

Industry 4.0-ready Building Automation System

System design and commissioning for HAMK Sheet Metal Center building



Bachelor's thesis

Degree Programme in Automation Engineering

Autumn 2017

Khoa Dang

Automation Engineering
Valkeakoski

Author	Khoa Dang	Year 2017
Subject	Industry 4.0-ready Building Automation System	
Supervisor(s)	Timo Väisänen	

ABSTRACT

The thesis was commissioned by the Degree Programme of Automation Engineering and the Sheet Metal Center research unit of Häme University of Applied Sciences for the “Healthy Digital House” project. The empirical targets for this thesis project included the commissioning of a programmable logic controller based system for data acquisition from the building automation system of the Sheet Metal Center. In addition, the system is designed to be capable of controlling of the lighting system and communicating with the wireless indoor environment sensor network. Finally, the targets also included deployment of a data server system to record and supply collected data for visualization and research purposes.

To realize the solution, theoretical concepts concerning sensors, actuators, programmable logic controllers and industrial data communications were reviewed in addition to concepts concerning software engineering, including database management systems and web application programming interfaces. The discussed theoretical concepts were focused on a smart building automation system for zero-energy buildings and capable of integration into the Internet of Things and Industry 4.0

The outcome of this thesis project consisted of a Beckhoff PLC system developed for data communication with an existing building automation system, as well as a DALI-based lighting control system and EnOcean-based wireless indoor sensor network ready for implementation. In addition, a data server unit was deployed, which utilized the InfluxDB time series database management system and a REST API for data and web services. The data acquired from this project was used to support energy optimization researches and 3D web visualization model of Sheet Metal Center. The targets of this thesis project were achieved, with further development planned for implementation in the following year.

Keywords Building Automation System, IoT, DALI, EnOcean, Backend, API

Pages 33 pages including appendices 52 pages

ACKNOWLEDGEMENT

I would like to express my gratitude to my family for their relentless support in my pursuit of education, research and happiness. Mom, dad, brother, girlfriend, I am really lucky to have you in my life, for that I am forever grateful. Also, to the whole staffs of HAMK for accompanying and guiding me through my bachelor degree. To my thesis supervisor Timo Väisänen, my greatest gratitude for your trust and support in my work. To Niina Valtaranta, thank you for putting up with my writings and my time schedule. To Susan, thanks for all the parties, you really cared for us like family. To Katariina, Annina, Päivi, Leena, Antti, Timo, Juha, Jan, Raine and Juhani for your efforts to my education, my career and for all the good times. To Atte and Minh, best teammates ever.

I would like to also express my gratitude for HAMK Sheet Metal Center and Ruukki Construction Oy for their support during my thesis work. Especially, for Kimmo Hilden and Jarmo Havula from HAMK, Erkki Honkakoski and Jyrki Kesti from Ruukki. Without you, there wouldn't be this thesis, so please accept my gratitude for supporting and encouragements to my ideas.

Finally, thank you, to my classmates and friends who made my bachelor journey at HAMK a wonderful ride.

Khoa Dang
Valkeakoski, 21.12.2017

CONTENTS

1	INTRODUCTION	1
1.1	Outline of thesis	1
1.2	Project background	1
1.3	Empirical targets	2
2	THEORETICAL BASIS.....	2
2.1	Industry 4.0 and Internet of Things (IoT)	2
2.2	Sensors and actuators in automation systems	4
2.2.1	Sensors.....	4
2.2.2	Final Control Elements - Actuators.....	5
2.3	Programmable logic controller (PLC)	7
2.4	Fieldbus and data communication protocols.....	8
2.4.1	Open Systems Interconnection (OSI) model	8
2.4.2	Digital Addressable Lighting Interface (DALI).....	10
2.4.3	EnOcean.....	12
2.4.4	Modbus TCP and RTU	13
2.4.5	TCP/IP	15
2.5	Data access and presentation in software engineering.....	17
2.5.1	Database management system	18
2.5.2	Application Programming Interface (API)	20
2.6	Building Automation System.....	23
2.6.1	Heating, Ventilation and Air Conditioning (HVAC) control system	23
2.6.2	Lighting control system and occupancy management.....	24
3	EMPIRICAL PROCESS.....	25
3.1	Preliminary design data collection and component selection.....	25
3.1.1	Preliminary design data collection	25
3.1.2	Component selection	26
3.2	Network infrastructure	26
3.3	PLC program	28
3.4	Backend deployment	28
4	RESULTS	30
5	LIMITATIONS AND EXPANSION POSSIBILITIES.....	31
6	CONCLUSION	31
	REFERENCES.....	32

LIST OF FIGURES

Figure 1. IoT platform components and communication channels (InfluxData, 2017) ...	3
Figure 2. Control Valve Flow Characteristics (Inc, 2017).....	6
Figure 3. OSI seven-layer model (Cisco, 2017)	9
Figure 4. DALI light control installation (Beckhoff Automation GmbH, n.d.).....	10
Figure 5. Analog 1-10V light control installation (Beckhoff Automation GmbH, n.d.)...	11
Figure 6. DALI query and response telegrams (Beckhoff Automation GmbH, 2010)	12
Figure 7. EnOcean communication frames (Dang, Lupea, Multaniemi, & Tran, 2016, p. 4)	13
Figure 8. Content of IPv4 header (Mackay, Wright, Park, & Reynders, 2007, p. 260) ...	16
Figure 9. Write throughput comparison – InfluxDB vs MongoDB (Persen & Winslow, 2016, p. 6).....	19
Figure 10. Disk storage comparison – InfluxDB vs MongoDB (Persen & Winslow, 2016, p. 6).....	19
Figure 11. Query throughput comparison – InfluxDB vs MongoDB (Persen & Winslow, 2016, p. 7).....	20
Figure 12. API URI demonstration	21
Figure 13. Air handling unit with air recycling PI-diagram	24
Figure 14. Connection scheme for empirical process	27
Figure 15. Node-RED flow for Caverion data sent from PLC in SMC.....	29
Figure 16. Node-RED flow for displaying weather data at SMC building.....	29
Figure 17. Node-RED dashboard display for the weather data at SMC building	30

LIST OF TABLES

Table 1. OSI model functionalities summary (Dye, McDonald, & Ruffi, 2008, p. 53; Microsoft, 2017)	9
Table 2. Modbus data point address ranges and functions (Mackay, Wright, Park, & Reynders, 2007, p. 98)	14
Table 3. Format of Modbus message frame (Mackay, Wright, Park, & Reynders, 2007, p. 97)	14
Table 4. Comparison of Modbus RTU/ASCII and TCP	15
Table 5. REST API actions and descriptions	22
Table 6. REST API request and response example	22

1 INTRODUCTION

1.1 Outline of thesis

This thesis is divided into six chapters as follows.

Chapter one discusses the background of the problem, the author's motivation for solving the problem and the targets of his empirical work. This included the involved parties in this thesis, overview on the target building and the current situation before this thesis was commissioned.

Chapter two reviews and discusses the theoretical aspects for the solution, which includes the structure of an automation system, data access and communication, the structure of the software system and building automation systems.

Chapter three describes in detail the solution which was commissioned at the target building, the structure of the software system solution and the author's reasoning for designing these solutions.

Chapter four describes the results of the commissioning process and preliminary data collection.

Chapter five discusses current limitations, as well as near-future goals for development continuation of this thesis in the following year.

1.2 Project background

The Sheet Metal Center building (SMC), operated by Ruukki and Sheet Metal Center Research Unit of Häme University of Applied Sciences (from herein referred to as HAMK), was one of the pilot case studies for the "Healthy Digital House" (Terveellinen Digitalo) project, which involved using building data to improve the occupancy experience of the inhabitants, as well as optimizing the energy consumption. The heating system of SMC was designed and implemented to be highly efficient and to provide the best possible indoor experience. For example, the ceiling radiant elements and the floor heating system were used for providing the comfortable indoor environment, whereas heat recovery from operating industrial machinery and geothermal pumps were used for energy storage and recycling. The ventilation of the building also affected the heating process, due to the massive doors frequent opening and closing. Furthermore, all the indoor space of the building was equipped with windows, for the purpose of taking advantage of natural lighting to decrease the consumption of electricity for lighting, but simultaneously also caused increased heat losses. Ruukki and Sheet Metal Center came to

the conclusion, that collecting data from the heating and air conditioning system, as well as increasing the quantity of indoor air quality measurements would assist in the building modeling process, thereby supporting the energy optimization process. The data collection process would involve recording data from the building automation system, as well as installing more sensors to perform measurements which were not available with the current system, e.g. temperature, air pressure, humidity and carbon dioxide (CO₂) levels in the offices and common areas.

The building's heating and ventilation system was commissioned and is being operated by Caverion Oy, which offers connectivity to other data infrastructures through the OPC DA standard. Originally, a data connection program was written by the Tampere Unit for Computer-Human Interaction from Tampere University, and further revised by automation students Khoa Dang and Minh Tran from HAMK. Due to the network structure within HAMK, and the security as well as reliability requirements of the control room computer, it was decided that the data connection program should not be used, and the data collection task would be conducted using a Beckhoff Industrial PC (IPC) system, which also allows additional sensors and other functionalities, namely light control, to be easily implemented at a later point. The Beckhoff IPC system also allows developing additional functions without affecting the operation of the Caverion system, therefore not compromising the current operation process of the building.

1.3 Empirical targets

The targets of this thesis work included:

- Commissioning a Beckhoff control system for the following functions:
 - Light control for Sheet Metal Center common area with the Beckhoff IPC, for performing cost analysis of a smart lighting control system
 - Extensive indoor quality measurement systems
 - Establish the connectivity to the existing building automation system of Caverion and collect all the building data from Caverion SCADA
- Designing and deploying an application programming interface (API) and backend data storage system to store and provide the data for different parties and purposes

2 THEORETICAL BASIS

2.1 Industry 4.0 and Internet of Things (IoT)

The Internet of Things (IoT) is a term, referring to the trend of enabling connectivity for all devices, to allow more information provided for human

and optimization of device operation, whereas Industry 4.0 specifically applies the IoT trend to industrial scenarios and use cases. Such advances are possible thanks to the development of the data communication in general, specifically the Internet and wireless technologies, as well as increase in computational and storage capability of computers. Figure 1 describes components of an IoT platform.

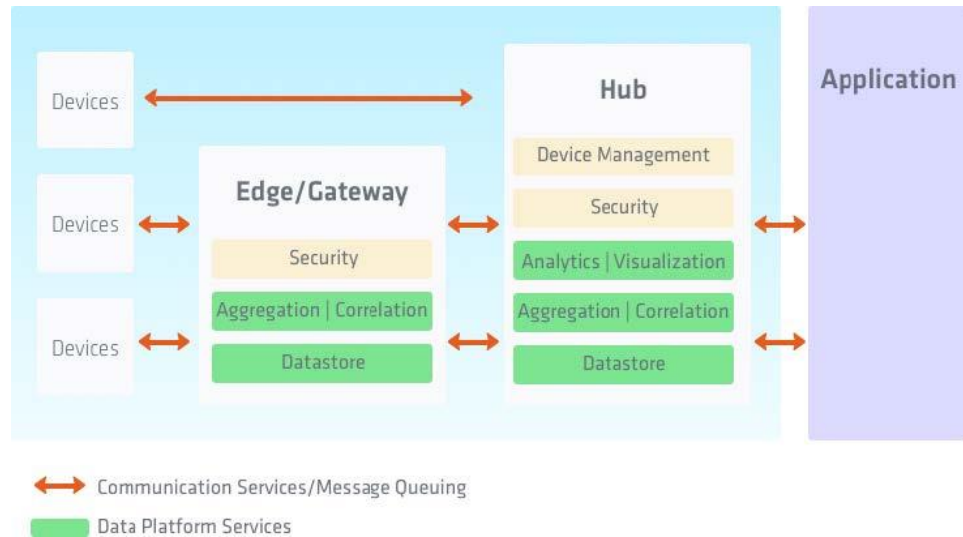


Figure 1. IoT platform components and communication channels (InfluxData, 2017)

In figure 1, data flows from end devices to a hub, optionally through a gateway for standardization purposes, then stored and served for applications. Examples of end devices are sensors, actuators, programmable logic controllers and smart appliances like phones, tablets computer or smart watches. In general, any electronic hardware or software that generate data can be classified as “devices” in the IoT architecture. Edge devices, or gateways, are devices equipped with ability to communicate with and control end devices through different data communication protocols, as well as condition and format the data received from end devices for sending to the hub. The hub, usually a server or a cluster of server computers, stores all the data as well as performs analytics on said data, ultimately finalizing and serve the data. The client for the final data can be end-users, other applications or services, and back to the devices themselves. In addition to the data flow, figure 1 also depicts the power picture of an IoT architecture. End devices and gateways often work with a local small power supply, such as batteries, renewable energy sources or low voltage power supplies, capable of only small data storage and simple calculations. On the other hand, the minimum power supply required for hubs are wall sockets and own power plant, in case of data centers, where much larger demand for computational power must be met.

Data collection has always been necessary in the industrial world, where extensive analytics of end devices would lead to economic benefits such

as precise production planning, downtime reduction and preventive maintenance. Data from end devices and production lines would be integrated with the Enterprise Resource Planning (ERP) and Manufacturing Execution System (MES) and translated into Key Performance Indicators (KPIs) which would aid the process of optimizing the business and production operation. With IoT, data collection expands itself to the consumer usage, i.e. collection of data from all kinds of devices ranging from household appliances to building management systems to increase the performances of said devices and improve the living conditions of human beings. Furthermore, the collection of said data goes hand-in-hand with the deployment of Application Programming Interfaces, allowing software developers to turn data into value-added services and further increases the benefits of IoT.

2.2 Sensors and actuators in automation systems

This chapter discusses the components of an automation system and their functionalities. In general, a classic automation system consists of sensors and actuators interfacing with a Programmable Logic Controller (PLC) via input/output cards. The PLC is responsible for automation and regulation processes, by controlling actuators based in the data received by different sensors. Vital information from these processes is then passed to a control room, where a supervisory control and data acquisition (SCADA) system is located. The SCADA system, in general, allows an operator with process expertise to monitor and regulate the processes, as well as performs the task of data storage and connectivity for further analytics of processes. In the following sections, each component of an automation system is further described.

2.2.1 Sensors

This chapter discusses the definition of a sensor, the typical challenges associated with metrology and smart measurement systems. Metrology, as defined by the International Bureau of Weights and Measures, “is the science of measurement, embracing both experimental and theoretical determinations at any level of uncertainty in any field of science and technology” (Mesures, n.d.). In the context of this thesis, it can be understood as applied metrology or measurement technology applied in the field of automation engineering. Generally, in automation systems, measurements are achieved by using sensors to measure and convert physical phenomena signals into electrical signals, which can be read and reacted to appropriately by the control system.

A sensor consists of a sensing element and signal conditioning components. According to McMillan (2010), the sensing element is responsible for the conversion of a process variable into a quantifiable output, which can be passed on to another sensing element, a transmitter

or a controller input. The output signal of a sensing element is called a measurand (McMillan, 2010, p. 10). In this case, the sensing element is connected to the signal conditioning components, which is responsible for converting the measurand into standardized signals or digital values, such as 4 – 20 mA or 0 – 10 V. In the case of digital values output, the definition of such values must be documented in the sensors datasheet. The values can be read using a microcontroller or a control system through communication protocols specified by the manufacturer.

Typically, the following signals are measured using a sensor, with their respective nominative units denoted in brackets:

- Temperature (°C or K)
- Relative humidity (%)
- Pressure (Pa, bar, hPa or mmHg)
- Force (N)
- Level (m)
- Flow speed and mass flow (m/s, m³/s or l/s)
- Concentration (kg/m³, ppm, ppb)

The quality of any given measurement is often reduced due to noise, interference and sensor self-fault. Hence, modern sensors often include features such as multiple measurements, real-time compensation, remote configuration and extensive embedded digital signals and information. Also, advances in wireless communication and sensing element technology have further evolved the sensor industry, allowing more measurements to be made with less commissioning effort and higher measurement quality.

In a building context, usually the following measurements are conducted: temperature, pressure, flow, volatile organic components, carbon dioxide (CO₂) and carbon monoxide (CO) concentrations. Said measurements are essential to the functioning of a building automation system, directly affecting the indoor environment quality and habitant experience. For example, temperature and pressure of air and water flow are measured in the heating and air ventilation control (HVAC) system. Gas component measurements can be used to adjust the power of the air ventilation system to achieve on-demand ventilation, enabling the optimal habitant experience while saving energy when there are no habitants in the building.

2.2.2 Final Control Elements - Actuators

This chapter discusses the definition and functionalities of the final control elements, also known as “effectors” or “actuators”. An actuator is responsible for converting the controller output from the control system into physical actions, to control the process. Usually, the action is achieved by manipulating one or multiple flows of material in the process, namely water or air flow. The flow manipulation is typically achieved using control

valves, motors and pumps (Hughes, 2007, p. 275). Additionally, motors and pumps can be connected to variable speed drives (VSD) to achieve precise speed and/or torque control.

Control valves can be separated into five types based on their mechanical construction: globe, gate, diaphragm, butterfly and ball valves; each with their own application areas depending on control function, flow material and pipe size. For example, gate valves are typically used for manual on-off use cases, diaphragm valves are used for liquid flow manipulation and butterfly valves are used for large pipe size (Hughes, 2007, p. 276). Another way to classify valves is based on the characteristic curve which describes the relationship between a valve's opening percentage to the change in flow through the valve (Hughes, 2007, p. 277). The valve's opening percentage can be changed using solenoids, hydraulic or pneumatic actuators. Figure 2 describes different types of characteristic curves.

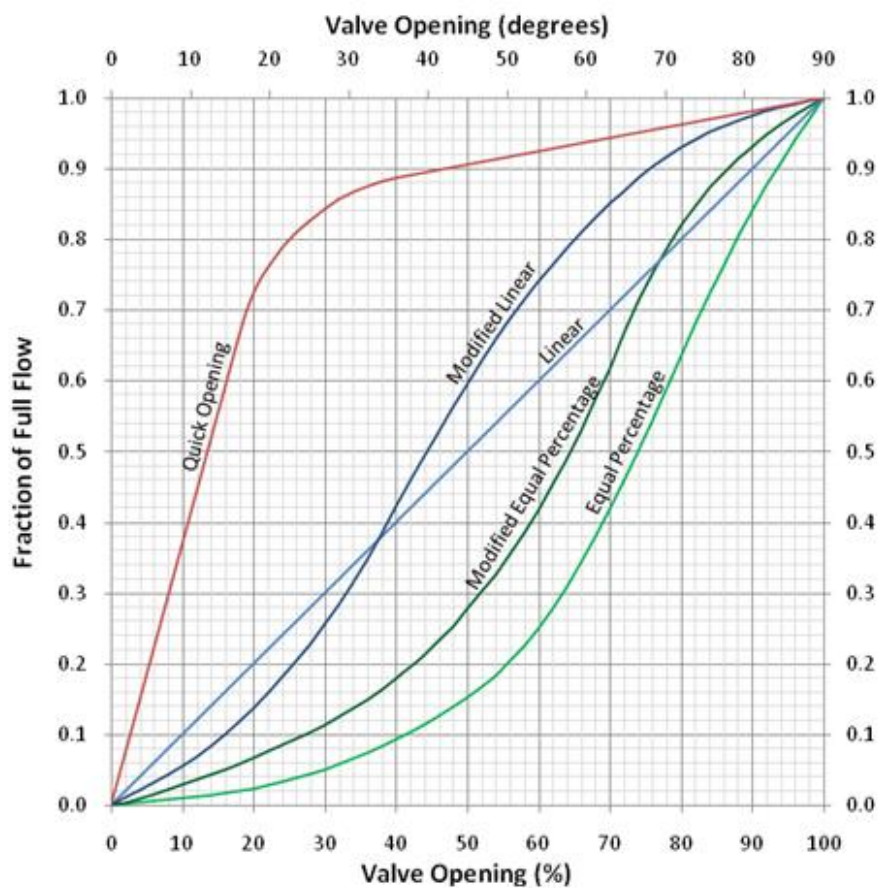


Figure 2. Control Valve Flow Characteristics (Inc, 2017)

Motors and pumps are necessary in scenarios where speed, pressure or position of the flow of material needs to be controlled. Quite often they are coupled with a variable speed drive (VSD) for speed control and local process control applications, and the whole unit is then referred to as a "drive". Communication between the drive and the control system is achieved via the use of analog or digital I/O, and mostly nowadays through fieldbuses.

Typically, selecting a drive requires knowledge on process mechanical requirements: mechanical load, pressure, angular velocity, etc. After the motor has been selected using the previously mentioned criteria, the speed drive is then chosen based on the motor's power requirements.

Finally, there are other types of actuator. For example, heating resistors and lights can also be considered as actuators.

2.3 Programmable logic controller (PLC)

This chapter discusses the definition, functionalities and programming of a programmable logic controller. A programmable logic controller, as defined within IEC 61131-1 (2003) standard, is "digitally operating electronic system, designed for use in an industrial environment, which uses a programmable memory for the internal storage of user-oriented instructions for implementing specific functions such as logic, sequencing, timing, counting and arithmetic, to control, through digital or analogue inputs and outputs, various types of machines or processes" (IEC, 2003).

The program on a PLC is developed using elements defined in the IEC 61131-3 standard. Data types, programming languages, program organization units (POUs), keywords and variable type definition are some examples of the content of the standard. Typically, a programmable logic controller system consists of a central processing unit (CPU) and I/O cards for interfacing with other devices, namely sensors, actuators and other PLCs. Nowadays, both the CPU unit and the I/O cards are designed to be modular, allowing hot-swap and a quick replacement in case of failure.

Modern PLC systems are equipped with a real-time runtime (PLC runtime), responsible for the process logics and an operating system (OS) runtime, usually Microsoft's Windows Embedded, Windows 10 IoT Core or a Linux distribution. The OS runtime is meant to perform more advanced programming tasks, allowing high-level programming language application to be developed and run on the controller.

On the aspect of programming languages, the PLC runtime applications (real-time applications) are developed using languages and elements defined in IEC 61131-3 (2013). Software and controller vendors, such as Codesys, Beckhoff, Wago or Siemens implement their own development tools following the standard and add their own specific applications and brandings as part of their product portfolio. For example, applications such as interlocking control, sequence control and motion control are implemented on the PLC runtime. These applications require less than 20 milliseconds of cycle time, which is the time for the output to react according to the input, often referred to as real-time capabilities.

On the other hand, high level applications on the OS runtime have no language limitations, as the programmable logic controller is fundamentally identical to any full-scale computer, although the OS runtime lacks real-time operation capability, a necessity for process control application. The types of application on the OS runtime may include e.g.: machine vision, webservice or information exchange services; implemented with C/C++, Python or JavaScript programming languages. Such applications serve the data integration process in the enterprise environment and does not require real-time capability.

In general, the term programmable logic controller has been understood to represent the PLC runtime for process control. Recent technological advances allowed the programmable logic controller to execute high level applications, bridging the gap between a PLC programmer and a software developer.

2.4 **Fieldbus and data communication protocols**

This chapter discusses fieldbuses and data communication in automation systems. In particular, the Open Systems Interconnection seven-layer model is described as the fundamental framework, followed by the descriptions and use cases of different protocols for data communication used in the empirical part of this thesis work.

2.4.1 Open Systems Interconnection (OSI) model

The OSI model provides a common basis for consistency between all types of network communication protocols. First published by the International Organization for Standardization (ISO) in 1984 as standard ISO 7498 (1984), the OSI model defined the seven-layer abstract model to profile any communication protocols. The functionalities of the layers are shown in figure 3.

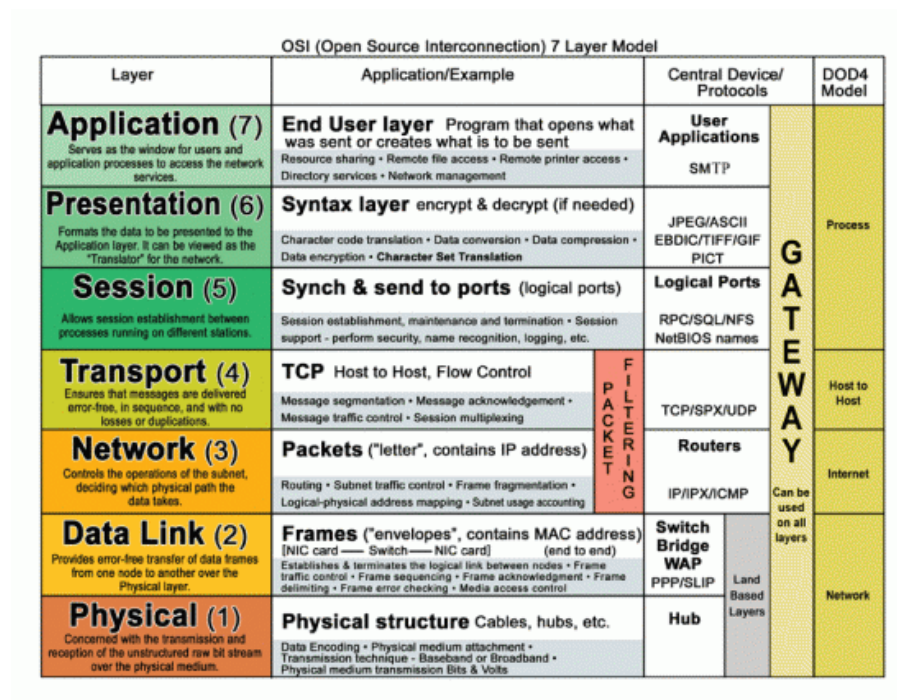


Figure 3. OSI seven-layer model (Cisco, 2017)

In general, the functionalities of OSI layers, from 1 to 7, are summarized and listed in Table 1.

Table 1. OSI model functionalities summary (Dye, McDonald, & Rufi, 2008, p. 53; Microsoft, 2017)

Number	Layer name	Functionality
1	Physical	Definition of how data is transmitted, encoding of the signal and transmission physical media (cable, wireless, etc.) and the connection topology of devices in the network
2	Data Link	Definition of nodes addressing, how data should be encapsulated or "framed", methods of controlling communication traffic and error reduction
3	Network	Definition of the communication path based on network conditions and how frames are routed to their destination
4	Transport	Definition of how data are transmitted with least amount of errors, losses and duplication
5	Session	Definition of how nodes can establish, maintain and terminate connections; optionally logging and security of said connections
6	Presentation	Definition of how data is formatted, compressed and encrypted
7	Application	Definition of services available to end-users and end-devices

The layers are usually referred to by their number, and any given protocols can be defined using some or all the OSI layers (Dye, McDonald, & Rufi, 2008, p. 53). The missing layers in the implementation could either indicate a lack of implementation, or an open implementation. For example, the Modbus protocol definition only match the application layer, lacking the definition of the lower layers. Due to this, there exist multiple implementations of the lower layers ranging from twisted pair cable communication (Modbus RTU or ASCII) to TCP/IP over ethernet or internet communication (Modbus TCP). Regardless, all implementations provide services according to guidelines defined in the Modbus protocol.

2.4.2 Digital Addressable Lighting Interface (DALI)

Digital Addressable Lighting Interface was developed in the 1990s, based on the IEC 60929 (2011) standard, for controlling lighting ballasts, drivers and relays. In a DALI network, each lighting device is assigned its own address and communication is achieved using low voltage digital signaling on two conductors. Before DALI was developed, lighting control was achieved using analog signals, which caused complexity in design and inferior communication signal quality. Each DALI master (or controller) can assume command of up to 64 addresses and 16 groups of addresses. (Sinopoli, 2010, p. 57). Examples of DALI installation and analog control installation are shown in figures 4 and 5.

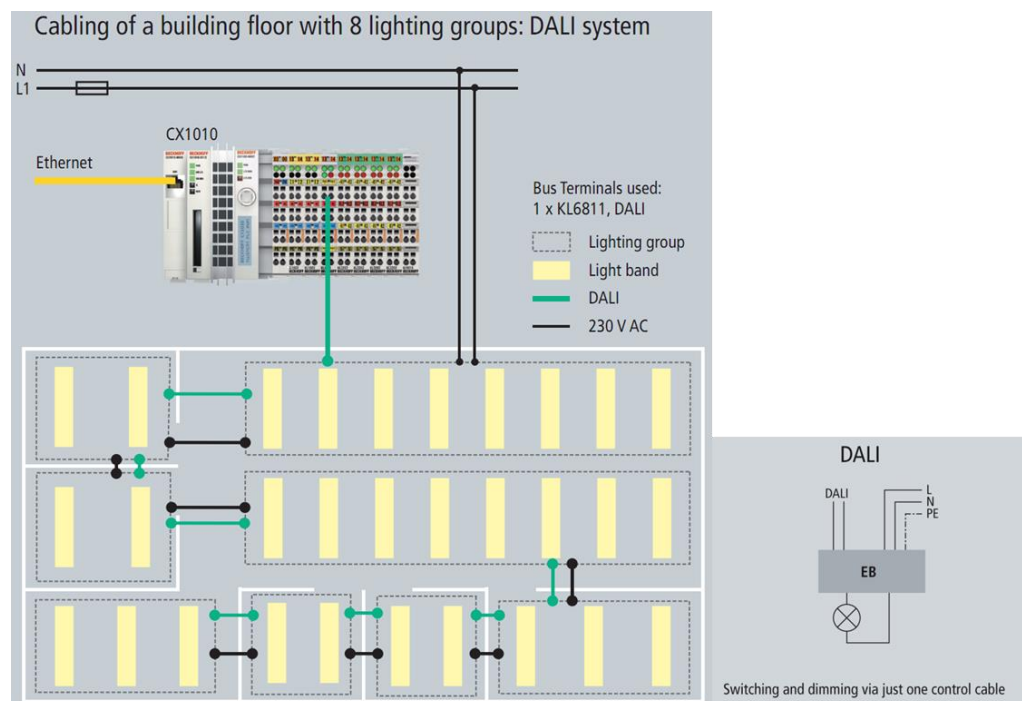


Figure 4. DALI light control installation (Beckhoff Automation GmbH, n.d.)

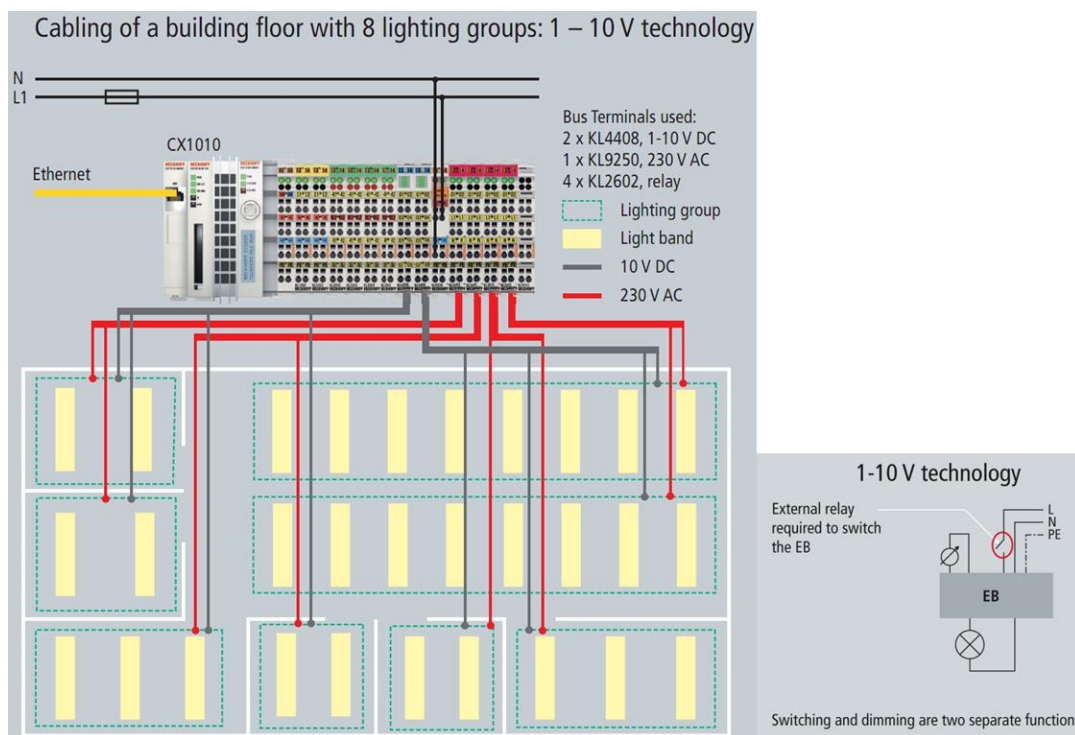


Figure 5. Analog 1-10V light control installation (Beckhoff Automation GmbH, n.d.)

As can be seen from figures 4 and 5, using DALI allows the cabling and functional design to be simplified significantly. The advantages of using DALI include flexible topology, interoperability between different manufacturers, real-time feedback of lighting values and flexible control of individual devices or groups. Firstly, the DALI network allows almost all types of topology, excluding only ring topology. Furthermore, the network requires no termination resistor, no polarity requirement and allows connection length of up to 300 meters. The DALI communication standard is maintained by DALI working group which includes reputable manufacturers, e.g. Siemens, Philips, Helvar, etc. therefore ensuring the continuous development and enforcement of the protocol. On the DALI network, the control and measurement values are repeatedly exchanged between the host controller and the end devices. Furthermore, said values are stored on both ends, allowing coherence and fast reaction within the network. Finally, each end devices are freely addressable and can be reconfigured programmatically, without physical change, allowing easy maintenance and flexible functionality throughout the lifecycle of the installation (Beckhoff Automation GmbH, n.d.).

The communication telegrams of the DALI network are illustrated in figure 6. Each block in the figure denotes one bit. The communication speed in the DALI network is achieved at 1200 bits per second.

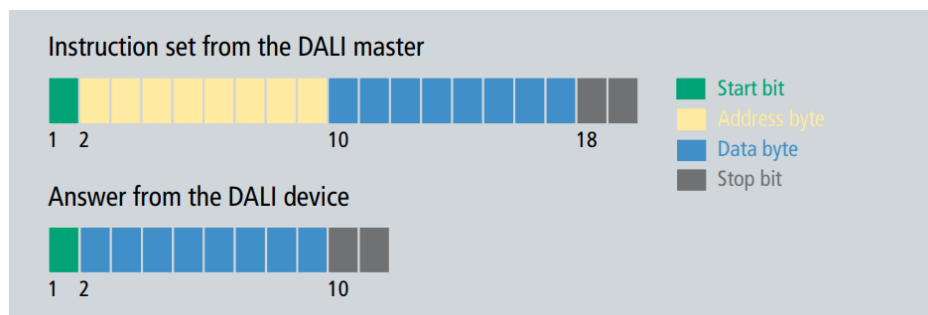


Figure 6. DALI query and response telegrams (Beckhoff Automation GmbH, 2010)

Each DALI device has two associated addresses: a 6-bit individual address (address range equals 0 to 63) and 4-bit group address (address range equals 0 to 15). Up to 16 scenes or output combinations are supported by one DALI network. Each device can be configured to respond to multiple group requests. The commissioning process of a DALI network is started by connecting all devices and randomizing the individual addresses of the end devices. Following that, each device address is reassigned according to engineering design, based on ease of programmability and grouping functions.

Typically, in building electrification, the power distribution to end devices is done with three-core cables (e.g. MMJ 3x1.5 cable), five-core cables are used for three phase electrifications. In the case of DALI, the designer can choose to use a five-core cable for both DALI communication and electrification, or run a separated cable for DALI communication and use the existing three-core cable for electrification. The latter method is applicable for upgrading of existing installations, where changing the cable would require efforts and downtime of the site. On the other hand, for new installations the first method should be applied due to economic reasons and maintainability.

2.4.3 EnOcean

EnOcean is innovative wireless technology based on energy-harvesting, in other words, efficient energy exploitation of mechanical motions and changes in the environment. After acquiring sufficient power, data is transmitted wirelessly. Most commonly used environmental effects used for energy harvesting are ambient lights and temperature differences. Data is transmitted and received through radio frequencies and is compatible with most major building automation protocols and PLC systems. The technology is in depth explained in the international standard ISO/IEC 14543-3-10. The protocol itself is regulated and maintained by the EnOcean Alliance, which consists of reputable manufacturer in the building automation field, e.g. EnOcean, Siemens, Texas Instruments, etc. Finally, typical use cases of EnOcean technology includes wireless control and indoor measurements. (Dang, Lupea, Multaniemi, & Tran, 2016, pp. 1,3)

The protocol employs the 868MHz frequency band in Europe, allowing up to 300 meters free space communication distance and 30 meters indoor communication distance. Due to its energy harvesting nature, the messaging rate between devices are typically from 5 to 15 minutes per message. Furthermore, the protocol is equipped with security mechanism such as encryption and message counting, allowing secured usage in sensitive environment applications. EnOcean data frames and security mechanisms are described in figure 7.


MESSAGE	Replay Attacks	Eaves-dropping	Energy & resource demand
R-ORG DATA	Yes	Yes	 Energy Harvester
R-ORGS DATA	Yes	No	
R-ORGS DATA RLC CMAC	No	Yes	
R-ORGS DATA RLC CMAC	No	No	

Figure 7. EnOcean communication frames (Dang, Lupea, Multaniemi, & Tran, 2016, p. 4)

An EnOcean device is typically identified via the chip ID and the sender ID. Chip ID is a 32-bit sequence, specific to each EnOcean device. The sender ID consists of the Base ID and a “sub” sender ID. Said IDs are reprogrammable, allowing faster device replacements without having to go through the teach-in process with the master module. The ID length is 32 bits in total, although the total number of reprogrammable addresses is only 65536 (16 bits). EnOcean devices communication is achieved by exchanging message packets conforming to EnOcean Equipment Profiles (EEP), which also provides the basis for programming EnOcean applications. The commissioning process of an EnOcean installation includes range planning and antenna placement to ensure the network coverage for all devices. Simultaneously, the list of devices, their IDs and profiles should be documented. After the installation of devices is done, the programming can be conducted.

2.4.4 Modbus TCP and RTU

Modbus was developed by Modicon (Schneider Electric) and has since become the de facto standard for multivendor device communication (Mackay, Wright, Park, & Reynders, 2007, p. 96). The protocol defined the message frame format between devices and the services which should be available on slave devices. In a Modbus network, each slave device is

assigned an address, which can be reconfigured if necessary. With Modbus RTU/ASCII, the slave device count is limited to 248, whereas with Modbus TCP the slave quantity is limited by the IP address range. The data points in each slave device is presented and classified into registers and coils, with registers representing numerical values (e.g. setpoint and measurement values) and coils representing binary values (e.g. alarm signals and contacts statuses). The master then interacts with the slaves through read and write operations on registers and coils, as the documentations of the slave devices often provide the definition and functionality of each coil and registers. Tables 2 and 3 describes the service (function) listing of the Modbus protocol and the message frame format, respectively.

Table 2. Modbus data point address ranges and functions (Mackay, Wright, Park, & Reynders, 2007, p. 98)

Data type	Addresses	Function codes	Function descriptions
Coils	0 → 9998	01	Read coil status
Coils	0 → 9998	05	Force single coil
Coils	0 → 9998	15	Force multiple coils
Discrete inputs	0 → 9998	02	Read single input
Input registers	0 → 9998	04	Read multiple input
Holding registers	0 → 9998	03	Read holding registers
Holding registers	0 → 9998	06	Write holding register
Holding registers	0 → 9998	16	Write holding registers
-	-	07	Read exception status
-	-	08	Diagnostic test

Table 3. Format of Modbus message frame (Mackay, Wright, Park, & Reynders, 2007, p. 97)

Address field	Function field	Data field	Error check field
1 byte	1 byte	Vary	2 bytes

Devices that support Modbus ranges from field devices such as sensors, variable speed drives to control room PLCs, SCADA and DCS systems. Typically, PLCs and control room devices act as master and field devices act as slaves in a Modbus network. Modbus RTU and ASCII uses physical layer defined in EIA-232 and EIA-485 standards, which is commonly known as twisted pair cables. Modbus TCP encapsulates the Modbus communication frame into a TCP message and uses the TCP network, usually Ethernet, to perform communication. A summarized comparison of Modbus TCP and Modbus RTU/ASCII is provided in table 4.

Table 4. Comparison of Modbus RTU/ASCII and TCP

	Modbus RTU/ASCII	Modbus TCP
Communication media	Twisted pair cable	Category 5/6 cable
Maximum speed	115 Kbps	57.6 Mbps with 100Mbps ethernet
Maximum length	1200 m at 9600 bps	100m
Redundancy	CRC16	Handled by TCP protocol
Concurrency	Half-duplex	Multiple session full-duplex
Topology	Single master network, typically line topology	Multiple masters network, line or star topology
Application notes	Baud rate and parity settings, termination resistor at furthest slave	TCP port and firewall settings

The commissioning of a Modbus network consists of installation and programming processes. The installation process should comply with information provided in Table 4, especially the application notes. Also, each slave device must be assigned a unique address. In the programming process, the datasheet and manual of each device in the network should be collected, as the register's and coil's map of each device should be listed in said documentations.

In building automation, Modbus is mainly used for communication between VSDs in air handling units or pumps, and the controller. Recently, heat pump units and solar inverters are also supporting Modbus as their communication method.

2.4.5 TCP/IP

Transmission Control Protocol (TCP) and Internet Protocol (IP), known together as the Internet protocol suite or TCP/IP, is generally acknowledged as the backbone for the Internet. The protocol suite was developed by the Advanced Research Projects Agency (ARPA), more commonly known nowadays as the Defense Advanced Research Projects Agency (DARPA). The protocol suite matches layer 3 and 4 in the OSI model, with IP acting as layer 3 and TCP acting as layer 4. (Mackay, Wright, Park, & Reynders, 2007, pp. 257,258).

The Internet Protocol's responsibility is the delivery of datagrams between hosts or devices in a network. This is achieved via the use of IP addresses for device identification and packet fragmentation, as networks might have different packet size requirements. Currently, the most widely used version of IP is IPv4, with plans for migration to version 6 (IPv6), due to the

limitation of the address pool with IPv4. Each IPv4 address is made up of 32 bits, usually denoted as four octets in their decimal presentation, e.g. "192.168.0.69".

To achieve the intended functionality of IP, a header of at least 20 bytes, i.e. five 32-bit combinations, is attached to the information passed down from the upper layers (Mackay, Wright, Park, & Reynders, 2007, p. 260). The header content for IP version 4 (IPv4) is attached in figure 8.

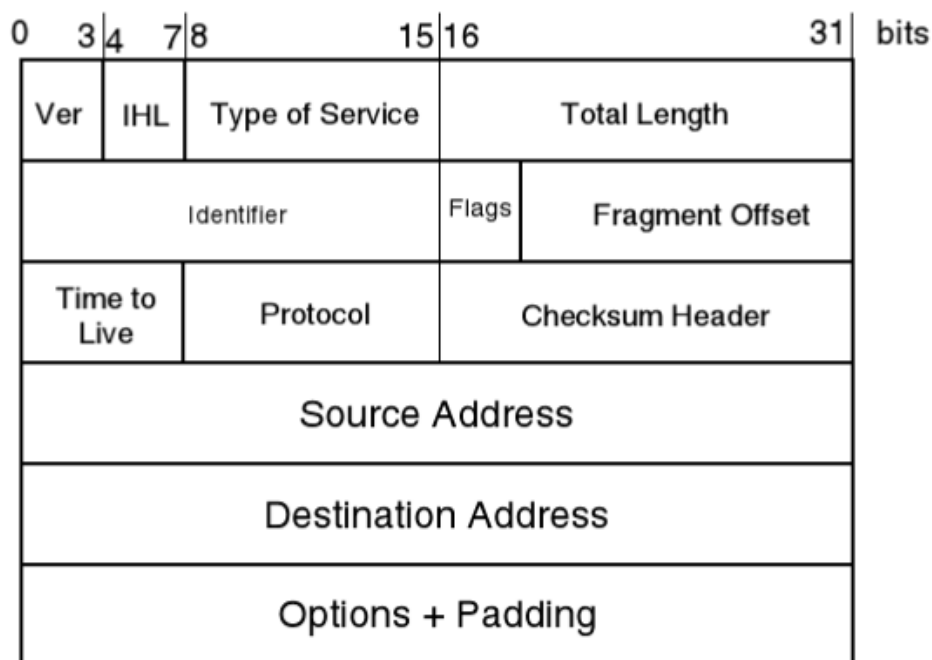


Figure 8. Content of IPv4 header (Mackay, Wright, Park, & Reynders, 2007, p. 260)

The fields shown in figure 8 is further explained as follows:

- Ver: indicates the protocol version. In IPv4 case, this field has the value of "4".
- IHL: Abbreviation for Internet Header Length. Indicates the length of the header in 32-bit segments.
- Type of Service: Used to indicate the quality of service (QoS) of the datagram, e.g. minimal delay, maximum throughput, maximum reliability, minimum monetary cost.
- Total Length: contains the length of the whole datagram. Combined with the IHL field, the host can determine the content of the datagram. The minimum length requirement accepted by all hosts is 576 bytes.
- Identifier: Used for unique identification of datagrams. In case of fragmentation, also used for reconstruction.
- Flags: Indicates whether a datagram could be fragmented or not.

- Fragment offset: indicates the position of one datagram for reconstruction use, in the case of fragmentation.
- Time to Live (TTL): Indicates the travel distance of one datagram. The value is decreased through each router that the datagram traversed. On the other hand, the field enforces deletion of datagram in case of undeliverable datagram, as a datagram will be discarded when the TTL value reaches zero.
- Protocol: indicates the protocol used in layer 4 for one datagram. Typical values are 6 for TCP and 17 for UDP.
- Checksum header: contains a check value for the IPv4 header. Recalculated at every point which the datagram passed through, due to constant changes in some parts of the header, e.g. TTL.
- Source and destination address fields: indicates the source and destination of a datagram, represented by their IPv4 address.
- Options and padding: used for appending additional information, or to fulfill the 20-byte length requirement.

Operating on the basis provided by IP, TCP provides the means for session establishment between two hosts, to ensure the reliability of data transmission. TCP allows large datasets to be transmitted partially and reconstructed, as well as provides verifications mechanism, flow control and socket services for multiple connections between two hosts. In addition to TCP, User Datagram Protocol (UDP) is an alternative to data transmission in use cases where synchronization is not required, or low data volume is needed. Both TCP and UDP uses the “port” concept for source – destination addressing and to achieve flow control as well as multiple connections between two hosts. The main difference between TCP and UDP is the reliability of data transmission. TCP headers are significantly larger than UDP headers, therefore allowing sequencing for large data transmission, error control mechanisms and retransmission for guaranteed data delivery. On the other hand, the small header size allows UDP to be used in situations such as network announcement, broadcasting and data streaming (Mackay, Wright, Park, & Reynders, 2007, pp. 270-272).

2.5 Data access and presentation in software engineering

This chapter provides information on the structure and abstract components of a software system, namely, the data access layer and presentation layer, often referred to as the backend and the frontend respectively. Specifically, the backend and frontend for the Internet of Things and web services use cases are discussed. The terms “data access” and “presentation” in this section do not refer to the meaning described in the OSI model, as these functionalities strictly belong to the application layer in the OSI model. In general, the backend is responsible for providing reliable data for different uses, e.g. monitoring, data collection, sharing and visualization, whereas the frontend is responsible for efficient human-

machine interaction, converting machine data into human-friendly useful indicators as well as interpreting human intent into machine directives and data manipulation.

In comparison with the backend, the frontend involves developing applications which are responsive, reactive and meaningful to the human user. Examples of backend products include computer programs, mobile applications and web applications. Within the scope of this thesis work, only backend development is discussed and was commissioned.

2.5.1 Database management system

Usually in the context of the IoT and web development, the backend involves the design and deployment of database systems and data connectivity platforms, commonly known as Application Programming Interfaces (APIs). The main challenges in backend development are scalability and reliability requirements, as it is necessary for the backend to store large quantity of data and handle data requests for multiple different clients with acceptable latency while maintaining high availability and uptime. With regards to web services, the main responsibility of the backend is to provide continuous interaction with the frontend and ensure synchronization between the stored data and the user interaction. Furthermore, with regards to IoT, the backend also handles machine-to-machine communication and decision-making functionalities to achieve maximum machine efficiency.

Data storage is achieved via the use of database management systems, often abbreviated as DBMS or commonly known simply as “database”. The performance of a database system heavily depends on the computing power and storage capability of the server computer. Modern database systems support distributed computation solutions, i.e. deployment of database systems in a large array of hardware, to achieve load reduction on each server unit and ease of hardware addition in case of increased computational requirement. In other words, this allows the server to “scale” as the application requirements grow, while also provides higher data availability.

Database model are unofficially classified into two categories: relational and Not-only-SQL or NoSQL. Relational database is also known as SQL (pronounced as “sequel”) database, as almost every relational database solution uses the Structured Query Language (SQL) for access and modification of data inside the database. This database model organize data into tables, using columns to describe different attributes of one dataset and rows to store different records of said dataset. NoSQL database system, on the other hand, offers data storage in different forms, e.g. key-value pair, document, graph and column.

The empirical work of this thesis involved processing and storing large quantity of time series (TS) data, i.e. data associated with time values. For example, a collection of different measurements recorded at different timestamps is considered a time series. SQL and NoSQL database systems could be implemented to warehouse TS data, at the cost of high latency and disk storage, due to their general purposed nature. Recently, time series database solutions were developed, offering higher data throughput and lower latency as well as optimized data warehouse algorithms, leading to lower storage requirements. Figures 9, 10 and 11 illustrates the performance comparison between InfluxDB and MongoDB, as examples of time series database system and NoSQL database system, respectively.

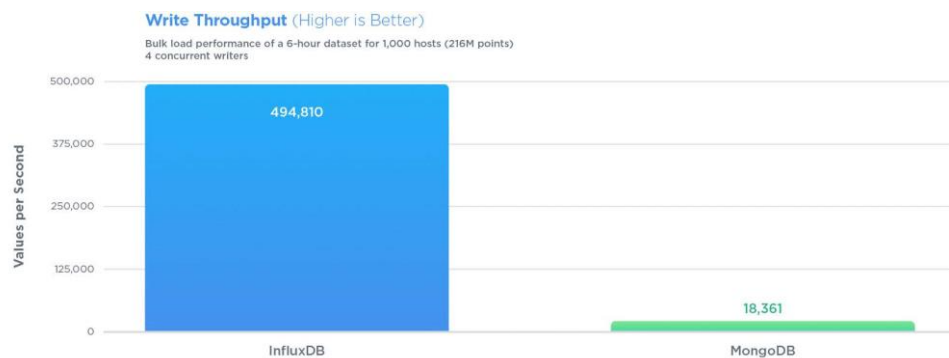


Figure 9. Write throughput comparison – InfluxDB vs MongoDB (Persen & Winslow, 2016, p. 6)

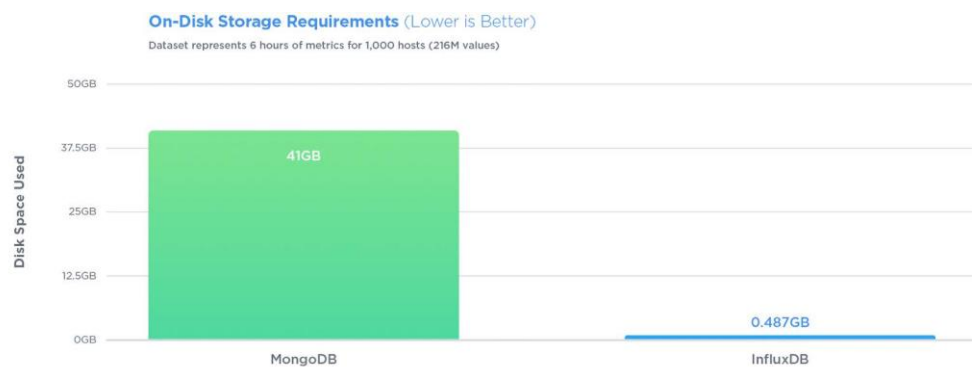


Figure 10. Disk storage comparison – InfluxDB vs MongoDB (Persen & Winslow, 2016, p. 6)



Figure 11. Query throughput comparison – InfluxDB vs MongoDB (Persen & Winslow, 2016, p. 7)

As can be seen from figures 9, 10 and 11, time series database system offers superior data writing throughput and disk storage optimization, while maintaining similar query performance. In conclusion, for the use cases of the IoT and Industry 4.0 data collection, time series database systems offer the most advantages compared to other database systems and therefore is the most optimal choice.

2.5.2 Application Programming Interface (API)

The term “application programming interface” can be understood as the platform for computer programs to interact and communicate with each other. As examples, Microsoft provides the Windows API for software developers to program applications which can be used on the Windows operating system and Google provides the Maps API for developers to build applications which needs location data. The first example represented the APIs provided for a specific platform, in this case machines which run the Windows operating system. The second example demonstrated APIs provided for multiple platforms, such as mobile operating systems and web-based systems. An API is usually associated with a ready-made software and typically defines a set of methods, protocols, data interpretation and access points so that software developers can write new software that interacts with the existing software. Consequently, an API provides encapsulation and controlled access to the existing software, i.e. allowing the existing software to be expanded and supportive of new software. This section discusses the Representational State Transfer web APIs, commonly known as REST APIs and their use case in the context of IoT and Industry 4.0. Any web services which provide a REST API are commonly denoted as RESTful web services.

With respect to the database systems discussed in section 2.5.1, it is necessary to develop an API so that devices and users can safely exchange data with the database, as well as assuring control over which party can have access to the data stored within the database. The API construction

can be separated into two parts, dealing with data input and output, respectively. The main clients of the input interface are devices, whereas the clients of the output interfaces are software developers, researchers and other software which can take advantage of the data.

The REST model was documented by Roy Fielding in his doctoral dissertation, as principles based on which web servers and clients could exchange data over the Internet. Those four principles included: identification of resources; manipulation of resources through representations; self-descriptive messages; hypermedia as the engine of application state. (Fielding, 2000; Wilde & Pautasso, 2011). To explain the principles, the term “resource” is explained as the data owned by the server, whereas the term “representation” is explained as the data received by the client from a request sent to the server through the “message”. The fourth principle can be understood as, the interaction between the server and client is contained within the hypermedia, i.e. web addresses or links. (Wilde & Pautasso, 2011, p. 37). REST API employs hypertext transfer protocol (HTTP) as the transport service, the same scheme with the world wide web. Links and web addresses are classified as Uniform Resource Identifier (URI), as standardized in the RFC3986 publication of the Internet Engineering Task Force (IETF). Figure 12 demonstrates an URI for retrieving all latest measurement points from the SMC building, provided by the API commissioned in the empirical part of this thesis.

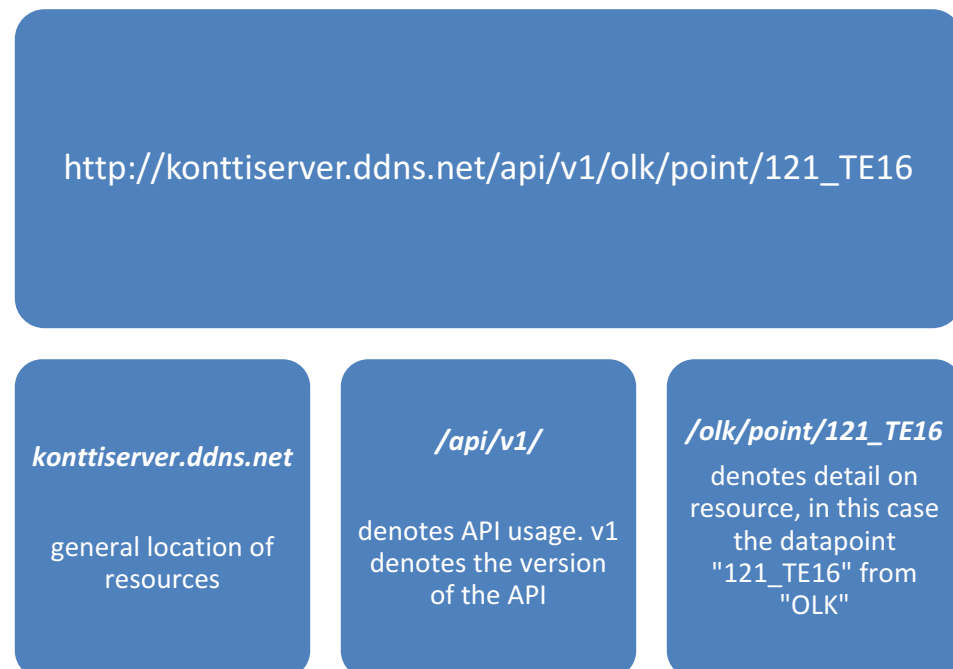


Figure 12. API URI demonstration

Typically, the client will send a HTTP request to the server in the format of "*{action} {URI} HTTP 1.1*", where the action could be one of those listed in Table 5 and the URI, usually referred to as the “REST endpoint”, in similar

format with one demonstrated in figure 11. Additional information such as details about the client, authorization or expected response content type can be, and very often is, embedded into the header of the request.

Table 5. REST API actions and descriptions

Action	Change resource on server	Usage
GET	No	Used to retrieve data from server
POST	Yes	Used to create new data on server
PUT	Yes	Used to update existing resource on server
PATCH	Yes	Used to update resource on server, partially
DELETE	Yes	Delete resource on server

Table 6 demonstrates a GET request sent to an API, and the response to said request.

Table 6. REST API request and response example

Request	Response
GET konttiserwer.ddns.net/api/v1/olk/point/ 121_TE16	{ "id":"121_TE16", "desc":"Air temperature in room 121", "unit":"°C", "value":20.39, "timestamp":"2017-12- 18T14:27:53.066Z" }

The example in table 6 is explained as an enquiry from a web client, asking for the temperature in room 121, which is identified by the string "121_TE16". The API server responded with a JavaScript Object Notation (JSON) string containing general information about the "121_TE16" point in the "desc" and "unit" keys; the measurement value and the timestamp at which the measurement was taken is stored in the "value" and "timestamp" key. JSON is a widely used string-based data representation, which allows different data structure to be represented in key – value notations while maintaining human-readability as it can be seen in the table 6 example. Furthermore, nowadays almost all programming languages have their own JSON parser, allowing developers from multiple programming paradigms to use JSON for data exchange in their applications (Ecma International, 2017). Finally, REST APIs are released with their own documentation, containing information on all endpoints; methods; descriptions of request and response; authorization requirements and limitations.

2.6 Building Automation System

The building automation system (BAS) is the network of sensors, actuators, controllers and computers used in the building environment, responsible for regulation and adjustment of the indoor environment. The responsibility of the BAS often includes heating, ventilation, air conditioning, lighting, alarms and access control of a property or a building. Modern BAS also includes features such as equipment management, and facility management, audio – visual control and video surveillance. In this section, the HVAC control and lighting control are emphasized and described, due to their vital necessity for building operation and energy optimization in Finland.

2.6.1 Heating, Ventilation and Air Conditioning (HVAC) control system

The HVAC system typically consists of two sectors: heating control and ventilation – air conditioning control. The heating system is responsible for controlling the temperature of radiator network, domestic water network and incoming air of the building through heat exchange or generation; whereas the air conditioning system is responsible for maintaining the indoor air quality by controlling the indoor air flow, air humidity and air filter.

In Finland, the heating energy supply for one building usually comes from a boiler unit or a district heating supply. A boiler-based system generates heat from burning different kind of fuels, whereas a district heating system would get the energy from local district heating network, which generates heat from various production activities. Recently, the use of heat pumps and solar heating systems in building heating system are increased, due to the combined effect of price reduction, performance, long-term benefits and energy resource scarcity. Following the heat generation section, the heat is carried through the whole building through the radiator network. In appendix 1, pages 11 to 14, the block diagram of the building heating system of Sheet Metal Center, which is equipped with geothermal heat pump and solar heating system is described.

The air conditioning system, or air handling unit (AHU), consists of the air piping network and fan units for supplying and removing the air from the building, with air filters for blocking small particles in the air circulation. The air supply process involves taking air from the outdoor environment, passed through a filter, heated up and then propagated inside the building. On the other hand, the air removal process takes the air out passively using pressure difference, or actively with the use of a fan unit and very often, the exhaust air is passed through an air heat recycling unit to take advantage of the heated air and reduce the heating costs. Figure 13 describes an air handling unit with heat recycling, in which the upper path is the air removal process and the lower path is the air intake process. The

full version of figure 13, included with the control signal and functional description (in Finnish) is included in Appendix 1, pages 30 to 32.

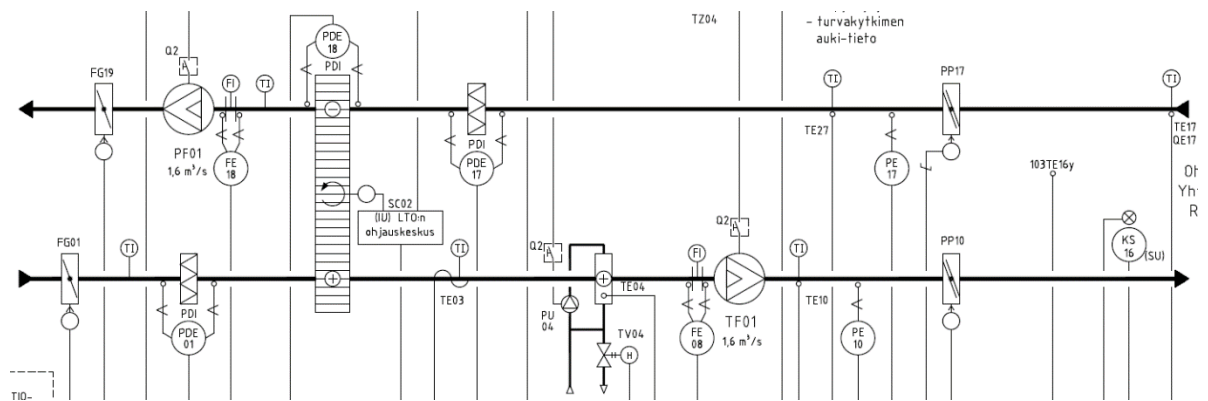


Figure 13. Air handling unit with air recycling PI-diagram

The HVAC control system is usually equipped with schedule control for optimal energy saving, which sometimes leads to excess energy wasting when the building is not occupied, as well as discomfort indoor environment when the control schedule does not meet with the current inhabitant capacity of the building. Therefore, on-demand HVAC control is necessary to maintain optimal indoor environment quality and energy saving targets. To achieve this, additional measurements such as occupancy and CO₂ concentration are needed. The traditional HVAC control system is often not equipped with such measurements, although recent advances in sensor technology and IoT could allow addition of these measurements, therefore making the further optimization of HVAC control system feasible.

2.6.2 Lighting control system and occupancy management

Smart lighting control and occupancy management contributes significantly to the energy optimization process of a building. As an example, Pirelli Deutschland GmbH in Germany performed a renovation to their warehouse lighting system, which involves adding occupancy control to only turn on the lights at areas which employees are working at. The warehouse has a floor area of 10000 square meters, where constant lighting 24/7 was used before the renovation. Furthermore, the company replaced conventional fluorescent lamps with LED lighting fixtures with DALI control for higher efficiency. The electricity power requirement for lighting reduced from 20 kW to 11.5 kW with the change from fluorescent to LED lighting and further went down to 1.5 kW with the addition of DALI control, directly translates into a 92.5 percent energy consumption reduction (Beckhoff Automation GmbH, 2016).

Smart lighting control involves multiple strategies to mobilize the lighting equipment based on real-time demands of the building occupants.

Sinopoli (2010) listed examples of those strategies which included scheduling, occupancy control, daylight utilization and windows coating usage. The first two strategies involved software solutions which can be commissioned after the building construction, whereas the latter two strategies were required to be executed during the construction planning phase. Furthermore, light intensity control, or “dimming”, is also widely used due to both comfortability and economic reasons (Sinopoli, 2010). A typical industrial lighting fixture can produce up to 10000 lumens of luminosity theoretically, whereas the illumination need for work areas are much lower, around 500 lumens (Finnish Standards Association - SFS, 2011). In conclusion, lighting engineering and precise control is necessary for satisfying the lighting requirement and achieving energy efficiency simultaneously.

3 EMPIRICAL PROCESS

3.1 Preliminary design data collection and component selection

3.1.1 Preliminary design data collection

To conduct the empirical work of this thesis, the following documentation were gathered:

- HVAC drawings of geothermal heat pump, solar heat generation, thermal energy storage, radiant roof heating system and air ventilation system of the SMC building. These drawings are included in Appendix 1.
- Electrical drawings of the SMC building, including lighting and distribution panels. The lighting system drawing is included in Appendix 2.
- List of measurement points from Caverion system of the HVAC control of SMC building.
- Datasheets of ABB weather station, solar inverters and datalogger

Before this thesis was commissioned, the data from HVAC control system is only available to the building owner on an annual basis, as a report requested from Caverion. Furthermore, even though the building was equipped with a weather station and solar inverter system, no data was collected to perform analysis on the performance of solar electricity system, as well as correlation of local weather with the performance of the HVAC system. Finally, the electricity consumption of the building could not be precisely analyzed, as the only information available to the owners was the monthly consumption from the electricity supplier. In general, there was a shortage of data, therefore energy researches conducted on the building, as well as data visualization for information purposes was limited severely.

3.1.2 Component selection

The component selection was conducted based on the connectivity requirement of existing systems, as well as operation targets of the project. The component selection for the PLC system and the functionality of each component were as followed:

- Beckhoff CX5130: Industrial PC with PLC runtime and Windows Embedded Standard 7. Responsible for data communication, lighting control and sensor communication. Used for Modbus TCP communication with Caverion system and data communication with the backend server.
- KL6811: Beckhoff I/O card for DALI communication. Used for lighting control.
- KL6581 and KL6583: Beckhoff I/O card for EnOcean communication and EnOcean antenna. Used for EnOcean sensor communications.
- KL6041: Beckhoff I/O card for RS485 communication. Used for Modbus RTU communication.
- Schneider Electric ABL8REM24030: 24 Volt 3 Ampere power supply module

On top of the PLC selection, the LED drivers of the current lighting system will be changed to DALI communication model for implementation of the smart lighting system in 2018. Also, EnOcean sensors will be added for more precise indoor air quality monitoring.

The backend server selected was a normal workstation desktop, due to possibility of moving the backend to cloud system. Regardless, the desktop was equipped with more memory and a solid-state drive to ensure performance requirement. In long term, cost analysis will be performed to decide whether a full-fledged server system or a cloud-based system will be used for production purposes.

3.2 Network infrastructure

The network connection was implemented based on the existing IT infrastructure of the building. The PLC was connected to HAMK IoT network, which was HAMK intranet with firewall rules implemented to support IoT communication.

Figure 14 illustrates the connection scheme which was commissioned for the thesis project.

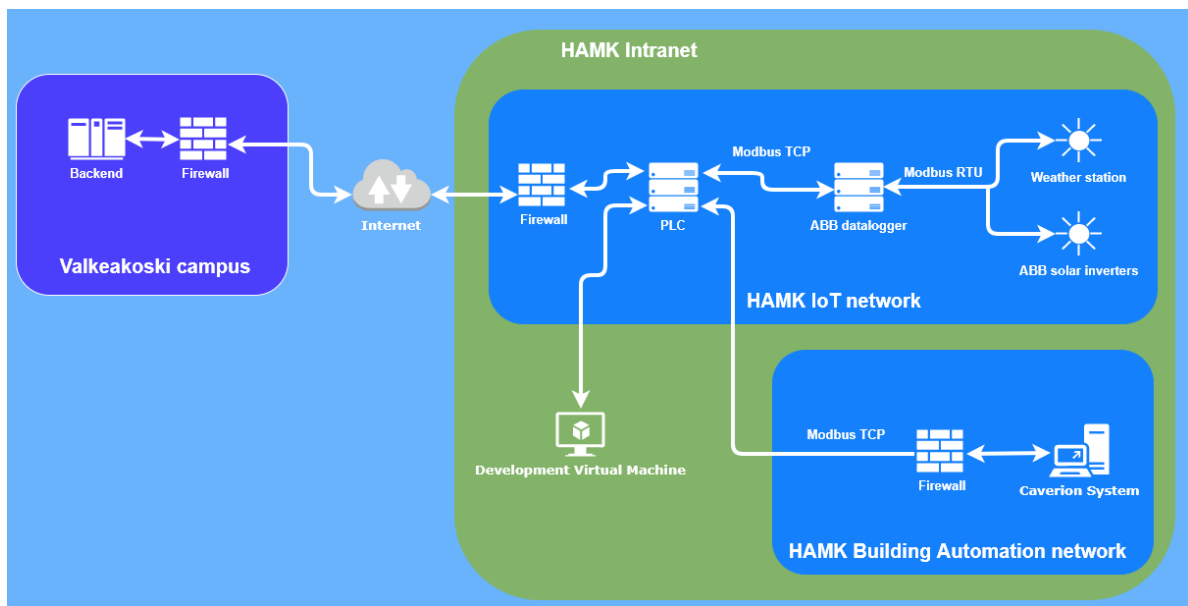


Figure 14. Connection scheme for empirical process

As can be seen from figure 14, there were four networks involved in this thesis project: HAMK IoT network, HAMK Building Automation network, HAMK Intranet and Backend network, noted as Valkeakoski campus. Their characteristics can be briefly summarized as:

- HAMK Intranet: production local network of HAMK, does not allow internal device communication regulation.
- HAMK IoT network: subnetwork of HAMK Intranet, allows flexible internal communication.
- HAMK Building Automation network: subnetwork of HAMK Intranet, for running critical infrastructure. Completely isolated from all other subnetworks and the internet.
- Backend network: development network, separated from Intranet. Interaction with the backend is only possible through the use of the API.

For communication with the Caverion system, a firewall exception had to be made, allowing a single-directional communication channel between the system and the PLC located in IoT network on TCP port 502. The Caverion system acted as the Modbus master device, writing all HVAC system data to the Modbus slave on the PLC runtime.

Inside the IoT network, communication regulations were more flexible, therefore the PLC could be connected with the ABB datalogger, which collected data from two solar inverters and the weather station. The Windows runtime from the PLC then aggregate the data from Caverion and the datalogger, then send them to the backend through the API.

Ideally, the PLC could not be accessed from outside of the HAMK IoT network. Currently, HAMK IT services configured and deployed a virtual

machine for remote development and troubleshooting purposes. This virtual machine can only be accessed by the author and will be shut down when the project is finalized.

3.3 PLC program

The PLC program consisted of two sections: PLC runtime and Windows runtime program. The PLC runtime program was responsible for receiving data from Caverion system and sending the data to the Windows runtime. Firstly, the PLC needed to be configured as a Modbus TCP slave, with mapping to PLC variables configured. In the PLC program, the registers data from Modbus communication was mapped into a structure, corresponded to the data point list of Caverion, for ease of later programming and readability. At the same time, the data structure was enabled for OPC UA access, as the Windows runtime program must be able to read the structure.

On the Windows runtime of the PLC, NodeJS was installed in order to run Node-RED, an IoT platform developed for data flow programming. The connection of data flow made in Node-RED was as followed:

- Caverion data structure: read using OPC UA, published to backend at endpoint `/api/v1/olk`
- ABB datalogger: read using Modbus TCP, published to backend at endpoints `/api/v1/olk_abb_weather`; `/api/v1/olk_abb_uno` and `/api/v1/olk_abb_trio` for the weather station, one-phase and three phase inverters respectively.

The data acquisition cycle time was set to five seconds and could be as low as two seconds. The cycle time was determined based on real performance of the systems and proposals for smart grid and smart building standards in Finland.

3.4 Backend deployment

The backend was deployed on a workstation desktop computer, located in Valkeakoski campus, with the network infrastructure handled by a 4G connection. In 2018, a decision will be reached on the new location for the backend, either on full-scale server hardware or cloud premises.

InfluxDB was selected as the database system, due to its open source nature and acceptable performance during the prototype phase. In the database schema, measurement points from different buildings were separated into different time series, even though this was not the recommended design practice provided by InfluxData (InfluxData, n.d.). The reason behind this design was due to incomplete API deployment, as the current API was not implemented with separated access for different buildings, therefore different time series was used to implement this

feature. Limitation and countermeasures of the current design will be further detailed in chapter 5.

Node-RED was selected as the platform for data connection and visualization, due to its ease of use for IoT prototyping. Firstly, REST API requests could be treated and handled as HTTP requests, as can be seen from figure 15 which demonstrated the data flow for data which came from the PLC in SMC building. The data from POST request will be extracted, parsed and inserted into the OLK time series in InfluxDB, while a HTTP return code 201 is sent back to the PLC, which indicates a successful request.

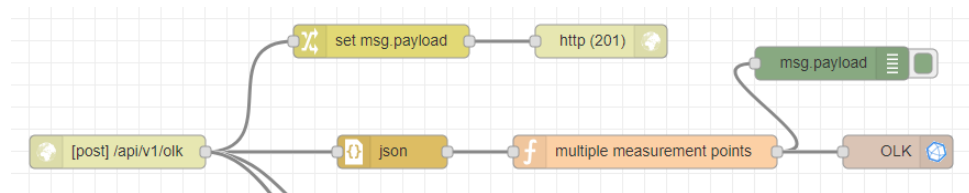


Figure 15. Node-RED flow for Caverion data sent from PLC in SMC

Secondly, Node-RED is able to visualize data fairly easily, using the dashboard module. Figures 16 demonstrates the visualization of data from the ABB weather station at SMC building. The flow operation is explained as a query made to the database for the latest weather record, then split into different items for display on the dashboard. The dashboard interface corresponding to the Node-RED flow in figure 16 is shown in figure 17.

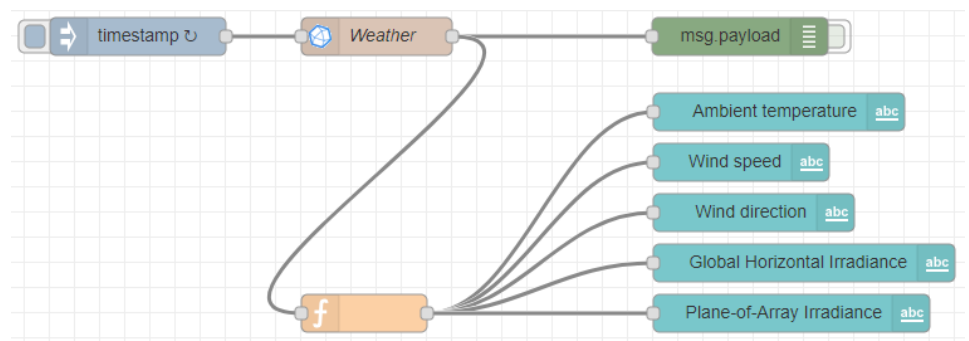


Figure 16. Node-RED flow for displaying weather data at SMC building

Weather data (ABB)	
Ambient temperature	0.70 °C
Wind speed	4.00 m/s
Wind direction	180.00 °
Global Horizontal Irradiance	15.00 W/m²
Plane-of-Array Irradiance	289.00 W/m²

Figure 17. Node-RED dashboard display for the weather data at SMC building

4 RESULTS

This chapter discusses the results of the thesis project, as of December 2017. The original target of collecting HVAC system data from Caverion control room was reached, as well as the deployment of the backend system. In addition to the original requirement of data service for Sheet Metal Center, the Valkeakoski Campus A-building and Hybrid Energy Module were also connected to the backend system. Therefore, data service of three entities are currently performed with the backend system of this thesis project. The A-building data was used for Ward Sohier's thesis project "Energy Efficiency of the A building", for the purpose of evaluating heat loss and energy performance of said building, as well as improvement suggestions for the building's HVAC system.

One of the usage for data collected from SMC building is verification of building simulation data. As building simulation could be used to test different energy optimization measures in the design phase, it is necessary to compare existing model in real scenario and in simulation to prove the merit of simulation techniques. For instance, the data collected from SMC building included the electrical consumption measurements. The collected data from November 2017 showed that the building consumed 16287 kWh. The simulated results published in "Energy Simulation of Sheet Metal Center with IDA ICE software" for November 2017 was 14559 kWh, which showed a 2 kWh difference with the real data (Nguyen, 2017). The difference was negligible, therefore with further developments and longer period studies, the simulation accuracy will be improved along with its merits. In addition, the data served by the backend is currently used in the development of the 3D web visualization model of Sheet Metal Center building.

5 LIMITATIONS AND EXPANSION POSSIBILITIES

Currently, the limitations of this system were as follows:

- Lack of API authorization
- Incomplete API documentation
- Load tests for the system not conducted
- Lighting control and sensor system not implemented

The development road map of 2018 for continuation of this thesis project was specified to address said issues. Firstly, the API will be finalized during January 2018, with proper authorization mechanisms and full documentations. Furthermore, more buildings are planned to be connected to the backend systems, which includes HAMK Visämäki campus S building and Päivölä boarding school, as those are pilot cases for projects commissioned by HAMK. Secondly, load tests will be conducted on the backend system to determine the final deployment location for the system which is either on commercial server hardware or cloud premises. Finally, the lighting control and extensive sensor system will be commissioned when the hardware is purchased. Also at this phase, the security audition for the PLC at SMC building will be conducted in cooperation with HAMK IT department.

6 CONCLUSION

In this thesis project, a PLC-based system was developed for the data communication with Caverion SCADA system, as well as ready for smart lighting control and indoor environment sensor network. In addition, a data server system was deployed for data storage and a REST API service was deployed for accessing stored data. The further development of the thesis project was planned and will be implemented in the following year.

All in all, the targets of this thesis project were achieved. The PLC system commissioned was able to communicate with the Caverion system and the data from the API was provided for energy researches at Sheet Metal Center, as well as the development of 3D web visualization model of Ruukki.

REFERENCES

- Beckhoff Automation GmbH. (2010). Application Note DK9222-0810-0031.
- Beckhoff Automation GmbH. (2016, October 14). Intelligent DALI lighting control system for highest energy efficiency. Retrieved from https://www.youtube.com/watch?v=pvF_gx5tHSY
- Beckhoff Automation GmbH. (n.d.). DALI technical training presentation.
- Cisco. (2017, December 11). *An OSI model for cloud*. Retrieved from <https://blogs.cisco.com/cloud/an-osi-model-for-cloud>
- Dang, K., Lupea, S., Multaniemi, S., & Tran, M. (2016). *EnOcean report*. Valkeakoski: Häme University of Applied Sciences.
- Dye, M., McDonald, R., & Rufi, A. W. (2008). *Network fundamentals : CCNA exploration companion guide*. Indianapolis, Ind: Cisco Press.
- Ecma International. (2017). Standard ECMA-404 - The JSON Data Interchange Syntax.
- Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Irvine: University of California.
- Finnish Standards Association - SFS. (2011, October 10). SFS-EN 12464-1 - Light and lighting. Lighting of work places. Part 1: Indoor work places. Helsinki, Finland.
- Hughes, T. (2007). *Measurement and control basics*. Research Triangle Park, NC: ISA.
- IEC. (2003). *IEC 61131-1 Programmable Controllers - Part 1: General Information*. Geneva: IEC.
- Inc, E. S. (2017, December 2). *Specifying Control Valve data*. Retrieved from <http://kb.eng-software.com/display/ESKB/Specifying+Control+Valve+Data>
- InfluxData. (2017, May). *Architecting for IoT: The Need for an IoT Data Platform*.
- InfluxData. (n.d.). *InfluxDB Version 1.4 Documentation*. Retrieved December 20, 2017, from <https://docs.influxdata.com/influxdb/v1.4/>
- Mackay, S., Wright, E., Park, J., & Reynders, D. (2007). *Practical Industrial Data Networks: Design, Installation and Troubleshooting*. Oxford: Newnes.
- McMillan, G. K. (2010). *Essentials of Modern Measurements and Final Elements in the Process Industry*. Research Triangle Park, N.C. : ISA.

Mesures, B. I. (n.d.). *BIPM - worldwide metrology*. Retrieved from <https://www.bipm.org/en/worldwide-metrology/>

Microsoft. (2017, December 11). *The OSI model's seven layers defined and functions explained*. Retrieved from <https://support.microsoft.com/en-us/help/103884/the-osi-model-s-seven-layers-defined-and-functions-explained>

Nguyen, N. (2017). *Energy simulation of Sheet Metal Center with IDA ICE software*. Hämeen ammattikorkeakoulu. Retrieved from <http://www.theseus.fi/handle/10024/136290>

Persen, T., & Winslow, R. (2016, September). Benchmarking InfluxDB vs. MongoDB for Time-Series Data, Metrics & Management.

Sinopoli, J. (2010). *Smart Building Systems for Architects, Owners, and Builders*. Amsterdam; Boston: Elsevier/Butterworth-Heinemann.

Wilde, E., & Pautasso, C. (2011). *REST: From Research to Practice*. New York: Springer.