5 %	± 2.5 %	+ 3 %	- 2 %	> 50 kW
AN3	± 5 %	± 5 %	+ 8 %	- 5 %	> 10 kW
AN4	± 10 %	± 10 %	+ 16 %	- 12 %	-

It should be noted, that the manufacturing tolerance grades only apply when the fan operating point efficiency is at least 0.9 times the stated best efficiency η_{opt}. Outside this range, tolerance grades are lower. With efficiency η in the range $0.8 \cdot \eta_{opt} < \eta < 0.9 \cdot \eta_{opt}$, tolerance grade is lowered by one grade. With $0.6 \cdot \eta_{opt} < \eta < 0.8 \cdot \eta_{opt}$, it's lowered two tolerance grades and for $\eta < 0.6 \cdot \eta_{opt}$, it's lowered three grades, if provided grades are still available. For the purposes of operating point estimation the changes in tolerances introduce further errors, as the efficiency in relation to the best efficiency must be known in addition to the operating point itself. (International Organization for Standardization, 2007)

As shown on Table 2.2, no negative limit is given to the fan shaft power. This effectively means that there is no limit on the negative deviation of power, leading to better efficiency. This is also shown on the efficiency tolerances, where there is no limit on how much the fan efficiency can exceed the given efficiency.

In addition to the error sources from the fan itself, the used drive system introduces error sources to the estimation. When using direct torque control (DTC), the rotational speed estimate given by the variable speed drive is shown to be within ±0.2 %, the shaft torque estimate within ±2.1 %, and the shaft power estimation within ±2.1 % of the nominal values (Ahonen, et al., 2011). Additional error sources caused by the drive system include the losses in bearings, possible belt drive, et cetera.

2.3 Fan system efficiency

The efficiency of a fan can be calculated using generated airflow and pressure in relation to power consumption using the equation

$$\eta = \frac{Q_{\mathrm{V}} \cdot p_{\mathrm{F}}}{P_{\mathrm{fan}}}. \tag{2.6}$$

However, this equation only gives indication of the fan efficiency. This is because even if the fan is operated at its best efficiency, most of the system losses are caused by ducting and other parts of the fan system. From a fan system energy efficiency viewpoint, a better indication is the fan system specific energy E_{s}, which indicates the fan energy consumption per transported air volume

$$E_{\mathrm{s}} = \frac{P_{\mathrm{total}}}{Q_{\mathrm{V}}}. \tag{2.7}$$

By using specific energy consumption as an indication of fan system performance, the efficiency of the whole fan system operation can be estimated. In general, a lower specific energy consumption equals better fan system efficiency. (Tamminen, et al., 2011)

The specific energy consumption can also be expressed as the specific fan power (SFP). The SFP is calculated by

$$SFP = \frac{P_{\mathrm{fan}}}{Q_{\mathrm{total}}} = \left[\frac{W}{l/s}\right] = \left[\frac{kW}{m^3/s}\right]. \tag{2.8}$$

The specific fan power also takes into account the whole system, including parts such as filters, heat exchangers, dampers, and ducting (Radgen, et al., 2008). The European Union has standardised the classification of fans based on the SFP in EN 13779 (European Standard, 2007). The specific fan power categories are listed in Table 2.3.

Table 2.3 Classification of specific fan power per fan (European Standard, 2007).

Category	Specific fan power $\left[\frac{W}{l/s}\right]$
SFP 1	< 0.5
SFP 2	0.50 – 0.75
SFP 3	0.75 – 1.25
SFP 4	1.25 – 2.00
SFP 5	2.00 – 3.00
SFP 6	3.00 – 4.50
SFP 7	> 4.50

There is no EU-wide legislation concerning the usage of SFP categories presented in Table 2.3. The categories are designed to standardise the way fan power consumption is represented. National regulations may however set requirements regarding the lowest accepted SFP category or a certain maximum SFP value for the whole building, individual fan system, or individual fans (European Standard, 2007). Many countries, such as Germany, Sweden, and United Kingdom have adopted the use of SFP to their legislation (Radgen, et al., 2008).

For example in the United Kingdom, legislation regarding the specific fan power have been taken into use. The requirements apply to the whole system, taking into account both the intake and exhaust fans. The SFP is calculated from the total circulated air and the power consumption of all the individual fans. Furthermore the requirements are for existing buildings as well and must be taken into account whenever air handling plant is provided or replaced. The requirements are shown in Table 2.4. (Department of Communities and Local Government, 2006)

Table 2.4 Maximum permissible specific fan power (Department of Communities and Local Government, 2006).

	New buildings $\left[\frac{W}{l/s}\right]$	**Existing buildings** $\left[\frac{W}{l/s}\right]$
Central mechanical ventilation including heating, cooling, and heat recovery	2.5	3
Central mechanical ventilation with heating and cooling	2	2.5
All other central systems	1.8	2
Local ventilation only units within the local area, such as window/wall/roof units, serving one room or area	0.5	0.5
Local ventilation only units remote the area, such as ceiling void or roof mounted units, serving one room or area	1.5	1.5
Other local units	0.8	0.8

When comparing Table 2.3 and Table 2.4 it can be seen that when a centralised system is used, the required specific fan power falling between categories SFP 4 and SFP 5. Local ventilation units have more strict requirements, with the required SFP in the range of categories SFP 2 and SFP 4.

3. MONITORING OF VARIABLE SPEED DRIVES

Traditionally the variable speed drives selected for heating, ventilation, and air conditioning applications are low-range and inexpensive units, with very limited features. As even the low-range products nowadays utilise sensorless estimates of rotational speed and shaft torque for motor control, these parameters are available in practically every variable speed drive (Holtz, 2000). As the processing power of variable speed drives has increased, communication interfaces providing these estimates to external devices have become more and more common. However, the estimates are only provided in real-time, with very short or no history available. To overcome this limitation, data logger, a separate device for logging the parameter values is normally required. Some variable speed drives do include basic logging capabilities, such as the load analyser found in the ACS580 by ABB (ABB Oy, 2015b). The load analyser logs the distribution of motor load and can be used to get a basic understanding of how the device operates over a longer period of time.

The data logger is commonly connected to the variable speed drive via fieldbus, a communication interface designed to allow the transmission of data between multiple devices in a private network. Variable speed drives commonly include a single fieldbus protocol as standard, with others being available through a separate communication module (ABB Oy, 2013; Vacon, 2014; Yaskawa America, Inc., 2015). By connecting a data logger to the fieldbus, the variable speed drive parameters can be recorded during a longer range of time. The available logging time is limited by the amount of storage in the data logger, the logging sample rate, and the number of parameters logged.

The current generation of stand-alone data loggers commonly use flash memory to save the data, which has reduced the effect of storage space constraints considerably (ABB Oy, 2014; ADFweb.com Srl, 2013; Vector Informatik GmbH, 2015). The use of exchangeable memory cards for storage has increased, allowing for easier data extraction and extension of storage capacity. Many of the devices also include a browser-based interface for extraction of data (ABB Oy, 2014; M2MLogger, 2015; Vector Informatik GmbH, 2015).

The current generation of remote data loggers typically send the gathered data either via File Transfer Protocol (FTP) or email (ABB Oy, 2014; Vector Informatik GmbH, 2015). State-of-the-art devices are also capable sending the data to a cloud service in real-time

(M2MLogger, 2015). The Internet connectivity is typically achieved either by Ethernet or a mobile network connection. One example of a device filling these requirements is the LogPRO m4 by M2MLogger, shown in Figure 3.1.

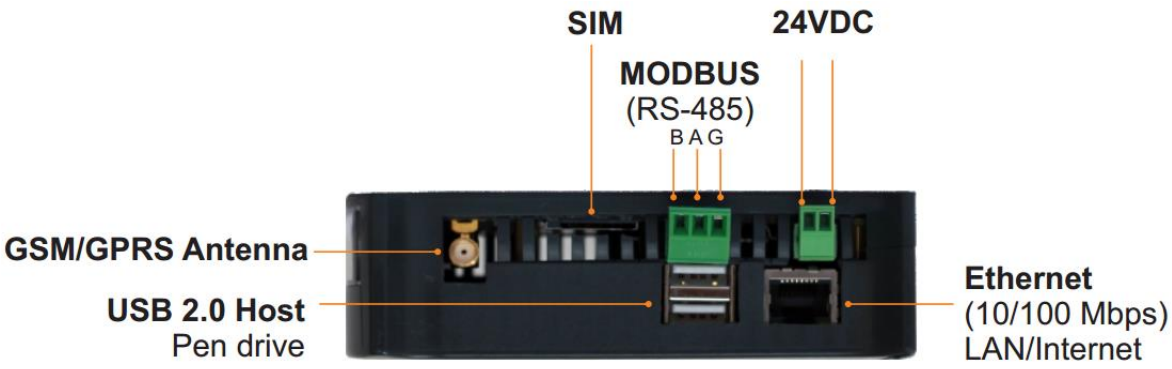


Figure 3.1 M2MLogger LogPRO m4 (M2MLogger, 2015).

As shown in Figure 3.1, the LogPRO m4 can be connected to Internet either by Ethernet or GPRS. The main restriction of the device is that the only protocol available for connecting to the variable speed drive is Modbus, either by RS-485 or Ethernet. When commissioning a completely new system, this limitation can in many cases be ignored as practically any VSD can have Modbus connectivity as an option. However in the case of retrofitting, the easiest and in many cases the only option is for the data logger to adjust to the existing system. As the protocols and interfaces can vary, a system with modular connection interfaces can be used. One such device is the NETA-21 by ABB, as shown in Figure 3.2.

Figure 3.2 NETA-21 data logger by ABB (ABB Oy, 2014).

The NETA-21 data logger is capable of communicating with variable speed drives through multiple protocols. The NETA-21 itself includes logging of the panel bus, Ethernet PC tool

communication, and Modbus/RTU via RS485. With the optional NEXA-21 expansion, logging via DDCS (Distributed Drive Communication System) with fibre optic cable can be added. The Internet connectivity is achieved either via Ethernet or an USB-connected 3G modem. The data is sent either by FTP or e-mail and is cached on a memory card between the send intervals. (ABB Oy, 2014)

4. FAN SYSTEM MONITORING USING VARIABLE SPEED DRIVE

The control of fan systems using variable speed drives is usually implemented on a dedicated control device. This device receives information gathered from sensors measuring values such as pressure and airflow generated by the fan, and are used to determine the required fan power. The control device then gives instructions to the variable speed drive using the various external reference inputs in the variable speed drive. In many cases measurement devices used by the external controller are not monitored. This can lead to an erroneous measurement, which can affect the whole system behaviour. By using the variable speed drive as a soft sensor, both the system behaviour and possible fault conditions can be monitored by using only one source of information; the variable speed drive.

The monitoring requirements on fan systems are quite low. Basic monitoring can be achieved by recording the motor rotational speed and the motor torque, both of which can be read from the variable speed drive. Power of the motor shaft can then be estimated with

$$P_{\text{shaft}} = \frac{2\pi nT}{60}, \tag{4.1}$$

where P_{shaft} is the motor shaft power in watts, n is the shaft rotational speed in revolutions per minute, and T is the motor torque in newton meters. As the motor rotational speed is in revolutions per minute, it must be divided by 60 seconds. Once the motor rotational speed and the motor shaft power are available, the fan characteristic curves can be utilised to calculate estimates of the fan operating point.

4.1 Model-based operating point estimation

Fan operating point is the base for almost all control and efficiency estimation and thus the knowledge of operating point is crucial. Operating point estimations also provide indications of points where the risk of a surge is possible. Traditionally operating point estimations are based on measurement of airflow rate and pressure, but not all fan systems are measured. Additionally, if measurements are only done during installation, shifting of the operating point will go unnoticed. This is also the case for modifications to the fan system, as it might be hard or even impossible to do new reference measurements after the modifications. By

using model-based sensorless estimations, it is possible to continuously monitor the fan system operating point without additional instrumentation. (Tamminen, 2013)

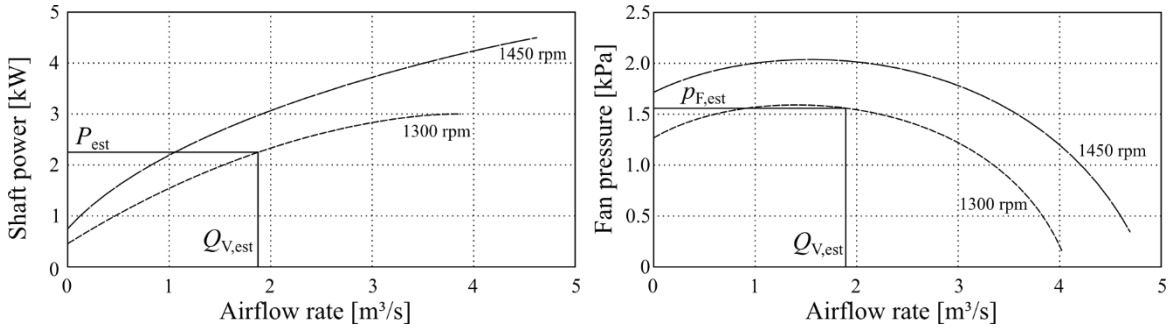
There are multiple methods for model-based operating point estimation, each of which require different input variables. These input variables commonly include shaft power and rotational speed estimates, both of which are nowadays available in even the most basic variable speed drives. However, it should be noted that especially larger fan systems commonly use belt drives as drive mechanism to lower the rotational speed. In this case, the motor rotational speed does not match the rotational speed of the fan and thus, cannot be used directly. Instead the ratio between the pulleys on the motor and fan shafts must be used to convert the motor rotational speed to the fan shaft speed. The fan rotational speed can be converted using

$$n_{\mathrm{F}} = \frac{d_{\mathrm{M}} \cdot n_{\mathrm{M}}}{d_{\mathrm{F}}}, \tag{4.2}$$

where subscript "F" stands for fan and "M" stands for motor, denoting the pulleys in question, *d* is the pulley diameter, and *n* is the rotational speed.

4.1.1 The *QP* method

The most basic estimation method, requiring only shaft power estimate, rotational speed estimate and *QP* curve, is called the *QP* method. By using affinity law for power estimation (2.4), the fan characteristic curve at known rotational speed can be shifted to the current rotational speed estimate. Airflow rate is then determined from the shifted *QP* curve by using the shaft power estimate. If needed, the fan pressure can then be determined by using the estimated airflow rate by using the shifted Qp_{F} curve. A graphic illustration of the method is presented in Figure 4.1.



(a) (b)

Figure 4.1 Estimation procedure using *QP* method. First, *QP* curve of known rotational speed (1450 rpm) is shifted to the current rotational speed (1300 rpm) using affinity laws. The airflow rate is then estimated from the shifted curve (a). If needed, estimated fan pressure can be determined from estimated airflow rate (b). (Tamminen, et al., 2011)

In Figure 4.1(a), the *QP* curve is monotonically increasing and because of this, the estimated airflow volume $Q_{V,est}$ can be found directly. In case the curve is nonmonotonic, assumption of the fan operational range is needed. In this case, only the monotonic part of operational range is included in the *QP* curve used when calculating $Q_{V,est}$.

4.1.2 The Qp_F method

The second fundamental model-based estimation method, called Qp_F method, uses the Qp_F curve to determine the airflow volume. This requires additional instrumentation to determine the actual generated pressure of the fan, and thus isn't possible to implement using only the estimations provided by the variable speed drive. On the other hand, having actual measurements from the process improves the accuracy of the model. The issues raised by nonmonotonic curves also applies to Qp_F method. The estimation procedure is fairly close to the one of *QP* method, first the Qp_F curve is rotational-speed-corrected using the affinity laws (2.2-2.4), after which the flow rate corresponding to the measured pressure is found from the corrected curve. (Tamminen, 2013)

4.1.3 The level correction method

The level correction method is an improvement of *QP* method, using a reference measurement of the fan flow rate, the rotational speed and the shaft power to fix one point of the

actual fan *QP* curve. This reference measurement can for example be done when commissioning the fan system. Based on the results of reference measurement, the difference between *QP* curve and the actual fan operation can be calculated using

$$P_{\text{bias}} = P(Q_{\text{meas}}, n_{\text{meas}}) - P_{\text{meas}}, \tag{4.3}$$

where the subscript "meas" denotes measured values and $P(Q_{\text{meas}},n_{\text{meas}})$ is the power given by the *QP* curve. The acquired P_{bias} is then used to correct the shaft power estimate with

$$P_{\text{corrected}} = P_{est} + P_{\text{bias}} \left(\frac{n}{n_{\text{meas}}}\right)^3. \tag{4.4}$$

This corrected estimate is then used as an input to the *QP* method. (Tamminen, 2013)

4.1.4 The Kernan method

The Kernan method is another improvement of *QP* method, with a correction measurement with pressure side valve closed (Kernan, et al., 2011). By closing the pressure side valve and running the fan system at different rotational speeds, a corrected exponent for affinity law (2.4) is calculated by

$$\kappa = \frac{\ln\left(\frac{P_{\text{SO},1}}{P_{\text{SO},2}}\right)}{\ln\left(\frac{n_1}{n_2}\right)}, \tag{4.5}$$

where the subscript "SO" is the shut-off condition, the subscript 1 the measurement at the rotational speed n_1, and the subscript 2 the measurement at the rotational speed n_2. The corrected affinity law (2.4) can be written as

$$P = P_0 \left(\frac{n}{n_0}\right)^{\kappa}. \tag{4.6}$$

Furthermore the power level correction of the *QP* curve is calculated with the shut-off power consumption at the nominal rotational speed by

$$P_{\text{bias}} = P_{\text{SO}} - P_{\text{SO,100\%}}, \tag{4.7}$$

where the subscript "SO,100%" is the measured shut-off power at the nominal rotational speed and "SO" the shut-off power given by the untreated model at the nominal rotational speed. P_{bias} is then used to correct the measurement similarly as in the level correction method. (Tamminen, 2013)

4.1.5 The p_F/P method

The p_F/P method uses the fan pressure divided by the fan shaft power to estimate the air flow rate. The method takes the shaft power, the pressure measurement, and the rotational speed estimate as inputs to the model and uses the p_F/P curve as model. It can be assumed that the p_F/P method is less influenced by the error in the affinity laws than for example QP method. However, the p_F/P method requires additional measurement of the fan pressure. (Tamminen, 2013)

4.1.6 The hybrid method

The hybrid method uses system curve estimated by the QP method in an operating region where the QP has preconditions to provide accurate flow rate estimates, e.g. close to the nominal rotational speed. Multiple reference measurements are done in the accurate range, and the system curve is formed by utilising the method of least squares. The hybrid method improves the estimations at lower rotational speeds compared to QP method, as the fan power consumption does not follow the affinity laws when the rotational speed has changed significantly compared to the nominal rotational speed. (Ahonen, et al., 2012; Tamminen, 2013)

4.1.7 The combined Qp_F/QP method

The combined Qp_F/QP method selects the method which is assumed to be more accurate from Qp_F and QP methods, and thus requires pressure measurement to be acquired. When the uncertainty of both methods is low, the flow rate estimates can be combined to achieve the final estimation. This can be accomplished by using the estimated uncertainties and weighting the estimates accordingly by

$$Q_{\text{est}} = \frac{\delta_{QP} \cdot Q_{\text{est},Qp_{\text{F}}} + \delta_{Qp_{\text{F}}} \cdot Q_{\text{est},QP}}{\delta_{QP} + \delta_{Qp_{\text{F}}}}, \tag{4.8}$$

where δ is the uncertainty of the method denoted by the subscript. (Tamminen, et al., 2014; Tamminen, 2013)

4.2 Detection of the impeller mass increase

As with any mechanical system, fan systems have abnormal operating conditions, which can lead to reduced lifetime. The main sources of such problems are for example aerodynamic or mechanical instability, dirt build-up on the fan impeller, et cetera. Traditionally these phenomena have been identified by using external monitoring equipment, such as pressure or vibration sensors. (Tamminen, 2013)

Many fan systems are used to transfer contaminated air and gases, which have the possibility to introduce contaminant build-up on the fan impeller. This build-up is traditionally been detected by visual inspection, which requires skilled personnel. As the contaminants attach to the fan, the rotational mass gradually increases. As the part of the contaminant build-up is removed either by vibration, external forces or careless maintenance, a mechanical imbalance of the fan impeller is caused. If the imbalance is not detected in time, it might lead to a fan failure, which in turn may lead to production losses. Therefore the detection of the contaminant build-up is vital. (Tamminen, et al., 2013)

By using the impeller mass increase as indication of contaminant build-up, the build-up can be detected prior to the resulting imbalance. The increase in the impeller mass is directly correlated to the inertia of the impeller, which can be detected without additional measurements. By accelerating the impeller with constant torque, the inertia of the fan impeller can be estimated, as shown in Figure 4.2. (Tamminen, 2013)

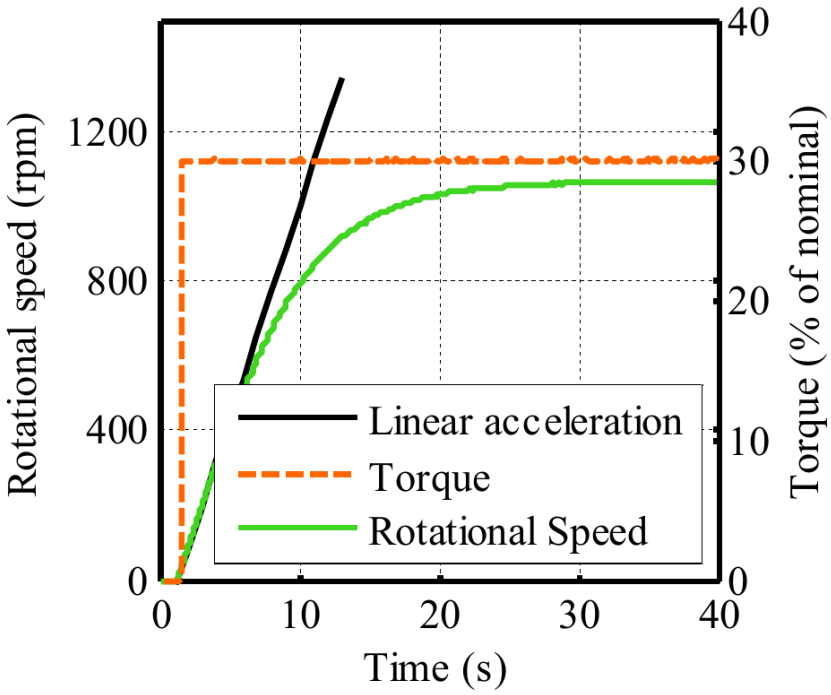


Figure 4.2 Start-up of a fan with constant torque, where "Linear acceleration" demonstrates the fan acceleration, if the aerodynamic effects are not taken into account. (Tamminen, 2013)

As seen from the figure, introducing a torque reference to the fan control causes the fan to start accelerating. At first, the acceleration is linear before aerodynamic effects start to slow down the acceleration. In addition, rotational speed estimates near zero can be erroneous, both because of the estimation methods and static friction. These properties limit the region used to estimate the angular acceleration α. The region is thus defined as the range between 30 rpm and $\sqrt{1-0.95}n_{\text{final}}$, e.g. for a fan operating typically at 1400 rpm the region would be from 30 to 313 rpm. (Tamminen, et al., 2013; Tamminen, et al., 2015a)

As the contaminants build up, angle α decreases as a result of larger inertia of the fan. As this method relies on comparison to previous values, the start-up procedure should be repeated multiple times before to ensure reliable results (Tamminen, 2013). As fan systems are usually run on a relatively similar cycle throughout their lifetime, the estimation can be done on every start-up of the fan.

The limitation of this method is the data acquisition interval in relation to the fan start-up duration. If the data acquisition is too slow, valid measurements from the linear acceleration range might be impossible to obtain. Also, as fan systems are generally run with rotational

speed control mode instead of torque reference control, some modifications to the method are required, as described in (Tamminen, et al., 2015b). An example of a fan start-up with linear rotational speed ramp is shown in Figure 4.3.

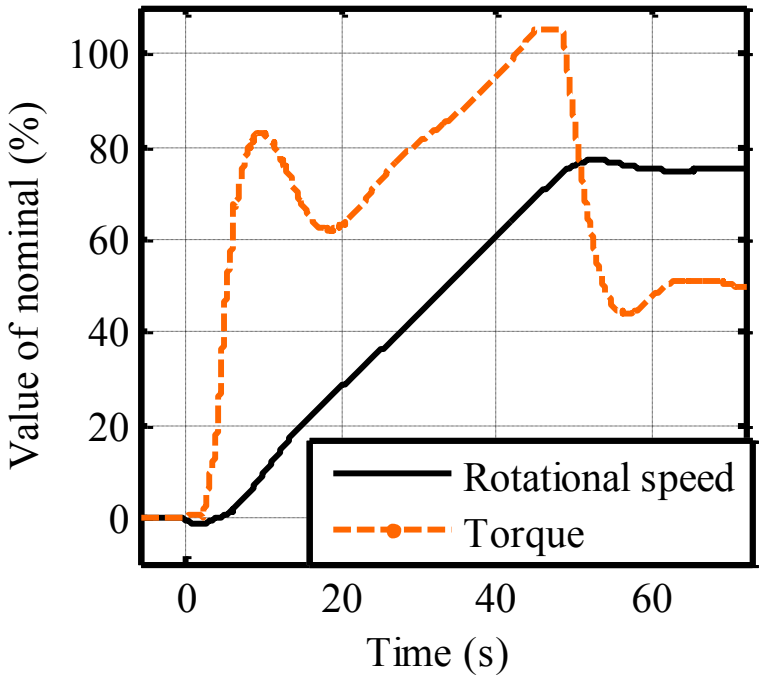


Figure 4.3 Example of a linear rotational speed ramp start-up of a fan. (Tamminen, et al., 2015b)

As seen from the figure, there are significant changes in the torque during the start-up of the fan, which makes the method described above unsuitable. Two methods for detecting inertia changes in such system are described in (Tamminen, et al., 2015b); the integrated torque method and the first peak method. The integrated torque method uses angular velocity difference during the linear portion of the start-up to estimate the inertia of the impeller. In contrast, the first peak method uses the torque peak (around 10 seconds in the figure) as the torque value. To further improve the estimation, an average of timeframe around the first peak can be used. (Tamminen, et al., 2015b)

In general, the inertia of a rotational speed controlled fans can be calculated with

$$J = \frac{\sum_{k=a}^{b} T(k)\Delta t}{2\pi \frac{n_b - n_a}{60}}, \tag{4.9}$$

where T is the torque in discrete time steps, b is the index where the discrete time rotational speed n_b corresponds to $\sqrt{1 - 0.95}n_{\text{final}}$, and a is the index n_a is 30 rpm and Δt is the sampling interval (Tamminen, et al., 2015a).

According to (Tamminen, et al., 2015a), the general expression gives good indication of the inertia increase but a simplified method results in more consistent estimates. In the start-up presented in Figure 4.3 the first peak occurs at 10 seconds. The impeller inertia can be calculated from ±2 seconds time frame around the first torque peak, corresponding to time indices a and b. The torque is averaged from these samples and the slope k of the rotational speed is linearly interpolated between the samples. The inertia is then calculated from

$$J = \frac{\frac{1}{b - a + 1}\sum_{i=a}^{b} T(i)}{k}. \tag{4.10}$$

As shown in (Tamminen, et al., 2015b), both the integrated torque method and the first peak method can detect the impeller mass increase over time, with the first peak method proving more accurate. Furthermore the estimation accuracy of a standard industrial frequency converter was proven sufficient for the implementation of these methods. However, because of the sampling time requirements, the methods are more suitable for implementation in the variable speed drive, rather than as a cloud service. Another possibility could be to use triggering to start faster sample collection during the system start-up.

5. CASE PITÄJÄNMÄKI

The case example used in this thesis was the Tellus building occupied by ABB Oy, located at Pitäjänmäki, Helsinki, Finland. The Pitäjänmäki unit is responsible for the development of electric motors, generators, variable frequency drives, and energy management solutions to name a few (ABB Oy, 2015a). The office building is built in 1999, and uses relatively modern technology. In addition to the traditional functions, the building is also used as a testbed for new technologies (J. Tolvanen, 2015, pers. comm., 20 March). The building consists of 10 floors, 6 of which are parking areas and 3 ½ floors of office and social areas (Kosonen, 2010).

The building is equipped with ABB ThermoNet building service platform, which controls the heating, air conditioning and cooling of the building (Kosonen, 2010). The fan system selected as testbed consists of a direct driven fan, GXAB-5-050 by IV Produkt AB, driven by an ABB 4-pole induction motor, and ABB ACS580 variable frequency drive. The data logger used is a development version of ABB NETA-21. The fan is used as an exhaust fan for the social areas, including the kitchen and dining area. The fan characteristic curves are presented in Figure 2.4 and the parameters of the electric motor in Table 5.1. (O. Alkkiomäki, 2015, pers. comm., 18 June and 19 August)

Table 5.1 The testbed fan system motor nominal parameters (O Alkkiomäki, 2015, pers. comm., 18 June).

Power	7.5 kW
Voltage	400 V
Current	14.9 A
Speed	1450 rpm
Frequency	50 Hz

The data was collected with the NETA-21 in the period of 17.03.2015 – 09.08.2015. The parameters logged are presented in Table 5.2.

Table 5.2 Parameters logged by the NETA-21.

Parameter name	Parameter unit	Parameter number
Motor speed used	rpm	01.01
Motor current	A	01.07
Motor torque	%	01.10
DC voltage	V	01.11
Output voltage	V	01.13
Actual flux	%	01.24
Control board temperature	°C	05.10

As shown in Table 5.2, the number of parameters logged is relatively high. With the slightly older firmware used during the measurement period, the fastest sample rate was around 2 seconds. However the later firmware versions allow logging of up to 20 parameters on 1 second interval (O Alkkiomäki, 2015, pers. comm., 27 November).

The parameters required for monitoring of the fan system are Motor speed used (number 01.01) and Motor torque (01.10). In addition to these parameters, the motor nominal torque is required, as the motor torque parameter reported by the VSD is presented as a percentage of the motor nominal torque. The nominal torque can be calculated with (4.1) by using the values presented in Table 5.1. The motor nominal torque was calculated to be 49.4 Nm.

5.1 Methods selected for implementation

The methods selected for the prototype platform were selected so that the parameters provided by the variable speed drive would be the only ones used. There are other measurements used on the ThermoNet system controlling the fan system, but these were only used to confirm the results of estimation methods.

The fan operating point estimation method used was the *QP* method, requiring only the motor rotational speed and torque estimates. The shaft power of the fan can be calculated using (4.1). The fan is typically run in the range of 1200 – 1500 rpm, with 1480 rpm being the most common rotational speed. The closest fan curve, presented in Figure 2.4, is 1400 rpm. By reading the produced airflow in relation to the power required, the equation

$$Q_{1400} = 1.84196 + 0.58682 \cdot P_{1400} + 0.08475 \cdot P_{1400}^2 - 0.06308 \cdot P_{1400}^3 \quad (5.1)$$

can be formed. The subscript 1400 denotes the values read from the 1400 rpm curve, Q is the airflow produced by the fan, and P is the power required. By using affinity laws 2.2 and 2.4, the QP curve can be estimated on other rotational speeds. In some cases the curve produced by the third-degree estimation may have multiple intersections and further assumptions are required.

The fan impeller mass increase detection was not implemented to the demo system for a variety of reasons. First of all, the data logging interval was fairly high in comparison to the fan start-up time. The fan system is quite small, allowing it the fan to achieve the range of aerodynamic resistance very fast. The NETA-21 revision available at the moment of writing can achieve logging intervals in the range of seconds, not milliseconds, which the demo case would require. Secondly the fan system at Pitäjänmäki is controlled by a rotational speed reference and the fan start-up utilises a fairly long ramp. The first peak method could be applied to tackle this issue, but it was not available until the very end of the development cycle. However it should be noted that with larger fan systems, the first peak method could be implemented even with the current NETA-21 revision. As the fan size increases, the start-up time is also increased, allowing the use of longer logging intervals.

6. CLOUD SERVICE FOR VARIABLE SPEED DRIVE MONITORING

We currently live in a data-driven society. More and more data is gathered every day, from larger variety of devices and sensors. By the end of 2015 it is estimated that almost 4.9 billion devices will be connected to the Internet of Things (IoT) and the Industrial Internet. By 2020 it is estimated that the number is in the range of 25 billion (Gartner, Inc, 2014) to 50 billion (Cisco Systems, Inc., 2015) and according to some sources, even these estimates are on the low side (Soley, 2014). Therefore, the amount of data available for observing the day-to-day life and operation of machines is mind-boggling. According to General Electric, the technical innovations related to Industrial Internet could find direct applications accounting for more than US$ 32.3 trillion in economic activity (General Electric, 2013).

"*If data doesn't change your behaviour, why bother collecting it?*" (Croll, 2015). Just by collecting data, not much can be gained. Instead, the data gathered needs analysing and processing to make it valuable for the end-user. The data can for example be used to increase efficiency, decrease downtime of machinery, and in integration with other production and enterprise data (Soley, 2014). As variable speed drives have integrated to practically everywhere in the recent years, they can provide a useful data source in both system optimisation and the development of smarter, more advanced control methods. Furthermore once the data is available, it can be easily combined with other measurements and observations. In the case of fan systems, these measurements could for example be related to air quality and quantity.

By processing the data gathered from variable speed driven systems, both system efficiency and the state of the system can be monitored more closely. From the processed data, it is then possible to optimise the system operation and thus reduce costs for the end-user. Previously this data has been collected and analysed on case-by-case basis from measurement done on-demand, usually during the commissioning of the systems. By using constant data gathering, analysis too can be performed constantly, which allows detection of changes in operation during the system lifetime.

Instead of processing the data on the VSD, the data processing can be done by the cloud service the data is sent to. This decreases the processing power required from the VSD, thus

reducing the cost of the drive itself. Furthermore if for example the formula used for estimating the operation proves to be incorrect, the original data is still available to re-analyse the history.

The main issue concerning the Industrial Internet and the Internet of Things is the lack of standardisation. Because they are in the early phases of evolution, there are no set methods and protocols for data transfer between devices and servers, let alone in machine-to-machine (M2M) communication. This means that even though there are many companies developing devices and applications for the Industrial Internet, the applications created by one vendor do not integrate with the applications from another vendor, at least not yet. This leads to a complex, fragmented, confusing, and less cost-effective integration of services. (Soley, 2014)

To try and overcome the issues caused by the fragmentation in the communication protocols, multiple consortiums have been formed, each competing to make their own idea to a standard. Some specifications have been published, mainly focusing on the overall architecture and system requirements (Muhonen, 2015). The model of vertical software industry evolution suggests that there are five phases in the evolution of software industries. In the first, Innovation phase, the software development focuses on automating the core business. The second, Productization and Standardization phase involves adopting the best practises of their competitors, which may happen in-house (typically in the case of market leader) or as a collaboration between competitors via a joint venture. The third, Adoption and Transition phase, typically has a growing user base and market share of the standardised offerings. In the fourth phase, Service and Variation, one dominant design emerges, which in turn attracts the majority of the subsequent development. In Renewal phase, new software-related opportunities are looked into as a source of gaining a competitive advantage, which leads to a new evolution cycle. (Tyrväinen, et al., 2008)

The Industrial Internet and Internet of Things can be considered to be somewhere in-between of the Innovation and Productization and Standardization phases. The technology is available and being deployed but standardisation lacks behind, slowing down the adaptation of the technology. With the development of a standardised application programming interfaces,

developers can concentrate on developing the features of the device instead of the communication profiles. Furthermore the full potential of the networked devices can be unleashed more freely, as the same communication standards can be used instead of developing interfaces to interact with multiple device manufacturers.

6.1 Web service design

As there are currently no standards for the Industrial Internet, every system is developed as a separate entity. The communication method used in the Industrial Internet can be virtually anything, but by using the Hypertext Transfer Protocol (HTTP) multiple issues can be overcome. For example, firewalls in industrial environments are commonly configured to be fairly strict, allowing only certain protocols to pass through, HTTP being one of the most common. Furthermore HTTP has fairly simple communication profile and is therefore simple to implement in multiple platforms. (Laurent, et al., 2001)

There are multiple design principles involving web service design for monitoring and controlling applications, such as Remote Procedure Call, SOAP, and Representational State Transfer. All of the methods can be implemented to work through multiple transport routes, such as HTTP. In addition, they allow functionality to implement both, control and monitoring of the devices.

6.1.1 Remote Procedure Call

As the protocol name suggests, the Remote Procedure Call (RPC) protocol is a protocol most commonly used to call remote functions. The request made by the client must include a method name, any number of parameters, and a target server. The server then runs the defined method in the request with the parameters, and returns the result in response. (Laurent, et al., 2001)

The Remote Procedure Call itself doesn't define data transport or encoding methods, thus there are multiple transport and encoding methods available for use with RPC. When using HTTP as transport protocol, RPC is most commonly implemented as XML-RPC. XML stands for Extensible Markup Language, a markup language for encoding documents, which is both human- and machine-readable. XML-RPC defines all the most commonly used data types, such as boolean, integer, double, string, and datetime. In addition, the data types can

be constructed to arrays and structures. Binary data can be transferred using Base64-encoding. Listing 6.1 shows an example of XML-RPC request. (Laurent, et al., 2001)

```
1  POST /LocalWeather HTTP/1.0
2  Host: www.mindstrm.com
3  Content-Type: text/xml; charset="utf-8"
4  Content-Length: 209
5
6  <?xml version="1.0"?>
7  <methodCall>
8    <methodName>WeatherStation.GetCurrentTemperature</methodName>
9    <params>
10     <param>
12         <value><string>Celsius</string></value>
13     </param>
14   </params>
15 </methodCall>
```

Listing 6.1 An example of XML-RPC request.

Lines 1-4 of the example show the required HTTP headers, defining the request method, API endpoint, request content type and length. The payload itself starts on line 6 with the XML version definition. Lines 7-15 include the RPC method call itself, calling the method `GetCurrentTemperature` from `WeatherStation` namespace. The only parameter for this request is the unit of temperature measurement. The response for the request is presented in Listing 6.2.

```
1  HTTP/1.0 200 OK
2  Content-Type: text/xml; charset="utf-8"
3  Content-Length: 150
4
5  <?xml version="1.0"?>
6  <methodResponse>
7    <params>
8      <param>
9          <value><double>26.6</double></value>
10     </param>
12   </params>
13 </methodResponse>
```

Listing 6.2 An example of XML-RPC response.

The response format presented in Listing 6.2 follows the request format. The headers indicate that the request was successful, the response content type, encoding, and length. The response itself returns the requested temperature as a double. The total payload transferred in both the request and response is 359 characters.

6.1.2 SOAP

SOAP, originally an acronym of Simple Object Access Protocol, is an extension of XML-RPC, with some new functionalities. SOAP is recommended by the World Wide Web Consortium, which means that the protocol has gone through a review process and is stable, thoroughly tested, and is encouraged for widespread implementation (World Wide Web Consortium, 2007). The protocol itself doesn't define a transport route, even though HTTP is the only one described in the specification. Thus, the protocol can be used via any transport route. SOAP is designed to allow information exchange in distributed computing environment and thus is well-suited for M2M-communication. There is no concept of central server in SOAP, which means that different nodes can be considered equal. (Englander, 2002)

The main feature of SOAP in comparison to XML-RPC is an envelope containing the message. The envelope allows confining any existing XML data into logical units, such as a header and a body. It doesn't however define the message structure itself, nor does it force the document itself to do so. An example of SOAP request is presented in Listing 6.3. (Englander, 2002; Richardson & Ruby, 2007)

```
1  POST /LocalWeather HTTP/1.0
2  Host: www.mindstrm.com
3  Content-Type: text/xml; charset="utf-8"
4  Content-Length: 323
5  SOAPAction: "WeatherStation"
6
7  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
8    <SOAP-ENV:Body>
9      <m:GetCurrentTemperature xmlns:m="WeatherStation">
10       <m:scale>Celsius</m:scale>
11     </m:GetCurrentTemperature>
12   </SOAP-ENV:Body>
13 </SOAP-ENV:Envelope>
```

Listing 6.3 An example of SOAP request (Englander, 2002).

The request presented in Listing 6.3 is essentially the same as presented in Listing 6.1, using SOAP instead of XML-RPC. The only major change in the header information is the `SOAPAction`, indicating the intent of the message. This can for example be used to determine the required resources without parsing the full payload. The payload itself starts from the line 7 with the SOAP envelope. Inside the envelope is the request body, requesting the current temperature reading in Celsius degrees. An example of SOAP response is presented in Listing 6.4 (Englander, 2002)

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: 348

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetCurrentTemperatureResponse xmlns:m="WeatherStation">
      <m:temperature>26.6</m:temperature>
    </m:GetCurrentTemperatureResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 6.4 An example of SOAP response (Englander, 2002).

The SOAP response follows the basic structure of the request, responding with the current temperature. The only major change is the replacement of `scale` element with `temperature` element. The total payload transferred in both the request and response is 671 characters, almost double compared to the same request in RPC-XML. This is mainly because of the additional schema definitions included in the SOAP.

As SOAP allows for peer-to-peer communication, it is well-suited for the Industrial Internet. However SOAP doesn't define any data structures or message format, which leads to the issue of standardisation. As there isn't any standardisation of the messages, M2M communication cannot be implemented without the knowledge of the features and message formats of both devices.

6.1.3 Representational State Transfer

Web-based application programming interfaces implemented with Representational State Transfer (REST, RESTful API) have increased rapidly in recent years. Companies such as Google, Twitter, and Facebook have all implemented their application programming interfaces in REST. This is mainly because this style of API fits fairly well together with object-oriented programming, a programming style designed to deal with representing real life objects. In comparison to RPC and SOAP, APIs designed to be RESTful are often simpler and easier to understand for the programmer utilising the API. REST doesn't define the transport protocol or data format, but is commonly associated with XML or JSON transferred via HTTP. (Richardson & Ruby, 2007)

RESTful APIs make use of the methods defined in HTTP specification whereas RPC and SOAP only utilise POST method. Table 6.1 shows the methods and their definition as defined in standard RFC7231 (Internet Engineering Task Force, 2014).

Table 6.1 HTTP methods defined in RFC7231 (Internet Engineering Task Force, 2014).

Method	Description
GET	Transfer a current representation of the target resource.
HEAD	Same as GET, but only transfer the status line and header section.
POST	Perform resource-specific processing on the request payload.
PUT	Replace all current representations of the target resource.
DELETE	Remove all current representations identified by the target resource.
CON-NECT	Establish a tunnel to the server identified by the target resource.
OPTIONS	Describe the communication options for the target resource.
TRACE	Perform a message loop-back test along the path to the target resource.

The methods GET, HEAD, OPTIONS, and TRACE are considered safe, meaning that the target resource is not supposed to be modified with such request, a design principle often ignored (Richardson & Ruby, 2007). Most of the methods defined in RFC7231 are fairly self-explanatory, GET is used to fetch data, POST to post new data, and DELETE to remove data. RESTful services utilise these methods to define the functionality in the API endpoints, making the API more consistent and simpler (Richardson & Ruby, 2007). In comparison both XML-RPC and SOAP use only the POST method and define the requested function in the message content instead (Laurent, et al., 2001; Englander, 2002). The temperature request presented earlier with XML-RPC and SOAP is presented with REST in Listing 6.5.

```
1  GET /LocalWeather HTTP/1.0
2  Host: www.mindstrm.com
3  Content-Type: application/json; charset="utf-8"
4  Content-Length: 21
5
6  {"scale": "Celsius"}
```

Listing 6.5 An example of REST request.

In comparison to the XML-RPC and SOAP requests, the request method used in REST request is GET instead of POST, indicating fetching of data. To reduce the overhead of the payload, the content type is changed from `text/xml` to `application/json` and the content is changed accordingly. The overall length of the request payload is reduced to 21 characters, in comparison to 209 characters with XML-RPC and 323 characters with SOAP. An example of the REST response is shown in Listing 6.6.

```
1 HTTP/1.0 200 OK
2 Content-Type: application/json; charset="utf-8"
3 Content-Length: 29
4
5 {“currentTemperature”: 26.6}
```

Listing 6.6 An example of REST response.

The response to the current temperature request is similar to the request, with the content changing to report the current temperature reading. The payload of the response is 29 characters, whereas XML-RPC required 150 characters and SOAP 348 characters. The total payload of both the request and response is 50 characters, in comparison to the 359 and 671 characters of XML-RPC and SOAP, respectively.

6.2 Implementation of fan system monitoring methods as a web service

The web service for implementing basic variable speed drive monitoring was developed using REST design practices with JSON as the data encoding format, as the total amount of VSDs monitored was assumed to be huge. As many of the cloud service providers charge for the data transferred, savings can be achieved by minimising the amount of overhead per request (Leinwand, 2009). However, with all major cloud computing providers only the amount of outgoing data transfer is billed for and incoming data is considered free (Amazon Web Services, Inc, 2015; Google, 2015; Microsoft, 2015). This effectively means that the amount of overhead in the data coming from the variable speed drives doesn’t matter and therefore the data sent from the VSD to the web service could also utilise SOAP or XML-RPC. Although the received data is free from the perspective of the cloud service, the cost of overhead can be quite large when considering the bandwidth usage of the device sending the data. In many cases, the data logger is connected to the Internet via metered connection with strict limits on the amount of transferred data.

Another reason for selecting REST over SOAP or XML-RPC is the processing power requirements. The complex structure and additional overhead of SOAP also increases processing power requirements, a major component in the pricing of cloud computing (Amazon Web Services, Inc, 2015; Google, 2015; Microsoft, 2015). In practice the decrease in the processing power requirements means that one instance can serve higher number of clients, reducing costs per client.

While the reduced amount of overhead and processing power are both valid reasons for selecting REST, the main reason for the selection was the simplicity of the REST based design. As the system consists of real-life objects, using object oriented architecture for the web service implementation is both simple and logical. The API is divided into multiple endpoints, all providing functionality related to different components of the full system.

6.2.1 The NETA-21 API

The data from variable speed drives is collected with ABB NETA-21 remote monitoring tool, which can monitor up to 64 variable speed drives (ABB Oy, 2014). The REST API endpoints for the NETA-21 and associated methods are presented in Table 6.2.

Table 6.2 The NETA API endpoints and associated methods.

/neta/	
GET	List all NETAs available to the client.
/neta/<serial number>/	
GET	Fetch NETA parameters.
POST	Submit data from NETA, allows sending of multiple drives at once.
PUT	Update NETA parameters.
DELETE	Remove NETA and all associated drives.

As Table 6.2 shows, by utilising the RESTful design principles the number of API endpoints can be kept minimal while still keeping the API logical. At the endpoint `/neta/`, the only method available is GET, listing all the NETAs and their parameters available to the client.

By appending the NETA serial number to the endpoint, access to the specific NETA is granted. The device specific endpoint exposes additional methods POST, PUT, and DELETE to the API. The POST method allows sending data from multiple drives connected to the NETA at once. The response to the data submit is the status code 201 indicating creation of new resources or an error code accompanied by an error message. The PUT method allows the update of device specific parameters, exposed by the GET request. The PUT method response consists of a HTTP status code and the updated data. The DELETE method does what is expected and removes the NETA, drives associated with it, and their data from the

system. The delete method responds with either the status code 204 or an error code with associated error message. The status code 204 indicates that the delete was successful and there is no data in the endpoint anymore. The response of the GET method to the device specific endpoint is shown in Listing 6.7.

```
HTTP/1.0 200 OK
Content-Type: application/json; charset="utf-8"
Content-Length: 178

{
    "name": "Demo device",
    "location": "ABB Pitäjänmäki",
    "description": "",
    "coordinates": {
        "latitude": 60.221644,
        "longitude": 24.874988
    }
}
```

Listing 6.7 The response of a GET request made to the device specific NETA API endpoint.

As shown in the Listing 6.7, the response to a GET request is fairly simple and remains human readable. However, in a production environment the response would be condensed to a single line with the unnecessary whitespaces removed, reducing the amount of data transferred and making the response less readable.

6.2.2 The variable speed drive API

The variable speed drive application programming interface provides access to the variables of the VSD. The API is fairly similar to the NETA API, as shown in Table 6.3.

Table 6.3 The VSD API endpoints and associated methods.

/neta/<serial number>/drive/	
GET	List all drives connected to the NETA.
/neta/<serial number>/drive/<serial number>/	
GET	Fetch drive variables.
PUT	Update drive variables.
POST	Submit data from single drive.
DELETE	Remove drive and all associated data.

As each drive is connected to a NETA, the VSD API expands the NETA API by introducing a new endpoint to it. In addition to keeping the API structure logical, this allows one to list all drives connected to a specific NETA. The GET, PUT, and DELETE methods exposed by the API are roughly similar to the ones of the NETA application programming interface. A new introduction to the API is the POST method, allowing submit of the data collected by the NETA. By using this endpoint instead of the one exposed by the NETA API, data from a single drive can be sent. This for example allows using different submit intervals for different drives, depending on the requirements of the application. An example of the GET method to the drive specific endpoint is shown in Listing 6.8.

```
HTTP/1.0 200 OK
Content-Type: application/json; charset="utf-8"
Content-Length: 483

{
    "name": "VP4 G3 308 TF",
    "location": "Tellus",
    "description": "308 TTK, GXAB-5-063-1-1-3-0, ABB Fläkt",
    "coordinates": {"latitude": 60.221644, "longitude": 24.874988},
    "drive_type": "ACS580",
    "system": {
        "type": "fan",
        "motor": {"rpm": 1450, "power": 7500},
        "curve": {
            "multipliers": [1.84196, 0.58682, 0.08475, -0.06308],
            "rpm": 1400}
    }
}
```

Listing 6.8 The response of a GET request made to the device specific VSD API endpoint.

As the Listing 6.8 shows, the response of the VSD API is similar to the one of NETA API. The name, location, description, and coordinates are presented similarly to the NETA API. Coordinates for each drive are included as the NETA-21 can connect to devices through Ethernet, allowing great distances between the data logger and the variable speed drive. The drive serial number is not included in the message body to keep all the fields defined in the field editable, reducing the API complexity. The general endpoint exposes the same data as key-value pairs, with the key being the drive serial number and the value the same as in the device specific endpoint. This way, the number of requests is minimised as each device doesn't require a separate request when initialising the user interface (UI).

New additions compared to the NETA API include the drive type and system description. The drive type can for example be used as a method to lookup the parameters required for the calculations, in case they differ from one drive type to another. The system parameters provide the information required for implementing different calculation methods. The type parameter defines the system type, in this case a fan system. Other options could for example include compressor or pump systems. Furthermore by allowing an empty value in the field, the system could be used to monitor variable speed drives in general, without a specific function.

The motor parameters described in the GET request are the nominal values as presented in Table 5.1 and are for example used to calculate the motor nominal torque. The curve parameters are used to implement the *QP* method. The curve multipliers determine the multipliers in (5.1), with the list index describing the exponent of power in ascending order. In this case,

the *QP* curve was formed by using a function of the third degree, but by defining the exponents as a list, any degree can be used. The curve RPM defines the rotational speed used to determine the curve, allowing the use of affinity laws to estimate behaviour on different rotational speeds.

6.2.3 The parameter API

The parameter application programming interface is used to fetch parameter values logged by the NETA. The API extends the VSD API, appending an additional arguments to the URI. The parameter API endpoints and the methods associated are presented in Table 6.4.

Table 6.4 The parameter API endpoints and associated methods.

/neta/<serial number>/drive/<serial number>/parameter/	
GET	Fetch list of parameters logged from the drive.
/neta/<sn>/drive/<sn>/parameter/<parameter id>/	
GET	Fetch parameter values, defaulting to last 24 hours and 10 second averaging period. Other time ranges can be fetched by defining query parameters: `start` datetime in ISO 8601 format `end` datetime in ISO 8601 format `avg_period` period for value averaging (in seconds) `verbose` include minimum, maximum, and standard deviation
DELETE	Remove parameter values from database, time range defined similar to GET method.

By making a GET request to the general parameter API endpoint a list of parameters logged from the drive is returned. In contrast to the other APIs, the parameter API doesn't include any methods for modifying the data. The only method affecting the content of the database is DELETE, which removes the parameter values from the database. Another difference comes in the data returned by the API, as the general and the parameter-specific API endpoints differ from each other, while in the APIs introduced earlier the data returned by the endpoints remained the same. An example response of a request made to the general parameter API endpoint is shown in Listing 6.9.

```
HTTP/1.0 200 OK
Content-Type: application/json; charset="utf-8"
Content-Length: 185

[
    {
        "id": "01.01",
        "name": "motor speed used",
        "unit": "rpm"
    },
    {
        "id": "01.10",
        "name": "motor torque",
        "unit": "%"
    }
]
```

Listing 6.9 The response of a GET request made to the general parameter API endpoint.

As shown in the Listing 6.9, the general API endpoint returns a list of parameters logged from the variable speed drive. The data contains the parameter ID, name, and unit, which are essential when building the UI. An example of request made to the API endpoint with the parameter ID defined is shown in Listing 6.10.

```
HTTP/1.0 200 OK
Content-Type: application/json; charset="utf-8"
Content-Length: 371603

{
    "number_of_values": 8639,
    "processing_time": 1841,
    "values": [
        ["2015-09-25T00:00:05Z", 1424.23],
        ["2015-09-25T00:00:15Z", 1423.27],
        ["2015-09-25T00:00:25Z", 1424.56],
...
        ["2015-09-25T23:59:35Z", 1432.46],
        ["2015-09-25T23:59:45Z", 1448.71],
        ["2015-09-25T23:59:55Z", 1436.67]
    ]
}
```

Listing 6.10 The response of a GET request made to the parameter-specific API endpoint.

As demonstrated by Listing 6.10, the response when calling the parameter specific endpoint is totally different compared to the endpoint listing the available methods. The parameter specific endpoints lists the variable values on the defined time range, using averaging interval dependent on the request. The averaging interval is used both to reduce the amount of data returned and to get the same timestamps on each of the fetched parameters. In addition the query parameters include the verbose flag, allowing fetching of the minimum, maximum, and standard deviation during the averaging interval.

In addition to the values, the number of values returned and the processing time in milliseconds are returned. The number of values is defined as

$$n_{\text{expected}} = \frac{\Delta t}{t_{\text{avg}}} - 1, \tag{6.1}$$

where n_{expected} denotes the expected number of results, Δt is seconds between start and end times, and t_{avg} is the averaging interval. This equation can be used to double check the actual number of results versus the expected. The processing time is mainly used for debugging and monitoring purposes.

Further optimisations can be achieved by compressing the JSON data returned. By the removal of unnecessary whitespaces used in Listing 6.11, the content length reduces from 371 604 bytes to 285 180 bytes, reducing the payload amount by 23 %. In comparison, the payload of a similar response done with the XML-RPC is 1 866 311 bytes with whitespaces removed, over 6 times the size of the payload of REST using JSON as data encoding method.

6.2.4 The calculation API

The calculation application programming interface is effectively an extension of the parameter API. However, as it relies on multiple parameters and the methods are more related to the operation of the variable speed drive system as a whole, the API endpoint is appended to the VSD API. The calculation API endpoints and the methods associated with it are presented in Table 6.5.

Table 6.5 The calculation API endpoints and associated methods.

/neta/<serial number>/drive/<serial number>/calculate/	
GET	Fetch list of calculations available for the drive system.
/neta/<sn>/drive/<sn>/calculate/<calculation type>/	
GET	Fetch calculation values, defaulting to last 24 hours and 10 second averaging period. The query parameters same as in the parameter API.
DELETE	Remove calculation values from database, effectively forcing a re-calculation. Time range defined similar to GET method.

As shown in Table 6.5, the calculation API is effectively the same as the parameter API and uses the same methods. The GET method initially tries to fetch past calculation results from the database. By comparing the number of results to the expected number by using (6.1), the API determines the need for a recalculation. If the number of results matches the expected, the results are returned immediately. If the number does not match, the whole result set is calculated again, returned to the browser, and saved to the database. The main difference between the parameter and calculation application programming interfaces is in the DELETE method, which is used to remove the cached calculation results. In comparison to the DELETE method in the parameter API, the calculation method removes only the cached values, but doesn't prevent recalculating the results. An example of a GET request to the general endpoint is shown in Listing 6.11.

```
HTTP/1.0 200 OK
Content-Type: application/json; charset="utf-8"
Content-Length: 442

[
    {
        "id": "power",
        "name": "Shaft power",
        "unit": "W"
    },
    {
        "id": "energy",
        "name": "Energy",
        "unit": "Wh"
    },
    {
        "id": "flow",
        "name": "Flow",
        "unit": "m³/s"
    },
    {
        "id": "pressure",
        "name": "Pressure",
        "unit": "Pa"
    },
    {
        "id": "sfp",
        "name": "Specific fan power",
        "unit": "kW/m³/s"
    }
]
```

Listing 6.11 The response of a GET request made to the general calculation API endpoint.

In addition to the same message structure as in the parameter application programming interface, Listing 6.11 shows the methods implemented to the prototype system. The shaft power is calculated using (4.1) and reported in watts. By further processing the power calculation, the energy consumption in watthours can be estimated. It should be noted that this calculation doesn't take into account the losses in the variable speed drivetrain and the motor,

thus representing only the energy consumption of the fan itself. The flow calculation is achieved by using the *QP* method. The flow is estimated using the shaft power, the motor torque, the affinity laws (2.2 - 2.4), and the formed *QP* curve (5.1). Furthermore by dividing the power used with the airflow, the specific fan power can be formed.

The pressure generated can be estimated using characteristic curves in a similar manner to the air flow estimation. However, in addition to the power and rotational speed, the air flow must also be known. In the simplest form, the fan pressure can be estimated using

$$p = \frac{P}{Q}, \tag{6.2}$$

where p is the pressure increase, P is the fan power consumption, and Q is the delivered airflow (CEATI International Inc., 2008). This however doesn't take any losses into account and is therefore only valid for an ideal system. To receive more accurate estimation results, an equation must be formed based on the characteristic curves, e.g. *QP* curve. Even in this case the uncertainties of the method remain quite high, if the produced airflow is estimated instead of a measurement.

6.3 Browser based user interface

As the application programming interface uses Hypertext Transfer Protocol for communication, the user interface (UI) can be built on virtually any platform and programming language. Moreover as all communication with the database goes through the API, the user interface is a completely separate entity and can therefore be implemented by multiple different clients. This allows separate clients to be developed to serve different applications and end-users, for example different mobile device platforms. This is vital, as the different mobile platforms all have their unique design principles, which affect the usability, available functionality, and the overall appearance of the application.

Another approach, one used during the making of this thesis, is to create a generic, browser-based user interface. With the current technologies available, it is possible to create a single user interface to serve both the mobile and desktop users. The UI can be automatically scaled depending on the used device, its screen resolution, and display size.

6.3.1 The MVVM architectural pattern

The Model View ViewModel (MVVM) architectural pattern focuses on separating the development of user interfaces from the development of the logic and behaviour of an application. This allows working on both the UI and the logic and backend of the application simultaneously, even within the same codebase. The development can be done simultaneously thanks to the separation between different parts of the code. Additionally the MVVM pattern encourages the reuse of the existing codebase, as the components can be modified to fit a specific use case. The MVVM pattern also allows unit testing of the components to be done separate from each other, reducing the amount of tests to be conducted per modification. Essence of the MVVM pattern is presented in Figure 6.1. (Osmani, 2012; Microsoft, 2012)

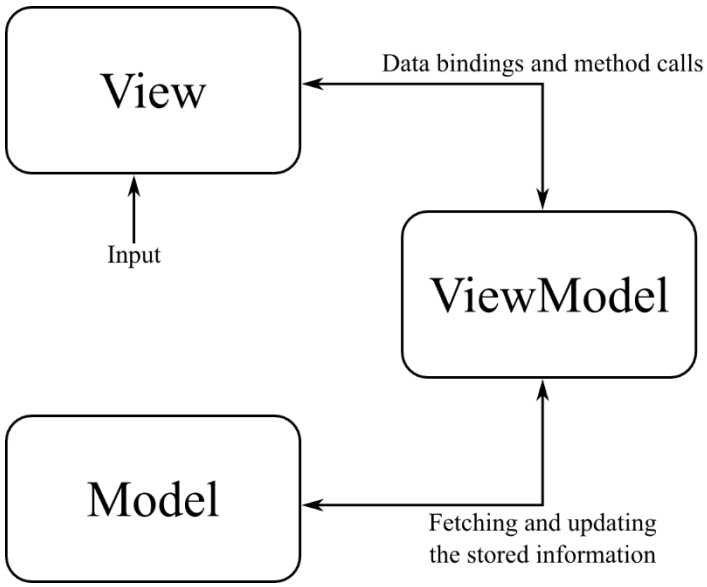


Figure 6.1 The basic principles of the MVVM design pattern.

As Figure 6.1 shows, the MVVM pattern consists of three main components, the model, viewmodel, and view. In the MVVM pattern the model is typically the data storage, used to hold the information. The model does not affect the way the data is represented in the user interface or affect the behaviour of the application directly. The only commonly accepted exception to these rules is the validation of data – the model can and in many cases should validate the data being inserted to the storage. (Osmani, 2012; Timms, 2014)

The view defines the structure, layout, and the overall appearance of the user interface. In the MVVM pattern the view is typically active and represents the current state of the view-model. The active View contains data-bindings, events and behaviours which are mapped to the properties of the viewmodel, allowing for an instant reaction to the changes in the view-model. Despite of this the view doesn't hold any information itself, the only responsibility of it is to keep itself in sync with the viewmodel. In addition to visualising the data to the user, the View can call the methods defined in the viewmodel. (Microsoft, 2012; Osmani, 2012; Timms, 2014)

Perhaps the most essential part of the MVVM pattern is the viewmodel, which provides the essential link between the view and the data stored in the model. The viewmodel is the logic layer, used to fetch data from the database, which is then transformed to a format the view can easily use. In addition it exposes methods to the View, allowing the user to interact with the model. For example, when a user clicks a button in the UI, a method is called from the viewmodel, which handles the transaction. (Microsoft, 2015; Osmani, 2012; Timms, 2014)

In the context of this thesis, the database used to store the data can be considered as the model of the MVVM architecture. In addition to storing the data and checking the data types of the input values, the database provides very little functionality. The only functionalities covered by the database used are fairly basic mathematical expressions and aggregations to calculate statistics from the data.

The application programming interface sits somewhere in between the model and the view-model. It does contain a lot of functionality and is not used to store data, so it does not fall under the model definition. On the other hand it does not communicate directly with the view, but does contain methods for manipulating the data, which is commonly implemented in the viewmodel. One example of this manipulation is calculating new results based on the existing data.

6.3.2 The viewmodel

The more traditional viewmodel is implemented in JavaScript and runs in the user's browser. The viewmodel is implemented using object-oriented programming, selection based on the fact that almost all of the components in the system can be thought as a real-life object. The

JavaScript uses Knockout, "*a JavaScript library that helps you to create rich, responsive display and editor user interfaces with a clean underlying data model*" (Knockout, 2015). Knockout is used to link the model and user interface together. The data fetched from the application programming interface is linked to observables, a special data type defined by Knockout. The observables are objects, which notify the user interface of any changes, allowing for an interactive user experience. The structure of the viewmodel is presented in Figure 6.2.

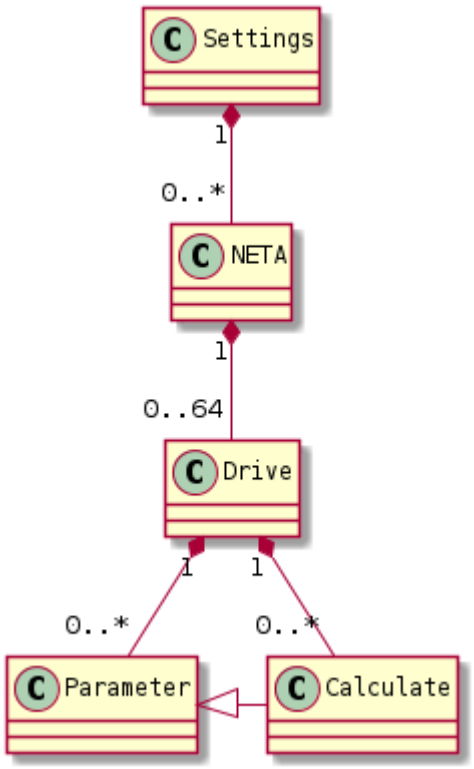


Figure 6.2 The viewmodel class structure.

As Figure 6.2 shows, the viewmodel closely represents the API structure. Each API endpoint is mapped to a JavaScript object, allowing easy access to the API. The Settings class provides user specific settings, such as the application programming interface address and authentication details. It is the only class not using the API to store its data, instead the data is stored to the users' browser. The Settings class provides the user interface with methods to authenticate to the API and fetch NETA details. As the system may consist of multiple NETAs, the Settings object may include anywhere from 0 to infinite number of NETAs.

The NETA class is linked to the NETA API endpoint and serves as a representation of the data provided by the API. The variables exposed by the API are mapped as observables, providing real time updates to any changes occurring in the model. These changes can be for example the user (or another user) updating the name of the NETA. In addition the NETA provides methods to delete itself from the system and fetching of the variable speed drives connected to it.

The drives fetched by the NETA objects are mapped to Drive objects, which shares many of the qualities used by the NETA class. In a similar manner, the variables exposed by the API are mapped to observables and are used to dynamically update the UI. In addition to removing the drive from the system, the object exposes methods for fetching the list of logged parameters and calculations from the API.

The parameters fetched via the parameter API are mapped to Parameter objects, which are used to access the parameter specific endpoints of the API. The methods exposed by the class provide access to the parameter values, which are then stored in the object itself. In addition to fetching the values, they can also be removed from the database. As the calculations performed with the logged data are exposed in a similar manner compared to the parameters, the Calculate class extends the Parameter class. The Calculate class then provides methods for accessing the calculation data.

6.3.3 The view

As a way of visualising the calculation results for an easier validation, a view was implemented as part of this thesis. The view is implemented in HyperText Markup Language (HTML), with Knockout providing dynamic updates based on the data received from the application programming interface. The variables mapped in the viewmodel are used to build the user interface based on a template system. In the template system, each class shown in Figure 6.2 has its own template providing the style of the presentation.

As the user enters the website, the initialisation function from the JavaScript is run. The initialisation first fetches NETAs the user has rights to observe. After this the drives connected to each NETA are fetched, as well as the calculations of each NETA. These are mapped to the corresponding partitions in the viewmodel, as presented in Figure 6.2. The viewmodel in turn is presented in the user interface by the template system, as shown in Figure 6.3.

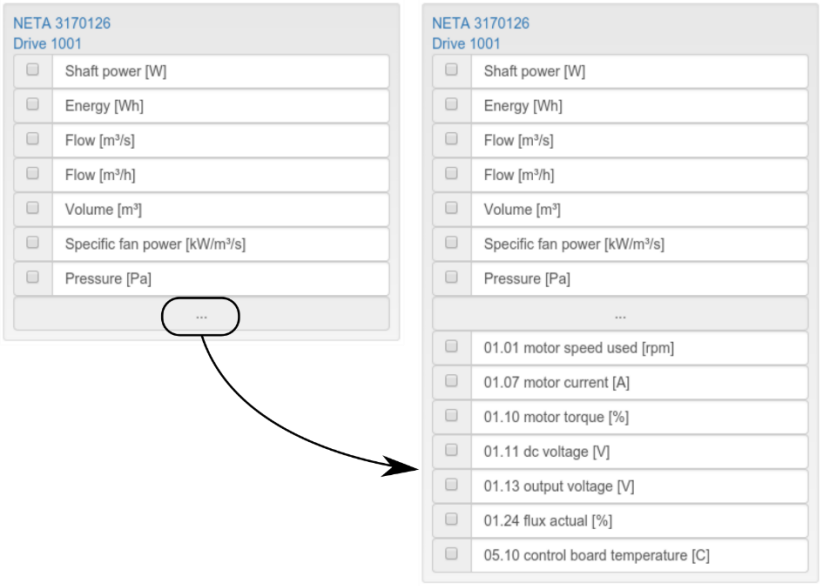


Figure 6.3 NETA with single drive in the UI, first with only the estimation results shown, and second with parameters.

As shown in Figure 6.3 the actual parameter values are not shown by default. However they can be accessed by expanding the user interface module by clicking on the expand icon. The user can choose any number of results from any number of drives to be plotted in the user interface. The plotted values are presented with the timestamps synchronised, allowing for

easy access to the values. Furthermore the plots can be zoomed and moved, and they stay synchronised to allow easier observation of the operation. An example of operation during a single day is shown in Figure 6.4.

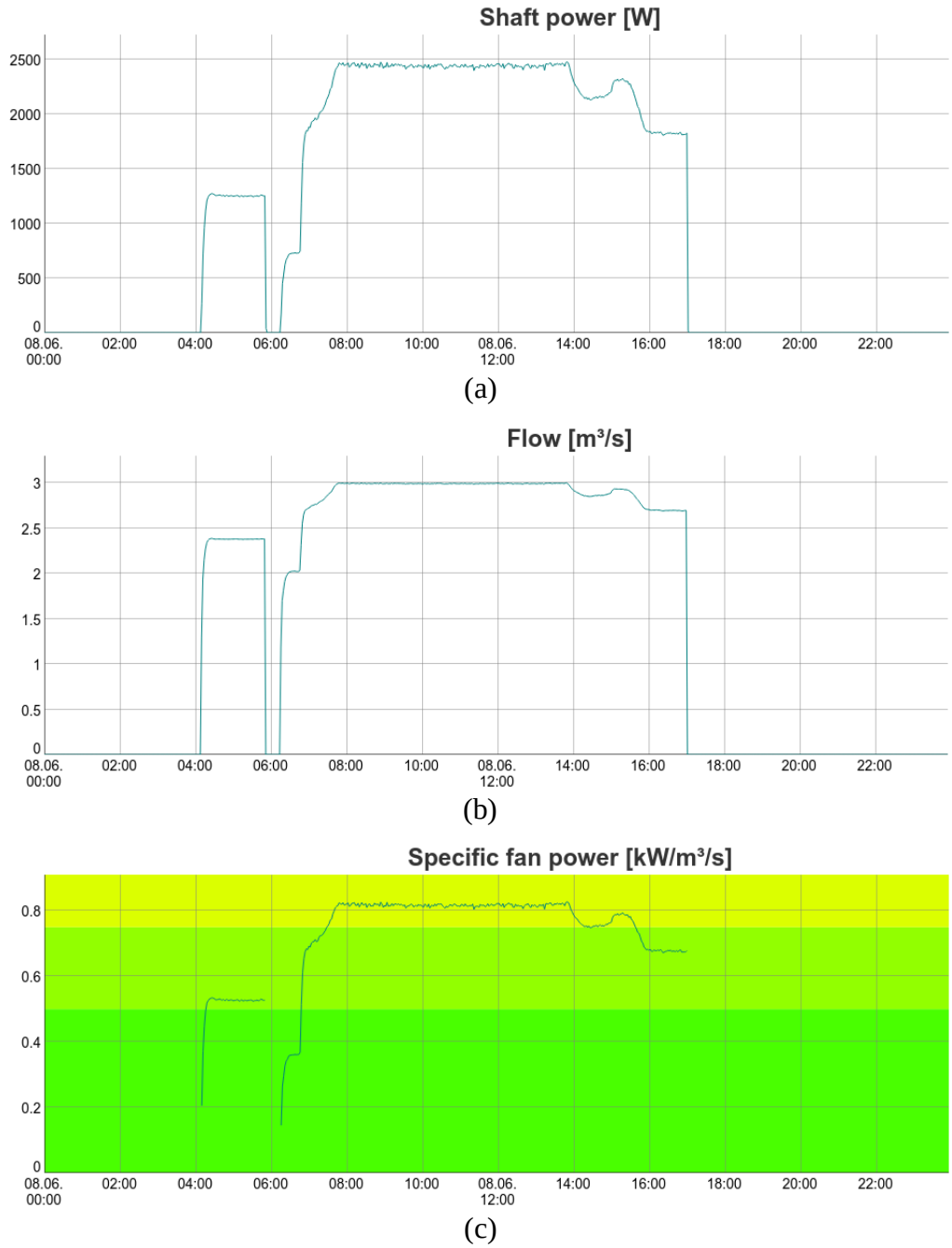


Figure 6.4 Fan operation during a single day. (a) The shaft power. (b) The produced airflow. (c) The specific fan power.

As Figure 6.4 shows, the fan is operated mainly during office hours. The fan is first started around 4 a.m. and run on a relatively low power till 6 a.m. Around 6.15 a.m. the fan starts

once again, reaching typical operating power around 8 a.m. It is then run on a constant power till 2 p.m., when the external control takes over. At 5 p.m. the fan system is stopped for the night. The shaft power is around 2.5 kW at its peak, as shown in Figure 6.4a. The airflow generated by the fan at the peak power is around 3 m^3/s, shown in Figure 6.4b. This leads to specific fan power of 0.83 $kW/m^3/s$, presented in Figure 6.4c. The specific fan power corresponds to the SFP category 3, with partial load operation being in the lower categories.

7. SUMMARY AND CONCLUSIONS

The aim of this thesis was to develop a cloud service to allow remote monitoring of variable speed drives used in fan systems. This involved a review of the current methods for operating point estimation based on mathematical models of the fan behaviour, fan system efficiency estimation, and monitoring of the variable speed drives using data loggers. The development of the cloud service involved an overview of the data transfer protocols and methods available, as well as the amount of overhead caused by each method. An application programming interface was developed using these findings, capable of both receiving data from the data logger and serving the data to the user interface. A basic user interface was developed to utilise the API. The user interface was designed to be responsive to the end device properties, such as screen size.

By constantly monitoring the variable speed drive, it is possible not only to detect the fault in the system but also observe the changes in behaviour leading to the fault. Even if the estimated values are not entirely accurate, the change in behaviour can be detected by observing the values on long term. This can provide valuable information when trying to discover the possible reason leading to the fault. Even if the reason for fault situation cannot be determined, the next occurrence maybe forecasted by analysing the circumstances resulting to the fault. Once the occurrence is forecasted, the required maintenance can be done pre-emptively.

In the demonstration case, the constant monitoring allowed observation of phenomena which could lead to better optimisation of the whole fan system. As Figure 6.4 suggests, optimisation related to the fan system may be possible even in the demonstration system. As shown in Figure 6.4, the fan stops around 6 a.m. for a brief period. The reason for this stop is unknown and should be looked into. It could be beneficial to start the fan system a bit later than 4 a.m. and keep it running through the day, reducing the stress caused by an additional start of the system. It should however be noted that the fan system used for the demonstration is located at a common area of the building, which includes for example a kitchen. The behaviour may be caused by the specific needs introduced by the kitchen area and thus a deeper knowledge of the behaviour is required.

As shown in Figure 6.4c, the specific fan power used by the fan is relatively low. The colours in the plot indicate the categories presented in chapter 0. The fan system used in the demonstration operates in the range of categories SFP 1 and SFP 3. However it should be noted that to get the full specific fan power category, the information from the intake fan should also be looked into. When comparing to the requirements set in the United Kingdom (see Table 2.4), the fan system falls under the category "Local ventilation only units remote the area, such as ceiling void or roof mounted units, serving one room or area". The maximum allowed SFP for both existing and new buildings in this category is 1.5. This effectively means that if the intake fan operates in a similar fashion the exhaust fan, the whole system would go slightly above the limit when operating at full power.

The current state of data loggers is enough for monitoring fan system behaviour, but further development would allow for real-time monitoring. In this scenario the data logger would send data to the application programming interface either on a set interval or when significant changes in the values are detected. Furthermore the data logger used in the demo case is not capable of obtaining the parameter values from the variable speed drive fast enough to implement the more complicated estimation methods, such as the detection of the impeller mass increase. It should however be noted that even the current hardware generation may be used to implement the method in some specific cases, such as with very large fans which start slowly. The cloud service created in this thesis only covers one type of data logger, but could easily be expanded to house different data loggers via similar API. This would require abstraction of the data logger type, so the NETA API would change to a general data logger API.

The issues with logging intervals et cetera raise the question of where each method should be implemented and how. Some methods require such high speed data collection, that it might be more beneficial to utilise either the data logger itself to calculate the values, the programmable logic controller (PLC), or even an integrated data logger found in some variable speed drives. A study on implementing the detection of impeller mass increase method utilising the integrated PLC found in ACS880 has already began, with the aim to expose the estimation result to the NETA-21 as a single variable. With this approach, the NETA-21 can be used to send the estimation result to the cloud in a similar fashion as the other parameters. Furthermore the future generations of NETA-21 firmware will include

reading of the internal data logger found in some variable speed drives (O. Alkkiomäki, 2015, pers. comm., 30 September). This would allow the internal data logger to be triggered by an event, such as change in the rotational speed reference. The internal data logger could then be used to read the values on a higher sampling rate and the NETA-21 would act as a link between the cloud and the variable speed drive.

Security is always a great concern with any monitoring system, especially with systems in industrial environments. With the methods described in this thesis, the only protocol used for transferring the data is the HTTPS. It is found to be secure enough for banking applications and thus should be enough for industrial applications as well. The data logger itself is not used to alter the behaviour of the variable speed drive in any way. It also does not require any traffic to be sent towards the data logger, therefore not requiring any incoming ports to be opened from the firewall protecting the industrial network. Furthermore the monitoring system described in this thesis can be deployed on customer premises as well, if security is a major concern.

The major advantage of cloud services comes from the capability of combining the data from various sources. For example the data gathered from a fan system could be combined with weather observations or a sensor monitoring carbon dioxide content in the air. With the current trend in Internet of Things, sensors are directly connected to a cloud service of their own. These cloud services can then provide access to the data via application programming interfaces, allowing either the client or a third party service provider to build services on top of the data.

REFERENCES

ABB Oy, 2013. *F-series fieldbus adapter modules.* s.l.: ABB Oy.

ABB Oy, 2014. *User's manual - NETA-21 remote monitoring tool.* s.l.: ABB Oy.

ABB Oy, 2015a. *ABB Suomessa.* [Online] Available at: http://new.abb.com/fi/abb-lyhyesti/suomessa [Accessed 11 September 2015].

ABB Oy, 2015b. *ACS580 standard control program - Firmware manual,* Helsinki: ABB Oy.

ADFweb.com Srl, 2013. *Modbus Master/Slave - Datalogger.* Mareno: ADFweb.com Srl.

Ahonen, T., Tamminen, J., Ahola, J. & Kestilä, J., 2012. Frequency-Converter-Based Hybrid Estimation Method for the Centrifugal Pump Operational State. *IEEE Transactions on Industrial Electronics,* December, 59(12), pp. 4803-4809.

Ahonen, T., Tamminen, J., Ahola, J. & Niemelä, M., 2011. *Accuracy study of frequency converter estimates used in the sensorless.* Birmingham, European Conference on Power Electronics and Applications.

Amazon Web Services, Inc, 2015. *Amazon EC2 Pricing.* [Online] Available at: https://aws.amazon.com/ec2/pricing/ [Accessed 23 September 2015].

Armbrust, M. et al., 2009. *Above the Clouds: A Berkley View of Cloud Computing,* Berkeley: Electrical Engineering and Computer Sciences, University of California.

Bureau of Energy Efficiency, 2005. *Book-3: Energy efficiency in electrical utilities.* 2nd ed. s.l.: Bureau of Energy Efficiency.

Carrier Corporation, 2005. *Variable Frequency Drive - Operation and application of VFD technology,* Syracuse: Carrier Corporation.

CEATI International Inc., 2008. *FANS & BLOWERS - Energy Efficiency Reference Guide.* [Online] Available at: http://www.ceati.com/freepublications/7027_guide_web.pdf [Accessed 7 July 2015].

Cisco Systems, Inc., 2015. *Internet of Things (IoT) Opprotunities.* [Online] Available at: http://www.cisco.com/web/solutions/trends/iot/portfolio.html [Accessed 10 September 2015].

Croll, A., 2015. *The Internet of Things has four big data problems.* [Online] Available at: http://radar.oreilly.com/2015/01/the-internet-of-things-has-four-big-data-problems.html [Accessed 26 May 2015].

de Almeida, A. T., Fonseca, P., Falkner, H. & Bertoldi, P., 2003. Market transformation of energy-efficient motor technologies in the EU. *Energy Policy,* Volume 31, pp. 563-575.

de Almeida, A. T. et al., 2000. *Improving the Penetration of Energy-Efficient Motors and Drives,* Coimbra: University of Coimbra.

Department of Communities and Local Government, 2006. *Non-Domestic Heating, Cooling and Ventilation Compliance Guide.* 1st ed. London: NBS for the Department of Communities and Local Government.

Dereyne, S. et al., 2015. *An efficiency measurement campaign on belt drives.* Helsinki, eeemods'15.

EnEV, 2006. *Verordnung über energiesparenden Wärmeschutz und energiesparende Anlagentechnik bei Gebäuden,* Berlin: EnEV.

Englander, R., 2002. *Java and SOAP.* 1st ed. Sebastopol: O'Reilly & Associates, Inc..

European Standard, 2007. *EN 13779 - Ventilation for non-residential buildings - Performance requirements for ventilation and room-conditioning systems,* Brussels: European Committee For Standardization.

Ferreira, F., 2008. *Strategies to improve the performance of three-phase induction motor driven systems,* Coimbra: University of Coimbra.

Gartner, Inc, 2014. *Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015.* [Online] Available at: http://www.gartner.com/newsroom/id/2905717 [Accessed 26 May 2015].

General Electric, 2013. *Industrial Internet - A European Perspective,* s.l.: General Electric.

Google, 2015. *App Engine Pricing.* [Online] Available at: https://cloud.google.com/appengine/pricing [Accessed 23 September 2015].

Holtz, J., 2000. *Sensorless Control of Induction Motors - Performance and Limitations.* Cholula, Proceedings of the 2000 IEEE International Symposium on Industrial Electronics.

InfluxDB, 2015. *InfluxDB Docs v0.9.* [Online] Available at: https://influxdb.com/docs/v0.9/introduction/overview.html [Accessed 30 September 30].

International Organization for Standardization, 2007. *Industrial fans - Tolerances, methods of conversion and technical data presentation.* 2nd ed. Geneva: International Organization for Standardization.

Internet Engineering Task Force, 2014. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.* [Online] Available at: https://tools.ietf.org/html/rfc7231 [Accessed 18 September 2015].

IV Produkt AB, n.d. *Fläktar, K-faktorer och fläktkurvor,* Växjö: IV Produkt AB.

Kernan, D. J., Sabini, E. P., Ganzon, N. W. & Stavale, A. E., 2011. *Method for determining pump flow without the use of traditional sensors.* United States of America, Patent No. 7,945,411 B2.

Knockout, 2015. *Knockout documentation.* [Online] Available at: http://knockoutjs.com/documentation/introduction.html [Accessed 5 October 2015].

Knox, D. L., Leuthen, J. M. & Lockyear, K. W., 1985. *Motor variable frequency drive.* United States of America, Patent No. 4,491,778.

Kosonen, E., 2010. *Toimistorakennuksen energiankulutustarkastelu - Case: ABB Kiinteistöt, Helsinki Pitäjänmäki, Tellustalo,* Lappeenranta: Lappeenranta University of Technology.

Laurent, S., Johnston, J. & Dumbill, E., 2001. *Programming Web Services with XML-RPC.* 1st ed. Sebastopol: O'Reilly & Associates, Inc..

Leinwand, A., 2009. *The Hidden Cost of the Cloud: Bandwidth Charges.* [Online] Available at: https://gigaom.com/2009/07/17/the-hidden-cost-of-the-cloud-bandwidth-charges/ [Accessed 23 September 2015].

M2MLogger, 2015. *Industrial Data-Loggers, Cloud Gateways & Machine-to-Machine Cloud.* Shahdara: M2MLogger.

Microsoft, 2012. *The MVVM Pattern.* [Online] Available at: https://msdn.microsoft.com/en-us/library/hh848246.aspx [Accessed 30 September 2015].

Microsoft, 2015. *Azure pricing.* [Online] Available at: http://azure.microsoft.com/en-us/pricing/ [Accessed 23 September 2015].

Muhonen, T., 2015. *Standardization of Industrial Internet and IoT - Perspective on condition-based maintenance,* Oulu: University of Oulu.

Osmani, A., 2012. *Learning JavaScript Design Patterns.* 1st ed. Sebastopol: O'Reilly Media, Inc..

Radgen, P., Oberschmidt, J. & Cory, W. T. W., 2008. *EuP Lot 11: Fans for ventilation in non residential buildings - Final Report,* Karlsruhe: Fraunhofer Institute.

Richardson, L. & Ruby, S., 2007. *RESTful Web Services.* 1st ed. Sebastopol: O'Reilly Media, Inc..

Soley, R. M., 2014. The Industrial Internet: The Opportunities ... and the Roadblocks. *Cutter IT Journal,* November, 27(11), pp. 6-10.

Tamminen, J., 2013. *Variable speed drive in fan system monitoring.* Lappeenranta: Yliopistopaino.

Tamminen, J., Ahonen, T., Ahola, J. & Kestilä, J., 2011. *Sensorless Flow Rate Estimation in Frequency-Converter-Driven Fans.* Birmingham, The 14th European Conference on Power Electronics and Applications.

Tamminen, J. et al., 2013. Detection of Mass Increase in a Fan Impeller With a Frequency Converter. *IEEE Transactions on Industrial Electronics,* 60(9), pp. 3968-3975.

Tamminen, J. et al., 2015a. *Case Study on the Variable Speed Drive-based Fan Impeller Contamination Build-up Detection.* Helsinki, eeemods'15.

Tamminen, J., Ahonen, T., Ahola, J. & Tiainen, T., 2015b. *Variable Speed Drive-Based Fan Impeller Contamination Build-Up Detection: Industrial Case Study*. Geneva, European Conference on Power Electronics and Applications.

Tamminen, J. et al., 2014. Comparison of model-based flow rate estimation methods in frequency-converter-driven pumps and fans. *Energy Efficiency,* 7(3), pp. 493-505.

Taranovich, S., 2012. *Teardown: The nuances of variable-frequency drives.* [Online] Available at: http://www.edn.com/design/analog/4371295/Teardown-The-nuances-of-variable-frequency-drives [Accessed 26 October 2015].

Tilastokeskus, 2015. *Energian hankinta ja kulutus, 2014, 4. neljännes,* Helsinki: Tilastokeskus.

Timms, S., 2014. *Mastering JavaScript Design Patterns.* 1st ed. Birmingham: Packt Publishing Ltd..

Tyrväinen, P., Warsta, J. & Seppänen, V., 2008. Evolution of Secondary Software Business: Understanding Industry Dynamics. In: G. León, et al. eds. *Open IT-Based Innovation: Moving Towards Cooperative IT Transfer and Knowledge Diffusion.* Madrid: International Federation for Information Processing, pp. 381-401.

U.S. Department of Energy, 2003. *Improving Fan System Performance.* Washington D.C.: U.S. Department of Energy.

Vacon, 2014. *Vacon NXS Brochure.* s.l.: Vacon.

Waide, P. & Brunner, C. U., 2011. *Energy-Efficiency Policy Opportunities for Electric Motor-Driven Systems*, Paris: International Energy Agency.

Vector Informatik GmbH, 2015. *Product Information - GL Logger Families.* s.l.: Vector Informatik GmbH.

World Wide Web Consortium, 2007. *SOAP Version 1.2.* [Online] Available at: http://www.w3.org/TR/soap12/ [Accessed 16 September 2015].

Yaskawa America, Inc., 2015. *Z1000 HVAC Drive.* s.l.: Yaskawa America, Inc..