TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

AHMED FARAHAT

TEST APPLICATION OF THE INTERNET OF THINGS FOR ENERGY EFFICIENT OUTDOOR SMART LIGHTING

Master of Science Thesis

Examiner: Professor José L. Martínez Lastra

Examiners and topic approved in the Engineering Sciences Faculty Council meeting on 9th of October 2013.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Master’s Degree Programme in Machine Automation

FARAHAT, AHMED: Test application for the Internet of Things for energy efficient outdoor smart lighting

Master of Science Thesis, 67 pages, 9 Appendix pages

Feb 2014

Major: Factory Automation

Examiner: Prof. José Luis Martínez Lastra

Supervisor: Anna Florea

Keywords: Smart lighting, energy efficient, Internet of Things, outdoor, smart metering, Device Profile Web Service, automation, Key Performance Indicators, Web Services, Service Oriented Architecture

The term Internet of Things (IoT) is immerging lately in many researches and publications with the purpose of standardizing the concept. The IoT tends to connect the physical world virtually through the internet by creating a virtual identity for every physical object. Huge amounts of devices with numerous amounts of data will be connected by the year 2020. Researches are aiming to implement the IoT paradigm in several domains such as smart lighting, traffic control, waste management, and air pollution. Due to the immense necessity for reducing energy consumption and creating adaptive lighting solution, smart lighting earned most of the attention of researchers.

This thesis work aims to apply the IoT paradigm for controlling and monitoring of a smart lighting application. The implementation aims to reduce the energy consumption in addition to, achieve adaptability of the lights according to the surrounding environment. The implementation uses Service Oriented Architecture (SOA) to allow heterogeneity and interoperability between components.

In this implementation a control algorithm is proposed, which takes into account most of the surrounding environment conditions and conducted tasks. The main purpose of control is to reduce energy consumption and loss by delivering sufficient amount of lighting required without affecting the visibility. On the other hand, energy consumption measuring and monitoring were achieved by automatic subscribe and discovery of energy meters by means of Device Profile Web Service (DPWS).

Moreover a set of specific KPIs are designed in order to give a holistic view of the system to the facility managers, in order to evaluate and analyse the performance. In addition, it raises the awareness of personnel with the impact of light usage. Finally a web based dashboard application is developed in order to present system data in real time and display KPIs in a visualized way. The dashboard consumes Web Services (WS) for retrieving data.

PREFACE

This thesis work was made in the Factory Automation Systems and Technology laboratory at TUT university. First I would like to begin by thanking professor Jose L. M. Lastra for giving me the opportunity to work in the FAST lab and providing me with everything I need for making this thesis work. Second I want to thank all of the FAST staff and members for providing the best work environment and, helping me to evolve and develop intellectually; including everyone without mentioning names so as not to forget anyone. And of course a very special thanks and appreciation to my supervisor Anna Florea for always pushing me one step further to do my best. As well as giving me guidance through writing this thesis and, teaching me the research methods. Thank you for babysitting me.

I would also like to thank all of my friends who were supportive all the way through. Special thanks to Borja, Sergii, Zoran, Anton, Mohamed, Manu, Oscar, Luis and Gerardo. Also my friends from Egypt for their long distance support, motivation and convincing me to go to Finland at the first place.

Finally and the most important of all a big thank you to my father Amr Bahey, mother Hanan Ahmed and, brothers for their support and love. All my family is god’s gift for me to walk through this life and be where I am right now. At the end, thanks to you, reader. If you are reading this line after the others, you at least read one page of my thesis. Thank You.

Tampere, January 11th, 2014
Ahmed Farahat

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACN	Architecture for Control Networks
BACnet	Building Automation and Control Networks
CAN	Controller Area Network
DALI	Digital Addressable Lighting Interface
DPWS	Device Profile Web Service
DSL	Domain Specific Language
EUI	Energy Use Intensity
EXI	Efficient XML Interchange
HTTP	Hypertext Transfer Protocol
IoE	Internet of Energy
IoS	Internet of Services
IoT	Internet of Things
IPSO	IP Smart Objects
JMEDS	Java Multi Edition DPWS Stack
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
KRI	Key Result Indicators
LED	Light Emitting Diode
LENI	Lighting Energy Numeric Indicator
LPD	Light Power Density
MAS	Multi-Agent System
OLED	Organic Light Emitting Diodes
PI	Performance Indicator
PLC	Power Line Communications
RDM	Remote Device Management
RFID	Radio Frequency Identification
RI	Result Indicator
RTU	Remote Terminal Unit
SOA	Service oriented Architecture
SOAP	Simple Object Access Protocol
THL	Temperature Humidity Light
WS	Web Service
WSDL	Web Services Description Language
XML	Extensible Markup Language
ZLL	ZigBee Light Link

1. INTRODUCTION

1.1. Background

Energy efficiency and renewable energy are becoming significantly important trends for research field lately, due to the global shortage in fossil fuels. Consequently reducing energy costs will be the main target of every business organization. The EU is targeting by 2020 to reduce energy consumption by 20%, increase renewable energy by 20%, and reduce CO2 emissions by 20% [1]. Lighting is the major sector consuming energy in buildings; it is estimated to be around 25% of total energy in US commercial buildings. On a global scale more than 19% of energy is consumed by lighting, this is considered to be more than energy produced by hydro or nuclear power plants and, equivalent to energy produced from natural gas [2].Thus drastic measures should be taken in order to save wasted lighting energy.

Smart lighting is considered the one of the main solutions for energy reduction by means of controlling lighting level according to desired need with minimum energy consumption. Smart-Lighting systems utilize motion and light sensors for performing the control algorithms, most of the reviewed solutions have shown superior results in energy reduction by 30-40%.

The Internet of Things (IoT) allows devices to be connected with the internet and, communicate and interact with each other. The IoT is connecting the physical world with the virtual one, by creating a digital entity for each object or device [3]. Addressing billions of devices will not be a problem anymore due to the IPv6 addressing protocol. Applying Smart-Lighting and Building Automation System (BAS) are one of the main challenges for the IoT. The numerous amounts of data exchanged and the interoperability between all devices and meters require a strong network infrastructure capable of holding huge bandwidth and a standardized protocol for communication.

This implementation will give a holistic view of the energy consumption and will aid the management to analyse and monitor energy data, as well as report results to the enterprise level. These reports will guide the enterprise to take decisions regarding its consumption behaviour and controlling methods.

1.2. Problem Definition

1.2.1. Justification of work

Lighting control systems are developed due to the immense need to reduce energy consumption and, provide adequate luminance level for working in industrial and commercial building. Different protocols are used for lighting objects to interact with each other and give real-time feedback, in order to achieve a distributed control system moreover, protocols used for energy metering.

The problems emphasis for huge complex networks of several devices of different vendors and protocols. Integrating all these devices in a heterogeneous network require specific gateways and translators which, require a lot of resources. In order to overcome this complexity, this thesis work tries to present a Service Oriented Architecture (SOA) lighting control solution, which will help to reduce the heterogeneity of the system and improve the interoperability.

1.2.2. Problem statement

"*Analyse the feasibility of IoT for controlling the lighting applications aiming to increase system efficiency.*"

1.3. Work description

The main objective of this thesis is to increase efficiency of lighting system by applying the IoT paradigm and using SOA. An autonomous distributed lighting control system will be developed. This system will reduce energy consumption by controlling the light intensity depending on motion, light and weather from surrounding environment.

The Smart lighting system will use web services (WS) for communicating, data monitoring and storing in the database for further analysis. Another objective is to define KPIs for analysing the results and, calculate the actual efficiency. A monitoring application will be developed in order to collect real-time data from the devices and visualize them in a simple interface for the end user together with the analysed data and KPIs.

1.3.1. Objectives

Considering the aforementioned description the objectives are defined as follows:

- Develop a control algorithm for controlling the system that reduces the energy consumption.
- Design the communication rules and messages between the IoT objects.
- Design the services needed to monitor and analyse the system.
- Design KPIs for energy efficiency analysis.
- Develop and design a dashboard application to visualize and monitor the results.
- Testing and modification for achieving higher efficiency.

1.3.2. Methodology

The thesis work starts with the research phase:

- Researching for smart-lighting applications, and studying their algorithms, benefits and limitations.
- Background study of the IoT, WS and DPWS.
- Investigate KPIs used for lighting systems and, design KPIs required for the implementation.

Then the implementation phase:

- Design and implement the control algorithm in the devices.
- Designing web-services and messages required for energy measurements and data collection.
- Designing the dashboard application needed for monitoring and visualization.
- Work will conclude with verifying the results and checking possibilities for achieving better efficiency.

1.3.3. Assumptions and limitations

The following assumptions and limitations were taken into account when making this thesis work:

- Geographical constrains affecting the lamps temperature and heat dissipation is not taken into consideration.
- During testing of the implementation it was assumed that the daylight duration is constant, although the difference in duration between different year seasons should be taken into account.
- The implementation only relates to outdoor lighting.
- It is assumed that all devices are calibrated and gives correct readings.

1.4. Thesis Outline

In the following chapter a theoretical background will present basics of smart-lighting, SOA and IoT in addition to some KPI implementations. Chapter 3 presents the methodology of work. In chapter 4 the actual implementation and control procedures of the smart-lighting are presented. Chapter 5 shows the validation techniques, used metrics and results. Chapter 6 presents conclusion and outlines future work.

2. THEORETICAL BACKGROUND

2.1. Historical Background on lighting technologies

In 1879 Thomas Edison invented the first commercial incandescent lamp. Later "General Electric" replaced the Carbon filament with tungsten for improving lifetime and withstanding high temperatures. More improvements were done later such as using inert gas and coiled filament [4].

Peter Cooper Hewitt patented the first mercury vapour lamp in 1900. Later on General Electric introduced low-voltage fluorescent lamps in 1930 for decorative purposes. These lamps where launched in the general lighting market in 1938 by GE and Westinghouse Electric Corporation. The main advantages of these lamps are prolonged life and efficiency. Finally fluorescent lamps became our main source for general lighting dominating the market until recent years when "Solid-State Lighting" started to emerge [4] [5].

In 1962 N. Holonyak successfully generated the first visible red light from a p-n junction as known right now by "Light emitting Diode" (LED) [6]. In 1994 S. Nakamura was able to produce white light from LEDs which is considered as a breakthrough, replacing regular lamps as a main source of lighting because of their low energy consumption and high brightness [5]. LEDs are now integrated in many of our daily applications such as cars, street lighting and, screens.

The next generation for lighting will be the Organic Light Emitting Diodes (OLED), which can provide high efficiency around 64lm/W, long life-time of 10,000 hours, full-colour, high brightness, wide view angle, small sickness and, flexibility [7]. Table 1 and Figure 1 show a comparison between the LED and OLED technologies. OLED is a newly born technology and still in the development phase, thus it didn't reach the development made for LEDS for more than 50 years.

Parameter	**OLED**	**LED**
Efficiency	Lower	Higher
Life Time	Lower	Higher
Cost	Higher	Lower
Design	Flexible	Rigid
Light Quality	Higher	Lower

Table 1, *Comparison between LED and OLED*

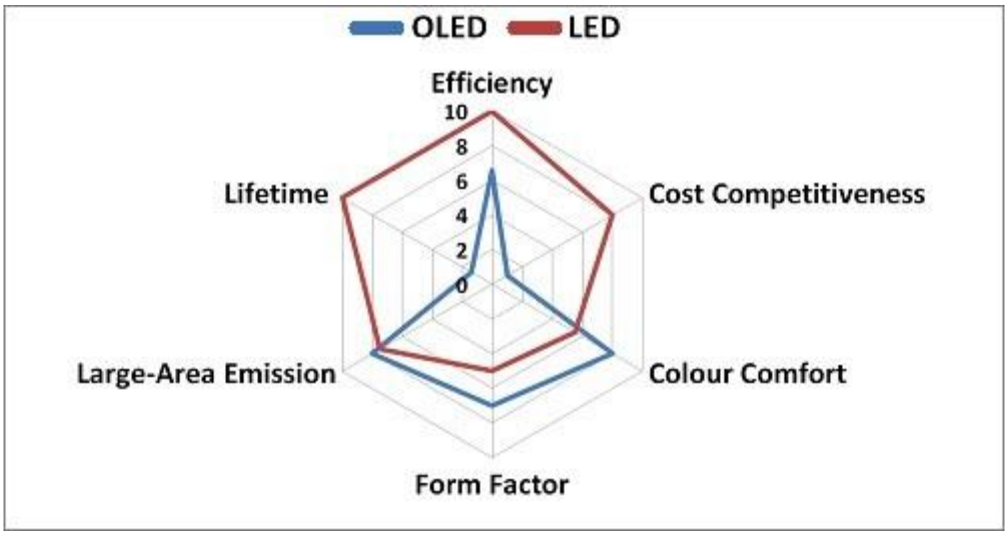


Figure 1, *Attributes of OLED and LED lighting* [8]

2.2. Smart Lighting

The terms "*Smart lighting*" or "*Intelligent lighting system*" is defined in paper [9] as a system where multiple lighting fixtures are connected to a network and, these fixtures cooperate to satisfy the user needs. Smart lighting is considered to be the next generation of standard lighting systems, as it will provide better performance, efficiency and sustainability.

Development of smart lighting is done on two phases [10]: "*First wave*" and "*Second wave*". The "*First wave*" stands for the replacement of standard bulbs with the new LED technology. The progress in this wave is done in the form of researching new technologies for reaching a better efficiency, cost reduction, lifetime and chip design for LED technology. The "*Second wave*" of development is concerned with the controllability, fast response and control communications of the LED, which will lead to revolutionizing the adaptive lighting to reach higher efficiency and user comfort.

The two main methods used for communication in lighting control are "Power Line Communications" (PLC) and "Wireless Sensors Network" (WSN). Although the wiring in PLC has been reduced significantly, it might still face some problems caused by noise attenuation. Therefore research is more directed towards WSN, especially ZigBee for lighting control [11]. The comparison of the two technologies was done in paper [12] and it is demonstrated in Table 2.

Advantages		**Disadvantages**	
WSN	**PLC**	**WSN**	**PLC**
WSN devices cost less than PLC devices.	No obstacles in the communication path.	WSN is dependent on the environment that could affect changes in the communication path.	High cost of devices compared to WSN devices.

Highly efficient for sink node applications and scalability that enables better control over the network and maintains a high performance levels.	PLC does not require installing additional wires or signal repeaters.	Installation of repeaters devices in order to avoid obstacles.	A part of the network may remain isolated in case of a short circuit. Low resistance to interference/noise and impediments in transition of signal through transformers.
Integrate a high number of nodes (over 500) and are redundant communication paths.	PLC is not affected by wireless interference.	WSN are affected by devices that function within the same band (Wi-Fi, Bluetooth).	Incorporates a low number of nodes (under 70) and has limited communication distance (300-350m).

Table 2, *WSN vs. PLC* [12]

2.2.1. Outdoor implementations

Many implementations have been done for smart lighting in outdoor environments. In paper [13] an approach for controlling street lights by means of WSN was introduced.

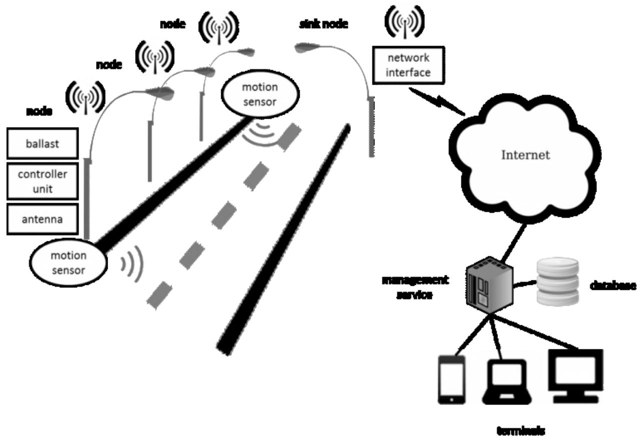


Figure 2, *Concept of the remote management and monitoring system* [13]

The basic concept of this implementation is maintaining lighting level at minimum until a motion is detected via wireless sensors. In case of motion detection the lights are lighting level is increased to maintain good visibility for passers-by for a given amount of time. The wireless nodes communicate using 6LoWPAN protocol, which provides reliable communication, low power consumption and, also it does not define routing protocols. The project presented in paper [13] presents the results for testing for 1 month; comparing the results of the managed energy to 100% and 60%

consumption Figure 3. The system has achieved 37% energy reduction by traffic based control of lighting, without causing problems or affecting the visibility of passers-by. In addition the system is capable to be integrated in any lighting system in the future.

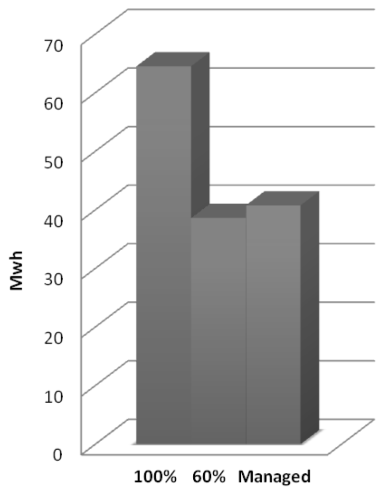


Figure 3, *Annual overall energy consumption* [13]

Paper [11] presents a different approach for applying outdoor street lighting control based on mesh topology of ZigBee nodes, with the aim of reducing energy consumption and achieving user satisfaction. The system used a gateway to convert ZigBee to standard TCP/IP protocol for interpreting the data on the server side. Monitoring of the system was based on 3 main parameters: lamp temperature, power consumption and luminance level. Each lamp has its own controller, that is connected to the sensors also has pulse width modulation (PWM) for controlling the LED lamps. A prototype was done for the network with software used for monitoring and controlling. The system might be expanded or integrated with a traffic management system. The results were satisfactory providing energy minimization also, positively affecting the user adaption and, environment.

2.2.2. Control Protocols

Protocol is defined from previous studies and readings, as “a set of predefined rules as syntax and scheduling that allows communication between two entities and, it improves interoperability between different devices”. Standardization of digital and, analogue protocols used for lighting control was carried out by many private research groups. Here is a brief overview of these protocols:

0-10V:

The demand for remote control of lighting by the entertainment industry has led to the emergence of 0-10v protocol which was standardized in [14]. This protocol is the simplest way for controlling light where, 0V is for minimum level and 10v is for maximum level. The main advantages of this protocol are that it is simple, safe due to low voltage and compatibility with most of control devices. On the other hand the disadvantages are that it needs hard wiring for each dimmer, voltage drop over long distances and, low voltages range near zero could not be achieved due to diode forward drop [15] [16].

Analogue Multiplexing (AMX192):

This protocol is based on having a paired wire; one wire carries synchronizing clock signal while the other pair carries the multiplexed analogue signal 0-5V. The main advantage of this protocol is eliminating wiring a cable for each dimmer instead, with one control signal 192 dimmers can be controlled, this is duo to 50μs multiplexing. But it is still considered as an analogue signal therefore it is becoming absolute [15].

Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories (DMX512):

As digital era dominated the electronics sector all analogue methods became absolute by time, interoperability was a critical issue in using digital control where DMX512 was trying to solve. DMX512 is based on EIA-RS-485 specification and, it can address 512 dimmer with 8 bits giving 256 lighting level [15]. DMX512 can support 32 physical devices with 4000 feet (1,219.2m), where one device acts as master sending the control signal while; the others are slave receiving the signal [17]. DMX512 dominated the entertainment industry because of its main advantage of is the interoperability across different manufacturers; moreover it was used for effects control even though it was not designed for it. DMX512 has also few drawbacks such that it is a unidirectional, resolution of 256 levels is not smooth, deficient in error correction, hard addressing using switches and, it is stated in the specifications that in practical the range should be less than the 1,200 meters [15].

Remote Device Management (RDM):

RDM is an extension for the DMX512 to support bidirectional communication for monitoring, discovery, configuration and address setting. The limitation of RDM is that it can only support a network of 32 devices [16].

Architecture for Control Networks (ACN):

ACN is a suit of 17 protocols used for theatrical lighting, audio, video and, effects control. ACN can be used over a variety of networks, including Ethernet. The main features of ACN are bidirectional, sequencing and reliability. ACN has unlimited number of devices that could be connected in the network [16].

Building Automation and Control Networks (BACnet):

Originally BACnet was developed to solve interoperability of heating, ventilating, air-conditioning and refrigeration equipment in building automation environment, later on an addendum was proposed to include fire safety and security. BACnet uses objects for modelling devices in the control system, where each object has a set of properties to describe it [15]. BACnet objects use services such as Who-is, Who-Has and, I-Am to facilitate discovery and integration of devices [18]. Objects used to model lighting devices are Binary Output Object (BO) representing relays or Analogue Output Object (AO) representing dimming lights, also a new Lighting Output Object is being developed specifically for lighting control [16].

The main advantages of BACnet are scheduling, monitoring, load management and, it provides a top level management of the controlled environment. BACnet contain a device ID for each device that can hold up to 4 million devices but, the limitation of BACnet/IP is depending on the bandwidth and total size of data being transmitted while, for BACnet MS/TP the protocol restricts 127 MAC address per segment and 32 device because of power consumption reasons [16].

Local Operation Network (LonWorks):

LonWorks is a network platform for control applications of HVAC and lighting developed by Echelon Corporation. LonWorks is based on peer to peer communication between embedded devices called neuron cores. Each neuron core contains two processors for communicating and, one for the control process of the node. LonWorks has a benefit of decentralized control so; it is not Susceptible to one point failure if one of the nodes is unreachable. The restriction of this platform is that it can hold up to 256 groups where, each group can have up to 64 nodes with 2 addresses for each node, this make it possible for 32,000 devices per network [16].

ZigBee Light Link (ZLL):

ZigBee is considered as one of the most commonly used wireless protocols for small and low power embedded devices. ZigBee can be connected in many topologies as mesh, star and, peer to peer. The main reasons for using ZigBee are easy to install, cheap low power consumption, does not require wiring and, can be easily integrated with other protocols as TCP/IP. The problems facing ZigBee are only limited bandwidth is used, due to low power and operating on battery also, short range of 75m outdoors which can be lower indoors due to walls and barriers [16].

Digital Addressable Lighting Interface (DALI):

The DMX protocol was targeting the entertainment business alternatively; DALI is an open protocol developed as an extension to IEC60929, for architectural and commercial lights. The main components in DALI network are controller or more, sensors and, lighting loads, where each lamp has its own addressable ballast as shown in Figure 4. DALI network require only one control cable that allows bus and star topologies [15].

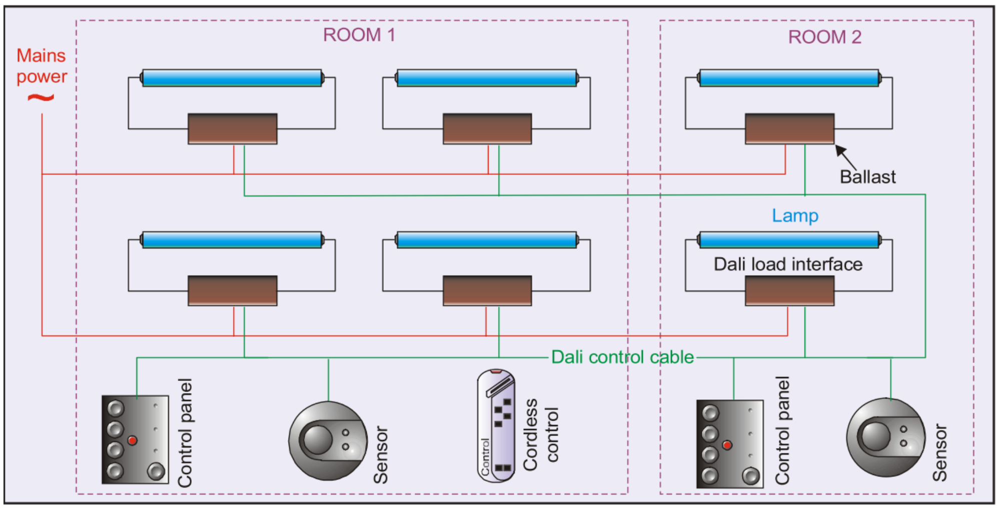


Figure 4, *Example of DALI Network* [15]

DALI protocol is bidirectional this allows monitoring by querying about lamps status, current level and failure. There are two types of messages; forward message frame and, backward message frame. Forward frame consists of 19 bits; 1 start bit, 1 address byte(1 individual or group address bit, 6 address bits, and 1 select bit), 1 data byte and, 2 stop bits as in Figure 5 ,while the backward frame consists of 11bits; 1 start bit, 1 data byte and, 2 stop bits as in Figure 5 [19].

Forward frame:

Start	Y	A	A	A	A	A	A	S	X	X	X	X	X	X	X	X	I	I

Backward frame:

Start	X	X	X	X	X	X	X	X	I	I

Figure 5, *DALI forward and backward frames*

There are 3 ways for addressing in DALI commands [19]:

- Broadcast: If a broadcast command is sent to the DALI network all connected ballasts should respond at the same time.
- Group Addressing: DALI network might be divided up to 16 groups in order to ease control of rooms or sections. Configuration is prerequisite to set groups which the ballast belongs to.
- Individual Addressing: Each ballast in the DALI network has its own unique address. A network can have up to 64 devices; this is due to 6bits of address. Individual addressing is the most commonly used.

The main advantages that DALI offers are; simple wiring, different addressing methods, no interference from the power line, bidirectional for querying lamp status, automatic discovery, logarithmic dimming for human eye’s sensitivity, built-in stored scene in the ballast, control over fading and dimming time, [20] off control signal which is not available in 0-10V, Manchester encoding this gives the advantage of 0 D.C. value of the signal and synchronization of the clock [15]. The only limitation of DALI is having only 64 devices per loop but, currently a new digital protocol is being developed to work simultaneously with DALI and allow 64 controllers to be added to the DALI bus [16].

2.3. KPIs

A general definition of Key Performance Indicator (KPI) is given in [21] as "*a set of measures focusing on those aspects of organizational performance that are the most critical for the current and future success of the organization*". Another more precise definition for KPI specifically for energy performance stated in paper [22], "*KPIs are accessible numeric metrics of energy usage or observed building characteristics that can be associated with better or worse than expected energy performance*".

Most people confuse between KPI and performance indicator (PI), which is defined in [23] as "*a variable that expresses quantitatively the effectiveness or efficiency or both, of a part of or a whole process, or system, against a given norm or target*". The four types of indicators mentioned in [21] are Key Result Indicators (KRI), Result Indicator (RI), Performance Indicator (PI) and Key Performance Indicator (KPI), they are used to measure or assess performance of any system. KRI are similar to the outer shell of the system as it gives a general idea of what has been done regarding critical issues. RI and PI go deeper and give more information about the system, describing what have been done and what should be done. KPI are considered as the core giving precise information about what should be done to increase the performance dramatically.

2.3.1. KPI Implementation for Energy Assessment

There are many implementations done for applying energy KPIs, because improving the energy performance will result in enormous savings and huge impact on the environment. The report [24] introduces a number of metrics, which can be used to assess the energy consumption of indoor lighting such as: Code Baseline Lighting Energy, Code Baseline LPD, Day-lighting and Occupancy Energy Savings, Lighting Energy Cost and, many more. The report gives a detailed description for each metric and how to calculate it also; it provides a form and guidelines that can be filled for any project to assess its energy savings.

Lighting Energy Numeric Indicator (LENI) is an index introduced in [25] to estimate the total lighting energy required by a building in one year and, it is calculated as in equation 2.1. The benefit of this index lies in comparison of energy consumption of buildings of different sizes regarding that they are from the same sector or having same function. LENI number was used in [26] to estimate the energy efficiency of a smart lighting implemented in classrooms. It was observed that LENI number decreases significantly when applying control actions for energy reduction. A similar KPI in English units can be found in [27], named "Energy Use Intensity" (EUI) and, has a unit of KBtu/ft^2.

$$\mathrm{LENI} = \mathrm{W/A}\,[\mathrm{kWh}/(\mathrm{m}^2 \times \mathrm{year})] \qquad (2.1)$$

Another implementation presented in [28] for energy KPI some metrics were defined for total plant energy assessment. The KPIs were the monthly Million British

Thermal Unit (MBTU) and MBTU/Ton. These metrics will include electrical and gas energy consumption. Energy data was collected from utility bills, shadow metering and sub-meters. A target reference was made 20% less consumption and, data were compared to this target value and results were presented. Having an EMS contributes significantly in cutting costs.

In [29] KPIs for small commercial buildings where defined for Designers, Operators, and Occupants. Lighting, gas and HVAC energy consumption were considered for calculation of the building KPIs. Defined KPIs included but not only limited to: Schedule Visualized EUI, Daylight Effectiveness Indicator and Overall Lighting Performance Indicator.

The Schedule Visualized EUI indicator shown in Figure 6 presents in a simple and visualized way the contribution of each system in the total consumption density, together with the total hours being active per day. The Daylight Effectiveness Indicator shows the lighting energy consumption density co-plotted versus the night length in hours, as shown in Figure 7.

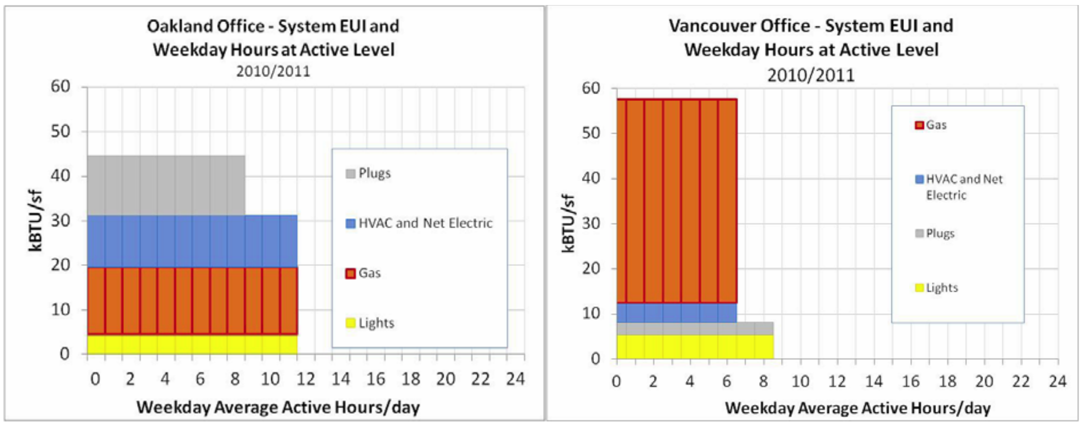


Figure 6, *Schedule Visualized EUI Indicator Results* [29]

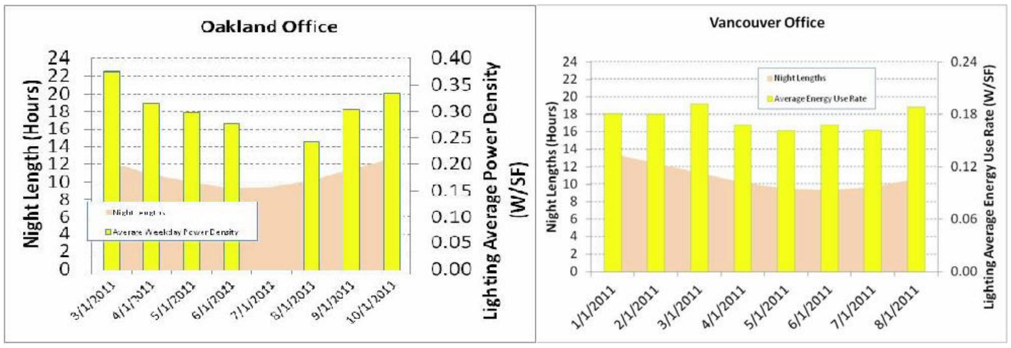


Figure 7, *Daylight Effectiveness Indicator Results* [29]

2.4. Internet of Things (IoT)

The term "*Internet of Things*" was brought up for the first time in 1999 by Kevin Ashton in a presentation for "Procter and Gamble" introducing the use of RFIDs in supply chain [30]. IoT is defined in [3] as a network infrastructure with self-configuring capabilities. Another definition found in [31] as: "*A global network which allows the communication between anything in the world by providing a unique digital identity to* *each and every object.*” IoT tends to connect the physical and virtual objects. These objects are smart so that they can configure themselves and operate without any human assistance.

The applications for IoT are divided into 3 domains; industry, environment and society. These 3 domains are not totally isolated but, they are interfering in some aspects. There are a lot of attempts to apply IoT paradigm in applications like climate monitoring, health care, home automation, traffic monitoring and, supply chain [3].

2.4.1. IoT Implementations for Energy Awareness

State and private research groups are trying actively to bring IoT in our daily activities. An urban lighting model described in [32] was presented in the “International science park” in china implementing the IoT. The model represents the IoT in 5 main layers: objects layer, sensing layer, perceptron layer, transport layer, and information layer as in Figure 8. The object layer is subcategorized to moving objects, stationary objects and area. Each of these subcategories is divided to several kinds and has certain arguments identifying the object parameters. The sensing layer is responsible for converting all analogue signals such as temperature, light and motion into digital signal. Perceptron layer provide information about objects from RF tags which is then sent to the transport nodes. The transport layer then sends this information to the control layer where the control algorithm and processing of information takes place. The information layer can discover, delete and configure objects; moreover it can apply control action on lights. The power consumption for each individual lamp is measured and analysed in the control centre.

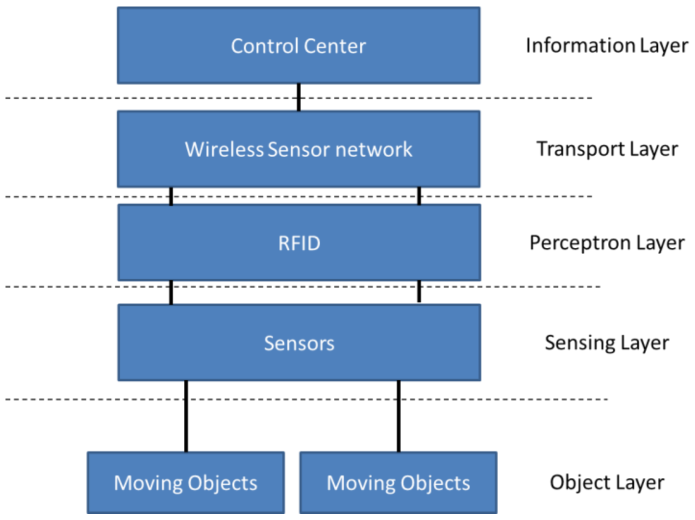


***Figure 8**, IoT model adapted from* **[32]**

Another application for IoT in building energy consumption monitoring is described in [33]. The application defines the IoT structure in 3 levels as shown in Figure 9. The collection level consists of smart-meters with RS-485 port and collectors for gathering energy consumption data. Each collector can connect up to 128 meters. Afterwards the data are converted in the transport level using translators to TCP/IP to enable flexibility and reliable connection. After that the data are forwarded to the management level via network switches, where data is analysed and stored for real-time monitoring and mapping of the data to an energy map. Old meters models that do not have digital connectivity can be integrated as well. Meter readings are visually detected and converted into digital signal sent through RS-485.

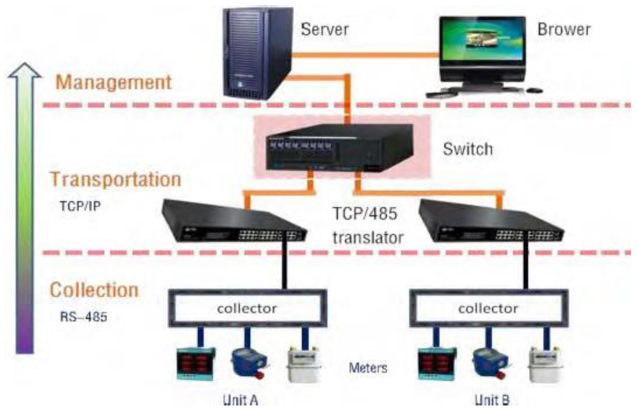


Figure 9, *Energy Management System Construction* [33]

2.4.2. IoT challenges

The IoT is newly born and still not completely developed, thus there are still some obstacles and challenges for the broaden use of IoT in different fields. There is no orchestration between research approaches of the US, EU and, China. If collaboration between these approaches is applied, it will lead to faster growth and standardized protocols. The IoT is facing other problems such as security, privacy and ethics. If devices are hacked, they can provide private information and life style habits of users [34]. In addition manufacturers still do not share their knowledge or access to control since there is no any financial motivations exist for doing so [35].

Another technological challenge is standardization of communication between devices. There will be some limitations for adoption of IP in the IoT. IPSO suggests that IP will be integrated in every smart device but, this will be a problem for real time applications because small and cheap devices cannot handle complex protocol such as IP. Also different field applications does not have speak same language, such medical, industrial and, home appliances [34].

If the research of IoT continued in the same trend and attitude, it will lead in having huge amount of data trapped in huge storages. These storages eventually will have to be connected by certain protocols but, it may also cause network congestion, slow communication and security issues. However, the proper way would be having “Internet of Islands” where, all devices in a certain space can communicate together then with the community around it. Unfortunately the research trend does not support this vision [35]. There is an urgent call for divergence in the research methodologies towards the global picture of researching the IoT and the possible collapses that might happen, instead of being sluggish and instrumental [34].

2.5. Service Oriented Architecture

Service Oriented Architecture (SOA) is considered the state-of-the-art standard used to facilitate the integration of different software components providing services. These services can be published, described and, discovered autonomously in order to develop fast, reliable and, interoperable application. [36] Web services (WS) are the most common way for realization of SOA.

2.5.1. Web-Services

WS are defined in [37] as "*a software system designed to support interoperable machine-to-machine interaction over a network*". There is a service provider and consumer entities that exchange messages, these entities can be a human, machine or an enterprise. The provider entity describes the services that it can provide and the operations within these services in Web Service Description Language (WSDL). The service consumer searches for the required service by certain criteria. After the provider and consumer agree on the WS description and semantics they start exchanging messages in SOAP format.

There are three main discovery mechanisms, in order for the provider and requester to be known to each other:

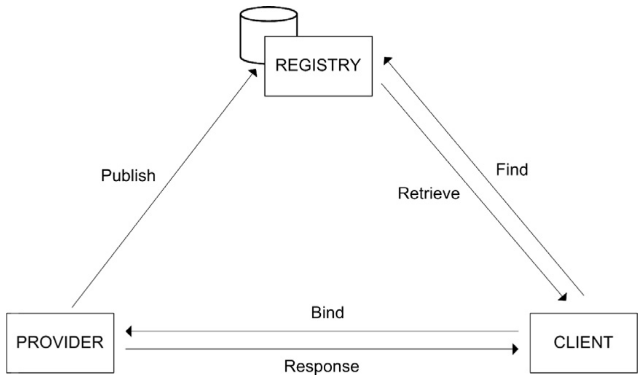


Figure 10, *Registry discovery method* [36]

- **<u>Registry:</u>** The registry holds a list of all available services and their description and, it is implemented using Universal Description, Discovery and Integration (UDDI) standard. Provider should publish a description of his services to the registry shown in Figure 10. The registry acts as a broker providing introductory service to both parties.
- **<u>Index:</u>** The provider provides the description to the services publically available on the web. Any entity can implement its own index and, have the flexibility of choosing which services to include. The publisher has no interaction with the index providers.
- **<u>Peer 2 Peer:</u>** The P2P provides a way for services to discover each other dynamically without any intermediate point. The requester quires its neighbours in the network for the service. If the neighbour has the service he will reply back

if not he will propagate the query to the neighbours and so on until a match is found.

Comparing the P2P to the Registry methods, The P2P method doesn’t have single point failure such as Registry method however, it has some drawbacks regarding performance and not guaranteeing the propagation of the path, in addition to the overheads required by each node even if it is not involved in the service exchange. For dynamic application where propagation is limited it usually preferred to use P2P system while, centralized registry is used for systems where it is more statically so that services does not vary frequently. [37]

The advantages of applying a service oriented applications are enumerated in [38] as: integration, agility, flexibility, re-configurability, cost reduction, complexity encapsulation and fault tolerance.

2.5.1.1 SOAP

SOAP stands for Simple Object Access Protocol but however, this definition was obsolete since SOAP 1.2, later it was substituted with Service Oriented Architecture Protocol. SOAP is a message format used for exchanging message in a distributed and decentralized way and it is the main messaging protocol for WS. SOAP is based on XML which make it platform and language independent and easily extensible. SOAP provides a messaging framework with reliability, security, correlation, routing and message exchange pattern features.

The SOAP message structure contains SOAP-Envelope, SOAP-Header and SOAP-Body shown in Figure 11 and elements are described in Table 3 [39].

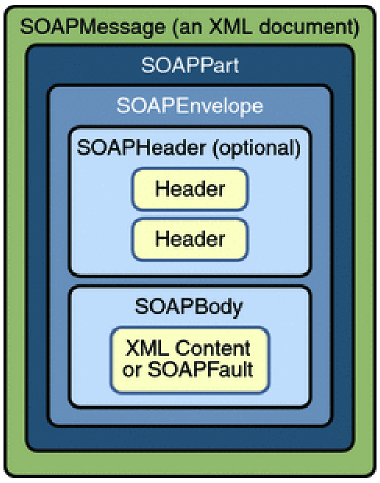


Figure 11, *SOAP Message Structure* [40]

Element Name	Description
SOAP Envelope	The envelope defines the boundaries of the SOAP message and it is the root element that identifies a SOAP message. It contains the xmlns namespace and encoding style attribute.
SOAP Header	The header contains information about the SOAP message. It must have namespace property. It may have any attributes of the following encodingStyle, role, mustUnderstand or relay.

SOAP Body	Containing the actual data required to be transmitted.
SOAP Fault	Might be used to describe message errors. It contains faultcode, faultstring, faultactor and details elements to describe the fault.

Table 3, *SOAP Message Elements*

2.5.1.2 WSDL

WSDL stands for Web Services Description Language and, it is an XML format based language for defining the web-service. It is simple so it can be processed by human or machine. WSDL is used to discover services and operations by clients, once the WSDL is agreed upon the exchange of message initialized. [41]

There are two versions of WSDL commonly used 1.1 and 2.0. The difference between them is that, in WSDL 2.0 there is no message component the messages are referenced from the types directly, in addition to that PortTypes are called interface in WSDL 2.0. [38] In Figure 12 shows WSDL 1.1 structure components described in Table 4: types, message, portType, binding and service.

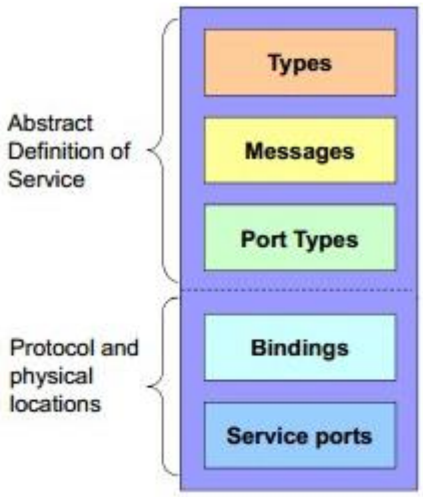

Abstract Definition of Service
Types
Messages
Port Types
Protocol and physical locations
Bindings
Service ports

Figure 12, *WSDL 1.0 Structure* [42]

Component	Description
Types	Describe the data type that will be transmitted in a schema format
Messages	List the message names that will be consumed by the service. Message elements are referenced from the type element.
PortType	List the operations that can be used by the service. Inside the operation input and output messages are defined.
Binding	This component defines the message protocols and binding, how it will be transported in the network interface.
ServicePorts	Defines the services and ports where the service is located in an URL string.

Table 4, *WSDL Components*

2.5.2. DPWS

DPWS stands for Device Profile Web Service is a profile developed for implementing WS on device level and achieves interoperability between different devices from different vendors. The profile was made so as to set minimum set of WS specifications for implementing secure messaging, dynamic discovery, description, and eventing. DPWS consists of WS-Discovery, WS-Eventing, WS-MetadataExchange, WS-Policy, WS-Transfer, and WSAddressing (WSA) The device contains meta-data about the device and the services offered. [43]

The main features that DPWS offers [44]:

- Device discovery and the services offered.
- Exchange of messages with devices.
- Services are described in a WSDL file.
- Service interaction.
- Event Subscription from WS.

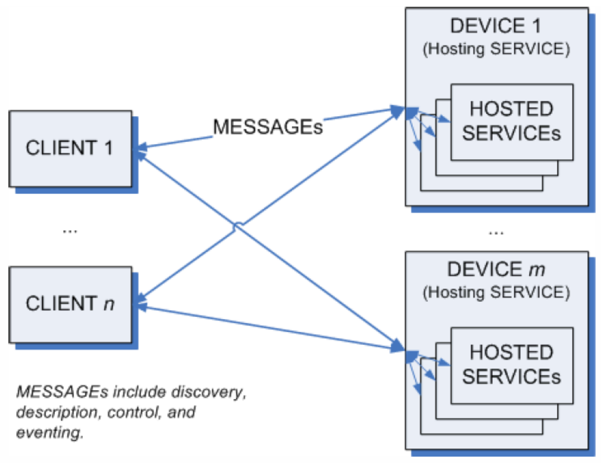


Figure 13, *DPWS Devices discovery and services* [43]

There are several tools developed for implementing DPWS clients in order to discover devices and consume services such as: Web Services on Devices API (WSDAPI) and Java Multi-Edition DPWS Stack (JMEDS).

In [45] an approach for using DPWS smart-metering system was proposed, in order to achieve heterogeneity and interoperability between metering devices in a smart-grid. Mapping between Smart Message Language (SML) and DPWS was conducted to make a comparison between protocols. Calculating the overheads of DPWS was much higher than the SML as expected. The overheads are not a problem for broadband internet but, in case of Power Line Communication (PLC) compression should be taken into consideration. The paper suggested Constrained Application Protocol (CoAP) for HTTP header compression and Efficient XML Interchange (EXI) for XML compression; overheads were reduced by 90%. Security requirements were satisfied by DPWS as it has already WS-security integrated. The paper succeeded in presenting the advantages of DPWS for smart-metering application.

From another aspect [46] was testing the implementation of DPWS on several metering devices and simulating 5000-10000 devices working simultaneously. The timing behaviours of the meters were analysed and the results showed that one DPWS meter can provide its service to thousands of devices simultaneously but as the number of subscriptions increase the processing capabilities of the device should increase.

2.5.3. Integration of SOA in the IoT

The integration of web services in the IoT became a significant challenge for the research field for the emergence of the next generation of internet. This integration will facilitate the homogeneity and interoperability of the network. The combination of contributions to the IoT and Internet of Services (IoS) will play an important role in the development of the Internet of Energy (IoE) [46] presented in Figure 14.

SOCRADES architecture was introduced in [47] for effective integration of IoT in enterprise services. This architecture eliminates the heterogeneity between devices hardware and software, communication protocols and data formats in the form of services. A middleware was proposed in [48] based on SOA and Multi-Agent System (MAS) to emphasis the collaboration, interaction and homogeneity of devices of different protocols and vendors in the network.

Despite the wide spread of SOA, still it cannot be used outside the boundaries of the enterprise. The term "*Internet of Services*" (IoS) is introduced in [49] as a combination of the Web 2.0 and SOA technologies. This combination will lead to the next step for internet information where, users and companies can use these interoperable services at the same time seamlessly.

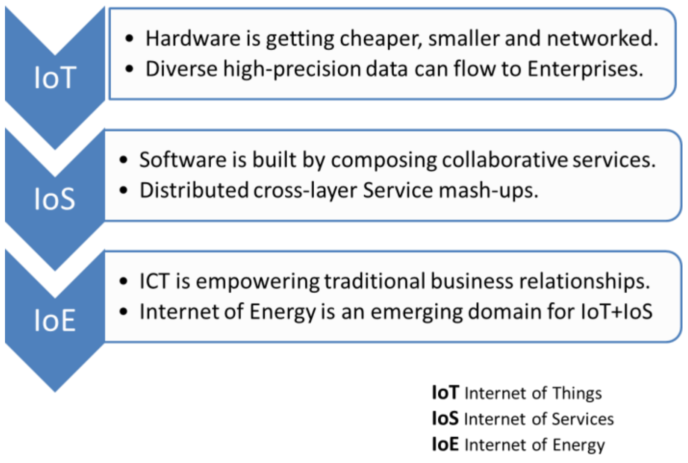

IoT
• Hardware is getting cheaper, smaller and networked.
• Diverse high-precision data can flow to Enterprises.
IoS
• Software is built by composing collaborative services.
• Distributed cross-layer Service mash-ups.
IoE
• ICT is empowering traditional business relationships.
• Internet of Energy is an emerging domain for IoT+IoS
IoT Internet of Things
IoS Internet of Services
IoE Internet of Energy

Figure 14, *Relation between the IoE, IoS and IoT adapted from* [46]

3. METHODOLOGY

Several implementations for the IoT were presented in the background section. Studying of these implementations resulted in the development of the architecture shown in Figure 15. This section describes the technologies and tools used in the development of the architecture such as, service discovery and subscription tools and, web development toolkits and frameworks.

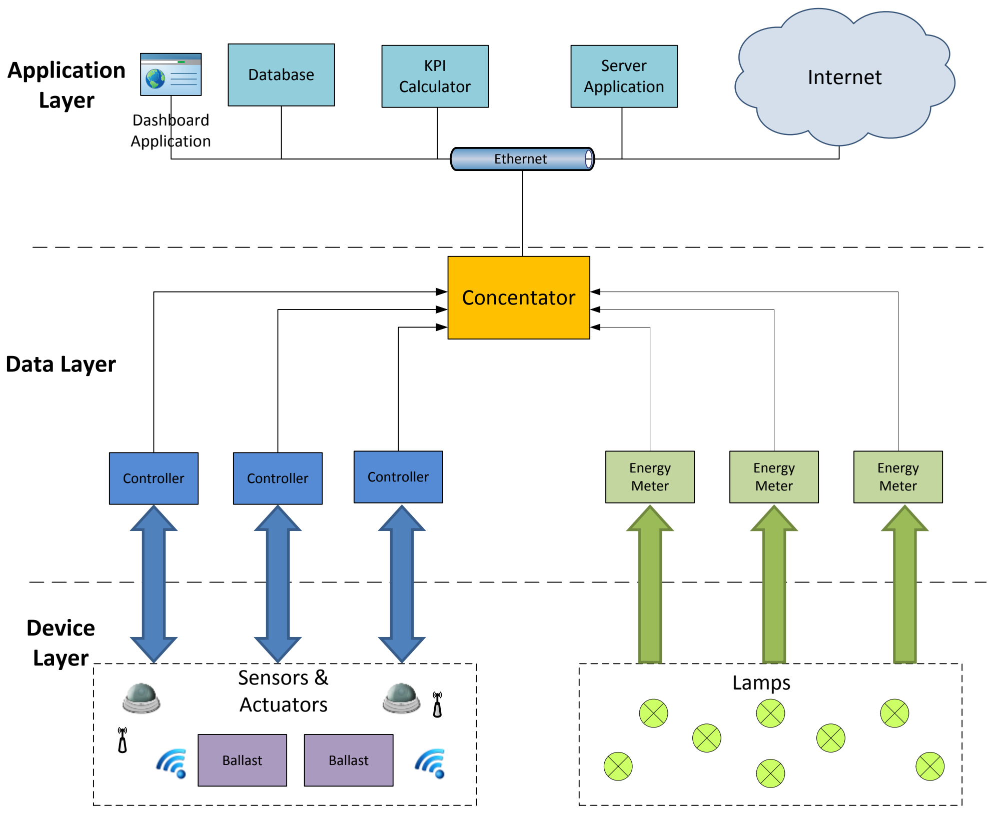


Figure 15, *Smart Lighting IoT Architecture adapted from* [50]

Energy analysers and Main controller devices are continuously sending messages containing energy and system data. The "Event-hub" is a java application that acts as a concentrator, subscribing to all events collecting messages and forwarding them to the application server, database and, any other interested monitoring application. The application server is hosting a web application that receives message parse them and display them in a visualized dashboard. The application acquires historical data and KPI data from the database and KPI-service respectively via web-services.

3.1. JMEDS v2.0

The "*Java Multi Edition DPWS Stack*" (JMEDS) framework facilitates the implementation of web-services based on DPWS in Java environment. It enables WS-messaging, WS-discovery and eventing. The "Dashboard Application Server" utilizes JMEDS framework, by creating a client that is capable of device and service discovery. This client is used for sending parameters required for the control by invoking certain operations. The JMEDS architecture divides the client and server side into three layers: Application, Dispatching and Communication layer as shown in Figure 16.

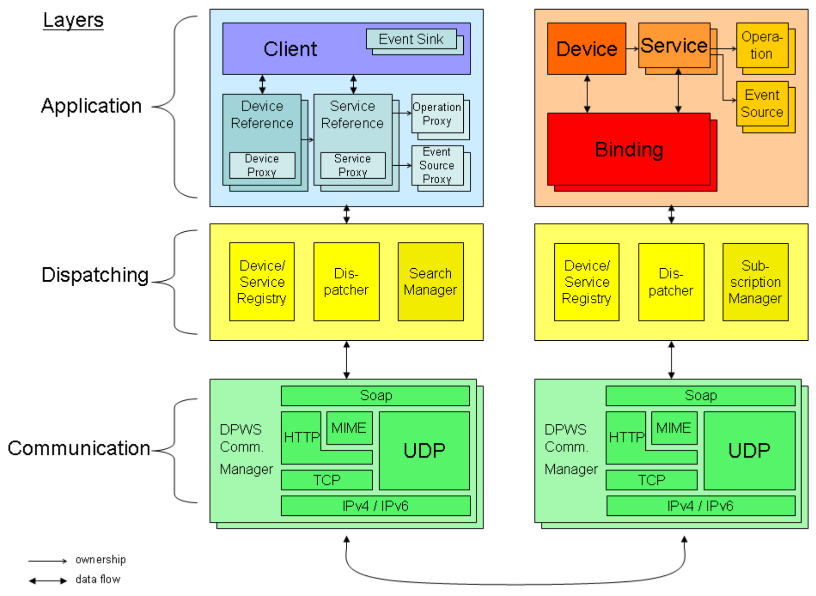


***Figure 16**, JMEDS architecture* [51]

The communication layer handles low level communication technologies through different communication manager for each technology, which facilitates migration of technology with different manager so that the developer does not have to make changes to upper layers. The communication manager enables the exchange of SOAP messages via TCP or UDP, besides it has IPv4 and IPv6 capabilities.

The Dispatching layer is very closely related to the communication layer. It is consisting of three main parts. A search manager in case of client that is responsible for discovering devices and services on the other hand, the device has a subscription manager which manages event subscribers. The dispatcher is responsible for message refactoring before handling them to the communication manager. The device/service registry stores and indexes all known devices and services, this facilitate for the client to fetch for a specific device instead of using the discovery method each time.

The actual implementation of the clients, devices and services is done in the application layer. The client has an event-sink for receiving events, while devices hold the services offer operations and events. [51]

Also a DPWS-client is implemented in the Event Hub application, which searches for the energy meters and subscribes to the energy services they are offering.

Events are collected in the event-sink, which is then forwarded to different end points via Camel framework that will be described in the following section.

3.2. Camel Framework

Camel is an open source framework developed to facilitate the integration between different projects in the enterprise. The framework is based on a routing engine for routing messages between different endpoints. Camel is based on “CamelContext” which is a runtime object containing several components and routes. The components are endpoint factories producing endpoint instances, while routes are the messages path through different processing points until it reach a final destination. Routes can be defined either by Routing Domain Specific Language (DSL) or an Xml Configuration. A Camel endpoint acts as URI or URL of a web application which can be communicated with, either by; producing or consuming messages. [52] [53]

In the implementation the “Event-Hub” utilizes Camel context creating an event-sink endpoint that pre-processes the messages and re-routes them to different applications. Likewise the application server implements an end point for receiving messages from the hub for parsing and processing them.

3.3. Spring Framework

Spring is a Java development framework offering several modules to simplify and facilitate application development according to the need. Spring is considered as the core for the monitoring application running on the server in this implementation. The main benefits of using spring is the Dependency Injection (DI), Model View Controller (MVC), Security and, many other useful features but not relevant to this implementation.

Spring offers an easy way for declaring beans in an XML format and injecting them in any class or controller needed through annotations. Then Spring handles the bean creation, initialization and lifecycle until the bean destruction. In this implementation important beans containing essential energy and weather data are declared in the application-context xml, these data are updated automatically whenever a new message is received in the camel end-point. These beans are injected in the controllers that are responsible for handling user requests.

Spring MVC is an essential tool for web development; its work flow is shown in Figure 17. Servlets are declared in an XML file and their mapping. When the user makes a request it first reaches the Dispatcher-Servlet, which refers to the handler-mapping to know the request destination controller. At that moment the request reaches the controller that is responsible for arranging the data in the required format which is then sent back to the servlet with the view-resolver name. The view-resolver is responsible for mapping the view name to a certain view. [54] In this implementation the Dispatcher-Servlet is receiving URL and AJAX requests and forwarding them to the appropriate controller.

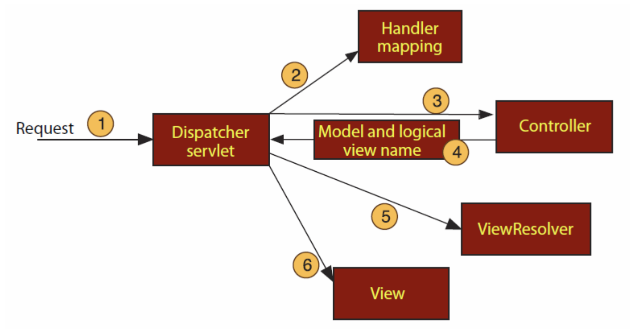


Figure 17, *Workflow of Spring MVC* [54]

3.4. Yahoo Weather Java API

This API is an apache licenced library used to facilitate accessing the Yahoo RSS feed service in simple Java classes. The library is based on Yahoo API documentation [55]. It is easier to use this API because by just passing the city code and degrees unit parameters to the “getForcast” method you can access all weather data without having to do a GET request and parse the response.

4. IMPLEMENTATION

The Smart Lighting implementation is located in the Factory Automation Systems and Technology laboratory at Tampere University of Technology while, the lamps are used outdoors to illuminate the backyard of the laboratory. The system uses motion and light sensors for detecting the surrounding environment. There are 6 lamps controlled with the specific lighting level in order to supply the adequate amount of lighting required without affecting the user visibility. The lamps locations are chosen such that they cover the whole backyard. The backyard and lamps fixation and numbering is shown in Figure 18. The objective of this implementation is to present a use case for the IoT objects collaborating in order to create a smart lighting environment capable of saving energy.

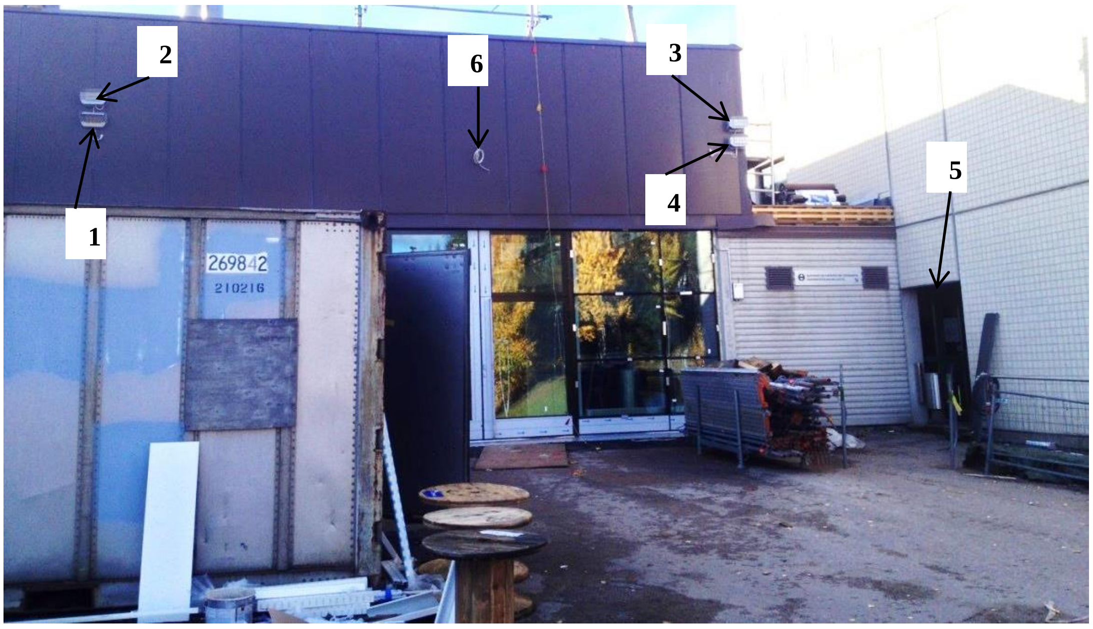


Figure 18, *Backyard view and Lamps fixation*

4.1. Operating Scenarios

Certainly the required lighting level is strongly dependent on the weather conditions. In clear weather at night might require more luminance than cloudy one, due to the reflection from the clouds. While during mist and foggy weathers require the highest possible lighting level, as the visibility reaches its lowest. On snowy weather it might require an intermediate level between clear and foggy. During night it requires high lighting levels, while at day it needs just fade level to provide guidance or turn off if the weather is clear. The lighting concentration in the yard is affected by the above conditions, this section describe several possible scenarios and the required lighting level and coverage.

4.1.1. Scenario 1

When an employee or a student goes outside of the building, he needs guidance until he gets out of the backyard area and reaches his car. The person is detected as soon as he gets out of the door by the motion sensor. A flag will be triggered in the controller, which will consequently brighten up the lamps near the door (lamps 3, 4, 5 and 6). These lamps provide the best coverage for the walking passage as shown in Figure 19. This flag will have wait time 20 seconds until the motion sensor is scanned again. If no presence is detected the flag goes down and lights dim to its initial value for safety reasons.

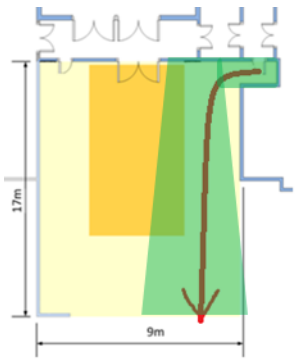


Figure 19, *Scenario1 Motion and Covered Area*

4.1.2. Scenario 2

When a person is detected coming from the parking area, he will be within the range of the motion sensor at about 18m distance. The coverage area is similar to scenario1 but the motion is in the opposite direction. Same lamps (lamps 3, 4, 5 and 6) will brighten up waiting for no presence to dim back again.

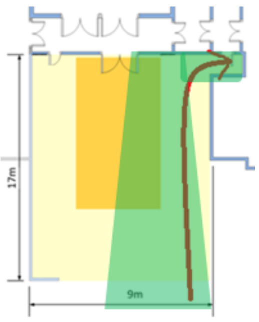


Figure 20, *Scenario2 Motion and Covered Area*

4.1.3. Scenario 3

The operator presses the truck “loading/unloading” button, to trigger the loading mode. All lamps except the door lamp (lamps 1, 2, 3, 4 and 6) will brighten up to 60%.

This combination provides the best coverage for the working area shown in Figure 21. When the motion sensor detects a presence in addition to the truck loading mode, in this case the previously mentioned lamps will brighten up to their highest levels to provide the operators with the adequate lighting level for working. When the operation is finished the operator turns off the “loading/unloading’ button. The lamps will return to a lower level, until there is no presence detected the lamps will return to their 40% minimum level.

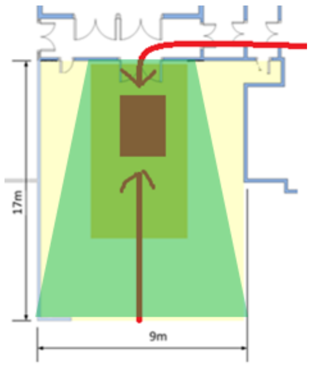


Figure 21, *Scenario3 Motion and Covered Area*

4.1.4. Scenario 4

This is the least energy consuming mode where there is no presence of personnel or truck. In this mode all lamps should be maintained at 40% level for safety regulations.

4.2. Hardware Architecture

The Smart-Lighting implementation is consisting of 6 fixtures of LED lamps, 3 S-1000 controllers, 1 Valopaa master unit, 1 THL sensor, 1 personnel counter, 1 light and temperature sensor, 1 motion sensors and 1 push button connected as shown in Figure 22.

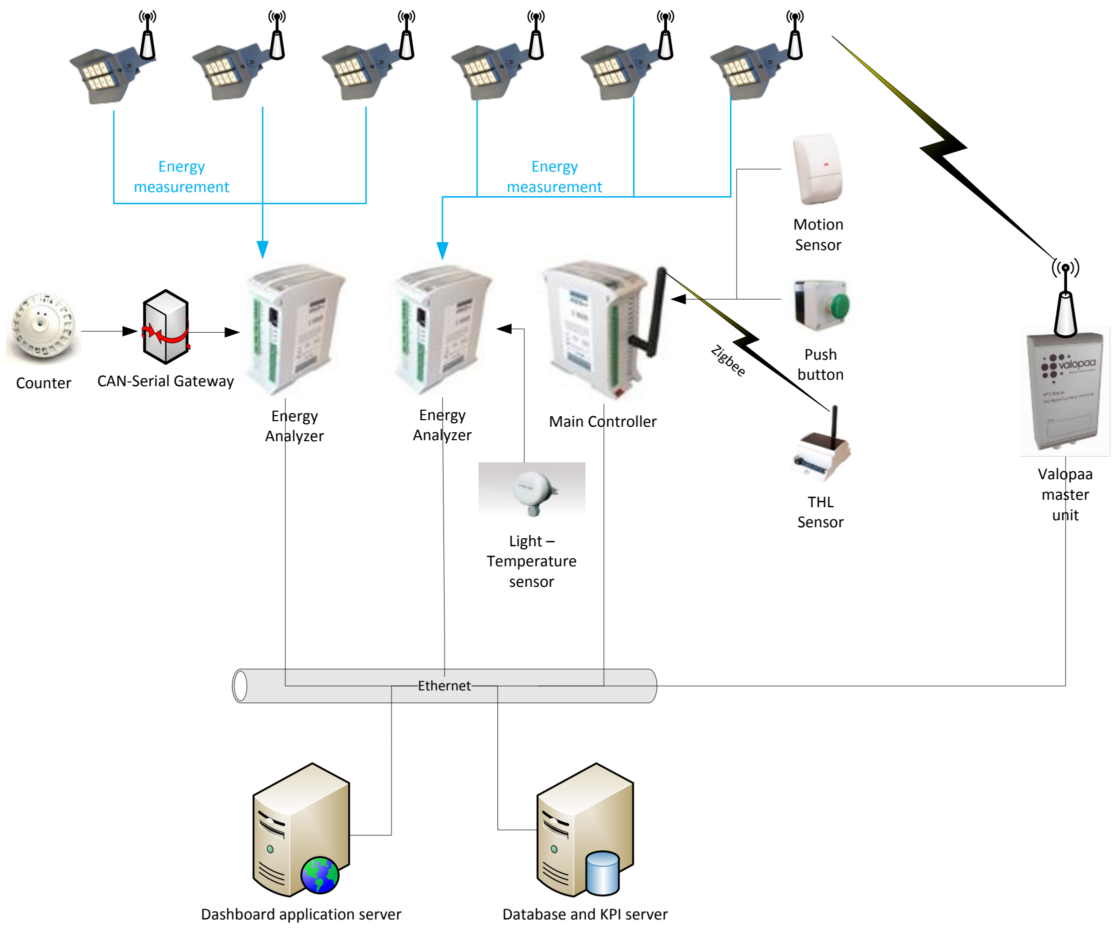


Figure 22, *Smart Lighting Hardware Architecture*

4.2.1. Lamps

The lamps are of type Valopaa VP3411 LED floodlight fixtures shown in Figure 23 are capable of providing the adequate amount of luminance with minimum energy dissipation. The lighting fixture can be installed outdoor as it is sealed and durable for use in extreme weather conditions. The aluminium casing provides efficient heat dissipation for the electronics inside. [56] The lamps are controlled wirelessly by radio signals from the Valopaa Master Unit.

Figure 23, *VP3411 LED floodlight fixture* [56]

4.2.2. Valopaa Master Unit

The Master Unit is used for controlling the lamps wirelessly through radio waves. It can also generate schedules and give statistical feedback for energy consumptions and lamps temperature through Ethernet. A wireless sensor module can be connected for motion and light sensors but, this module is not used in this implementation. It can also have optional capabilities of GPRS and WLAN for receiving control data and sending statistics. It is well encapsulated so that it can withstand harsh weather environment and can be installed outdoors.

Figure 24, *Valopaa Master Unit* [57]

The “Master Unit” is modified to handle specifically tailored commands for this implementation, for controlling the lamps through Https POST requests. This will help the application server for sending control commands and directly controlling the lamps.

4.2.3. S-1000 Controllers

S-1000 is a smart programmable RTU with Ethernet connectivity that has DPWS capabilities and can publish web-service events to other subscribers and SOAP messages to devices. In this implementation two controllers will be extended with energy module E-10 that provides energy measurement capabilities for 3 different nodes. The main controller will be extended with wireless router module to be able to connect to the THL sensor nodes. These controllers also have analogue and digitals I/O for connecting to the outer world, in addition to an optional serial or CAN ports might be possible.

Figure 25, *S-1000 with Wireless Module*

Figure 26, *S-1000 with E-10 Module*

4.2.4. THL sensor

This sensor combines a temperature, humidity and light sensors all integrated in one board. The sensor has wireless capabilities and can send the values in a SOAP message over 6lowpan network. The “Main controller” has a wireless module to act as a wireless router for this sensor. The scan cycle of this sensor is set to 15 minutes, this insures battery life saving and utilizing of energy. The light intensity measured by this sensor plays an important role in the control algorithm for calculating variable Z (4.4.3).

Figure 27, *THL Sensor*

4.2.5. Motion detector

The motion sensor used is of Busch-Watchdog 280 MasterLINE model shown in Figure 28. It is a passive infrared detector that detects movement of thermal objects. It has a detection range of 280° which can cover almost all required area. The sensor acts as a digital relay with normally open digital output. The output of this sensor is connected to the input of the main controller.

Figure 28, *Motion detector*

4.2.6. Personnel counter

The IRC1004 counter concept depends on thermal image processing for counting personnel going in or out of retail store. The sensor is often fixed indoors above the entrance doorway in order to count personnel from above in the direction towards downwards. The sensor has three types of outputs: CAN (Controller Area Network), Serial COM or normal relay output. In this implementation the sensor used was immigrated from another project therefore, it is using a CAN to serial gateway in order to communicate with the S-1000. All ST programs for communicating and resetting the counter are imported from the old project.

Figure 29, *Irisys IRC1004 Personnel Counter*

4.2.7. Light and Temperature sensor

The “Produal lux34” is a combination of light and temperature sensors in the same fixture. It is capable of producing 0-10V analogue signal representing the data. A signal conditioning is needed at the output because the S-1000 controller analogue inputs have to be 0-5V. It has severe operating temperatures which make it suitable for working under harsh weather conditions in Finland.

Figure 30, *”Produal lux34” Light and Temperature Transmitter*

4.3. Software Architecture

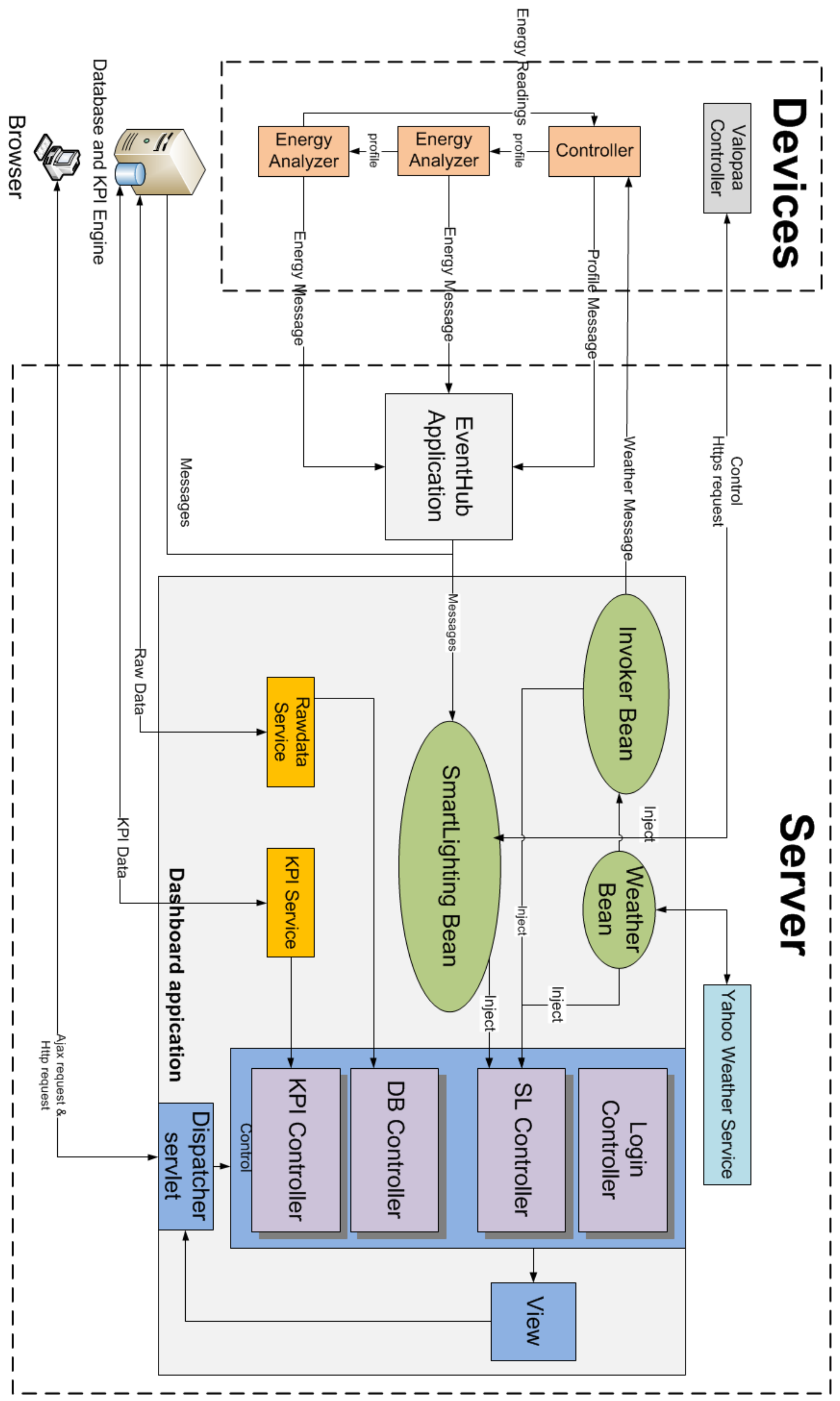


Figure 31, Smart Lighting Software Architecture

Then the software architecture of the smart lighting implementation shown in Figure 31 was designed. It consists of the dashboard application, Event hub application, database, KPI engine and, programs hosted on the s-1000 devices.

The Event Hub application works as a concentrator for the messages coming from the devices. It implements a DPWS client for discovering devices and subscribing to certain events with a given filters. The hub application then forwards the messages to the database and Dashboard application through a Camel endpoint.

The Dashboard application used to visualize the system data and results in a web application. The application contains a Smartlighting Spring bean that has all relevant values of the implementation. In the initialization method of the Smartlighting bean a camel context is initialized with and end point for receiving the messages from the hub. The endpoint then forward the message to parseMessage() method, where uses the parser interface for parsing the message and generating the message data depending on the message name. If the message was a control message, the control parameters are passed together with the profile value to the Valopaa manager object, which is responsible for sending Https control requests to the lamp controller. The Smartlighting bean is injected in the main controller for retrieving the data when needed.

The Weather bean is also injected to the main controller to provide the required weather data for the Ajax requests. The weather bean has an initialization method that gets the current weather from the Yahoo service every one hour.

There are four controllers in the Dashboard application responsible for handling login and Ajax requests made by the user client. The DB and KPI controllers use services described in section 4.8.7 to get the raw data and KPI data from the database and KPI engine respectively. The controllers responsible for arranging the data and convert them to data table format.

The Invoker bean also implements a DPWS client for sending weather and manual parameters to the s-1000 controller device. The bean is injected into the main controller in order to invoke manual operations on the device when a manual parameter is sent from the user.

4.4. Control Algorithm

There are three main variables X, Y and Z taking part in the control, they represent different conditions. Variable X presents the weather condition, variable Y shows personnel presence or truck loading and, variable Z represent the day or night depending on the light level. There are 32 combinations between these variables giving 32 possible profiles. These variables are going to be illustrated in details in this section.

4.4.1. Weather Condition (X)

For determining the weather conditions, the server gets the data from Yahoo RSS feed mentioned in section 3.4. Then the server invokes the weather operation in the main controller with the weather parameters. There are 48 possible weather conditions available in Yahoo feed, for simplification in this implementation only four weather conditions were defined; snow, rain, mist and, clear. This simplification is done by approximating similar or partially similar conditions that require almost same lighting level as shown in Table 5. A numerical value shown in Table 6 is assigned to the variable X according to the approximated weather condition. The logic for determining X is shown in section 4.5.1.3.

Code	Description	Approximated Condition
0	tornado	Snow
1	tropical storm	Inapplicable
2	hurricane	Inapplicable
3	severe thunderstorms	Rain
4	thunderstorms	Rain
5	mixed rain and snow	Snow
6	mixed rain and sleet	Rain
7	mixed snow and sleet	Snow
8	freezing drizzle	Snow
9	drizzle	Rain
10	freezing rain	Snow
11	showers	Rain
12	showers	Rain
13	snow flurries	Snow
14	light snow showers	Snow
15	blowing snow	Snow
16	snow	Snow
17	hail	Snow
18	sleet	Snow
19	dust	Mist
20	foggy	Mist
21	haze	Mist
22	smoky	Mist
23	blustery	Mist
24	windy	Clear
25	cold	Clear
26	cloudy	Clear
27	mostly cloudy (night)	Clear
28	mostly cloudy (day)	Clear
29	partly cloudy (night)	Clear
30	partly cloudy (day)	Clear
31	clear (night)	clear
32	sunny	clear
33	fair (night)	clear
34	fair (day)	clear
35	mixed rain and hail	Rain
36	hot	clear
37	isolated thunderstorms	Rain
38	scattered thunderstorms	Rain
39	scattered thunderstorms	Rain
40	scattered showers	Rain
41	heavy snow	Snow
42	scattered snow showers	Snow
43	heavy snow	Snow
44	partly cloudy	clear
45	thundershowers	Rain
46	snow showers	Snow
47	isolated thundershowers	Rain
3200	not available	Inapplicable

***Table 5,** Weather condition simplification*

Simplified Condition	Value of X
Clear	1
Fog	2
Rain	3
Snow	4

***Table 6,** Values Assigned for X Variable*

4.4.2. Personnel and Truck (Y)

The value of this variable is depending on the personnel and truck Boolean values as shown in Table 7. When the motion sensor detect presence the personnel value is set to true. The truck value is determined by a push button triggering the “loading/unloading” mode. A simple logic is applied for determining the Y value in section 4.4.1.4, by detecting if both or none or only one of the values is true.

Industrial Condition	Value of Y
Personnel	1
Truck	2
Both	3
Neither nor	4

Table 7, *Values Assigned for Y Variable*

4.4.3. Light Intensity (Z)

The value of Z is determined by the light level detected by the THL sensor (4.2.4). If the light level is above a threshold value, Z equals one. On the other hand if the light value is below, Z equals two.

Illumination level	Value of Z
>threshold value	1
<threshold value	2

Table 8, *Values for Z*

4.4.4. Profiles

The combination of the previous variables gives 32 possible profiles. Lamps lighting levels are assigned carefully for each profile in order to achieve maximum user satisfaction. All possible scenarios described in section 4.1 were taken into consideration while assigning the lighting level. Table 9 shows all possible profile combination and their assigned lighting level.

Variables				Correspondent Illumination Level %						
X	Y	Z	Profile No.	Lamp 1	Lamp 2	Lamp 3	Lamp 4	Lamp 5	Lamp 6	Average Lighting Level %
Clear	personnel	Day	1	0	0	0	0	0	0	0
		Night	2	40	40	80	80	80	70	65
	Truck	Day	3	0	0	0	0	0	0	0
		Night	4	60	60	60	60	40	60	56,66666667
	Both	Day	5	0	0	0	0	0	0	0
		Night	6	90	90	90	90	60	85	84,16666667
	non	Day	7	0	0	0	0	0	0	0
		Night	8	20	20	20	20	20	20	20
Mist or Fog	personnel	Day	9	40	40	60	60	60	40	50
		Night	10	80	80	100	100	100	100	93,33333333
	Truck	Day	11	60	60	60	60	50	60	58,33333333
		Night	12	70	70	70	70	60	70	68,33333333
	Both	Day	13	80	80	80	80	70	80	78,33333333
		Night	14	100	100	100	100	90	100	98,33333333
	non	Day	15	40	40	40	40	40	40	40
		Night	16	50	50	50	50	50	50	50
Cloudy	personnel	Day	17	20	20	30	30	30	20	25
		Night	18	40	40	60	60	60	50	51,66666667
	Truck	Day	19	20	20	20	20	20	20	20
		Night	20	50	50	50	50	40	50	48,33333333
	Both	Day	21	60	60	60	60	40	60	56,66666667
		Night	22	80	80	80	80	60	80	76,66666667
	non	Day	23	10	10	10	10	10	10	10
		Night	24	20	20	20	20	20	20	20
Snow	personnel	Day	25	20	20	40	40	40	30	31,66666667
		Night	26	60	60	90	90	90	80	78,33333333
	Truck	Day	27	40	40	40	40	30	40	38,33333333
		Night	28	70	70	70	70	50	70	66,66666667
	Both	Day	29	50	50	50	50	40	50	48,33333333
		Night	30	90	90	90	90	70	90	86,66666667
	non	Day	31	20	20	20	20	20	20	20
		Night	32	30	30	30	30	30	30	30

Table 9, *Profiles and Lamps Required Lighting Level*

4.4.5. Control Action

The initial design was to control the lamps directly from the devices using serial port and a gateway which will translate the commands into DALI protocol. Since neither the gateway nor the DALI lamp was available by the time of writing this thesis, it was demanding to find an alternate solution for controlling the lamps. The solution was to control the lamps from the server side, because the devices are not capable for making https requests. When the server receives a control message (4.6.8), it automatically generates an Https POST request using the "Valopaa Manager" (4.8.6). It was observed that the response time was higher than required thus this control method is just temporarily and need to be modified.

4.5. ST programs

The S-1000 controllers are programmed in structured text (ST) language. A controller can have one or more logic program running at the same time in parallel. These programs are executed in a cyclic behaviour. This section explains and describes programs used in the "Main controller" and "Energy meters".

4.5.1. Main Controller

4.5.1.1 Function program

This ST program is designed to calculate the function “F” by using the equation 4.1 in order of determining the profile. Then the program iterate through a series of if conditionals to check for the profile, this is due to that the S-1000 does not support “Switch Case” commands. When a profile match found, the “profile_new” variable is set as well as the corresponding lighting levels. The program is executed in a cyclic manner as shown in Figure 32.

$$F = 100 * X + 10 * Y + 1 * Z \qquad (4.1)$$

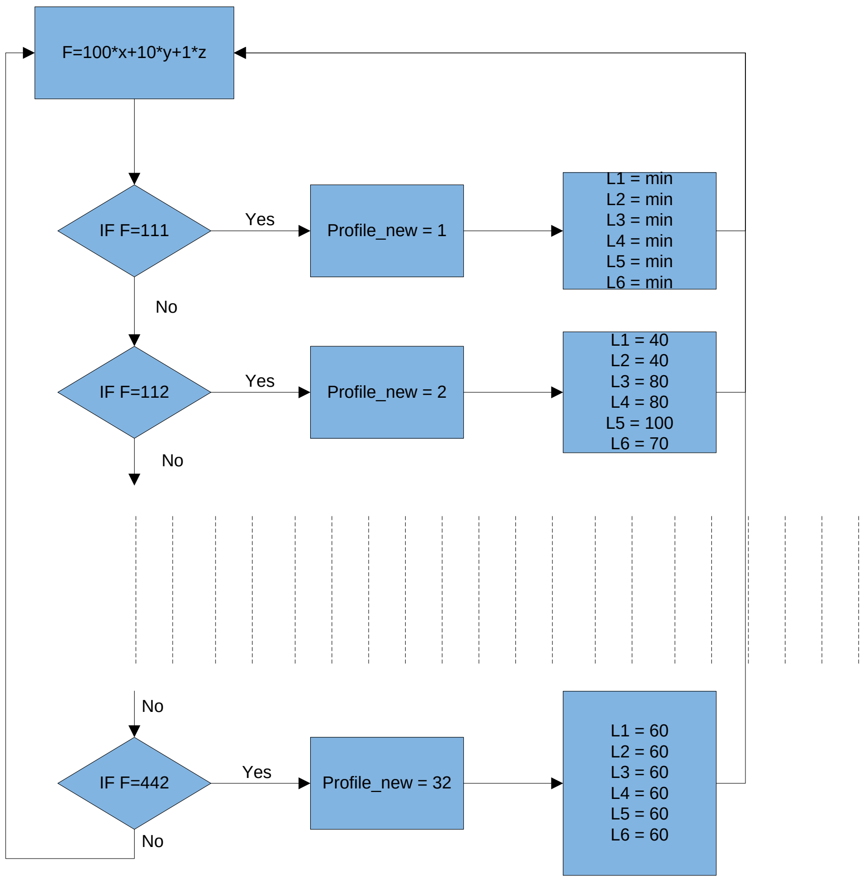


Figure 32, *Function Program Flowchart*

4.5.1.2 Compare Profile

This ST program is used in order to detect the moment of profile change by comparing the new profile value to the absolute one. If change is detected an internal service is sent to the next energy meter informing with the new profile. Finally the current value of profile is updated with the new one.

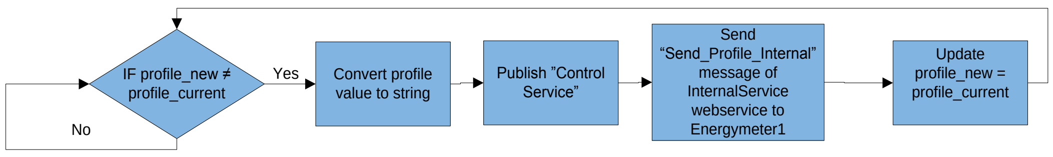

IF profile_new ≠ profile_current
Yes
No
Convert profile value to string
Publish "Control Service"
Send "Send_Profile_Internal" message of InternalService webservice to Energymeter1
Update profile_new = profile_current

Figure 33, *Compare Profile ST Program Flowchart*

4.5.1.3 Determine weather

This program is triggered when the weather operation is invoked in the device. Fist a response string is set to be replied back, and then the weather condition is converted into a numerical value. The condition is checked through a series of if conditionals then, the value of X is set according to the current condition.

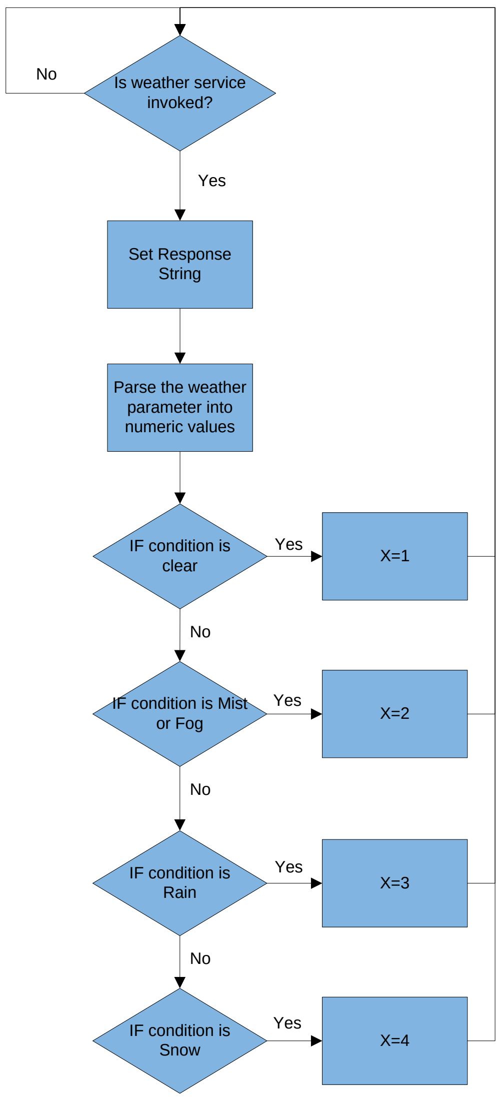


Figure 34, *Determine weather ST Program Flowchart*

4.5.1.4 Operating mode

This program is used to determine the y variable by detecting the presence of personnel and truck mode concurrently or separately. This done by checking the AND, OR and XOR values of these two Boolean parameters and setting the Y value accordingly. Finally the personnel and truck values are converted to string to prepare them to be sent through the web service.

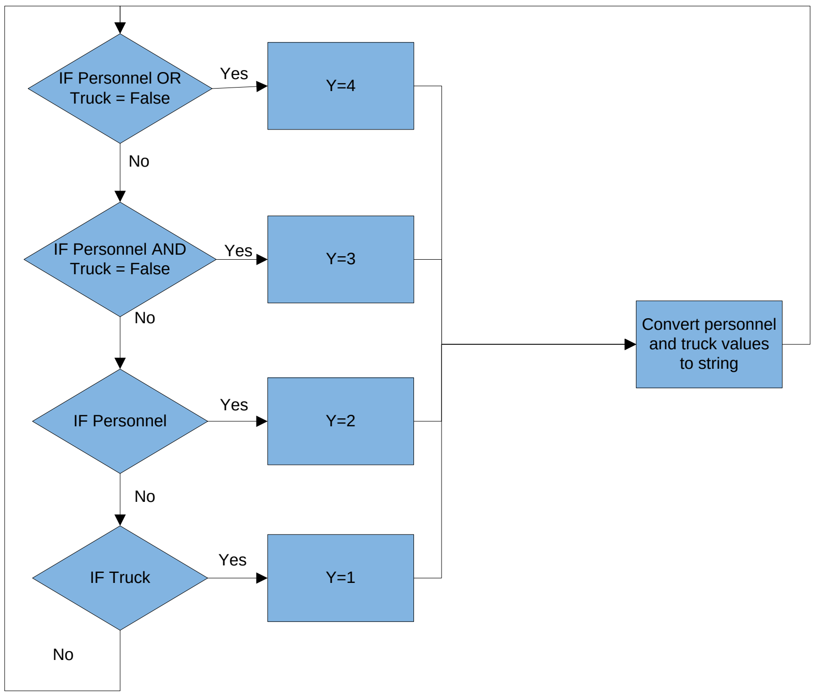

IF Personnel OR Truck = False
Yes
Y=4
No
IF Personnel AND Truck = False
Yes
Y=3
No
IF Personnel
Yes
Y=2
No
IF Truck
Yes
Y=1
No
Convert personnel and truck values to string

Figure 35, *Operating mode ST Program Flowchart*

4.5.1.5 Ambient Light level

The Z value is determined by a threshold value for the temperature light and humidity sensor. If the threshold value is exceeded then it is day and Z equals 2, else it is still night and Z equals 1.

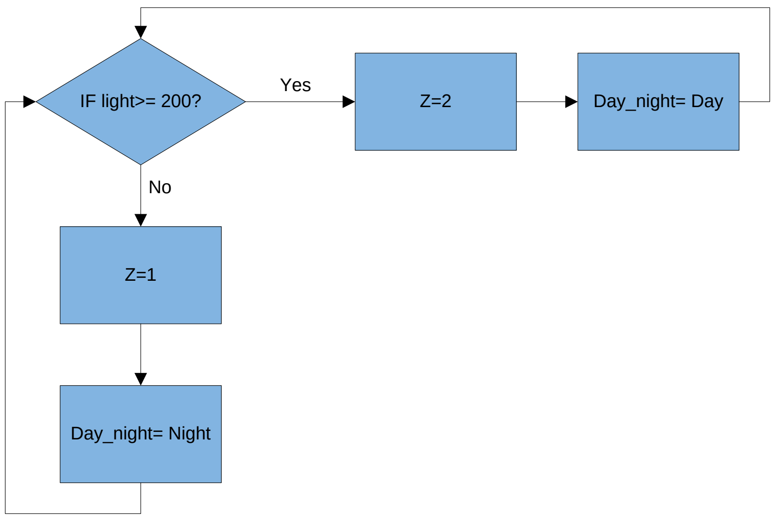


Figure 36, *Ambient Light level ST Program Flowchart*

4.5.1.6 Energy readings received

This program is initiated when the “Main controller” receives all energy readings from energy meters. The controller then sends a response back acknowledging message received and sets the timestamp to make the variable ready to be sent. Then the Profile message is sent to the hub and the serial commands are sent to all lamps simultaneously.

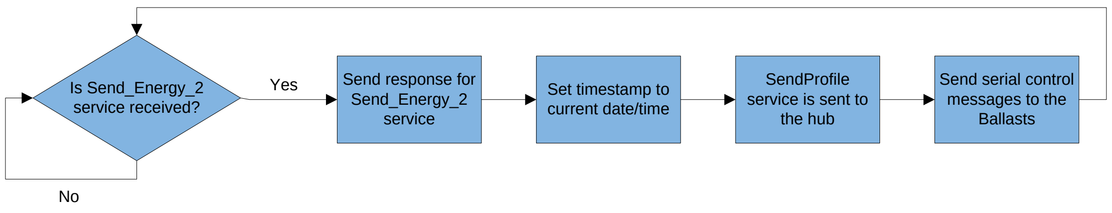

Is Send_Energy_2 service received?
Yes
Send response for Send_Energy_2 service
Set timestamp to current date/time
SendProfile service is sent to the hub
Send serial control messages to the Ballasts
No

Figure 37, *Energy readings received ST Program Flowchart*

4.5.1.7 Personnel detection

This program is waiting for the triggering of the motion sensor, if the assigned input true then, there is personnel presence and the personnel flag is set to true. After that the program is suspended for a configurable time value to insure that there is no presence.

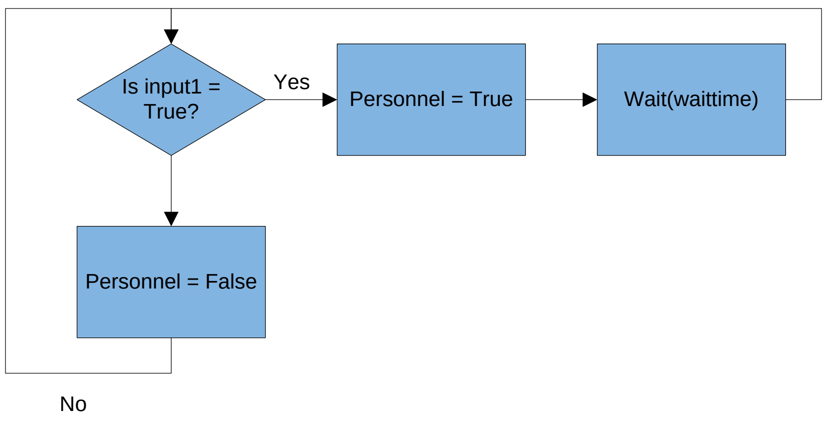


Figure 38, *Personnel detection Program Flowchart*

4.5.1.8 Truck detection

This program is similar to the previous one but it is detecting the truck mode button input instead of personnel. The rest of the logic is similar to the previous program.

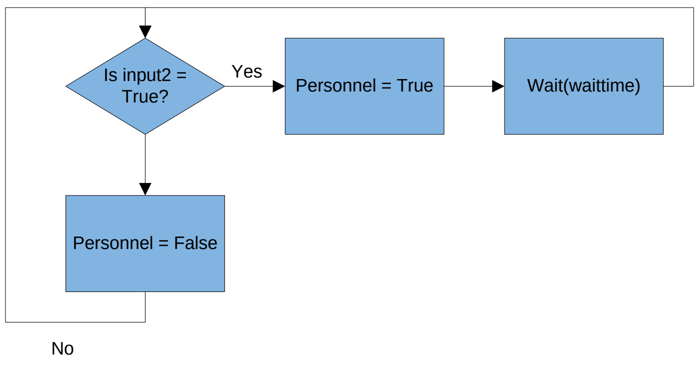
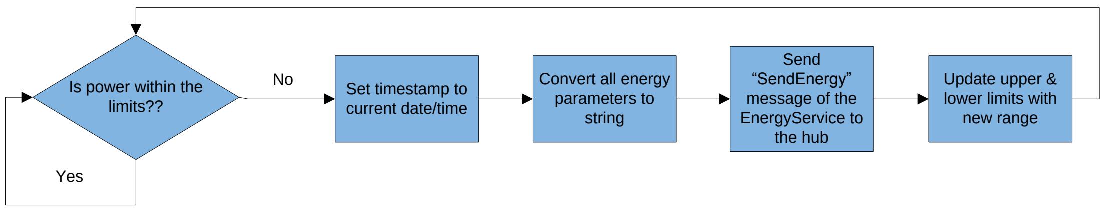


Figure 39, *Truck detection Program Flowchart*

4.5.2. Energy Meter

4.5.2.1 Send energy event based

This program is used for sending the energy message based on the "Event-based Variable Sampling rate" explained in section 4.6.1. The program is waiting for the power to exceed the range given. If exceeded, the timestamp is set to current date and time then, energy parameters are prepared and converted to string format to be sent. Finally the



Is power within the limits??
No
Yes
Set timestamp to current date/time
Convert all energy parameters to string
Send "SendEnergy" message of the EnergyService to the hub
Update upper & lower limits with new range

Figure 40, *Send energy event based ST Program Flowchart*

4.5.2.2 Send energy fixed sampling rate

On the contrary this program sends energy message based on fixed sampling rate as long as the power value is within the range. But first the program checks for the sampling rate value because it should be at least 100ms or else, it might have implications during execution. Then it checks the power value if still within the limits, it prepares the energy message by setting the timestamp, converting the parameters to string values and, finally send the energy message. If the power exceeded the range a new limits are assigned.

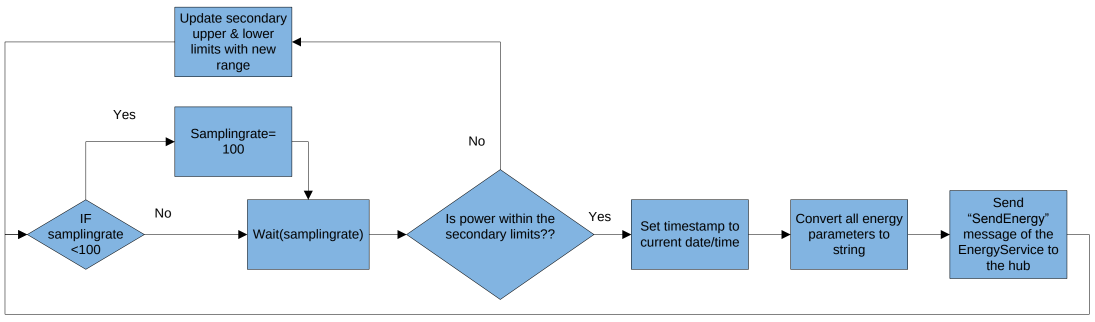

Update secondary upper & lower limits with new range
Yes
Samplingrate= 100
No
IF samplingrate <100
No
Wait(samplingrate)
Is power within the secondary limits??
Yes
Set timestamp to current date/time
Convert all energy parameters to string
Send "SendEnergy" message of the EnergyService to the hub

Figure 41, *Send energy fixed sampling rate ST Program Flowchart*

4.5.2.3 Profile Change

This program is triggered when a new profile is received by the "Internal Service". When triggered it send a response back and convert its energy readings to string values and send them to the next device with the profile in the "Internal Service".

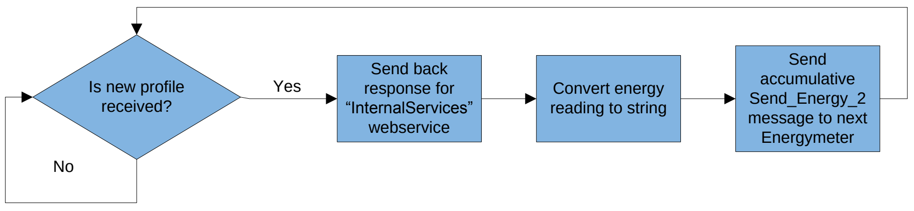

Is new profile received?
Yes
No
Send back response for "InternalServices" webservice
Convert energy reading to string
Send accumulative Send_Energy_2 message to next Energymeter

Figure 42, *Profile Change ST Program Flowchart*

4.5.2.4 Power Compensation

Due the lack of calibration for the energy meter, the power and energy values need to be programmatically compensated. For power an offset values are added to the original value.

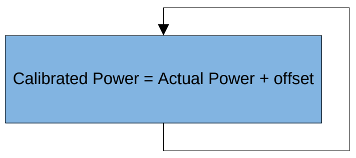
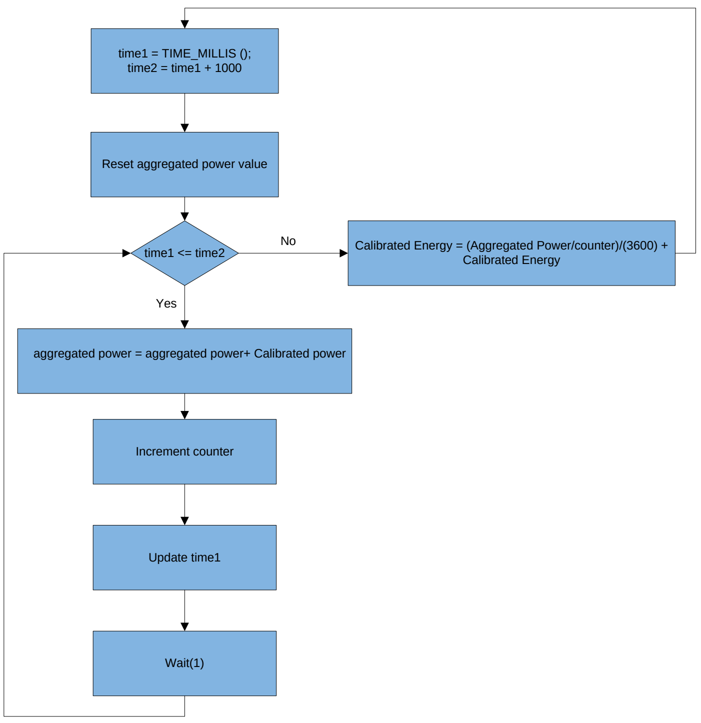


Figure 43, *Power Compensation ST Program Flowchart*

4.5.2.5 Energy Compensation

For compensating the energy values, it requires a complex algorithm and computing resources. At first two timers are fixed 1000ms apart and reset aggregated values. Then check if the timer1 is less than timer2 the power is aggregated with the calibrated power value from the previous program and counter is incremented. Then timer1 is updated with the new value and then the program is suspended for 1ms, this is because without the suspension it might cause crash. This loop is broken when timer1 is larger than timer2 then the calibrated energy is calculated and aggregated using equation 4.2.

$$Calibrated\ Energy = \frac{{}^{Aggregated\ Power}/_{counter}}{3600} + Calibrated\ Energy \quad (4.2)$$



time1 = TIME_MILLIS ();
time2 = time1 + 1000
Reset aggregated power value
time1 <= time2
No
Calibrated Energy = (Aggregated Power/counter)/(3600) + Calibrated Energy
Yes
aggregated power = aggregated power+ Calibrated power
Increment counter
Update time1
Wait(1)

Figure 44, *Energy Compensation ST Program Flowchart*

4.6. Web-Services

The devices have several services sending events to give real time data of the system status and energy consumption. These services are described in a WSDL file and it can be easily discovered and subscribed to the events. Some of the services are used to set some parameters in the devices such as weather conditions and manual control parameters. The following table list all services available on the devices.

ID	Service	Description	Device
1	Energy service	Publishes energy readings for each lamp based on interval time or event based.	Energy Meters
2	Profile service	Publishes profile and sensors data whenever the profile changes.	Main Controller

3	Weather Service	Used to invoke the “sendweather” operation in the main controller with weather parameters.	Main Controller
4	Internal service	This service is used for exchange of profile, energy and sensors data between the devices.	Energy Meters and Main Controller
5	Sensors service	Publishes sensors data periodically for monitoring.	Energy Meters
6	Manual service	Used to invoke the manual operations on the device with manual parameters from the application user.	Main Controller
7	Reset service	Used to initiate the reset program to reset energy readings.	Energy Meters
8	Control Service	Used to initiate the control command and provide the required lighting levels.	Main Controller

Table 10, *Web Services description and their location*

4.6.1. Energy service

The energy service consists of one event for sending the energy message, which contains a list of lamp element where each lamp has its own unique ID. Each lamp contains energy data such; voltage, current, power and energy. The energy message is published on two different intervals:

- **Event-based Variable Sampling rate:** This method is used to reduce network traffic and unnecessary messages. There are higher and lower limits for power consumptions of the lamps. If these limits are exceeded in both directions the message is published and, the limits are shifted with to new values. Thus this method provides less overheads and network usage enabling more meters to be connected in the network.
- **Fixed Sampling rate:** This method is required to ensure device connectivity and in good working conditions. This method is based on marking two limits as well but, with a larger span. Energy message is published on a fixed sampling rate whenever the power reading is within the range. All ranges and sampling rate can be adjusted by the user for fine tuning.

4.6.2. Profile service

The profile service contains an event having all the profile data such as profile value, weather, current energy consumption, personnel and truck. The profile event is triggered when the main controller receive all energy readings from the internal service. The data sent by this event is used later on to be sent to the server for displaying in the dashboard.

4.6.3. Weather service

The weather service has only one input message defined, used for setting the weather parameters from the server to the device. The Weather message contains the following parameters: condition, temperature, visibility, pressure, wind speed and direction. Currently, only weather condition parameter is used in the control algorithm, but in the future other parameters can be involved as well.

4.6.4. Internal service

The Internal service is defined in all devices and used for internal messaging between them. Each device has one input and one output message. The device receives a message containing the profile and energy from previous devices then appends its own readings and forwards the message to the next device. The message propagates until it reaches the main controller again, at this moment the controller is aware of the energy consumptions of all lamps. Then the "main controller" sends the profile message. Although the energy readings are currently not included in the control algorithm, it might be involved for complex algorithms for energy reduction in the future.

4.6.5. Sensors service

The sensor service is defined in the energy devices and it includes a sensors event, which contains data about all sensors connected to this device. The devices send sensors service periodically for the dash board for visualization purposes. Each sensor has unique ID attribute and the value is in the element.

4.6.6. Manual service

The manual service resides in the SL controller and it includes three main messages setProfile, setLamps and setManualmode. These messages are sent by the application server whenever the user set a manual parameter from the control panel. Currently there is no manual mode implemented in the main controller only automatic but, it can be implemented in the future.

4.6.7. Reset service

The reset service has one input message for requesting the reset. When the message is received a reset program is triggered that reset all energy readings on the device. This is a useful service to reset the readings in the end of each month or whenever desired by the operator.

4.6.8. Control service

The control service contains one event for control message. The event is published whenever the profile is changed and, it contains the new profile value and desired lighting level. This service was introduced instead of using the profile message to minimize the response time, so as not to depend on the “Internal service” message which has to go through all devices and the possibility for the message to be lost. When the server receive the control message it parse the desired lighting level and pass them to the Valopaa Manager object (4.8.6) in order to set the lamps by Https request.

4.7. KPIs

Metrics defined in this section are used for measuring the system overall performance and compare values to calculate its efficiency. The metrics were chosen to give the best holistic view of system energy performance for the management level. Results based on these KPIs will aid researchers to improve the control algorithm and lamp design in order to reach better results.

ID SL001	Title: Average Power per hour
Mathematical expression	$APH = \frac{\sum_{hour} TP}{n}$
Notations	APH: Average power per hour in Watt. TP: Total power of all lamps. n: number of readings.
Description	This KPI calculate the average power used in one hour by summing all power readings in one hour and divide by the number of readings.
Comments	This KPI shows the average consumption per hour in a specific day, from which peak hours can be found.
Input data	Power readings from the database

ID SL002	Title: Average Power per day
Mathematical expression	$APD = \frac{\sum_{Day} TP}{n}$
Notations	APD: average power per day in Watt. TP: Total power for all lamps. n: Number of readings
Description	This KPI calculate the average power used in one day by summation of all power readings and dividing by number of readings.
Comments	The average power consumed per day can be compared between several days and, any abnormality in the lamps behaviour can be easily detected.
Input data	Power readings at all timestamps of the day

ID SL003	Title: Light Power density per day
Mathematical expression	$LPD = APD/Area$

Notations	APD: Average power per day (SL002) Area: Total parking area = 152m^2 LPD: Light power density in Watt/m^2
Description	This KPI Calculates the LPD by dividing the average power by the area of usage.
Comments	This KPI will help comparing different sites and rooms regardless the size.
Input data	SL002 Area covered by lamps

ID SL004	Title: Total Energy Consumption per day
Mathematical expression	$TECD = \sum_{Lamps} ERED - \sum_{Lamps} ERBD$
Notations	TECD: Total Energy consumption per day in Watt.hr ERED: Energy Reading at End of the Day. ERBD: Energy Reading at Beginning of the Day.
Description	This KPI calculate the total energy consumption per day by subtracting the total energy in the beginning of the day from the energy reading at the end of the day.
Comments	This KPI will give exact figure of the energy consumption for each day and based on it the cost per day can be estimated in SL005.
Input data	Energy of lamp(id) @ 23:59:00 timestamp from Database Energy of lamp(id) @ 00:00:00 timestamp from Database

ID SL005	Title: Total Cost per day
Mathematical expression	$TCPD = TECD \times P$
Notations	TECD: Total Energy Consumption per day (SL004) P: Price of 1 kWh = 0.08 euro/kWh
Description	This KPI calculate the cost for each day by multiplying TECD by the price of Kw.hr
Comments	This KPI will give help in the calculation of the total cost in the end of the month and estimating the lighting bill.
Input data	SL004

ID SL006	Title: CO_2 equivalent per day
Mathematical expression	$\mathrm{CO2\ eq} = \mathrm{TECD} \times 7{,}0555 \times 10^{-4}$ [58]
Notations	TECD: Total Energy Consumption this month(SL004) CO_2 equivalent: The amount of CO_2 in Kg produced from fuel combustion for the same equivalent amount of electricity.
Description	This KPI calculate the equivalent amount of CO2 emitted for one day.
Comments	This KPI will determine the environmental impact of the lighting system.
Input data	SL004

ID SL007	Title: Profile Percentage

Mathematical expression	$$PP = \frac{\sum_{profile(i)}(NPTS - CPTS)}{TP} \times 100$$
Notations	PP: Profile Percentage. NPTS: Next Profile Timestamp. CPTS: Current Profile Timestamp. TP: Total Period of calculation.
Description	This KPI calculate the percentage of occurrence of each profile by summing the total duration and dividing by total period multiply by 100.
Comments	This KPI will help determining profile occurrence, the most common profiles. Determining unused profiles to be able to eliminate them or reduce them to simplify the control algorithm.
Input data	Timestamp of Profile(i) from profile table Timestamp of next profile in the table.

ID SL008	Title: Profile Energy
Mathematical expression	$$P_iE = \sum NP_iE - CP_iE$$
Notations	PiE: Profile number(i) Energy NPiE: Next Profile Energy CPiE: Current Profile Energy
Description	This KPI calculate the total energy consumption for a certain profile (i) for a certain period of time by, subtracting current profile reading from the next one.
Comments	This KPI will determine how much energy is consumed by certain profile keeping in mind that the duration of the profile SL007 is directly affecting the energy consumed by it.
Input data	Sum of Energy of lamps at certain profile(i). Sum of Energy of lamps at next profile from table.

4.8. Dashboard Application

Developing the dashboard application for monitoring, analysing and controlling was one of the objectives of this thesis work. It is developed in Java language and, it is based on Spring MVC described in section 3.3. The main features of this application are: stability, security, reliability, user friendly, easy and fast learning curve. Security and user authentication were achieved using spring security library. There are seven main partitions in the application code as follows:

4.8.1. SmartLightingApp

The SmartLightingApp class is considered as the core of this application, containing all useful data. It is defined as a Spring bean in the application context to easily inject in any controller if required. It has an initialization method to start the camel context and initialize the endpoint used for receiving the messages and, a destroy method to shut down the context when the bean is destroyed.

The camel context has one route defined from the endpoint, then the SOAP header is removed and body of the message is transformed into a Document class. Finally, the document is passed to the parseMessage method which uses the parser interface for extracting the useful information from the document and generating objects and setting them in the variables.
The main variables in this class are:

Variable	Description
profile	An instance of ProfileMessage class where all profile data are located.
energymeters	A hash map of available energy meters.
lamps	A hash map of all lamp objects contained in the energy meters.
sensors	A hash map of all sensor objects.
valopaa	A valopaa manager that is used to send https requests to the valopaa controller.

Table 11, *SmartLightingApp variables*

4.8.2. Parser Interface

This interface contains certain methods used for simplifying parsing of the message and generating different objects. The main requirement for this interface is flexibility and easily generating different objects with different ID, so that adding more energy meters and lamps in the future can be easily done. The MessageParser” class implement this interface’s methods.

An UML diagram that describes the SmartLighting and MessageParser is shown in Figure 45. The main methods for this interface are:

Method	Description
generateProfileMessage()	This method generates a ProfileMessage object from the document. This object contain all the parameters describing the profile such as: profile value, weather parameters, alarms, sensors, personnel, truck, lamps energy and the required lighting levels.
generateEnergyMessage()	This method generates an EnergyMessage object that contains the time stamp and list of Lamp objects inside the message.
generateLamp()	This method used to generate Lamp object that has all values regarding the lamp such as: ID, volt, current, energy, power and temperature.
getDeviceId()	This method is used to identify the device sending the message by a unique device ID string.
generateSensor()	This method is used to create a sensor object from the sensors message. Each sensor has a unique ID and value.

Table 12, *Parser Interface methods*

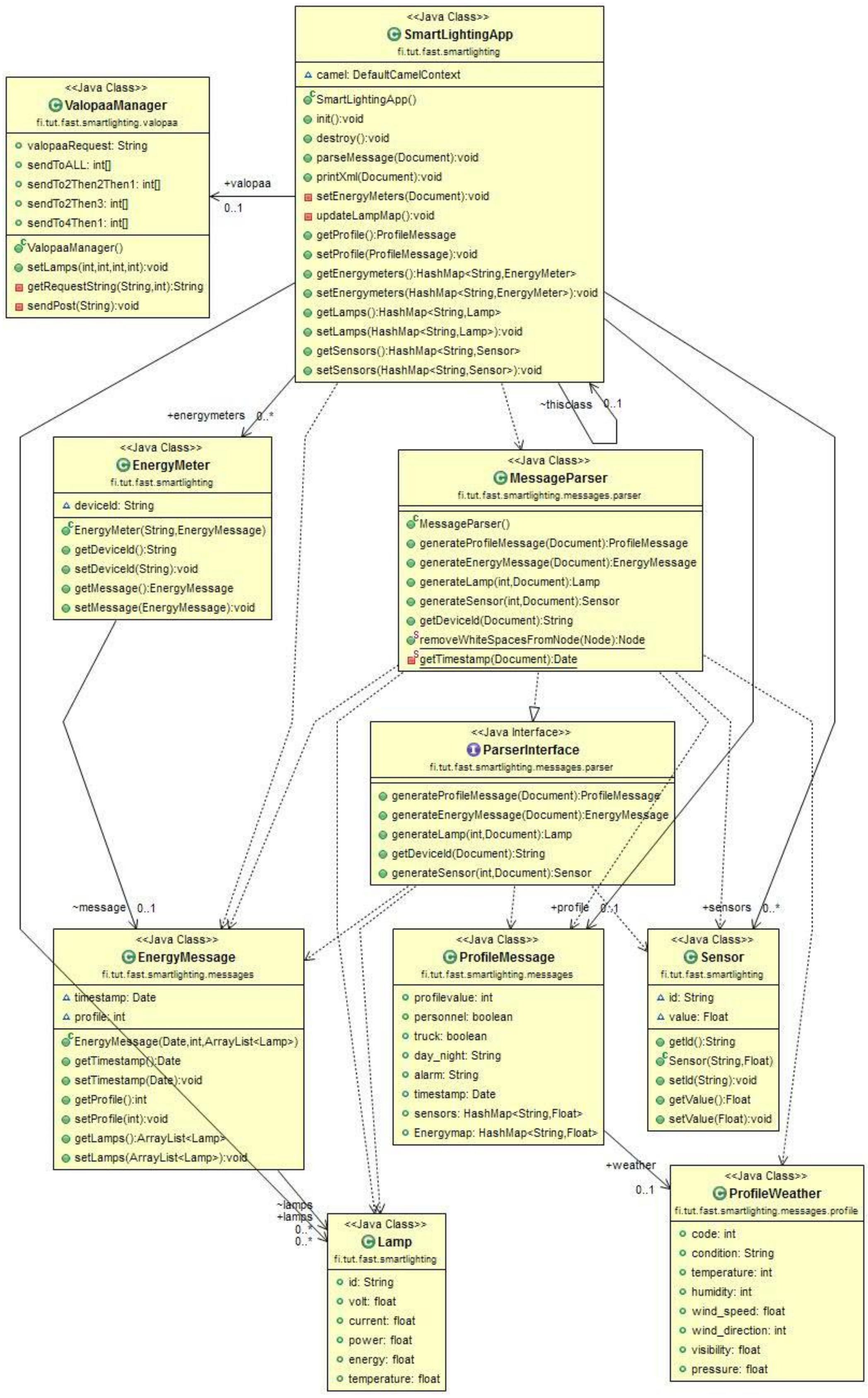

<<Java Class>>
SmartLightingApp
fi.tut.fast.smartlighting
camel: DefaultCamelContext
SmartLightingApp()
init():void
destroy():void
parseMessage(Document):void
printXml(Document):void
setEnergyMeters(Document):void
updateLampMap():void
getProfile():ProfileMessage
setProfile(ProfileMessage):void
getEnergymeters():HashMap<String,EnergyMeter>
setEnergymeters(HashMap<String,EnergyMeter>):void
getLamps():HashMap<String,Lamp>
setLamps(HashMap<String,Lamp>):void
getSensors():HashMap<String,Sensor>
setSensors(HashMap<String,Sensor>):void
<<Java Class>>
ValopaaManager
fi.tut.fast.smartlighting.valopaa
valopaaRequest: String
sendToALL: int[]
sendTo2Then2Then1: int[]
sendTo2Then3: int[]
sendTo4Then1: int[]
ValopaaManager()
setLamps(int,int,int,int):void
getRequestString(String,int):String
sendPost(String):void
+valopaa
0..1
+energymeters
0..*
~thisclass
0..1
<<Java Class>>
EnergyMeter
fi.tut.fast.smartlighting
deviceId: String
EnergyMeter(String,EnergyMessage)
getDeviceId():String
setDeviceId(String):void
getMessage():EnergyMessage
setMessage(EnergyMessage):void
<<Java Class>>
MessageParser
fi.tut.fast.smartlighting.messages.parser
MessageParser()
generateProfileMessage(Document):ProfileMessage
generateEnergyMessage(Document):EnergyMessage
generateLamp(int,Document):Lamp
generateSensor(int,Document):Sensor
getDeviceId(Document):String
removeWhiteSpacesFromNode(Node):Node
getTimestamp(Document):Date
<<Java Interface>>
ParserInterface
fi.tut.fast.smartlighting.messages.parser
generateProfileMessage(Document):ProfileMessage
generateEnergyMessage(Document):EnergyMessage
generateLamp(int,Document):Lamp
getDeviceId(Document):String
generateSensor(int,Document):Sensor
~message
0..1
+profile
0..1
+sensors
0..*
<<Java Class>>
EnergyMessage
fi.tut.fast.smartlighting.messages
timestamp: Date
profile: int
EnergyMessage(Date,int,ArrayList<Lamp>)
getTimestamp():Date
setTimestamp(Date):void
getProfile():int
setProfile(int):void
getLamps():ArrayList<Lamp>
setLamps(ArrayList<Lamp>):void
<<Java Class>>
ProfileMessage
fi.tut.fast.smartlighting.messages
profilevalue: int
personnel: boolean
truck: boolean
day_night: String
alarm: String
timestamp: Date
sensors: HashMap<String,Float>
Energymap: HashMap<String,Float>
<<Java Class>>
Sensor
fi.tut.fast.smartlighting
id: String
value: Float
getId():String
Sensor(String,Float)
setId(String):void
getValue():Float
setValue(Float):void
+weather
0..1
<<Java Class>>
ProfileWeather
fi.tut.fast.smartlighting.messages.profile
code: int
condition: String
temperature: int
humidity: int
wind_speed: float
wind_direction: int
visibility: float
pressure: float
~lamps
+lamps
0..*
0..*
<<Java Class>>
Lamp
fi.tut.fast.smartlighting
id: String
volt: float
current: float
power: float
energy: float
temperature: float

Figure 45, *SmartLightingApp and MessageParser UML Diagram*

4.8.3. Controllers

The controllers function is to acquire and arrange the data needed by the client side as stated in section 3.3. For the dashboard application there will be 4 controllers defined by annotation.

4.8.3.1 SL Controller

The purpose of the SL controller is to handle most of the requests by user, such as page request or AJAX request to get the weather, profile or lamps status. The main controller has SmartLightingApp and Weather beans injected inside of it, to be able to instantaneously get any data whenever needed. The main controller is responsible for responding to (/dashboard), where it check the user role and return the appropriate page for the user.

This controller is also responsible for responding to getLamps, getWeather, getProfile AJAX requests. The controller gets the data from the beans injected inside of it and, converts them into JavaScript Object Notation (JSON format to be easily understandable by the browser in JavaScript.

The controller handles the requests for setting the manual parameters. The controller utilizes the Invoker class in order to send the parameters requested by the user to the device. Finally the controller responds with the response from the device or invocation error if the device couldn’t be found.

4.8.3.2 Login Controller

The login controller intercepts all requests and it delivers pages required for user authentication. This controller handles requests for login, logout and login failure.

4.8.3.3 DB Controller

The DB controller is in charge of handling the getHistoricalData Ajax request which contains the parameter, end date and start date parameters. Then the controller requests the historical data by these parameters from the database and responds them in the appropriate format to the browser. This controller uses the raw data web service explained in section 4.8.7 for acquiring the data. The controller needs to sort the data by lamp before it uses getMultiChart() method in the chart interface described in section 4.8.5. The interface formats the data in the appropriate JSON response format for plotting.

4.8.3.4 KPI Controller

This controller handles the getKpiData.kpi that as well contains kpiID, “end date” and start date. Then the controller retrieve KPI data using web service explained in section 4.8.7 and format them in the chart JSON format using the drawKPI() method. This controller doesn’t require sorting the data by lamps because; this is done in the chart implementation.

4.8.4. DPWS Client

The DPWS client extends JMEDS default client and, it is responsible for discovering and invoking operations on the devices with weather and manual parameters. The Invoker class is defined as a java bean and has an instance of the DPWS Client. The use of the Invoker as a helper class is because there was a conflict for declaring the client directly as a Spring bean in the application context. The main methods in this client are:

Method	Description
discoverDevices()	This method is used for device discovery and listening to hello messages.
deviceFound(DeviceReference, SearchParameter)	This method is used to add the device when found into the device list, so that it can be easily retrieved.
sendWeather()	This method is used to send the weather parameters to the devices.
setProfile(profile)	This method is used to manually send manual profile value to the device.
setManualMode(mode)	This method is used to set manual or automatic mode in the device.
setLamps(lamp1, lamp2, lamp3, lamp4)	This method is used for sending manual values to the device.
resetEnergy()	This method is used to invoke the reset operation in the reset service, in order to reset the energy readings.

Table 13, *DPWS Client methods*

4.8.5. Chart Interface

The Chart interface implements getMultiChart() and drawKPI() methods to create a ChartData in JSON format understandable by Google-charts. The ChartResponse is a custom class that contains a status string so that the browser is aware if data is available or not for the specified timeframe and, it also contains a ChartData object that has lists for column and rows in the specific Google-charts format as shown in Figure 46. The chart implementation creates columns and inserts the data into rows, then creates a ChartResponse object with status and data. Finally this response is converted into JSON format using google.GSON library and returned as string to be further plotted on the user side.

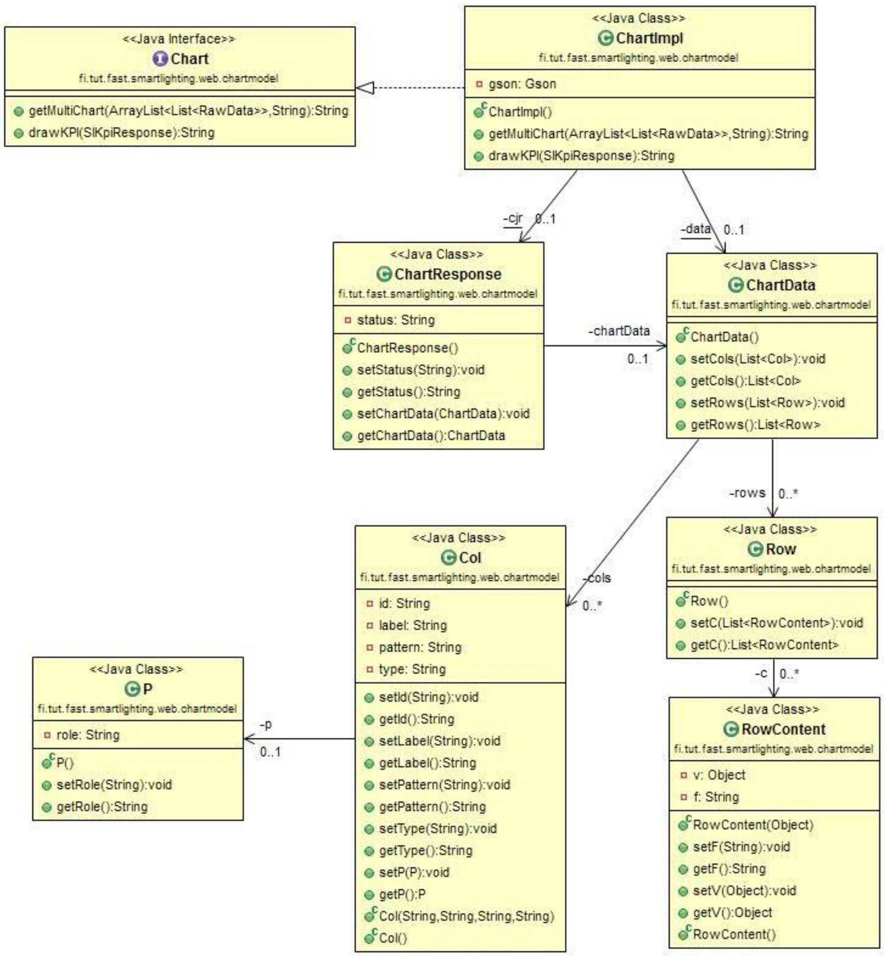


Figure 46, *Chart Interface UML Diagram*

4.8.6. Valopaa Manager

The Valopaa Manager is a custom classes made to facilitate the control of the lamps using simple java methods. The manager utilizes a custom made HTTPS client for making the POST requests.

The HTTPs custom client has custom methods for printing the response and disabling certificate validation. Although disabling certificate validation is not the most secure solution but, it was required for making the https POST requests. Further investigation should be done for using SSL secure connections.

Method	Description
setLamps(profile, level1, level2, level3)	This method is used for controlling all lamps by giving the profile value and desired lighting levels sent in the control message. The number of POST request and addresses depend on the groups required to be controlled.
getRequestString (groupaddress, level)	This method is used to generate the request string in a format that is understandable by the Valopaa master unit.
sendPost (request)	This method uses the custom Https client for sending the post request desired.

Table 14, *Valopaa Manager Methods*

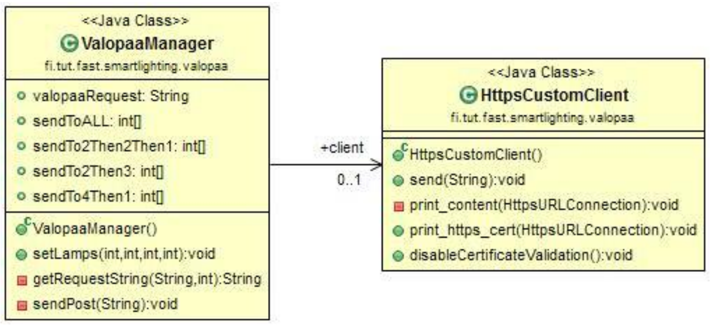

<<Java Class>>
ValopaaManager
fi.tut.fast.smartlighting.valopaa
valopaaRequest: String
sendToALL: int[]
sendTo2Then2Then1: int[]
sendTo2Then3: int[]
sendTo4Then1: int[]
ValopaaManager()
setLamps(int,int,int,int):void
getRequestString(String,int):String
sendPost(String):void
+client
0..1
<<Java Class>>
HttpsCustomClient
fi.tut.fast.smartlighting.valopaa
HttpsCustomClient()
send(String):void
print_content(HttpsURLConnection):void
print_https_cert(HttpsURLConnection):void
disableCertificateValidation():void

Figure 47, *Valopaa Manager UML Diagram*

4.8.7. Services

Data is retrieved from the database and the KPI engine using defined WS. Services are defined in WSDL files located on the network. wsimport is a built-in java tool used to generates JAX-WS services and interfaces from WSDL. These services and interfaces are then used for data retrieval through request and response. There are two kind of service defined in this implementation:

4.8.7.1 Raw Data Service:

It is used to get the raw data directly from the database. The service is requested by creating a SlRawDataRequest with the required parameters such as parameter, start date and end date. A SlRawDataResponse is responded back containing a list of RawData object that has all requested parameters. The RawData object contains a raw energy key that assigns each value to certain lamp and timestamp.

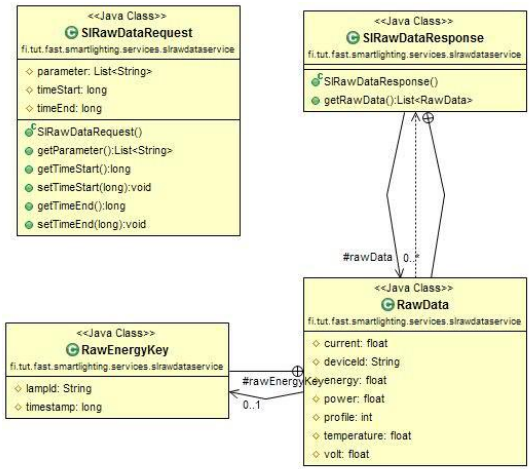


Figure 48, *SlRawData Service request/response UML Diagram*

4.8.7.2 KPI Service:

This service is used to get KPI values from the KPI engine. Similarly to the previous service a SlKPIRequest is made with the required parameters. The Response contains a list of KpiData which is defined by a specific KPI id, timestamp and KpiValue object that defines the consumer and numerical value of the KPI.

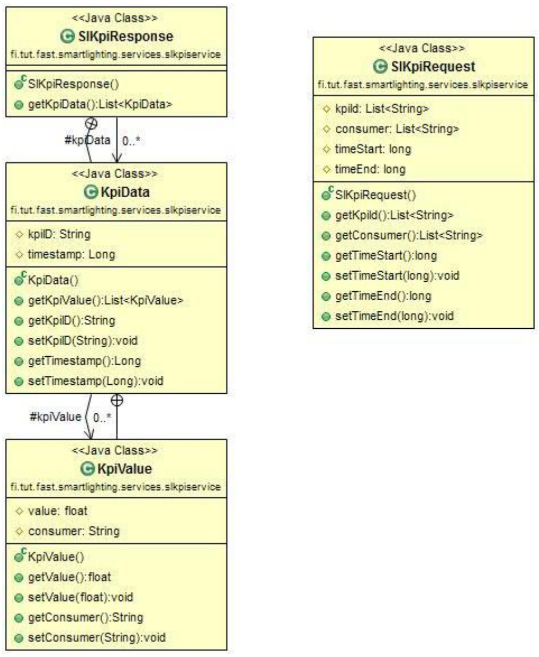


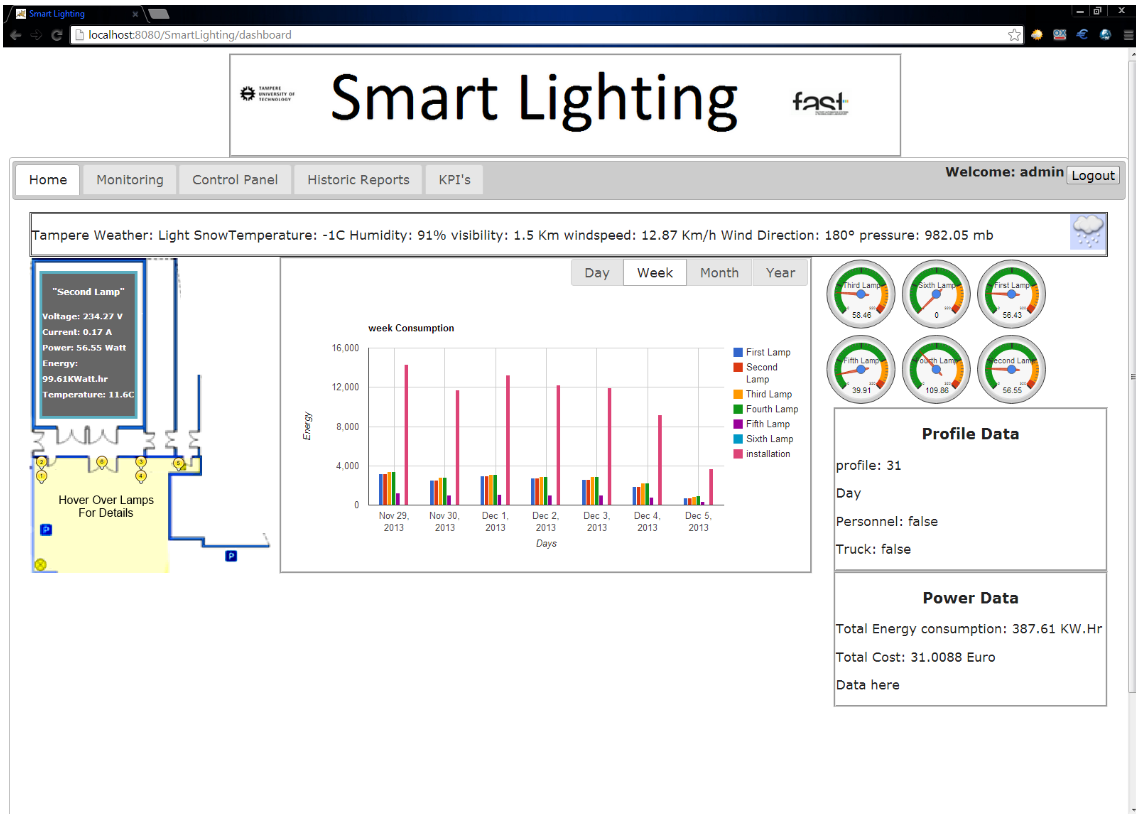
Figure 49, *SlKPI Service request/response UML Diagram*

4.8.8. Dashboard Visualisation

The main purpose of the visualisation is to present the real time data in an easy way for all the user types: administration, management, operators and researchers. The application is easy to use and has many data representation methods such as widgets, gauges and charts. These methods make it easier for the user to interpret what these data represents.

4.8.8.1 Home page visualisation

The home screen shown in Figure 50 is the first page shown to the user and it contains all the basic information to be displayed. There is a weather gadget on the top part showing the current weather condition. There is a map on the left hand side showing the lamps position and, it can show data for each individual lamp just by hovering the mouse over. The middle there is a chart showing the energy consumption per day, week, month or year depending on the user choice. There are 6 gauges on the right hand side showing real time power consumption by each lamp in watt. Additionally there are profile data and power data sections displaying profile and energy data respectively. All sections in this page are sortable so the user can change the layout to any view that comforts his need.



Tampere Weather: Light SnowTemperature: -1C Humidity: 91% visibility: 1.5 Km windspeed: 12.87 Km/h Wind Direction: 180° pressure: 982.05 mb

"Second Lamp"
Voltage: 234.27 V
Current: 0.17 A
Power: 56.55 Watt
Energy: 99.61KWatt.hr
Temperature: 11.6C

Profile Data

profile: 31
Day
Personnel: false
Truck: false

Power Data

Total Energy consumption: 387.61 KW.Hr
Total Cost: 31.0088 Euro
Data here

Figure 50, Home tab screenshot

4.8.8.2 Monitoring page visualisation

On the monitoring tab shown in Figure 51 all metering data are displayed in real time. In the right side there are 6 gauges showing real time power for each lamp and, one gauge for displaying total current consumed by the system in ampere. There is a real time plot chart on the left plotting any parameter the user chooses from the drop box.

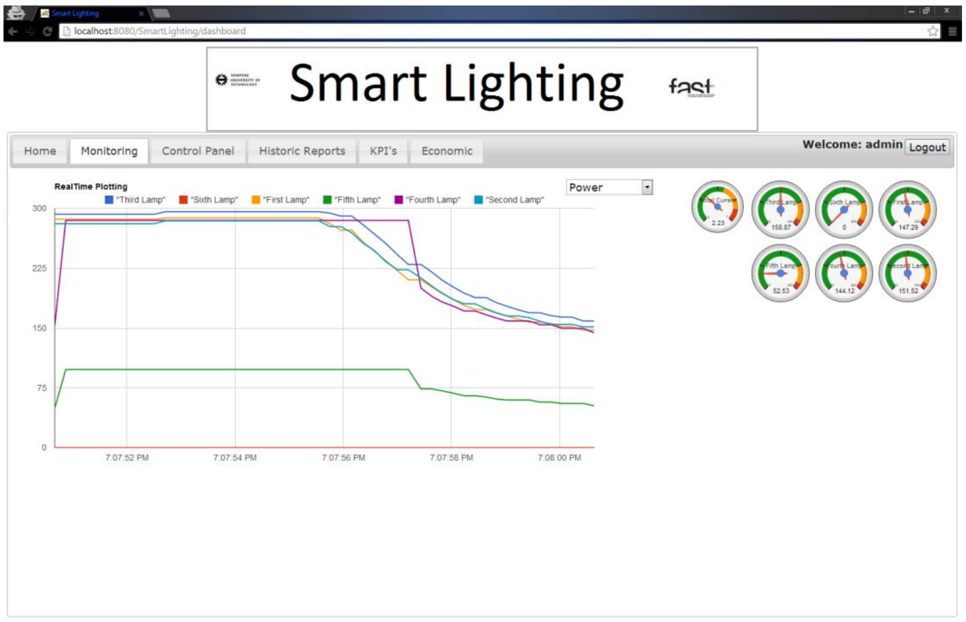


Figure 51, Monitoring tab screenshot

4.8.8.3 Control panel visualisation

The control panel shown in Figure 52gives the user control over the lighting system and to override the light levels. On the right side there is a small control panel with some sliders and buttons to override the system lighting levels, profile or reset the energy. On the right side there is a real time column chart showing the consumption by each lamp this to help the operator in taking the decision. Currently all the parameters are sent to the device through the manual service but, the manual mode is not implemented yet, it should be added in the future.

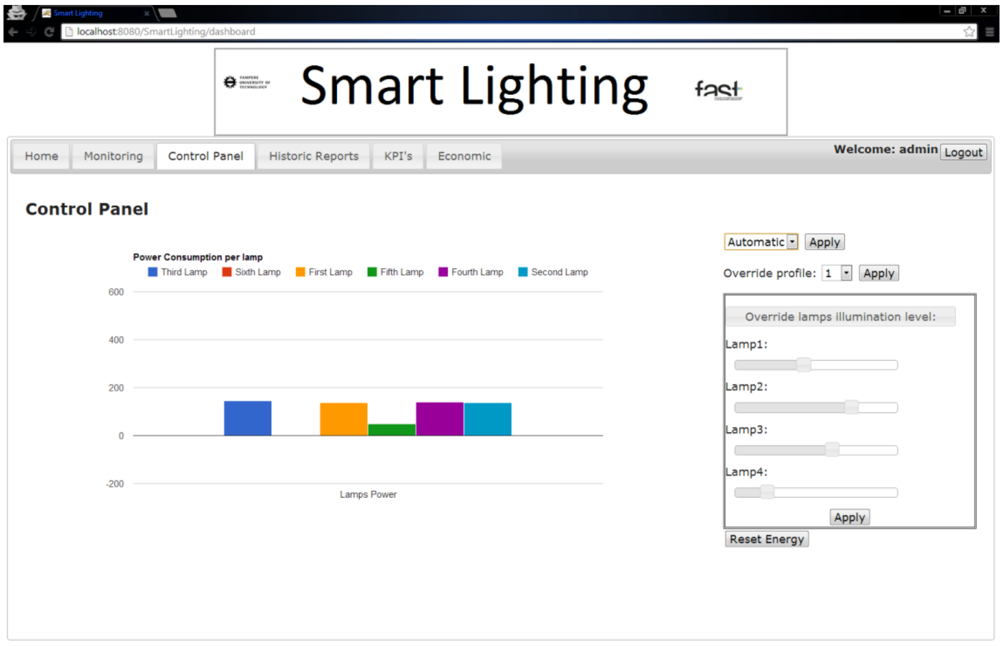
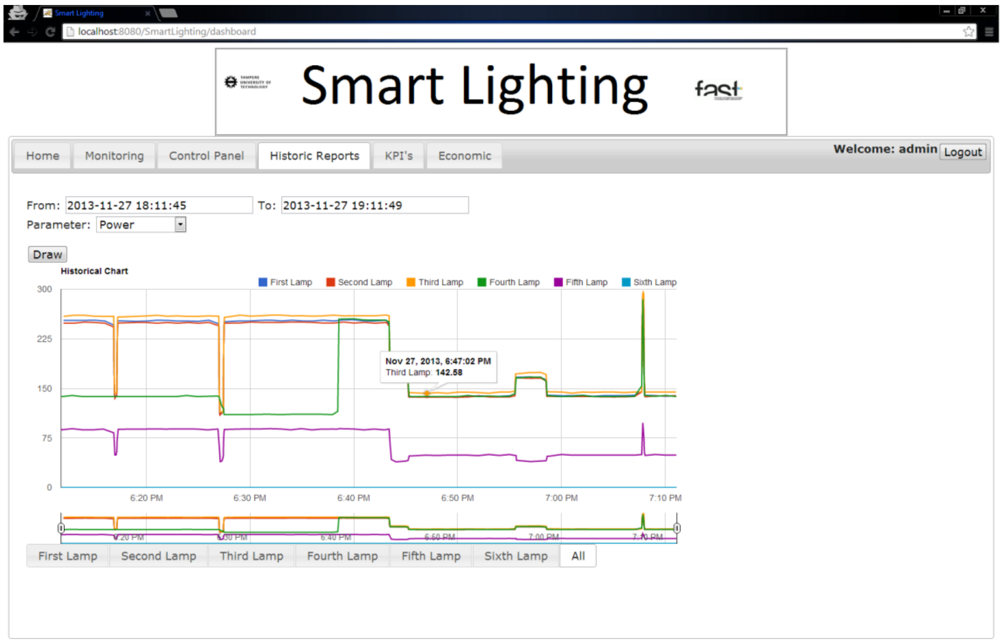


Figure 52, *Control panel tab screenshot*

4.8.8.4 Historical data visualisation

Historical values are retrieved from the data base using the raw data service (4.8.7.1) and drawn using google charts as shown in Figure 53. The user can control the time frame and the parameter to be shown. In addition there is radio button to display each lamp separately.

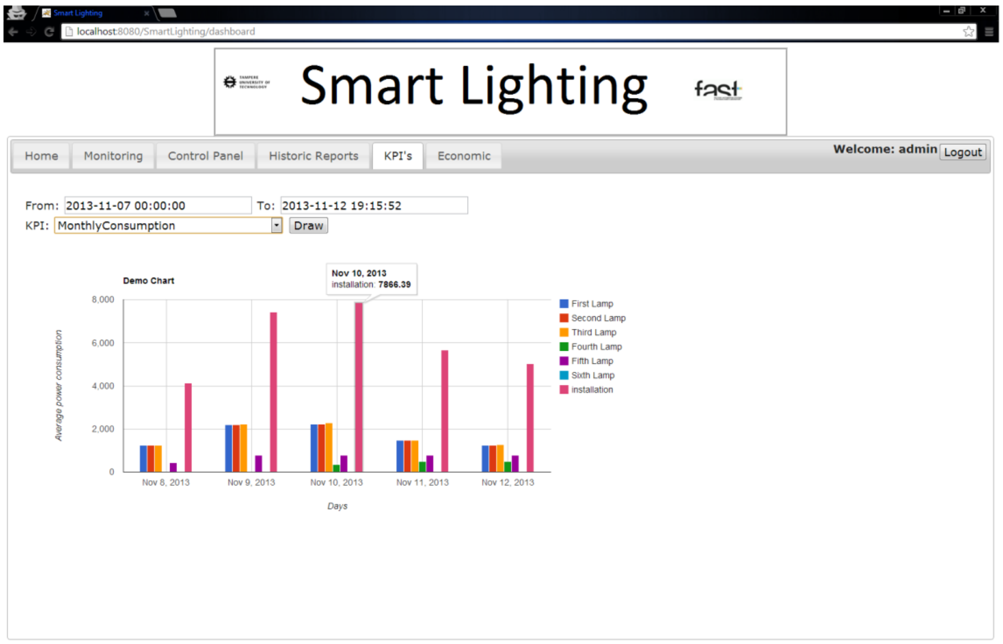

From: 2013-11-27 18:11:45 To: 2013-11-27 19:11:49
Parameter: Power

Nov 27, 2013, 6:47:02 PM
Third Lamp: 142.58

Figure 53, *Historical Reports tab screenshot*

4.8.8.5 KPIs visualisation

KPIs defined in section 4.8.8 are visualised in this tab. The user can choose a specific time frame and the KPI desired then the server utilizes the KPI service (4.8.7.2) to retrieve the data. An example of retrieving the data is shown in Figure 54.


From: 2013-11-07 00:00:00 To: 2013-11-12 19:15:52
KPI: MonthlyConsumption

Nov 10, 2013
installation: 7866.39

Figure 54, *KPI tab screenshot*

5. RESULTS

The implementation presented a proof of concept for the applicability of the IoT in a distributed control lighting solution. This section describes the tests conducted and the results for this implementation.

5.1. Scheduled Control

Tests were carried out for a certain period of time and, a comparison was conducted between scheduled and controlled mode. The scheduled lighting level was designed as a reference benchmark for energy consumption because, there were no past energy bills or readings that can be used.

According to the schedule the lamps should work with 90% lighting capacity from sunset until midnight then, the levels dim down to 30% until they turn off completely by sunrise. This schedule showed a constant consumption of 12.4kW.hr.

5.2. Automatic Control

Then automatic control was applied to the lights starting November 29th. During the testing period the average day length was six hours and the weather was almost clear. Daily energy consumption (SL004) for automatic controlled is plotted in Figure 55, where the vertical and horizontal axes represent the energy consumed in Watt.hr and days respectively. It was clear in the beginning the consumption was around 12kW.hr. The lighting levels in Table 9 was reassigned and the waiting time was reduced to 10s for fine tuning, consequently starting December 4th the energy consumption was reduced significantly to 8.4kW.hr. A problem in the KPI calculation was noticed during December 5th and 6th, thus readings are ignored for these two days.

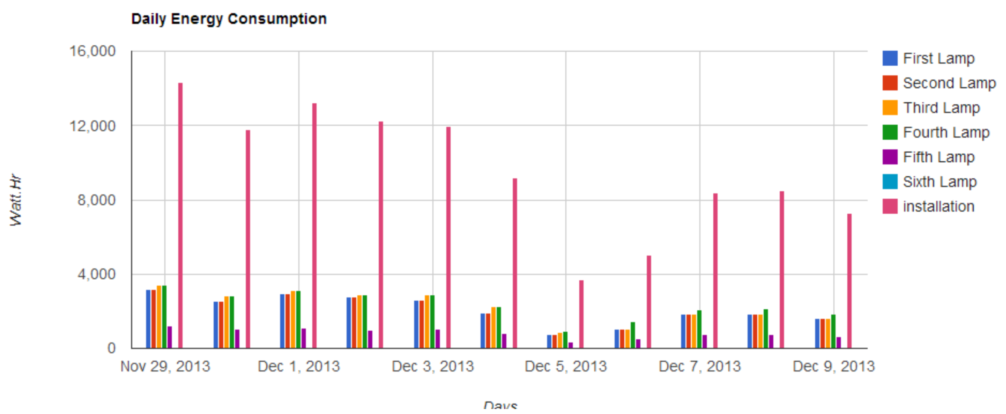


Figure 55, *Daily energy consumption for automatic control*

It can easily be deduced from FIG that, the environmental impact of the system has been reduced from 9Kg of CO2 per day to only 6kg recalibrating the parameters. Moreover from the plot of SL005 the daily cost is observed to be at 0.67 euro.

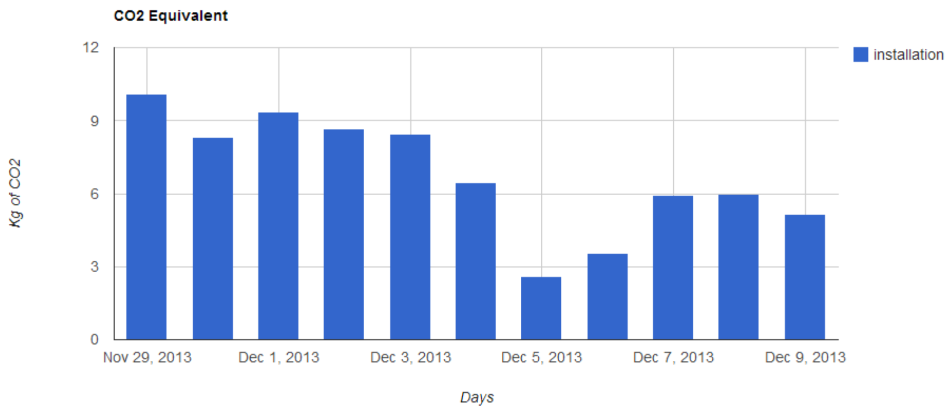


Figure 56, *CO2 equivalent plot*

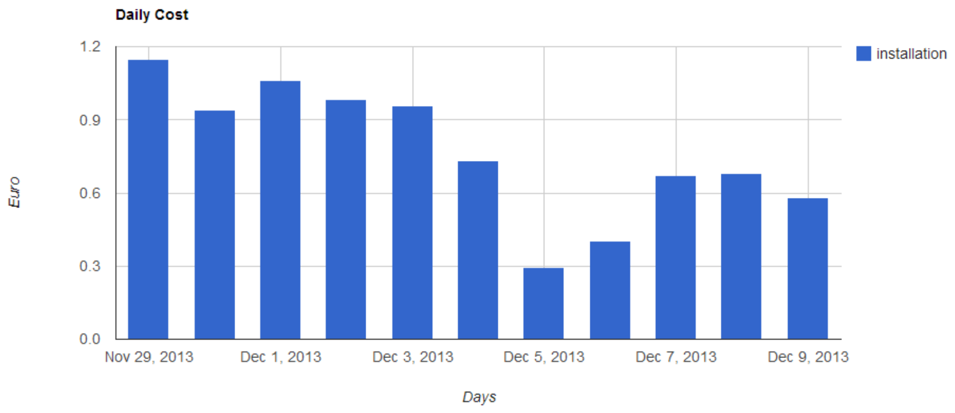


Figure 57, *Cost per day in euros*

5.3. Comparison of results

A comparison between scheduled and controlled control from several KPIs point of views is concluded in Table 15. The controlled system showed enhanced results in comparison to scheduled one, with efficiency of 33.62%. The energy efficiency is calculated by using equation 5.1, where “E” stands for energy. Similarly rest of parameters can be applied to the equation. The energy consumed is variables of the number of people going through, weather condition and day light length, so the efficiency is variable. More tests under different conditions will be conducted in the future work.

$$\eta_{\%} = \frac{E_{scheduled} - E_{controlled}}{E_{scheduled}} \times 100 \qquad (5.1)$$

Criteria per day	Scheduled	Controlled	Efficiency
Energy	12.655 kW.hr	8.4 kW.hr	33.62%
Average Power	104.845 W	84.678 W	12.24%
LPD	0.68 W/m^2	0.553 W/m^2	12.24%
CO_2 Equivalent	8.929 Kg	5.921 Kg	33.62%
Cost	1.012 €	0.672 €	33.62%

Table 15, *Scheduled and Controlled Comparison*

6. CONCLUSION

This section discusses the results of the implementation and highlights the objectives that have been achieved. Then it states future work required for further improvement in order to extend the implementation, achieve better response time and energy savings.

6.1. Conclusion of Implementation and Results

The automatic control of the lighting system provided the adequate amount of lighting at minimum energy consumption. The variables involved in the control covered the main parameters affecting the visibility of the user. However it showed unsatisfactory results for the comfort of the personnel and, this is due to the long response time. The downside for this implementation is the relatively high response time .The response time could be improved by allowing the controller to directly control the lamps without involvement of any third parties in the middle. Direct control was not available during the thesis work due to some complications in supplying of the gateway and modification of the lamp.

Communications between objects is essential aspect for the IoT paradigm. The Implementation of SOA for smart lighting application facilitated the interoperability between devices. Communications between devices in the system were done seamlessly, using messages defined in the internal service. All devices are aware of the current surroundings and lamps energy consumptions. This collaboration can be used for extending the control to several areas and more lamps. The use of a separate service for control helped in reducing the response time. However it is still recommended that devices directly control lamps without any intermediate nodes.

The use of DPWS facilitated automatic discovery and subscription to the metering devices. Energy messages provided the main data required for monitoring of the system. Thus the smart lighting implementation demonstrated an outstanding metering model based on DPWS. The model can be easily adapted for smart cities and, integrated in a smart grid in the future. Where utility companies easily subscribe to the DPWS meters and gather data autonomously.

The sampling rate for publishing the energy message is based on two methods: variable and fixed sampling rates. The fixed rate is constant time period for publishing the message while, variable rate is event based. The combination of these two methods reduces network congestion and processing resources by the server and database. Additionally this combination also offers a reliable message delivery and detection of significant power changes without data loss or noise.

The KPI plots generated in the application can assist the management level in taking decisions in order to improve the control strategy and reduce the environmental impact. The KPIs defined gives a holistic view of the performance of the system and energy behaviour. The KPI data are retrieved from the database using defined WS. The KPI data are plotted using Google charts library for representation.

For monitoring and analysing the results a web based dashboard application was developed. The dashboard was design such that it serves different kind of users with different purposes. Moreover the application is user friendly and has an easy and informative interface that can be understood without guiding.

The server receives several messages and parses them automatically, regardless of the device type or vendor. New meter devices can be added without any pre-configuration required as long as, it supports DPWS and have same message format.

The implementation showed astonishing results for energy savings. The energy consumption was compared to some predefined schedule as a benchmark. An energy efficiency of 33.6% was achieved. Tests were performed mostly for clear and cloudy weathers. The efficiency is variable depending on conditions during the tests were performed. Environmental impact and CO_2 equivalent were reduced subsequently.

The implementation presented a use case for smart lighting application. Also the implementation has proven the feasibility of applying the IOT for smart lighting application and, stated the drawbacks. Finally this thesis work has already contributed in the scientific society, as it facilitated publishing of paper [50] as a proceeding of the 4th international workshop CONET2013.

6.2. Future Work

This thesis work was performed to present a model for control and monitor of lighting installation. Due to the time constrain for this thesis it was difficult to test all possible conditions. The average day length was six hours and, the weather was almost clear. Only under these conditions the implementation was tested, thus it is advised to perform more test and analysis under different conditions.

The security subject was out of focus for this thesis work however, it is suggested applying Https requests in the future. Additionally more specific KPIs can also be added to have more generic view of the system performance. Furthermore it is advised to perform direct control from the devices over the lamps without intermediate nodes. This will help in reducing the response time and improve user experience. Moreover it will allow achieving distributed control using multiple devices thus, extension of the implementation areas.

The temperature affects the lifetime of the lamps, thus it is also advised to take into account lamps temperature in the control algorithm. This will allow reducing the dissipated energy through heating. Finally currently only automatic control is implemented however, in future work it is advised to apply manual control in order to override the system for meeting user needs and fine tuning.

REFRENCES

[1] "Energy efficient Europe," 22 June 2011. [Online]. Available: http://ec.europa.eu/news/energy/110622_en.htm. [Accessed 22 August 2013].

[2] Aalto University School of Science and Technology, Guidebook on Energy Efficient Electric Lighting for Buildings, Raisio: IEA publications, 2010.

[3] European Research Projects on the Internet of Things (CERP-IoT), "Internet of Things -Strategic Research Roadmap," 2009.

[4] I. L. Azevedo, M. G. Morgan and F. Morgan, "The Transition to Solid-State Lighting," *proceedings of the IEEE,* vol. 97, no. 3, pp. 481-510, 2009.

[5] M. Shur and A. Zukauskas, "Solid-State Lighting: Toward Superior Illumination," *Proceedings of the IEEE,* vol. 93, no. 10, pp. 1691-1703, 2005.

[6] N. Holonyak and S. F. Bevacqua, "COHERENT (VISIBLE) LIGHT EMISSION FROM Ga(As1−xPx) JUNCTIONS," *Appl. Phys. Lett.,* vol. 1, no. 4, pp. 82-83, 1962.

[7] S. Kunić and Z. Šego, "OLED technology and displays," in *ELMAR,* Zadar, 2012.

[8] N. Bardsley, "OLED vs LED lighting: Is there room for OLED lighting?," 29 May 2013. [Online]. Available: http://www.idtechex.com/research/articles/oled-vs-led-lighting-is-there-room-for-oled-lighting-00005477.asp. [Accessed 1 November 2013].

[9] M. Miki, T. Hiroyasu and K. Imazato, "Proposal for an intelligent lighting system, and verification of control method effectiveness," *Cybernetics and Intelligent Systems,* vol. 1, pp. 520-525, 2004.

[10] R. Karlicek, "Smart lighting - more than illumination," in *Asia Communications and Photonics Conference (ACP),* Guangzhou, 2012.

[11] A. Siddiqui, A. Ahmad, H. K. Yang and C. Lee, "ZigBee based energy efficient outdoor lighting control system," in *14th International Conference on Advanced Communication Technology (ICACT),* PyeongChang, 2012.

[12] A. Lavric, V. Popa and I. Finis, "The Design of a Street Lighting Monitoring and Control System," in *International Conference and Exposition on Electrical and Power Engineering (EPE),* Iasi, 2012.

[13] N. Zotos, C. Stergiopoulos, K. Anastasopoulos, G. Bogdos, E. Pallis and C. Skianis, "Case Study of a dimmable outdoor lighting system with intelligent management and remote control," in *International Conference on*

Telecommunications and Multimedia (TEMU), Chania, 2012.

[14] Entertainment Services & Technology Association (ESTA)ANSI E1.3 - 2001 (R2006), *Entertainment Technology - Lighting Control Systems - 0 to 10V Analog Control Specification.*

[15] R. S. Simpson, Lighting control: Technology and Applications, Oxford: Focal Press, 2003.

[16] T. I. C. P. Committee, *Lighting Control Protocols,* New York: Illuminating Engineering Society of North America, 2011.

[17] Elation Professional, "DMX 101: A DMX 512 HANDBOOK," May 2008. [Online]. Available: http://www.elationlighting.com/pdffiles/dmx-101-handbook.pdf. [Accessed 24 July 2013].

[18] W. Swan, "The Language of BACnet-Objects," *Engineered Systems,* vol. 13, no. 7, pp. 24-32, 1996.

[19] National Electrical Manufacturers Association (NEMA), *Digital Addressable Lighting Interface (DALI) Control Devices Protocol - PART 1-2004 - General Requirements, Draft V1.13.*

[20] Digital Addressable Lighting Interface Activity Group (DALI AG), "DALI-AG," 2011. [Online]. Available: http://www.dali-ag.org/c/manual_gb.pdf. [Accessed 24 July 2013].

[21] D. Parmenter, Key Performance Indicators (KPI): Developing, Implementing, and Using Winning KPIs, 2nd ed., John Wiley & Sons, Inc, 2010.

[22] New Building Institute (NBI), "Key Performance Indicators for Commercial Buildings," [Online]. Available: http://newbuildings.org/sites/default/files/What_is_a_KPI_1.pdf. [Accessed 30 July 2013].

[23] C. Lohman, L. Fortuin and M. Wouters, "Designing a performance measurement system: A case study," *European Journal of Operational Research,* vol. 156, no. 2, p. 267–286, 2004.

[24] M. Deru, N. Blair and P. Torcellini, "Procedure to Measure Indoor Lighting Energy Performance," National Renewable Energy Laboratory (NREL), Springfield, 2005.

[25] Technical Committee CEN/TC 169 "Light and Lighting", *PrEN 15193: Energy performance of buildings — Energy requirements for lighting,* 2006.

[26] L. Martirano, "Lighting systems to save energy in educational classrooms," in *10th International Conference on Environment and Electrical Engineering (EEEIC),* Rome, 2011.

[27] Energy Star, "What is energy use intensity (EUI)?," [Online]. Available: http://www.energystar.gov/buildings/facility-owners-and-managers/existing-buildings/use-portfolio-manager/understand-metrics/what-

energy?fuseaction=buildingcontest.eui. [Accessed 8 November 2013].

[28] J. C. Van Gorp, “Using Key Performance Indicators to Manage Energy Costs,” 2005. [Online]. Available: http://hdl.handle.net/1969.1/5643. [Accessed 7 October 2013].

[29] D. Harris and C. Higgins, “Key Performance Indicators and Analysis for Commercial Buildings with System Level Data,” in *Summer Study on Energy Efficiency in Buildings (ACEEE)*, Pacific Grove, CA, 2012.

[30] K. Ashton, “That 'Internet of Things' Thing,” 22 June 2009. [Online]. Available: http://www.rfidjournal.com/articles/view?4986. [Accessed 05 August 2013].

[31] S. Agrawal and M. Das, “Internet of Things — A paradigm shift of future Internet applications,” in *Nirma University International Conference on Engineering (NUiCONE)*, Ahmedabad, Gujarat, 2011.

[32] Y. Li and Y. Qian, “Regional model of internet of things based on urban lighting,” in *3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, Beijing, 2010.

[33] L. Wan, D. Sun and J. Deng, “Application of IOT in building energy consumption supervision,” in *International Conference on Anti-Counterfeiting Security and Identification in Communication (ASID)*, Chengdu, 2010.

[34] R. v. Kranenburg and A. Bassi, “IoT Challenges,” *Communications in Mobile Computing*, vol. 1, no. 9, 2012.

[35] B. Proffitt, “What's Holding Up The Internet Of Things,” 14 June 2013. [Online]. Available: http://readwr.it/qWr. [Accessed 09 August 2013].

[36] M. Palacios, J. García-Fanjul and J. Tuya, “Testing in Service Oriented Architectures with dynamic binding: A mapping study,” *Information and Software Technology*, vol. 53, no. 3, pp. 171-189, 2011.

[37] W3C, “Web Services Architecture,” 11 February 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/. [Accessed 7 October 2013].

[38] J. Minor, *Bridging OPC UA and DPWS for Industrial SOA*, Tampere: Tampere University of Technology, 2011.

[39] W3C, “SOAP Version 1.2 Part 1: Messaging Framework,” 24 June 2003. [Online]. Available: http://www.w3.org/TR/2003/REC-soap12-part1-20030624. [Accessed 7 October 2013].

[40] Oracle, “The Java EE 5 Tutorial,” Oracle, 2010. [Online]. Available: http://docs.oracle.com/javaee/5/tutorial/doc/bnbhg.html. [Accessed 7 October 2013].

[41] W3C, “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language,” 10 November 2003. [Online]. Available: http://www.w3.org/TR/2003/WD-wsdl20-20031110. [Accessed 7 October 2013].

[42] A. Cheung, “Web APIs - Service-oriented Architecture,” 19 July 2013. [Online].

Available: http://www.ansoncheunghk.info/article/web-apis-service-oriented-architecture. [Accessed 7 October 2013].

[43] OASIS, “Devices Profile for Web Services Version 1.1,” 1 July 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.pdf. [Accessed 7 October 2013].

[44] Microsoft Corporation, “Introducing Devices Profile for Web Services,” 2007. [Online]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=20902. [Accessed 7 October 2013].

[45] V. Altmann, J. Skodzik, F. Golatowski and D. Timmermann, “Investigation of the use of embedded Web Services in smart metering applications,” in *38th Annual Conference on IEEE Industrial Electronics Society (IECON)*, Montreal, QC, 2012.

[46] S. Karnouskos and A. Izmaylova, “Simulation of web service enabled smart meters in an event-based infrastructure,” in *7th IEEE International Conference on Industrial Informatics (INDIN)*, Cardiff, Wales, 2009.

[47] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. Souza and V. Trifa, “SOA-Based Integration of the Internet of Things in Enterprise Services,” in *IEEE International Conference on Web Services(ICWS)*, Los Angeles, CA, 2009.

[48] W. Zhiliang, Y. Yi, W. Lu and W. Wei, “A SOA Based IOT Communication Middleware,” in *International Conference on Mechatronic Science, Electric Engineering and Computer*, Jilin, 2011.

[49] C. Schroth, “The internet of services: Global industrialization of information intensive services,” in *2nd International Conference on Digital Information Management (ICDIM)*, Lyon, 2007.

[50] A. Florea, A. Farahat, C. Postelnicu, J. L. M. Lastra and F. J. A. Sánchez, “Smart Lighting in Multipurpose Outdoor Environments: Energy Efficient Solution using Network of Cooperating Objects.,” in *4th International Workshop on Networks of Cooperating Objects for Smart Cities 2013 (CONET/UBICITEC)*, Philadelphia, 2013.

[51] Web Services for Devices, “JMEDS Framework Overview,” 24 May 2011. [Online]. Available: http://ws4d.e-technik.uni-rostock.de/wp-content/uploads/2011/05/StackOverview.pdf. [Accessed 2 10 2013].

[52] C. Ibsen and J. Anstey, “Meeting Camel,” in *Camel in Action*, Manning Publications, 2011, pp. 4-6.

[53] Apache Software Foundation, “Apache Camel user guide,” 2010. [Online]. Available: http://camel.apache.org/manual/camel-manual-2.5.0.pdf. [Accessed 2 October 2013].

[54] C. Walls, Spring in Action, New York: Manning Publications, 2011.

[55] Yahoo, “Yahoo Weather RSS Feed,” [Online]. Available: http://developer.yahoo.com/weather/. [Accessed 18 October 2013].

[56] Valopaa Energy Efficient Lighting, “VP3411 LED floodlight fixture,” [Online]. Available: http://www.valopaa.com/led_lighting_products/intelligent_led_lighting_system/vp3411i_led_floodlight. [Accessed 21 September 2013].

[57] Valopaa, “VPS Master unit,” [Online]. Available: http://www.valopaa.com/led_lighting_products/intelligent_led_lighting_system/vps_master_unit. [Accessed 18 October 2013].

[58] US Environmental Protection Agency, “Calculations and References,” 19 September 2013. [Online]. Available: http://www.epa.gov/cleanenergy/energy-resources/refs.html. [Accessed 25 November 2013].

[59] A. Rakar, S. Zorzut and V. Jovan, “Assesment of production performance by means of KPI,” in *Proceedings of the Control* , Bath, 2004.

APPENDIX A – SERVICE DESCRIPTION

This section presents some of the WSDLs used to describe the services consumed by the implementation.

Weather Service:

```
<?xml version="1.0" encoding="UTF-8" ?>
 <definitions name="WeatherService"
targetNamespace="http://www.tut.fi/wsdl/weather"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://www.tut.fi/wsdl/weather"
 xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/">
 <types>
 <xsd:schema targetNamespace="http://www.tut.fi/wsdl/weather"
elementFormDefault="qualified">

    <xsd:element name="Response" type="xsd:string"/>
        <xsd:element name="Weather">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element name="Condition">
                                                        <xsd:complexType>

        <xsd:simpleContent>

        <xsd:extension base="xsd:string">
                <xsd:attribute name="code" type="xsd:string"/>

        </xsd:extension>

        </xsd:simpleContent>
                                                        </xsd:complexType>
                                </xsd:element>
                                <xsd:element name="Temperature">
                                                        <xsd:complexType>

        <xsd:simpleContent>

        <xsd:extension base="xsd:string">

        </xsd:extension>

        </xsd:simpleContent>
                                                        </xsd:complexType>
                                </xsd:element>
                                <xsd:element name="Humidity">
                                                        <xsd:complexType>

        <xsd:simpleContent>

        <xsd:extension base="xsd:string">

        </xsd:extension>
        </xsd:simpleContent>
                                                  </xsd:complexType>
                                 </xsd:element>
                                 <xsd:element name="Wind">
                                                  <xsd:complexType>

        <xsd:simpleContent>

        <xsd:extension base="xsd:string">
                <xsd:attribute name="direction" type="xsd:string"/>

        </xsd:extension>

        </xsd:simpleContent>
                                                  </xsd:complexType>
                                 </xsd:element>
                                 <xsd:element name="Visibility">
                                                  <xsd:complexType>

        <xsd:simpleContent>

        <xsd:extension base="xsd:string">

        </xsd:extension>

        </xsd:simpleContent>
                                                  </xsd:complexType>
                                 </xsd:element>
                                 <xsd:element name="Pressure">
                                                  <xsd:complexType>

        <xsd:simpleContent>

        <xsd:extension base="xsd:string">

        </xsd:extension>

        </xsd:simpleContent>
                                                  </xsd:complexType>
                                 </xsd:element>

                         </xsd:sequence>
                 </xsd:complexType>
        </xsd:element>

                 <xsd:complexType name="ConditionType">
                         <xsd:simpleContent>
                                 <xsd:extension base="xsd:string">
                                         <xsd:attribute name="code"
type="xsd:string"/>
                                 </xsd:extension>
                         </xsd:simpleContent>
                 </xsd:complexType>

 </xsd:schema>

  </types>
        <message name="WeatherMessage">
                <part name="messageK" element="tns:Weather"/>
         </message>
        <message name="ResponseMessage">
                <part name="messageK2" element="tns:Response"/>
        </message>


<portType name="WeatherServicePortType" wse:EventSource="true">
```

```
        <operation name="GetWeather">
                <input message="tns:WeatherMessage"/>
                <output message="tns:ResponseMessage"/>
        </operation>

</portType>

<binding name="WeatherServicePortType" type="tns:WeatherServicePortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />

        <operation name="GetWeather">
                <soap:operation style="document" />
                <wsdl:input>
                        <soap:body use="literal" />
                </wsdl:input>
                <wsdl:output>
                        <soap:body use="literal" />
                </wsdl:output>
        </operation>

</binding>
 <service name="WeatherService">
        <port name="WeatherServicePortType"
binding="tns:WeatherServicePortType">
        <soap:address location="http://192.168.2.62:80/dpws/ws01" />
  </port>
  </service>
  </definitions>
```

Profile Service:

```
<?xml version="1.0" encoding="UTF-8" ?>
 <definitions name="ProfileService"
targetNamespace="http://www.tut.fi/fast/smart_lighting/profile"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://www.tut.fi/fast/smart_lighting/profile"
 xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/">
 <types>
 <xsd:schema targetNamespace="http://www.tut.fi/fast/smart_lighting/profile"
elementFormDefault="qualified">
 <xsd:element name="Profile_change">
                    <xsd:complexType>
                      <xsd:sequence>
                        <xsd:element type="xsd:integer" name="Profile"/>
                        <xsd:element name="Weather">
                          <xsd:complexType>
                            <xsd:sequence>
                              <xsd:element name="Condition">
                                <xsd:complexType>
                                  <xsd:simpleContent>
                                    <xsd:extension base="xsd:string">
                                      <xsd:attribute type="xsd:byte"
name="code"/>
                                    </xsd:extension>
                                  </xsd:simpleContent>
                                </xsd:complexType>
                              </xsd:element>
                              <xsd:element type="xsd:integer"
name="Temprature"/>
                              <xsd:element type="xsd:integer"
name="Humidity"/>
                              <xsd:element name="Wind">
                                <xsd:complexType>
                                  <xsd:simpleContent>
```

```
                                     <xsd:extension base="xsd:float">
                                       <xsd:attribute type="xsd:short"
name="wind_direction"/>
                                     </xsd:extension>
                                   </xsd:simpleContent>
                                 </xsd:complexType>
                               </xsd:element>
                               <xsd:element type="xsd:float"
name="Visibility"/>
                               <xsd:element type="xsd:float" name="Pressure"/>
                             </xsd:sequence>
                           </xsd:complexType>
                         </xsd:element>
                         <xsd:element type="xsd:boolean" name="personnel"/>
                         <xsd:element type="xsd:boolean" name="truck"/>
                         <xsd:element type="xsd:string" name="Day_Night"/>
                         <xsd:element name="Energy">
                                    <xsd:complexType>
                                          <xsd:sequence>
                                            <xsd:element name="Lamp"
maxOccurs="unbounded" minOccurs="0">
                                                    <xsd:complexType>
                                                      <xsd:simpleContent>
                                                            <xsd:extension
base="xsd:integer">
                                                              <xsd:attribute
type="xsd:integer" name="Id" use="optional"/>
                                                            </xsd:extension>
                                                      </xsd:simpleContent>
                                                    </xsd:complexType>
                                            </xsd:element>
                                          </xsd:sequence>
                                    </xsd:complexType>
                                  </xsd:element>

                                  <xsd:element name="sensors">
                                    <xsd:complexType>
                                          <xsd:sequence>
                                            <xsd:element type="xsd:integer"
name="count"/>
                               <xsd:element type="xsd:integer"
name="n_people"/>
                                            <xsd:element type="xsd:float"
name="temperature_s"/>
                               <xsd:element type="xsd:float" name="light_s"/>
                                          </xsd:sequence>
                                    </xsd:complexType>
                                  </xsd:element>

                         <xsd:element type="xsd:string" name="Alarm"/>
                       </xsd:sequence>
                       <xsd:attribute type="xsd:dateTime" name="Timestamp"/>
                     </xsd:complexType>
                   </xsd:element>

 </xsd:schema>

  </types>
        <message name="ProfileMessage">
                <part name="messageK" element="tns:Profile_change"/>
         </message>



<portType name="ProfileServicePortType" wse:EventSource="true">

        <operation name="SendProfile">
                <output message="tns:ProfileMessage"/>
        </operation>

</portType>

<binding name="ProfileServicePortType" type="tns:ProfileServicePortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />

        <operation name="SendProfile">
                <soap:operation style="document" />
                <wsdl:output>
                        <soap:body use="literal" />
                </wsdl:output>
        </operation>

</binding>
 <service name="ProfileService">
        <port name="ProfileServicePortType"
binding="tns:ProfileServicePortType">
        <soap:address location="http://192.168.3.19:80/dpws/ws05" />
  </port>
  </service>
  </definitions>
```

Energy Service:

```
<?xml version="1.0" encoding="UTF-8" ?>
 <definitions name="EnergyService"
targetNamespace="http://www.tut.fi/fast/smart_lighting/energymeter"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://www.tut.fi/fast/smart_lighting/energymeter"
 xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/">
 <types>
 <xsd:schema
targetNamespace="http://www.tut.fi/fast/smart_lighting/energymeter"
elementFormDefault="qualified">
      <xsd:element name="EnergyMeter">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element type="xsd:integer" name="Profile"/>
        <xsd:element name="Lamp" maxOccurs="unbounded" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element type="xsd:float" name="volt"/>
              <xsd:element type="xsd:float" name="amp"/>
              <xsd:element type="xsd:float" name="watt"/>
              <xsd:element type="xsd:short" name="watthr"/>
              <xsd:element type="xsd:float" name="temperature"/>
            </xsd:sequence>
                        <xsd:attribute type="xsd:string" name="ID"
use="optional"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute type="xsd:dateTime" name="Timestamp"/>
          <xsd:attribute type="xsd:string" name="DeviceId"/>
    </xsd:complexType>
  </xsd:element>

 </xsd:schema>

  </types>
        <message name="EnergyMessage">
                <part name="messageK" element="tns:EnergyMeter_2"/>
         </message>
```

```
<portType name="EnergyServicePortType" wse:EventSource="true">

        <operation name="SendEnergy">
                <output message="tns:EnergyMessage"/>
        </operation>

</portType>
<binding name="EnergyServicePortType" type="tns:EnergyServicePortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />

        <operation name="SendEnergy">
                <soap:operation style="document" />
                <wsdl:output>
                        <soap:body use="literal" />
                </wsdl:output>
        </operation>

</binding>
 <service name="EnergyService">
        <port name="EnergyServicePortType"
binding="tns:EnergyServicePortType">
        <soap:address location="http://192.168.3.96:80/dpws/ws01" />
  </port>
  </service>
  </definitions>
```

Sensors Service:

```
<?xml version="1.0" encoding="UTF-8" ?>
 <definitions name="SensorsService"
targetNamespace="http://www.tut.fi/fast/smart_lighting/sensors"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://www.tut.fi/fast/smart_lighting/sensors"
 xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/">
 <types>
 <xsd:schema targetNamespace="http://www.tut.fi/fast/smart_lighting/sensors"
elementFormDefault="qualified">
 <xsd:element name="Sensors">
        <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Sensor" maxOccurs="unbounded" minOccurs="0">
          <xsd:complexType>
                        <xsd:simpleContent>
              <xsd:extension base="xsd:float">
                <xsd:attribute type="xsd:string" name="ID" use="optional"/>
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>

    </xsd:complexType>
                </xsd:element>
 </xsd:schema>
  </types>
        <message name="SensorsMessage">
                <part name="messageK" element="tns:Sensors"/>
         </message>
<portType name="SensorsServicePortType" wse:EventSource="true">

        <operation name="SendSensors">
                <output message="tns:SensorsMessage"/>
        </operation>
```

```
</portType>

<binding name="SensorsServicePortType" type="tns:SensorsServicePortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />

        <operation name="SendSensors">
                <soap:operation style="document" />
                <wsdl:output>
                        <soap:body use="literal" />
                </wsdl:output>
        </operation>

</binding>
 <service name="SensorsService">
        <port name="SensorsServicePortType"
binding="tns:SensorsServicePortType">
        <soap:address location="http://192.168.3.96:80/dpws/ws04" />
  </port>
  </service>
  </definitions>
```

APPENDIX B – MESSAGES

This section shows example for the messages exchanged to achieve the implementation goals.

Example of Energy message:

```
<EnergyMeter xmlns="http://www.tut.fi/fast/smart_lighting/energymeter"
xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" DeviceId="First
Energy Meter" Timestamp="2000-02-17T23:19:04.500">
      <Profile>1</Profile>
      <Lamp ID="First Lamp">
        <volt>234.56</volt>
        <amp>-0.01</amp>
        <watt>0.85</watt>
        <watthr>1894.11</watthr>
        <temperature>50.6</temperature>
      </Lamp>
      <Lamp ID="Second Lamp">
        <volt>235.14</volt>
        <amp>0.00</amp>
        <watt>0.27</watt>
        <watthr>1889.21</watthr>
        <temperature>11.6</temperature>
      </Lamp>
      <Lamp ID="Third Lamp">
        <volt>233.10</volt>
        <amp>0.00</amp>
        <watt>-1.20</watt>
        <watthr>2048.46</watthr>
        <temperature>11.6</temperature>
      </Lamp>
    </EnergyMeter>
```

Example of Profile message:

```
<EnergyMeter xmlns="http://www.tut.fi/fast/smart_lighting/energymeter"
xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" DeviceId="First
Energy Meter" Timestamp="2000-02-17T23:19:04.500">
      <Profile>1</Profile>
      <Lamp ID="First Lamp">
        <volt>234.56</volt>
        <amp>-0.01</amp>
        <watt>0.85</watt>
        <watthr>1894.11</watthr>
        <temperature>50.6</temperature>
      </Lamp>
      <Lamp ID="Second Lamp">
        <volt>235.14</volt>
        <amp>0.00</amp>
        <watt>0.27</watt>
        <watthr>1889.21</watthr>
        <temperature>11.6</temperature>
      </Lamp>
      <Lamp ID="Third Lamp">
        <volt>233.10</volt>
        <amp>0.00</amp>
        <watt>-1.20</watt>
        <watthr>2048.46</watthr>
        <temperature>11.6</temperature>
      </Lamp>
    </EnergyMeter>
```

Example of Control message:

```
<Control xmlns="http://www.tut.fi/fast/smart_lighting/control"
xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
      <profile>9</profile>
      <Group>40</Group>
      <Group>60</Group>
      <Group>100</Group>
    </Control>
```

Example of Sensors message:

```
<Sensors xmlns="http://www.tut.fi/fast/smart_lighting/sensors"
xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
      <Sensor ID="Temperature">-32.76</Sensor>
      <Sensor ID="Light">332.03</Sensor>
    </Sensors>
```